Pose optimized multiple camera systems for vehicle surround-view vision

Puligandla, Venkata Anirudh

Sveučilište u Zagrebu

University of Zagreb
Faculty of Electrical Engineering and Computing

Venkata Anirudh Puligandla

POSE OPTIMIZED MULTIPLE CAMERA SYSTEMS FOR VEHICLE SURROUND-VIEW VISION

DOCTORAL THESIS

Zagreb, 2024

University of Zagreb

University of Zagreb
Faculty of Electrical Engineering and Computing

Venkata Anirudh Puligandla

POSE OPTIMIZED MULTIPLE CAMERA SYSTEMS FOR VEHICLE SURROUND-VIEW VISION

DOCTORAL THESIS

Supervisor: Academician Sven Lončarić, F.C.A.

Zagreb, 2024

Sveučilište u Zagrebu

Sveučilište u Zagrebu
Fakultet Elektrotehnike i Računarstva

Venkata Anirudh Puligandla

Optimizacija Rasporeda Kamera u Sustavima za Panoramsku Vizualizaciju Okoline Vozila

DOKTORSKI RAD

Mentor: Akademik prof. dr. sc. Sven Lončarić

Zagreb, 2024

The doctoral thesis was completed at the University of Zagreb, Faculty of Electrical Engineering and Computing, Department of Electronic Systems and Information Processing.

Mentor: Academician Sven Lončarić, F.C.A.

The thesis has 97 pages.

Thesis number: _____

About the Supervisor

Sven Lončarić received Diploma of Engineering and Master of Science degrees in electrical engineering from the Faculty of Electrical Engineering and Computing in 1985 and 1989, respectively. He received Ph.D. degree in electrical engineering from University of Cincinnati, USA, in 1994. Since 2011, he has been a tenured full professor in electrical engineering and computer science at FER. He was a project leader on a number of research projects in the area of image processing and computer vision. From 2001-2003, he was an assistant professor at New Jersey Institute of Technology, USA. He founded the Image Processing Laboratory at FER and the Center for Computer Vision at University of Zagreb. Prof. Lončarić has been a co-director of the national Center of Research Excellence in Data Science and Cooperative Systems and the director of the Center for Artificial Intelligence at FER. With his students and collaborators he published more than 250 scientific papers. Prof. Lončarić is a Fellow of the Croatian Academy of Sciences and Arts. According to recent Stanford University studies he was ranked in the top 2% of the most cited world scientists in the category artificial intelligence – image processing. For his scientific work he received several awards including the National Science Award.

O mentoru

Sven Lončarić diplomirao je i magistrirao u polju elektrotehnike na Fakultetu elektrotehnike i računarstva, 1985. i 1989. godine. Doktorirao je u polju elektrotehnike na Sveučilištu u Cincinnatiju, SAD, 1994. godine. U zvanje redoviti profesor u trajnom zvanju u polju elektrotehnike i polju računarstva na FER-u izabran je 2011. godine. Bio je suradnik ili voditelj na brojnim istraživačkim i razvojnim projektima u području razvoja metoda za obradu slika i računalnog vida. Od 2001. do 2003. bio je Assistant Professor na Sveučilištu New Jersey Institute of Technology, SAD. Voditelj je istraživačkog laboratorija za obradu slike na FER-u. Osnivač je i voditelj Centra izvrsnosti za računalni vid na Sveučilištu u Zagrebu. Suvoditelj je nacionalnog Znanstvenog centra izvrsnosti za znanost o podatcima i kooperativne sustave i voditelj Centra za umjetnu inteligenciju FER-a. Sa svojim studentima i suradnicima publicirao je više od 250 znanstvenih i stručnih radova. Prof. Lončarić redoviti je član Hrvatske akademije znanosti i umjetnosti. Prema studijama Sveučilišta Stanford rangiran je u 2% najutjecajnijih svjetskih znanstvenika u kategoriji umjetna inteligencija i obrada slike. Za svoj znanstveni i stručni rad dobio je više nagrada uključujući Državnu nagradu za znanost.

Preface

This thesis summarizes the research work conducted during the time between 2019 - 2023 as part of the ImmerSAFE research project funded under the European Union's (EU's) H2020-MSCA-ITN-2017 call, part of the Marie Sklodowska-Curie Actions-Innovative Training Networks (ITN) funding scheme under project 764951, and supervised by Professor Sven Lončarić, PhD.

I would like express my sincere gratitude to Prof. Sven Lončarić, PhD, the head of the Image Processing Group at the Department of Electronic Systems and Information Processing at the Faculty of Electrical Engineering and Computing, University of Zagreb, for offering me the *early-stage researcher* position, and for his continuous support and encouragement throughout the course of the PhD, and also for his valuable contribution, as a co-author, to the research work published during this period. This work would not have been possible without his morale support beyond the professional aspect of my life. I would also like to thank my father, P.V.V.S. Murthy, my mother P.V. Rajyalakshmi, and, my brother, P.V. Kasyap for encouraging me to pursue a doctoral degree and for their continual support in all of my decisions in life. Special thanks to the rest of my family for their valuable support, motivation, and professional and personal tips. I would like to extend my sincere gratitude to Valentina Topolovčan for her support during my time at the University, and through a brief period of uncertainty during the latter half of my doctoral research. Lastly, I would also like to thank my friends and, all the colleagues at the Image Processing Group, FER, University of Zagreb, for livening up my time at FER to make it fun and enjoyable.

Abstract

Advanced Driver Assistance Systems (ADAS) are ubiquitous in present day vehicles. Among various ADAS systems, parking assistance systems offering novel views of the vehicle's surroundings, such as, 360^o and bird's-eye views are becoming a part of all modern vehicles. Presenting novel views involves multiple steps of image processing including, image/video capture, image registration and visualization. Capturing surrounding view requires the placement of multiple cameras on the vehicle as a first step. Precise placement and calibration of the cameras is important as minor errors may lead to significant artefacts during the subsequent steps of image registration and visualization.

Camera placement optimization (CPO) aims to optimize the poses of multiple cameras with an objective to increase the overall coverage of the target area, and/or to reduce the cost of the multiple-camera system. CPO can also eliminate the requirement for camera calibration, as the precise pose of the cameras is already estimated during the optimization step. Although CPO problems are well-studied for surveillance scenarios, there exists a dearth of literature in the context of applications to vehicle surrounding view capture. Compared to surveillance scenarios, CPO problems for vehicle surround view capture need to address additional challenges posed by the complex, non-convex structure of vehicles, and the requirement of a high degree of accuracy in the estimated camera's pose. CPO problems are simulated in discrete space by sampling the continuous space. Although modelling in discrete space is the favoured approach for their simplicity, few works use continuous space models or a mix of both for added accuracy.

The scope of this work includes CPO problem formulation for surround-view coverage for vehicles, CPO problem definitions in discrete as well as continuous space domains, and proposing a new heuristic algorithm to improve the performance of existing optimization algorithms. Firstly, new contributions are made towards formulating the CPO problem for surround-view coverage using a 3D discrete space model. A novel *multi-resolution* heuristic optimization algorithm is proposed to significantly improve the performance of existing discrete optimization algorithms. The CPO problem is then reformulated in the continuous space domain and compared against the discrete-space variant to highlight improved accuracy. Lastly, a super-voxel segmentation method, which was tailored for use in the *multi-resolution* optimization method, is introduced and validated on well-known 3D point cloud datasets. Experiments and simulation results on high-resolution 3D models of a variety of vehicles show that the proposed methods are effective in optimizing camera poses of multiple cameras for vehicle surround-view, meeting the demands of real-world scenarios in a reasonable amount of time.

Keywords: camera placement optimization, global optimization, image segmentation, computational geometry, 3D visualization

Optimizacija Rasporeda Kamera u Sustavima za Panoramsku Vizualizaciju Okoline Vozila

Razni napredni sustavi pomoći vozaču (ADAS) sve se više dodaju u sve vrste vozila. Sustavi pomoći pri parkiranju neki su od najpopularnijih ADAS sustava koji se koriste u današnjim vozilima. Sustavi pomoći pri parkiranju razvili su se iz jednostavnog ultrazvuka senzora dometa postavljeni na donji stražnji kraj vozila za otkrivanje objekata u blizini i upozoravanje operatera, do sustava s više kamera postavljenih oko vozila za snimanje okolnog pogleda i predstaviti nove poglede operaterima vozila. Sustavi s više kamera obično se sastoje od četiri kamere s ribljim okom postavljene na četiri strane vozila za snimanje pogleda od 360 stupnjeva na okruženje. Snimljeni videozapisi spajaju se i prikazuju zajednički prostor iz kojeg se novi prikazi kao što su pogled straga, pogled odozgo ili pogled iz ptičje perspektive, generiraju i prikazuju operateru kao pomoć pri upravljanju vozilom. Dok za mala vozila (npr. automobile) raspored kamere se sastoji od četiri kamere na četiri strane, za veća vozila (npr. kamion, teška vozila, građevinska oprema itd.) tako jednostavan raspored možda neće biti dovoljan za snimiti pogled od 260 stupnjeva. Pronalaženje optimalnog položaja i orijentacije kamere za surround prikaz s više kamera sustav snimanja za velika vozila može uključivati naporan proces od nekoliko ponavljanja ispitivanja i greška. Optimizacija postavljanja kamere (CPO) je tema u kojoj je cilj pronaći optimalni raspored za postavljanje više kamera u definirani prostor, s ciljem da se minimaliziraju troškovi sustava s više kamera ili, kako bi se povećala pokrivenost definiranog ciljnog područja ili, kako bi se postigla oba cilja istovremeno. CPO problemi su detaljno proučavani za različite primjene uključujući nadzor, praćenje materijala, praćenje ljudskog kretanja, snimanje ljudskog pokreta, 3D rekonstrukcija itd. u posljednja dva desetljeća. Unatoč svom književnom radu, CPO problemi još nisu proučavani u kontekstu snimanja vozila iz okruženja. S povećanjem broj senzora integriranih u vozila, CPO za pokrivenost prostornim pogledom može pomoći ADAS sustavima za poboljšanje kvalitete pokrivenosti i smanjenje troškova sustava s više kamera. CPO problemi sastoje se od pojedinačnih koraka stvaranja simuliranog prostora s regijama definiranim za postavljanje kamera i ciljne regije koje moraju biti pokrivene postavljenim kamerama, prikupljanje varijabli odluke iz definiranog prostora za proces optimizacije i definiranje objektivne funkcije koju je potrebno optimizirati kako bi se pronašle optimalne pozicije kamere. Za npr. u tipičnom scenariju unutarnjeg nadzora, simulirani prostor je modeliran kao tlocrt prostorije ili više soba. U najjednostavnijem slučaju, simulirani prostor sadrži samo zidove za pregradu prostora i nema drugih prepreka. U tom prostoru unutarnje površine zidova djeluju kao moguće lokacije za postavljanje kamera, dok je pod ciljno područje koje treba pokriti sa postavljenim kamerama. Točke se uzorkuju iz odgovarajućih regija kako bi predstavile diskretne skupove kandidatskih kamera i kontrolnih točaka, koje postaju dio problema optimizacije kao primarne varijable odluke. Primjer funkcije cilja za CPO problem može

biti definiran kao maksimiziranje broja kontrolnih točaka pokrivenih postavljenim kamerama. Ostala ograničenja kao što je nedopuštanje postavljanja više od jedne kamere na jednu uzorak sa zidova, maksimalno ograničenje ukupnog broja kamera koje se mogu postaviti itd., dodaju se kako bi se dovršio CPO problem. Metode za optimizaciju funkcije cilja uključuju točni i heuristički algoritmi. Uobičajen pristup rješavanju problema je korištenje binarnog programa formulacija temeljenog na cjelobrojnom programiranju. Iako je ovu formulaciju NP-teško riješiti, linearna funkcija cilja i linearna ograničenja omogućuju rješavanje problema unutar razumne količine vremena. Problem i prostor za pretraživanje moraju biti mali da bi se koristili točni optimizacijski algoritmi. Algoritmi točne optimizacije, kao što je algoritam grananja i vezanja, teoretski je zajamčeno da će postići globalni optimum kada je osigurano dovoljno resursa i vremena računanja. Međutim, složenost algoritma eksponencijalno raste s brojem the varijabli odluke i ograničenja problema, što rezultira ozbiljnim ograničenjima veličine modeliranog CPO problema. Ograničenje točnih algoritama u smislu resursa i vremenski ograničava njihovu primjenu na modele malog prostora, obično u dvije dimenzije s niskom frekvencijom učestalost uzorkovanja modeliranog prostora za prikupljanje mogućih položaja kamere i kontrole bodova. Modeliranje problema iz stvarnog života u 2D nije točno zbog ograničenja ili aproksimacija u smislu stvarnog ili efektivnog vidnog polja kamere. Efektivno vidno polje kamere reducira se na jednostavne geometrijske oblike, poput trokuta ili sektora, što rezultira lošim približnim slika modeliranog prostora. Dodatno, niska frekvencija uzorkovanja rezultira velikim prostornim praznine između mogućih lokacija kamera ili kontrolnih točaka, čime se nedovoljno predstavljaju naseljeni prostor. Zbog ovih ograničenja točnih optimizacijskih algoritama, razvijeni su mnogi približni ili heuristički algoritmi za rješavanje CPO problema. To uključuje Greedy algoritme koji rade iterativnim odabirom kamera s najvećom pokrivenošću sve dok nije postignut limit broja kamera, genetski algoritmi koji oponašaju biološki proces prirodne evolucije odabirom brojnih mogućih rješenja i kombiniranjem kamera između odabranih rješenja za proizvodnju novih rješenja, algoritama roja čestica koji usklađuju istraživanje i iskorištavanje prostora pretraživanja inicijaliziranjem određenog broja čestica koje se kreću prostorom pretraživanja i pokušati kolektivno postići optimalno, vjerojatno uzorkovanje algoritama koji odlučuju koje odabrano rješenje kandidata treba obraditi procjenom, a vjerojatnost da je uzorak dobro rješenje, algoritmi lokalnog pretraživanja koji naglašavaju stvaranje poremećaja u blizini naišlih dobrih rješenja kandidata, itd.

U ovom radu prvo se bavimo nedovoljno proučavanim problemom optimizacije položaja kamere za okružni pogled vozila. Kako je integracija ADAS sustava relativno nova i zahvaljujući činjenici da su sustavi kruznih kamera ograničeni na male automobile, problem CPO-a za pogled iz okoline vozila nije dovoljno proučen. Međutim, zbog sve većeg broja kamera i kombinacija različitih tipova senzora koji ulaze u snimanje prostornog pogleda sustavima, problem dobiva sve veću pozornost. Formulacija temeljena na BIP-u prvi put je predložena u 3D pomoću

svemirskog modela koji se sastoji od 3D modela vozila. Za diskretni BIP problem, poligonalni model vozila se vokselizira i prikuplja se skup mogućih lokacija kamere s vanjske površine modela vozila. Cilindrični prostor definiran je sa 360 stupnjeva oko vozila iz kojeg se uzorkuju kontrolne točke. Ograničenja u obliku ograničenja na broj kamera, vidljivost kamera, itd., dodaju se problemu maksimizacije pokrivenosti uzorkovanih kontrolnih točaka. FoV fotoaparata modeliran je pomoću 3D piramida opisano korištenjem pet ravnina, a pokrivenost se izračunava pomoću proračuna točke u ravnini. Za kontrolnu točku se kaže da je pokrivena određenom kamerom ako se kontrolna točka nalazi unutar svih pet ravnine piramidalnog vidnog polja kamere. Za kamere su definirane pozicije i kontrolne točke optimizacija pomoću binarnih varijabli odlučivanja. Brojne orijentacije oko terena i smjerovi skretanja svake kamere definirani su u koracima unaprijed definiranog kuta. CPO problem rješava se korištenjem algoritma grananja i vezanja (BnB) i Greedy algoritma. Dobiveni rezultati naglašavaju ograničenja točnih algoritama optimizacije i daju motivaciju za daljnje istrazivanje heurističkih algoritama. Ograničenja korištenja točnih algoritama leže u činjenici da se definirani CPO problem nije mogao riješiti korištenjem BnB algoritma za većinu dospjelih slučajeva zbog ograničenja RAM-a. Samo djelić minute definiranog skupa pozicije kamere i kontrolnih točaka morali su biti nasumično uzorkovani da bi se mogao riješiti CPO problem korištenjem BnB algoritma. Nasuprot tome, isti je problem riješen u nekoliko sekundi korištenjem Greedy algoritma bez nailaska na pogreške nedostatka memorije. Visok stupanj točnosti orijentacije kamera je potreban u ovim sustavima kamera s više kamera za njihovu upotrebu u naknadnoj obradi slike (spajanje slika/videozapisa npr.). Ovo zahtijeva visokofrekventno uzorkovanje modeliranog prostora, što rezultira velikim brojem varijabli odlučivanja, čime je problem nepraktičan za rješavanje korištenjem točnih optimizacijskih algoritama. Novi heuristički algoritam temeljen na optimizaciji za pozicije kamere u više rezolucija predloženo je da se omogući prikupljanje uzoraka iz modeliranih na visokoj frekvenciji i rješavanje problema korištenja ograničenih resursa u ograničenom vremenu. Metoda funkcionira tako da se opetovano klasterizira položaj kamere na temelju njihove prostorne lokacije i orijentacije te optimizaciju za poziciju kamere na skupinama točaka. U procesu grupiranja, svaka kandidatska pozicija kamere dodijeljena je normala površine točke (procijenjena s obzirom na vokselizirani 3D model vozila) kao njegova primarna orijentacija, a metrika udaljenosti koristi se za provjeru sličnosti točaka unutar susjedstva i dodijeljivanja unaprijed definiranim centrima klastera.

Nakon što je korak dodjele za iteraciju je završen, težište svih točaka koje pripadaju klasteru se koristi za predstavljanje prostornog položaja klastera dok je prosjek svih njihovih površina normala se koristi za predstavljanje orijentacije klastera za optimizaciju. Broj klastera tijekom svih iteracija održava se konstantnim, a metoda je nazvana multi-rezolucija jer, kako iteracije prolaze, klasteri postaju sve manji dok ne dosegnu točku gdje nema daljnje potrebe za grupiranjem, čime se površina vozila prikazuje u najvećoj razlučivosti. Metoda je testirana pomoću

pet različitih optimizacijskih algoritama, a rezultati ispitivanja se uspoređuju s rezultatima dobivenim rješavanjem istog problema, u jednom koraku, bez ikakvog grupiranja (naziva se jednostruka razlučivost). Rezultati predložene metode više razlučivosti pokazuju da je smanjuje vrijeme optimizacije do 160 puta za veće instance. Rezultati također pokazuju da ova metoda omogućuje korištenje tocnih optimizacijskih algoritama na velikim problemima što je inače nije moguće za određeni broj uzoraka prikupljenih iz prostora. Metoda je bila ispitana pomoću tocnih i heurističkih algoritama optimizacije, a predložena metoda smanjuje vrijeme računanja i za heurističke algoritme. Rezultati također pokazuju da predložena metoda neznatno poboljšava pokrivenost za neke slučajeve zbog sub-voxel točnosti postignute kroz klasteriranje.

Predložena metoda klasteriranja prilagođena je CPO primjeni s naglaskom na blizinu između orijentacije ili, normale površine, preko njihove prostorne udaljenosti. U usporedbi s drugim metodama segmentacije supervoksela, naša predložena metoda klasteriranja ima drugaciju shemu inicijalizacije sjemena klastera koja mu pomaže da ima više klastera u regijama s visokom gustoćom točaka, što rezultira boljim pripanjanjem granica. Kada je sličnost teksture ili boje točaka korišteno zajedno s orijentacijom i prostornom blizinom za grupiranje točaka, predložena metoda pokazala je bolju izvedbu u usporedbi s najsuvremenijim segmentima supervoksela segmentacijske metode. Metoda je testirana na RGBD skupu podataka otvorenog koda od 1400 oblaka točaka u zatvorenom prostoru i u usporedbi s tri najsuvremenije metode segmentacije supervoksela. Rezultati pokazuju da predložena metoda nadmašuje najsuvremeniju u tri od četiri metrike. Proizvodi kompaktnije supervoksele u usporedbi s drugim metodama zahvaljujući našim predlozenim shemama inicijalizacije, te je veća težina dana orijentaciji i sličnosti teksture. Na kraju, novi CPO problem temeljen na mješovitom cjelobrojnom programiranju također je predložen kao dio ovoga rada. Unatoč korištenju heurističkih metoda optimizacije, dobiveno rješenje možda neće biti od željene kvalitete, prvenstveno zahvaljujući uzorkovanju poziciji kamere. Poduzorkovanje možda neće odgovarati dobrom približavanju površini vozila, dok se prikupljanjem velikog broja uzoraka povećava složenost problema. Nekoliko relevantnih radova modelira varijable odluke u kontinuitetu prostora zbog jednostavnosti formulacije problema temeljene na BIP-u i složenosti geometrijskih izračuna uključena u izračun pokrivenosti kamerom.

Višestruki sustavi kamera za kruzni prikaz vozila koriste se za proizvodnju spojenih video izlaza od 360 stupnjeva gdje pozicija kamere mora biti poznata s visokim stupnjem točnosti. Za predloženi CPO problem u domeni kontinuiranog prostora, orijentacija i položaj kamere se definiraju pomoću varijabli koje mogu poprimiti kontinuirane vrijednosti unutar određenog raspona dok su kontrolne točke definirane u diskretnom prostoru, kao uzorci prikupljeni iz ravnine tla oko modela vozila. Formulacija problema s mješavinom diskretnih i kontinuiranih varijabli pomaže u izbjegavanju složenih geometrijskih izračuna za izračunavanje pokrivenosti kamere i smanjenju vrijeme računanja zbog značajno manjeg broja varijabli odlučivanja. U

ovom prikazu optimizacije mješovitog cijelog broja, definiranjem hemisfere u sfernim koordinatama oko 3D model vozila, kamera je predstavljena pomoću samo tri varijable. Međutim, zbog korištenje diskretnih i kontinuiranih varijabli, CPO problem više nije linearan. Mapiranje položaja kamere s hemisfere na površinu vozila čini CPO problem nelinearni. Stoga tocni algoritmi kao što je algoritam grananja i vezanja mogu se duže koristiti obzirom su definirane samo za linearne ili kvadratne ciljne funkcije s linearnim ili ograničenjima. Naš problem je riješen pomoću tehnika optimizacije crne kutije koje ne zahtijevaju sve informacije o funkciji cilja (npr. gradijenti) za napredak prema cilju optimuma funkcije.

Predloženi kontinuirani CPO problem uspoređen je s diskretnim CPO problemom koji koristi algoritam optimizacije roja čestica na više od stotinu realističnih 3D modela vozila različitih tipova, od automobila do teških strojeva ili građevinske opreme. Kontinuirani CPO problem testiran je korištenjem optimizacije roja čestica algoritam koji koristi neizrazitu logiku za procjenu vrijednosti parametara algoritma i Bayesian algoritam optimizacije. Naša metoda koristi matricu kamere za izračunavanje pokrivenosti koja nudi veću fleksibilnost u modeliranju različitih tipova kamera i, također pomaže u izbjegavanju skupih geometrijskih proračuna (kao što je točka u ravnini) za izračunavanje pokrivenosti. Različite vrste leća mogu se lako uklopiti u problem određivanjem parametara kao što su žarišna duljina, izobličenje itd., u matrici kamere. Predložena shema procjene položaja kamere na površini modela vozila rezultira sa samo pet varijabli za svaku kameru. Nasuprot tome, pozicija kamere definirane u diskretnoj domeni rezultiralo bi tisućama varijabli odluka za optimizaciju problema. Rezultati pokazuju da kontinuirani CPO problem postiže znatno bolju pokrivenost za isti broj kamera u usporedbi s diskretnim CPO problemima.

Ključne riječi: optimizacija položaja kamere, globalna optimizacija, segmentacija slike, računalna geometrija, 3D vizualizacija

Contents

Chapter 1

Introduction

Modern vehicles are increasingly relying on several types of ADAS systems to reduce on-road accidents, assist during parking, and to address driver fatigue among other uses. Parking assistance systems such as, rear-view cameras are present in most modern vehicles. With increasing demand for parking assistance, multi-camera systems such as surround-view camera systems are garnering interest in the automotive industry, [1, 2]. Capturing surrounding-view with multiple cameras enables generating novel views such as, top-view, rear-view, etc. A common approach to deploy vehicle surround-view camera systems is by placing 4 wide-angle/fish-eye lens cameras on the 4 sides of the vehicle (see Figure 1.1), [3]. While this approach may work well for small vehicles (e.g., cars) as they have a wide field-of-view (FoV) to also include sufficient overlap between different FoVs, it may not be suitable for larger vehicles such as, trucks, construction equipment, etc. Some other systems, in addition to four cameras, add range sensors such as, lidar and ultrasound for additional accuracy and reliability, [4]. Apart from providing assistance, surround-view capture can help in avoiding fatal accidents by ensuring visibility of blind-spots, i.e., the region around the vehicle that are not directly visible to the vehicle operator.

Capturing and displaying of the surroundings of a vehicle can be divided into four stages: 1) sensor placement planning, 2) multi-camera calibration, 3) image/video registration/stitching, and 4) novel view generation and/or 360^o display. Although the three stages of camera calibration, [5, 6, 7], image registration, [8, 9], and displaying, [10, 11] are well-studied, not enough research has been conducted on the first stage of sensor placement planning. The primary reason for this is that these systems are limited to cars where four cameras placed on the four sides of the vehicle suffice. However, intuition may fail when multiple cameras need to be placed on larger vehicles with complex, non-convex structure. Further, planned placement of the cameras on the vehicle can result in full surrounding-view coverage with fewer cameras thereby, reducing the cost of the multi-camera system. A well-defined algorithm designed to optimize the placement of cameras can scale easily for vehicles of any size and may help to avoid manually finding good locations through a lengthy trial-and-error process. Additionally,

Figure 1.1: An example of a vehicle surround-view capture system with four fish-eye lens cameras on the four sides of the vehicle, [2]. The four images around the center show the camera view from the four fish-eye lens cameras placed on the vehicle.

it may also prove useful during the camera calibration stage by providing an accurate estimate of the cameras' poses in real-world.

Literature for the Camera Placement Optimization (CPO) problem dates back to over a decade. While the general sensor placement optimization (SPO) problems are used in various fields including but not limited to, identifying defects in structures, [12, 13], drone navigation, [14], 3D reconstruction, [15], Internet of Things, [16], and video surveillance, [17], CPO problems, which are a subset of the SPO problem, also find applications in a diverse set of fields such as, 3D object reconstruction, [18], human behaviour monitoring and motion capture systems, [19, 20], multi-camera network design with VR interface, [21], etc., but their applications are largely concentrated to video surveillance, [22, 23, 24]. Compared to CPO problems for surveillance, a CPO problem for vehicle surround-view needs to address additional challenges such as, more degrees of freedom (DoF) in camera pose, non-convex space models and placement of cameras in such environments, and a requirement of high accuracy due to the dependence of subsequent image processing steps (calibration, image registration, etc.) on the optimized camera poses. Despite these challenges, only a few recent works, [25, 26, 27], address the camera (or heterogeneous sensors) placement optimization problem for the specific use-case of vehicle surround-view capture.

1.1 Problem Statement

The problem of camera placement optimization relates it's origins to the art gallery problem, [28], where a combinatorial theorem is used to identify the positions of security guards from where maximum area of an art gallery can be monitored. Modelling a CPO problem requires the definition of a space where the cameras are allowed to be placed, a target space that needs to be viewed by the placed cameras, and a model of the camera's FoV along with a cost of the camera. A common and an established practice is to model the CPO problem as a set-cover problem in discrete space, where samples are collected from the space where cameras can be placed and the target space, which are represented as sets of possible camera locations and target area points for the set-cover problem. Consider that we are given a set, $\mathbb{J}$, of samples form the target area represented as points in 3D Cartesian coordinate system and a set of samples, $\mathbb{I}$, representing the points in 3D Cartesian Coordinates where cameras are allowed to be placed. Consider that a visibility matrix, A, with elements a_{ji}, which maps the two sets, $\mathbb{J}$ and $\mathbb{I}$, is also given. Given these considerations, a simple CPO problem can be formulated using binary decision variables, x_i, which decide whether a camera position is selected or not. Then, the uni-cost set cover problem modelled using binary integer programming (BIP) to obtain full coverage of the points in set $\mathbb{J}$ is then given as,

$$\begin{aligned} &min \sum_{i=1}^{|\mathbb{I}|} x_i \\ &s.t. \sum_{i=1}^{|\mathbb{I}|} a_{ji} x_i \geq 1 \quad \forall j, 1 \leq j \leq |\mathbb{J}| \ . \end{aligned} \tag{1.1}$$

If a unit cost is associated with each camera, the above problem tries to minimize the cost of the multi-camera network while ensuring coverage of all of the target space samples in $\mathbb{J}$. Although this binary integer programming-based problem formulation is commonly found across CPO problems and SPO problems, the modelling of the space and collection of discrete samples is dependent on the use-case and varies from problem to problem. While the discrete CPO model is widely used due to it's simplicity, few works also consider describing the decision variables continuous space without any sampling of space, albeit at the cost of significantly higher complexity. For the case of optimization in continuous space, the decision variables are no longer binary or integer, but are allowed to take real values within a specified range (depending on the requirements of the use-case) which describe the position (and/or orientation) of the camera in the modelled space.

1.2 Contributions

There are two main contributions made by this thesis. Modelling the CPO problem for vehicle surround-view forms the basis for the two contributions. For the first, contribution, a multi-resolution optimization method is proposed which is a heuristic optimization algorithm that addresses the concerns of complexity and high optimization times of the discrete binary integer programming-based CPO problem. Given an input 3D voxel model of a vehicle, the multi-resolution approach repeatedly groups the voxels into subsets based on the proximity of the voxel centers and the surface normals of the voxels and optimizes camera poses on the subsets or segments. The size of the segments becomes smaller with each iteration until camera poses are optimized directly on the voxels of the vehicle's 3D model. This approach helps in reducing the number of decision variables for each optimization run, thereby reducing the complexity of the problem.

The voxels are grouped together using a supervoxel segmentation method that is tailored for this specific use-case. This segmentation method was compared against state-of-the-art supervoxel segmentation methods and is discussed in this thesis as a separate contribution. For the second contribution, a CPO problem is formulated as mixed-integer programming model in continuous space. In such models, some decision variables are allowed to take continuous values while the rest are modelled using discrete samples. For our problem, the target space that needs to covered is modelled using discrete points while the cameras' positions and orientations are modelled as continuous decision variables. This problem is complex and non-linear, and is optimized using global black-box optimization techniques. The main reason to formulate the camera pose in continuous space is to preserve the accuracy of pose estimation, particularly of the cameras' orientation as even small variations of camera orientation may result in significant artefacts while registering the images from multiple cameras.

1.3 Thesis Structure

The necessary principles for understanding camera placement optimization problems are briefly described in Chapter 3 within the context of state-of-the-art methods. This chapter introduces the concepts of space and camera modelling, CPO problem formulation, and deterministic and heuristic optimization for discrete as-well-as continuous decision variables. Necessary concepts related to 3D geometry, image segmentation and visualization are introduced in Chapter 2. The scientific contributions made by this thesis are detailed in Chapter 4 and the conclusions are presented in Chapter 5 along with future research directions.

Chapter 2

Overview of Computer Vision Principles and Visualization

Apart from mathematical principles of combinatorial and global optimization, the work presented in this thesis relies on the principles of 3D geometry, computational photography, 3D visualization, 3D image segmentation and machine learning. Defining cameras, their Field-of-View (FoV), and calculating coverage of control points by a camera, requires the use of 3D geometry and principles on camera-view projection. To verify the optimization results qualitatively, we developed a visualization tool to display the cameras at the optimized locations on the vehicle's 3D model along with the camera FoVs. To overcome the limitations of exact optimization algorithms on large datasets, a new heuristic algorithm was proposed. The algorithm uses image segmentation and machine learning principles to group together the vehicle's model into larger segments. Some optimization methods were implemented on the GPU to speedup computation using parallel programming techniques. All these principles are briefly introduced in the following sections.

2.1 3D Computational Geometry

Computational geometry is a broad field but, the work in this thesis is mostly focused on camera's field-of-view and camera projection matrices. Calculating camera coverage using the camera projection matrix or simply the *camera matrix*, [29], is discussed in this section whereas, the geometrical camera FoV models are introduced in Section 3.2. The camera matrix is a 3×4 matrix that allows mapping of 3D points in Cartesian space into 2D image coordinates ($\mathbb{P}^3 \rightarrow \mathbb{P}^2$). A 3D point, p_w, and it's pixel coordinates, p_c, in an image are related using the camera matrix M as,

$$p_c = sMp_w \ , \tag{2.1}$$

where, s is a scalar value. The camera matrix M has 11 degrees of freedom as multiplication of the matrix by any arbitrary scalar value s results in an equivalent camera matrix. The matrix encompasses the translation and rotation transformations of the point in world to camera coordinates, projection of the transformed point onto the image plane, and distortions (scale, skew, etc.) resulting from the camera hardware, lenses, etc.

The matrix M is a combination of two individual matrices, the intrinsic parameter matrix and the extrinsic parameter matrix. The intrinsic parameter matrix describes the properties of the camera and it's model. The intrinsic parameter matrix is an upper triangular matrix given as,

$$K = \begin{bmatrix} \alpha & -\alpha cot\theta & c_x \\ 0 & \frac{\beta}{sin\theta} & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.2}$$

where, c_x and c_y, describe the translation between the center of the image plane and the origin of the image in pixel coordinates, α and β describe the projection using the focal length of the lens f (for a simple pinhole model, $\alpha = \beta = f$), and $cot\theta$ and $sin\theta$ describe the skew of the sensor (a skewed camera coordinate system results when the angle between the two axes is not equal to 90^o) and the distortion from the lens, respectively. Skew parameter can be ignored in most use cases as most sensors are not skewed. Distortion is present in wide-angled lenses and can be ignored when modelling cameras with a standard lens.

The intrinsic parameter matrix maps the points from a 3D camera reference system to the image whereas, another transformation is needed to map 3D points from an arbitrary world coordinate system to the camera coordinate system. This transformation can be achieved by using the rotation and translation matrices, R, a 3×3 matrix, and T, a 3×1 vector, respectively. Collectively, the extrinsic parameter matrix is denoted is $[RT]$. Using the intrinsic and extrinsic parameter matrices, a 3D point, $p_w = (x, y, z)$ can be represented in image coordinates $p_c = (u, v)$ as,

$$\begin{bmatrix} u \\ v \end{bmatrix} = K[R\ T] \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{2.3}$$

The camera matrix can be used to calculate coverage of control points in CPO models. A control point is covered by a camera if it's image coordinates fall within the modelled image dimensions, and it is not covered otherwise.

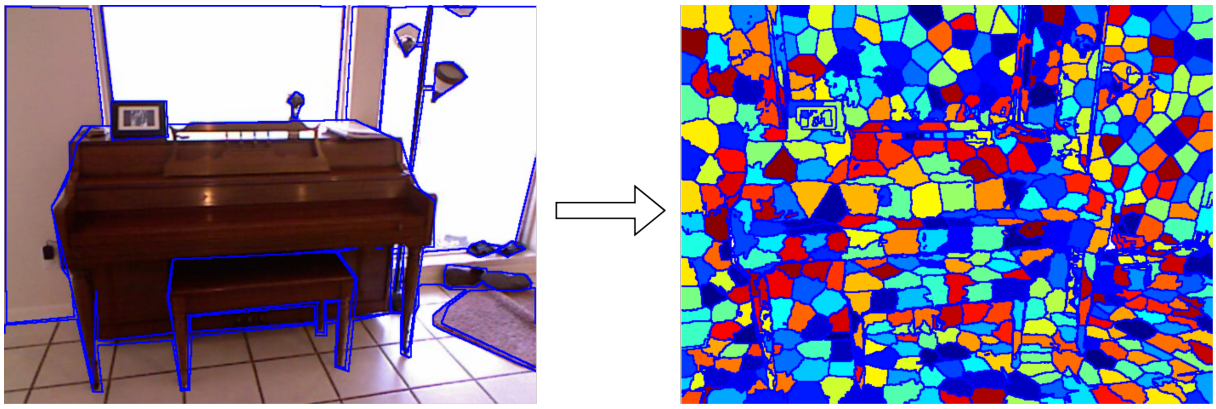


Figure 2.1: An example of supervoxel segmentation of an RGBD image. The image on the left shows ground truth edges overlayed on the RGB image.

2.2 Supervoxel Segmentation

Supervoxels are disjoint clusters of points in a point cloud which represent regions sharing common features such as, spatial location, surface normal orientation, and color. Image processing for large point clouds is computationally expensive as some operations have to be repeatedly applied on thousands or millions of points. Grouping regions into supervoxels allows to represent the set of points with shared attributes. Image processing operations can be applied on the supervoxel and the changes can be propagated to the underlying points at a later stage, as desired. Although, supervoxels are an approximation of a set of points, *good* supervoxels may try to maximize similarity between the points and preserve minute details. The following properties are desirable in *good* supervoxels: 1) boundary adherence - object boundaries should be preserved, 2) compactness - supervoxels should be regularly shaped, 3) efficiency - generating supervoxels should be computationally inexpensive. An example of supervoxel segmentation of an RGBD image is shown in Figure 2.1.

Supervoxels are a three-dimensional extension of superpixels, [30, 31, 32], which are over-segmentation of 2D images. One of the first methods of supervoxel segmentation was applied on videos, in which time is the third dimension. Moore et al., [33], proposed an over-segmentation method for videos to iteratively divide pixels into clusters by cutting a 3D grid horizontally and vertically. Achanta et al., [34] proposed an efficient k-means-based algorithm for over-segmentation of 2D RGB images and video sequences. Their method works by intializing seeds uniformly across an image and assigning pixels within a local radial neighbourhood to the closest cluster seeds using a distance metric based on color and spatial similarity. The spatial distance in the metric was extended to 3D distance for the case of segmentation of videos. Another supervoxel segmentation method using graph cuts to solve an energy minimization problem was proposed in, [35].

A method for supervoxel segmentation of RGBD videos to partition a graph constructed us-

ing color and surface normals into a spatio-temporal segments using a spectral graph clustering method, was proposed in [36]. Gao et al., [37], improved the segmentation results over existing methods by using non-uniform cluster seed initialization to obtain relatively dense seeding at salient regions. In [38], Papon et al., proposed one of the first supervoxel segmentation methods for RGBD images which works simialr to the SLIC algorithm, [34]. Their method was efficient and fast as they worked with voxelized point clouds and used voxel adjaceny graphs for fast access of voxel neighbours for assignment to cluster seeds. Lin et al., [39], proposed a supervoxel segmentation method where an energy function is optimized by formulating a subset selection problem. The subset selection problem formulation allows for segmentation without an input parameter for the number of supervoxels.

Newer methods came up with different strategies to improve boundary adherence of supervoxel segmentation methods. A modification to the vccs algorithm, [38], was proposed in [40], to work directly on point clouds without voxelizing. An algorithm absed on local allocation, with a cost function focussed on preserving boundaries was proposed by Ni and Niu, [41]. Some methods went a step further to use deep neural networks which learn geometrical features of an image and output supervoxels, [42, 43]. For this thesis, a new supervoxel segmentation method was proposed that is particularly suited for point clouds. The proposed methods takes advantage of points' orientation, color and spatial distance with an innovative cluster seed initialization strategy and improves compactness of the supervoxels, when compared to other methods. The method was developed with a focus on CPO problems for vehicle surround-view to reduce the time and resource complexity of exact CPO optimization algorithms.

2.3 3D Visualization & Parallel Computing

3D visualization tools were developed to assist in qualitative analysis of the new CPO methods proposed as part of this work. The tools were developed using open source software libraries including, OpenGL, [44], a toolkit in C++ programming language for computer graphics applications, Visualization Toolkit (VTK), [45], a C++ programming library with a clean interface to visualize 3D data, and Paraview, [46], a 3D visualization tool with parallel visualization capabilities. Our visualization workflow consists mostly of the input data in the form of 3D polygon mesh models, [47], which are processed to extract data from the vehicles' models and add a ground plane to the model to extract the control points. The optimized camera poses obtained as the solution of the optimization problem is marked on the vehicle's surface in the 3D model and the cameras' FoVs are drawn using lines to show the covered control points by the cameras placed at the optimal locations.

Our work uses two different visualization models for the discrete and continuous CPO problems. For the discrete CPO problem, the polygon mesh models were first voxelized using VTK

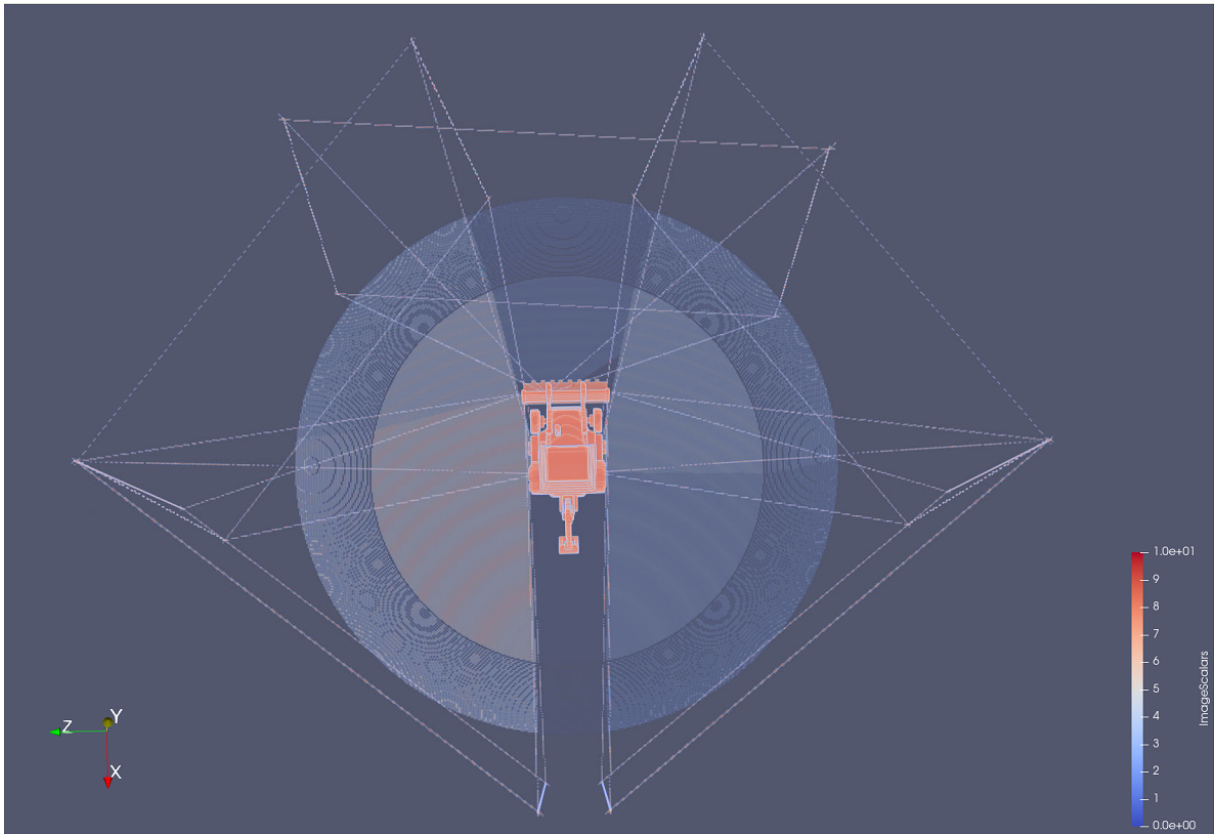


Figure 2.2: An example of the visualization of results for the discrete CPO problem.

to make collection of discrete samples from the vehicle's surface easier. A bowl-shaped shaped surface was defined around the vehicle to collect samples of control points from. An example visualization used for qualitative analysis of the results for the discrete CPO problem can be seen in Figure 2.2. In the figure, the region in red represents the vehicle's model while, the region in shades of blue shows the control points. The darkest part of the blue region indicates that the region is not covered by any cameras. It can also be seen that cameras' FoV are clearly visualized using lines. The visualization helps to visually verify the optimization results as well as to verify if the optimal camera poses are meaningful or if they lie at impossible locaitons on the vehicle's surface.

Similarly, another visualization model was developed for the continuous CPO problem. As sampling of vehicle's surface is not required for the continuous CPO problem, the polygon mesh models were used with voxelizing. The processing of input data for this visualization involved obtaining bounds of the vehicle's model, placing a hemispherical surface around the model and collecting samples from a defined ground plane around the vehicle's model. For the visualization model for the continuous CPO problem, the sampled control points were visualized as small spheres surrounding the vehicle's model. An example visualization of the results for the continuous CPO problem is shown in Figure 2.3. Also for this visualization, the optimized cameras' FoVs are shown using lines. Visualizations for both the problems were created using

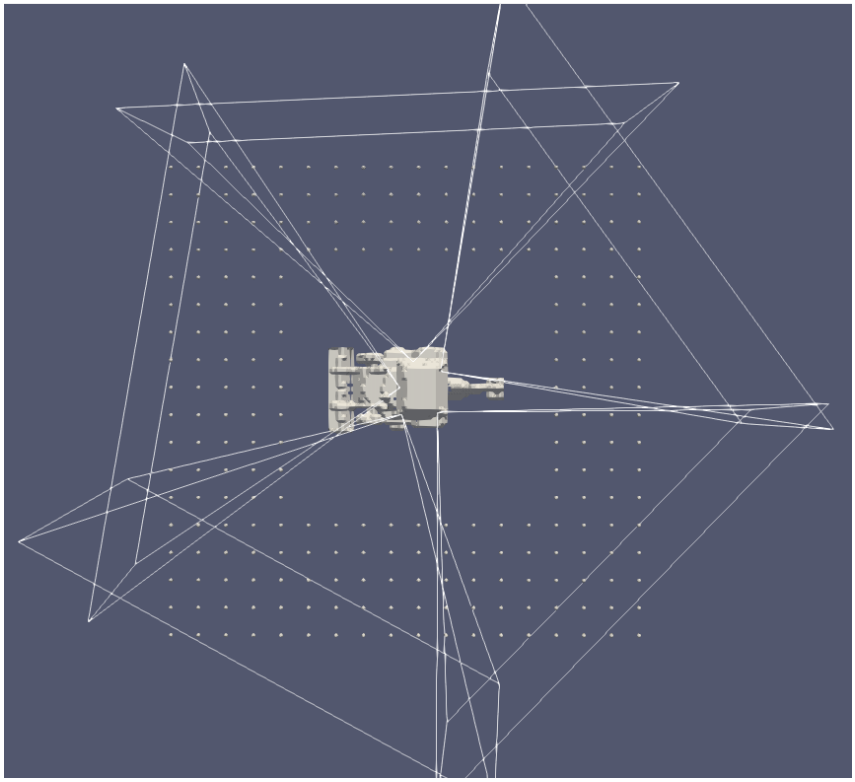


Figure 2.3: An example of the visualization of results for the continuous CPO problem.

VTK and visualized on Paraview.

Parallel computation was used to speed up some optimization algorithms used for this work. Parallel programming was done using the OpenCL, [48], library using C++ programming language, and numba, [49], library using python programming language. A greedy algorithm used for discrete CPO problem was programmed to run on the GPU using openCL library. The algorithm required recomputing the visibility of all available candidate cameras at each iteration, and calculating camera visibility parallelly provided a significant speedup. Similarly, a particle swarm optimization algorithm for the continuous CPO problem was also programmed on the GPU using numba library. As the particles in a particle swarm optimization algorithm, move independently of each other, the fitness for each particle at each iteration can be calculated parallelly, to provide a significant speedup.

Chapter 3

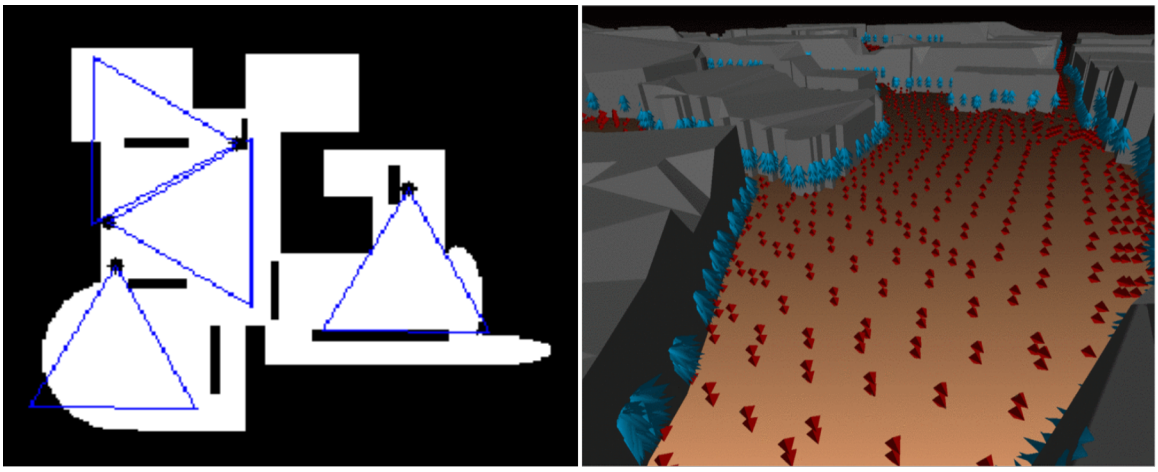
Overview of Camera Placement Optimization

Before formulating the CPO problem, the simulation space needs to be defined, from where the set of samples will be collected over which the decision variables will be defined. It is also necessary to define the camera FoV models and compute the visibility matrix before CPO problem formulation. The following sections describe these prerequisite steps. Before concluding this chapter, continuous space model and associated decision variables are introduced at the end.

3.1 Modelling Space

As stated earlier, it is a common practice in CPO literature to use discrete space models. A simple 2D space model may include a floor plan divided into regions where cameras can be placed and target areas that need to be covered by the placed cameras. This space is sampled using various sampling strategies to use them for generating the decision variables for the optimization problem. Some example space models are shown in Figure 3.1. Figure 3.1(a) shows a typical 2D space model used in CPO for surveillance applications. This space model was introduced by Hörster et al., in [50], which is one of the pioneering works in CPO literature. In the figure, the target area is shown in white and the region in black represents obstacles (e.g., walls). This space addresses a 2D CPO problem where a given number of cameras are to be placed on the black region, in an optimal way, such that they cover as much area of the white region as possible.

To model it as discrete CPO problem following a BIP-based model, it is first required to gather the sets $\mathbb{J}$ and $\mathbb{I}$. In this example, the target points that need to covered (commonly known as *control points* in CPO literature) are sampled from the white region, whereas, samples collected on the edges of the black region would represent the points where cameras can be placed. While this example is one of the simplest models, other works address additional

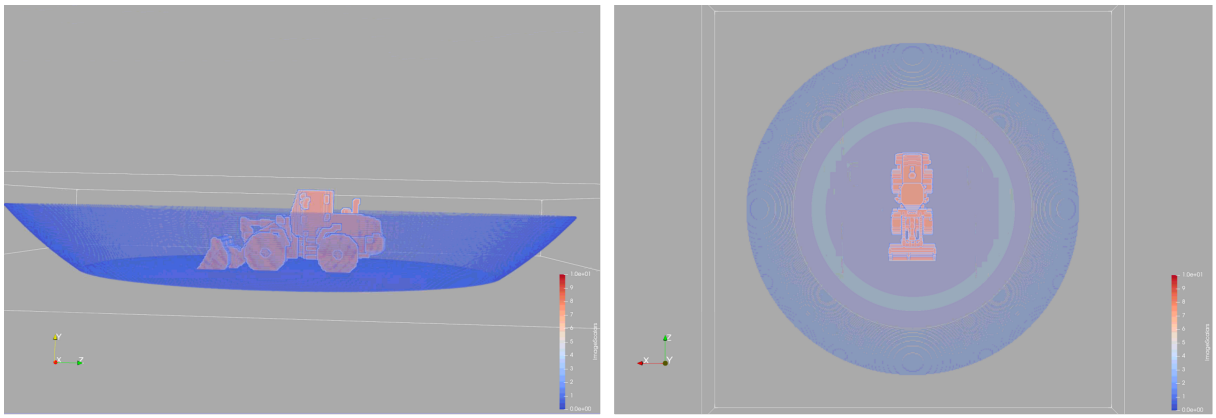


(a) 2D floor plan (b) 3D city plan

Figure 3.1: Examples showing common method of modelling space using floor plans for applications of CPO in surveillance scenarios. (a) A 2D indoor floor plan-based space model, [50], (b) A 3D space model of a city, [51].

complexities in the space. In [52], the authors considered a similar 2D floor plan with uniform weightage for all control points. David et al., [53] also considered a uniform 2D floor plan, but divided them region into zones based on various properties to simplify the problem and to influence camera placement. All these works also consider static obstacles which are traced into the camera's FoV using ray-tracing techniques. In [54], Hörster and Lienhart used weighted sampling to assign different levels of importance to different regions. In [55], Yabuta and Kisawa divided the region into smaller rectangles to reduce the complexity of the problem. Each rectangle is represented by it's center thereby eliminating the need for sampling. By changing the size of the rectangles they could control the resolution as smaller rectangles represent smaller part of the target region thereby implying higher accuracy of coverage. Zhao and Cheung, [56], extended the discretization of space strategy to apply the CPO problem for object tracking, e.g., tracking of tags, face, etc. They also employed different sampling strategies such as, random sampling, stratified sampling, and systematic sampling, to discretize the space.

Citing the limitations of modelling the problem space in 2D, Zhang et al., [57], proposed an into *2.5D* where they still preserved the floor plans, but allowed the cameras to be placed in two dimensions instead of one by adding a height parameter. Owing to more computational power, recent works started addressing the problem in three dimensions. Kritter et al., [51], modelled city plans in 3D to achieve the goal of optimizing camera poses for effective surveillance of the city. They used large space models and allowed the cameras to be placed on the buildings while the streets were modelled as target areas (see Figure 3.1). However, they highlighted that they had to use sparse sampling of the space due resource and time constraints for optimizing the problem on datasets of such scale. SOme other works with 3D space models include, [58], where Becker et al., proposed an algorithm based on voting scheme for SPO in a 3D volume,



(a) Side View (b) Top View

Figure 3.2: An example visualization of the space model used for the discrete CPO problem proposed in this work.

and [59], where Malhotra et al., to optimize 6 degrees of freedom (DoF) camera poses in 3D voxel volumes.

3.1.1 CPO for vehicle surround-view

Almost none of the above mentioned literature addressed the CPO problem for vehicle surround-view capture, as this application has not been studied enough. Some of the only works addressing CPO problems for autonomous vehicles include, [25], where Indu et al., proposed a CPO model in discrete space with weighted coverage function to emphasise critical areas in the vehicle's surrounding region, [26], where the authors optimized placement of distinct types of sensors for more ADAS purposes than only surround-view coverage, and [60], where Kim et al. proposed an approach similar to the previous one, but using only lidar sensors. The common feature in all the mentioned CPO problems lies in the space model where the possible camera locations are sampled on the vehicle which is placed at the center with the control points sampled from the region 360^o around the vehicle. For the work in this thesis, the space is modelled in 3D with 3D polygonal models of vehicles placed at the center and control points sampled on the ground around the vehicle until a certain distance from the center of the vehicle. An example visualization of the space model used for the work done in this thesis is shown in Figure 3.2. In the figure, the vehicle model placed at the center of the simulated space is shown in red and the control points sampled form the target area surrounding the vehicle's model are shown in blue. The target area is modelled to resemble a *bowl surface* to satisfy the requirements for image registration method as they project images from fish-eye lens cameras on bowl shaped surfaces for accurate image stitching.

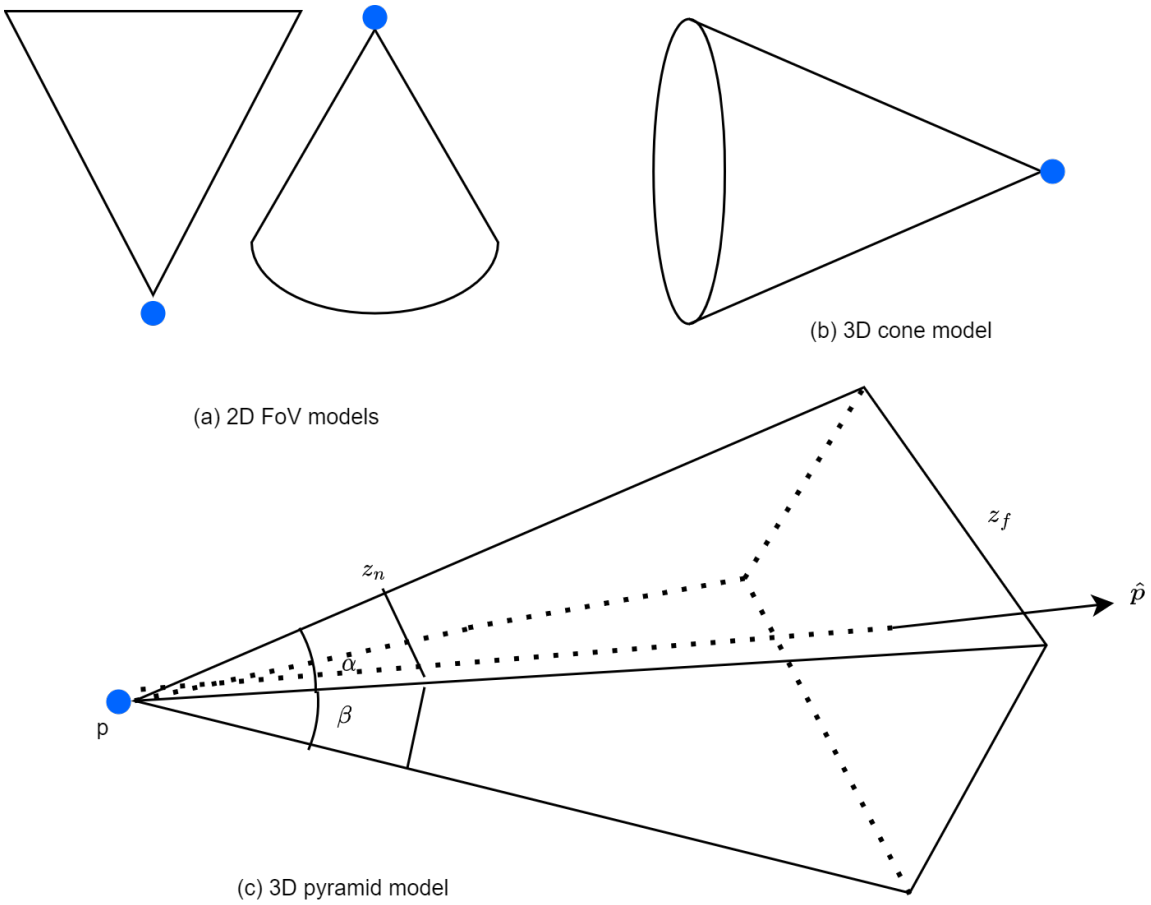


Figure 3.3: Some examples of camera FoV models. (a) triangular and sector FoV models in 2D, (b) Cone FoV model in 3D, and (c) 3D pyramidal FoV model which commonly used for 3D OCP problems.

3.2 Modelling Camera FoV

The next step before formulating an OCP problem is to define the cameras and FoVs. Similar to modelling space, these methods also vary from 2D to 3D. Camera FoV models are necessary to calculate coverage of the control points by the placed cameras. The calculated coverage value is plugged into the cost function to estimate the direction or the multi-camera placement combination for the next iteration of the optimization process. Figure 3.3 shows some example FoV models that are commonly used in OCP literature. Triangles and circular arcs were widely used for OCP problems modelled in 2D, [50, 61, 62, 63]. While these geometrical shapes work well for 2D camera models, the same shapes are extended into their 3D versions for 3D OCP problems. Cones and pyramidal FoVs are the two 3D extensions that are commonly used in 3D OCP problems, [51, 57, 64].

In our work, we use the pyramidal FoV. This FoV model is described using five to six planes, four planes representing the four sides of the pyramid wiht the remaining two planes representing the far plane (z_f) and near plane (z_n), Figure 3.3(c). The far plane defines the depth of field of the camera model while the near plane models the focal length. The length and

width of the far plane is related to the horizontal and vertical FoV angles, α and β, respectively, which together define the field of view of the camera model. In the figure, the camera origins are represented using a solid circle. In all cases, the view direction is defined by the line passing from the camera origin through the center of the opposite end. For e.g., $\hat{p}$ in Figure 3.3(c) is the view-direction vector of the camera which is the line from p passing through the center of the far plane z_f. Camera orientations at any given position can be modelled by simply rotating $\hat{p}$ at steps of a predefined angle.

Calculating control point coverage, for the pyramidal FoV model for e.g., is done by checking every control point against each of the planes of the FoV model. If a control point lies inside all the planes for a given camera model, that control point is said to be covered by that camera placed at position p with an orientation $\hat{p}$. The FoV model can be more sophisticated to model complex coverage criteria. For e.g., ray-tracing can be implemented in the FoV model to identify obstacles lying in the way, in which case, all the control points falling behind the obstacle can be marked as uncovered. Resolution can be implemented in the coverage model, by assigning higher weights to control points that lie within a certain range of distance, i.e., points farther away from resolution thresholds can be given lower weights to represent lower resolution. Additionally, the region within the FoV model can be assigned different real-valued weights which change by the gradient, for cases where a subject need to be placed at the center of the FoV. Some works also allow the cameras to pan and tilt, [65]. For a detailed survey on camera FoV and coverage models, please see [66].

3.3 Visibility Matrix

All discrete OCP problems require a visibility matrix which maps every possible camera pose with control points to describe coverage. An illustration of the visibility matrix is shown in Figure 3.4. This matrix is created by calculating the coverage for every possible camera pose. All the control points are checked against every camera pose and if a control point falls within the FoV of a given camera model, then the corresponding entry in the matrix is marked with one. Entries for control points that are not covered by a given camera, are marked with zeros. To use this visibility matrix in the OCP problem, a new binary variable is created for each entry of the visibility matrix. These variables can be incorporated into the OCP model visa constraints which help to ensure that only cameras that cover control points are selected during optimization.

Although constructing this matrix requires high amounts of computational time, it can be precomputed and saved to access during optimization. At this stage of constructing the visibility matrix, some of the unnecessary data points can be filtered to exclude them from the optimization process. The filtration step helps to reduce the size of the decision variables and

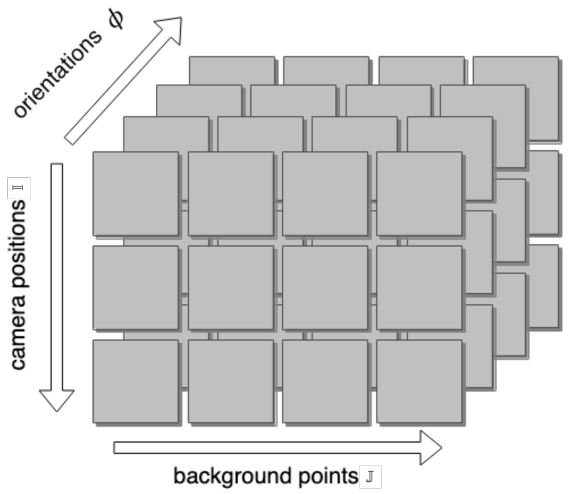


Figure 3.4: An Illustration of the visibility matrix.

constraints and can reduce the computational cost of optimization significantly. These steps can contain various general constraints and application specific constraints. Some common steps include, removing camera poses that do not cover any control points or less control points than a predefined threshold, removing camera poses whose orientations look away from the target area or directly into obstacles, and removing camera poses that have significant overlap with adjacent camera poses.

3.4 Variables and Problem Formulation

Camera placement is an offline optimization problem. It is often considered in two approaches: 1) maximize coverage while keeping the cost constant, and 2) minimize the cost while maintaining satisfactory coverage. For the simplest case, the cost function is usually the number of cases, but it can be the cost of the multi-camera network for advanced cases. Before formulating the OCP problem, all the data gathered in the previous steps needs to be modelled in terms of decision variables that can impact the cost function and the optimization process. For simplicity, we discuss only a simple uni-cost set-cover OCP problem. For this problem, it is assumed that the cameras are all of the same type and have the same cost. Considering that the set of possible camera locations have already been sampled, if the orientation of the camera at each sampled location is quantized into Φ rotations, the decision variables for a camera pose are defined as,

$$x_{i\phi} = \begin{cases} 1 & \text{if a camera is placed at location} \\ & i \text{ with orientation } \phi \\ 0 & \text{otherwise} \end{cases}, \tag{3.1}$$

where, $\phi = 1,\ldots,\Phi$ and $i = 1,\ldots,|\mathbb{I}|$. Similarly the control points sampled form the target area are assigned binary decision variables as,

$$c_j = \begin{cases} 1 & \text{if } \textit{control point } \text{j is covered} \\ & \text{by at least one camera} \\ 0 & \text{otherwise} \end{cases}, \tag{3.2}$$

where, $j = 1,\ldots,|\mathbb{J}|$. After visibility checks have been performed and the visibility matrix, A, has been constructed, assuming that no camera poses were filtered when constructing A, every entry of the visibility matrix is assigned a binary decision variable as,

$$a_{i\phi j} = \begin{cases} 1 & \text{if } \textit{control point } j \text{ is} \\ & \text{covered by a camera placed at} \\ & \text{position } i \text{ with orientation } \phi \\ 0 & \text{otherwise} \end{cases}. \tag{3.3}$$

The optimization process is influenced by the three sets of decision variables defined in (3.1), (3.2) and (3.3). Considering that we want to place n cameras on the vehicle, we define a simple objective or cost function as the maximization of coverage as,

$$max \sum_j c_j. \tag{3.4}$$

Some constraints need to be enforced to encourage coverage, place restrictions on the number of cameras etc. Firstly, the constraints to map the possible camera poses and control points to the visibility matrix are written as,

$$c_j \cdot \Big(\sum_{i\phi} x_{i\phi} \cdot a_{i\phi j} - 1 \Big) \geq 0 \,, \tag{3.5}$$

$$(1 - c_j) \cdot (1 - \sum_{i\phi} x_{i\phi} \cdot a_{i\phi j}) \,. \tag{3.6}$$

The above inequalities, which are non-linear as they involve product of two binary variables, can be linearized by replacing every occurrence of the product $c_j \cdot x_{j\phi}$ with a new binary variable $v_{i\phi j}$. Having introduced a new variable, the following constraints need to be added,

$$c_j + x_{i\phi} \geq 2 \cdot v_{i\phi j} \,, \tag{3.7}$$

$$c_j + x_{i\phi} - 1 \leq v_{i\phi j} \,. \tag{3.8}$$

The inequalities (3.5) and (3.6) can be rewritten in terms of $v_{i\phi j}$ as,

$$\sum_{i\phi} v_{i\phi j} \cdot a_{i\phi j} - c_i \geq 0 \ , \tag{3.9}$$

$$1 - \sum_{i\phi} x_{i\phi} \cdot a_{i\phi j} - c_i + \sum_{i\phi} v_{i\phi j} \cdot g_{i\phi j} \geq 0 \ . \tag{3.10}$$

To ensure that not more than n camera poses are optimized, the following constraint needs to be introduced,

$$\sum_{i\phi} x_{i\phi} = n \ . \tag{3.11}$$

Lastly, the following constraint needs to be introduced to ensure that only one camera is placed at a given sampled location,

$$\sum_{\phi} x_{i\phi} \leq 1 \ . \tag{3.12}$$

Having defined the decision variables, the objective function, and the constraints, a simple BIP-based CPO problem to maximize coverage given that n cameras are placed, is formulated as,

$$\begin{aligned} &\text{maximize } \sum_{j} c_j \\ &\text{subject to: (3.7), (3.8), (3.9), (3.10), (3.11), (3.12), and,} \\ &\qquad 1 \leq j \leq |\mathbb{J}|, \ 1 \leq i \leq |\mathbb{I}|, \ 1 \leq \phi \leq \Phi \ . \end{aligned} \tag{3.13}$$

3.5 Optimization

The optimal camera placement problem is commonly modelled in Discrete space and is tackled using combinatorial optimization algorithms. In this context, combinatorial optimizations work by selecting the best available combination of camera poses that represent the optimal or near-optimal value of the cost function while satisfying the defined set of constraints, from a preselected set of all possible camera poses. Discrete CPO problems are proven to be NP-hard which means that the global optimum is hard to find in polynomial time. Optimization algorithms used for CPO can be broadly classified into two categories: 1) deterministic algorithms, which always provide the same solution for a given problem with proven bounds around the optimal solution, and 2) heuristic algorithms, which provide an approximation of the optimal solution and may or may not provide proven bounds around the optimal solution. Linear programming-based methods, such as branch and bound algorithm, are capable of finding the optimal solution, [67], but, they are limited only to small-scale problems as the resource requirement for these category of optimization algorithms increases exponentially with increase in problem size. In discrete space problem formulations, there is always a trade-off between having a good approximation of the underlying space and limiting the size of the optimization

problem. If accuracy is desired, sampling becomes refined to produce a larger sized problem leading to impractical resource and time requirements for using linear programming-based optimization algorithms.

For this reason, heuristic algorithms find abundant mention in CPO literature. Heuristic do not suffer from the problems of high resource and time requirements as their linear programming-based counterparts but, only provide an approximate solution which may or may not be close to the global optimal solution. Nevertheless, many heuristic algorithms, specific to the CPO problem, have been proposed over the past decade that provide good quality or near-optimal solutions in only a fraction of the time compared to linear programming-based methods. In the next subsections we briefly introduce the linear programming (LP) method and the greedy and particle swarm heuristic optimization algorithms. Only the optimization methods studied as part of this work are included in this thesis but, a wider range of deterministic and heuristic algorithms can be found in literature, some of which are discussed in Section 3.8.1.

3.5.1 Linear Programming

Linear Programming is a mathematical formulation for a system of equations, i.e., the objective function and the constraints. It is one of the simplest ways to solve an optimization problem. The constraints expressed as inequalities define a feasible polyhedral set region in the search space. It is expected that the one of the vertices of this region represents the optimum value, thereby reducing the problem to one of finding the extreme points. Commonly used algorithms to solve LP problems include the simplex method, [68], and the branch and bound algorithm, [69]. Integer Linear Programming (ILP) is a type of LP which restricts some or all variables to integer values. If not all of the variables are integer, the problem is known as Mixed-Integer Linear Programming (MILP). MILP allows some variables to take integer values while the others take continuous values. As shown in (1.1), an LP in canonical form is written as,

$$\begin{aligned} \text{maximize} \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b, \\ & x \geq 0, \\ & x \in \mathbb{Z}^n \ , \end{aligned} \tag{3.14}$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and $A \in \mathbb{R}^{m \times n}$.

In the context of CPO, the problem is formulated such that each entry in a set of possible camera placements is assigned a decision variable that takes a value 0 or 1 indicating the selection of the corresponding camera pose. As the decision variables take a value of either 1 or 0, the resulting problem becomes a special case of ILP, known as Binary Interger Programming (BIP). The binary constraint, however, makes the problem NP-hard making it unsolvable

in polynomial time. Nevertheless, the interest in LP relaxation has made the branch and bound (BB) algorithm the first choice to solve this problem. The BB algorithm breaks the problem into smaller sub-problems and uses bounding functions eliminate sub-problems that cannot possibly contain the possible optimum. The BIP-based CPO problem formulation has been used in many works across various applications of CPO, [50, 62, 70, 71, 72].

3.6 Greedy Heuristics

One of the simplest heuristics method to obtain a feasible solution. It is also one of the fastest of all optimization methods for CPO problems. For the problem of maximizing coverage, greedy algorithm works by arranging all the possible camera poses in the descending order by the value of it's coverage and selecting the camera at the top (with highest coverage of all) to include it into the final solution. This process is repeated for iterations equal to the number of cameras that need to be placed. Suppose, if the placement of $n = 5$ cameras is to be optimized, the greedy algorithm will select the top 5 cameras with highest coverage. This implies that the algorithm selects a locally optimal solution at each iteration. Due to it's simplicity, this algorithm is significantly faster than any other CPO optimization method.

However, the algorithm selects cameras deterministically meaning that the algorithm selects the exact same cameras no matter how many times it is run. There is no provision in the algorithm to reconsider the selection or to explore other combinations of cameras. Therefore, the resulting solution, even though a feasible solution, is always a locally optimal solution. Selecting one camera at a time also destroys the combinatorial aspect of the problem. In [54], Hörster and Lienhart proposed a Greedy heuristic which preserves the combinatorial aspects to some extent. Their algorithm works by removing all the control points that are covered by an already placed camera. For the next iteration, the coverage for all possible camera poses is recalculated on the remaining control points from the set. This step ensures that the algorithm tries to cover a wider set of control points rather than clustering all the cameras at close-by locations.

Due to their low computational complexity, greedy algorithms have found an interesting use to estimate an initial feasible solution that can be passed as a starting point for more complex and efficient optimization algorithms, [51]. it has also been shown that the greedy algorithm is the best polynomial-time approximation for the set cover problem, [73]. The authors in [74, 75], proposed greedy algorithms for the partial set cover problem, in other words for the cost minimization problem which sets a threshold on the minimum required coverage. In [56, 72], the authors have proposed greedy algorithms suited for the coverage maximization problem for placing a predefined number of cameras. A detailed review of greedy algorithms and also other heuristics can be found in [76]. Zhao et al., in that article, have a also a proposed an adaptive

greedy algorithm that can optimize multiple number of cameras together instead of one by one, to better preserve the combinatorial aspect of the problem.

3.7 Particle Swarm Optimization

Particle swarm optimization (PSO) methods are a group of bioinspired algorithms that mimic the movement of swarms of flies, a flock of birds or a school of fish, [77, 78]. The idea behind these algorithms is that a group of particles are initialized in the search space with each particle representing a feasible solution. The number of dimensions of each particle is equal to the number of cameras that need to be placed. The particles iteratively move through the search space with individual velocities. At each iteration, the fitness values local to each particle is compared to the global best fitness value encountered so far by the swarm and the global best is updated accordingly. The particles move through the search space with varying velocity that is influenced by the particle's current fitness value, best recorded fitness value and the global best fitness value.

The primary idea behind PSO algorithm is to determine the velocity of each particle. The velocity of the i^{th} particle at a time-step or iteration $t+1$ is given as,

$$v_i(t+1) = wv_i(t) + c_1 r_1(p_i - x_i(t)) + c_2 r_2(p_g - x_i(t)) \ , \tag{3.15}$$

where, x_i is the particle's position and v_i is the particle's velocity, w is the inertia, p_i is this particle's current best fitness values, p_g, is the current best fitness value reached by the whole population, c_1 and c_2 are constants that emphasise the effect of local and global optima on the particle's velocity, and r_1 and r_2 are random factors in $(0,1)$ that perform similar function as c_1 and c_2 with randomness. The constants and random factors allow for dynamic control of the balance between exploration and exploitation of the search space. The position of i^{th} particle for the next iteration is calculated using the velocity from (3.15) and it's current position as,

$$x_i(t+1) = x_i(t) + v_i(t+1) \ . \tag{3.16}$$

Every particle in the swarm has a memory to store the the best fitness value and the associated position (or camera combination) encountered by the particle so far. The algorithm also has a memory to store the global best fitness value encountered by the whole swarm. At each iteration, if any of the particles encounter a better fitness than the current best global fitness value, then the global is updated accordingly. The algorithm moves the particles according to their local and global optima, while the random and constant factors ensure that the particles also explore the search space without getting stuck at local optima.

The PSO algorithm is a metaheuristic and is well-suited for problems with large search

spaces and non-linear objective functions. As a metaheuristic method, they cannot guarantee optimality of the solution but, can certainly find feasible solutions very close to the optimal solution by balancing exploitation and exploration of the search space. As the velocities of the particles play a key role in maintaining this balance, several variants of the PSO algorithm have been proposed, [61, 79, 80, 81, 82]. Some of the recently published variations of the PSO algorithm, although not specifically applied to the CPO problem, have shown to be efficient on large-scale real-world problems, [16, 83].

3.8 Sampling-based methods

Sampling-based methods work similar to greedy methods with an added randomness in search space exploration, thereby avoiding local optima and making these methods non-deterministic. The simplest of sampling-based methods is to randomly sample the search space for predetermined number of iterations or until a satisfactory solution is reached. Most problems are however, complex with large search spaces thereby rendering random sampling insufficient to find an acceptable solution. To overcome this limitation several other sampling strategies, such as, metropolis sampling (MS), [84], Gibbs sampling (GS), [85], and simulated annealing (SA), [86], were used for sampling based optimization problems. These methods work by sampling from a probability distribution by constructing a Markov chain with a distribution matching the equilibrium distribution. This is achieved by relating the fitness or the objective value of the sampled point to it's probability of sampling. This follows the idea that assigning a higher probability to sampled points with higher fitness values may lead to the sampling of points closer to the optima.

MS and GS are two commonly used Markov Chain Monte Carlo (MCMC) algorithms. These methods can be used in the context of optimization by modelling the objective function as a distribution from which samples are drawn repeatedly and storing the sample with the best objective value. MS method works by making a small perturbation around the current sample, calculating the gain of the perturbation and accepting the sample if the gain is higher than a sampled random number. Assume that x_i is a combination of cameras selected from the set S of all possible cameras, i.e., $x_i = [x_0, x_1, \ldots, x_n] \in S$, then according to MS, the probability of sampling x_i can be given using the probability function as,

$$P(x_i) = \frac{exp\ logf(x_i)}{\sum_{j \in S} f(x_j)}\ , \tag{3.17}$$

where, $f(.)$ denotes the objective function. The gain of the perturbation according to MS is given as, $logf(x_i^{'}) - logf(x_i)$, [76]. The GS method also works similar to MS but, instead of making a small perturbation, the Gibbs sampler selects a sample based on the conditional

probability,

$$P(x_i = 1|x_0,\ldots,x_{i-1},x_{i+1}\ldots,x_m) = \frac{exp\ logf([x_0,\ldots,x_{i-1},x_{i+1}\ldots,x_m])}{\sum_{j\in S} f([x_0,\ldots,x_{j-1},x_{j+1}\ldots,x_m])}\ . \tag{3.18}$$

As the GS sampler admits samples based on the conditional probability, rather than based on a random number, the GS sampler admits more samples than the MS sampler.

Simulated Annealing (SA), [86], is a technique that establishes a non-linear relationship in how a point is sampled with respect to the objective value. SA algorithms work by sampling only around the peaks of the objective function. A new variable, *temperature (T)*, is introduced in the probability function that controls the probability of a sample being accepted. Teh probability function can be written as follows,

$$P(x_i) = \frac{explogf(x_i)\cdot T}{Z}\ , \tag{3.19}$$

where, z is a normalization factor, [76]. The parameter T is usually set to high value at the beginning and is gradually decreased. When T is high, a greater number of samples are accepted resulting in exploration of the search space. As the temperature cools down, the acceptance rate of a sample decreases thereby shifting the focus of the algorithm on to searching for the optimum. When T is small, the density of samples is concentrated around the peaks. This algorithm functions similar to MS when T is high. The rate of decrease of T is controlled by the user through cooling function that is incorporated into the algorithm. This class of sampling-based algorithms have found numerous applications in CPO due to their accuracy is approximating the global optima and low computational complexity, [64, 87, 88, 89, 90].

3.8.1 Other Optimization Algorithms

Semidefinite Programming (SDP)

SDP is an alternative to LP that allows to solve nonlinear convex objective functions, and provides a tighter relaxation to the binary constraint, [65, 91]. An SDP problem can be defined using quadratic programming, [92] as,

$$\begin{aligned} &\min\ c^T x \\ &\text{s.t.}: Y \succeq 0\ , \end{aligned} \tag{3.20}$$

where $Y = Y_0 + \sum_{i=1}^{N} x_i Y_i$ and $c \in \mathbb{R}^N$, $Y_0,\ldots,Y_N \in \mathbb{R}^{M\times M}$ are symmetric matrices. The sign $\succeq$ implies Y is positive semidefinite, i.e., $z^T Y z \geq 0\ \forall\ z \in \mathbb{R}^n$. One advantage of using SDP is that they can be solved efficiently.

Genetic Algorithms (GA)

GA are a category of commonly used metaheuristic optimization algorithms that mimic the process of natural selection using techniques such as, mutation and crossover, [93]. GA are versatile and suited for discrete combinatorial optimization as they do not make any assumptions on the objective function or the search space such as, requirement of derivatives. Similar to some variants of PSO, they provide a balance between exploitation and exploration of the search space by using random processes such as, crossover and mutation. GA tend to find the best approximation to the optimal solution and they also list meaningful candidates. The convergence of GA has been theoretically proven in [94]. For these advantages, this group of algorithms has been well-studied in CPO literature, [53, 95, 96, 97, 98].

GA work by selecting a random initial solution from the list of candidate solutions in the search space. Subsequently, in each iteration, a subset of solutions are selected which are ranked based on their fitness values or cost. A *crossover* operation is applied to mix and match *parents* the subset of solutions to create a new solution belonging to the new generation. The *mutation* operation then changes one or more values in the current solution to create a new *chromosome*. These steps of natural evolution are repeated until a convergence criteria, such as, maximum number of iterations, minimum required coverage, etc., is achieved. In the context of CPO, a *chromosome* can be referred to as a subset of cameras while, *crossover* can be described as swapping this subset with a subset of same size from another set of cameras, [99].

3.9 Continuous CPO

As discretization of the simulation model provides an added advantage to linearize the model to use BIP-based problem formulation, the use of continuous variables has been neglected. Smith et al., [100], proposed a continuous optimization approach for view and path planning of an UAV for multi-view stereo reconstruction of an urban scene. To tackle the complexity of continuous optimization, they modelled the path planning problem as a view selection problem by selecting the optimal path from a predefined set of paths. While such an approach relates the problem to the CPO problem, it is a simple and application-specific approximation that cannot be generalized for CPO problems. Optimizing a CPO problem all variables modelled in the continuous space requires complex geometrical calculations (e.g., plane volume intersections) to calculate the view of a camera. Problem formulation based on Mixed-Integer Programming (MIP), where part of the variables are modelled in space while the rest are modelled in discrete space, allows to circumvent this limitation.

Kirchof, [101], proposed a method where the orientation of the cameras was modelled in the continuous space, while the control points and the camera positions are modelled in discrete space. A *quality function* was introduced and related to the objective function through

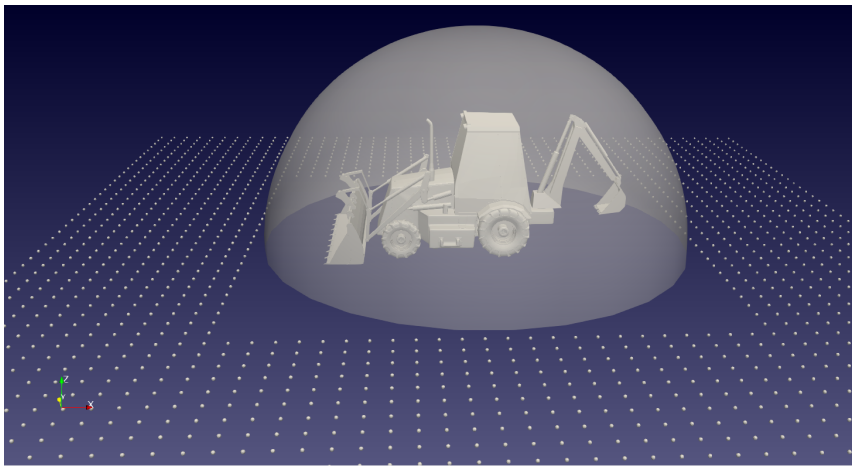


Figure 3.5: An illustration of the CPO problem simulation using continuous variables to represent camera pose.

a non-linear relationship and the problem was solved using non-linear programming methods. Similarly, Ismail et al., [102], proposed a sensor placement optimization problem to monitor defects in composite structures by modelling the sensor positions in continuous space with the rest of the decision variables modelled in the discrete domain. Sundeep and Geert, [103], argued about the limitations of sensor selection problem and proposed a method to represent the discrete grid with a continuous function using Taylor interpolation. Other works using continuous variables for SPO problems can be found in [12, 104, 105]. Although, CPO problems modelled in discrete domain simplify the problem to use linear programming, representing the camera pose using continuous variables may improve the solution in terms of coverage or cost, as sub-sample accuracy can be achieved using continuous variables.

In this work, we also propose a partly continuous space model for the CPO problem for vehicle surround-view by modelling the camera pose using continuous variables. To avoid complicating the problem, control points are sampled from the ground plane around the vehicle's model. An illustration of the modelled space is shown in Figure 3.5. The space model is defined with a 3D model of the vehicle placed at the center on the ground plane. Control points are sampled on the ground plane around the vehicle's model. A hemisphere, represented in spherical coordinates is placed surrounding the vehicle model. During optimization, points are sampled on this hemisphere which are then mapped onto the surface of the vehicle's model to represent a camera's pose. An illustration of the mapping of sampled points from the hemispherical surface to the surface of the vehicle's 3D model is shown in Figure 3.6, where the cameras in solid show invalid or rejected candidate camera poses while, the hollow cameras show valid camera poses. The mapping works by projecting a ray in the direction of the line passing through the ground plane and the center of the vehicle's 3D model. The intersection point between the projected ray and the 3D model denotes the position of the camera and the reverse direction of the ray denotes the orientation of the candidate camera. For an n-camera

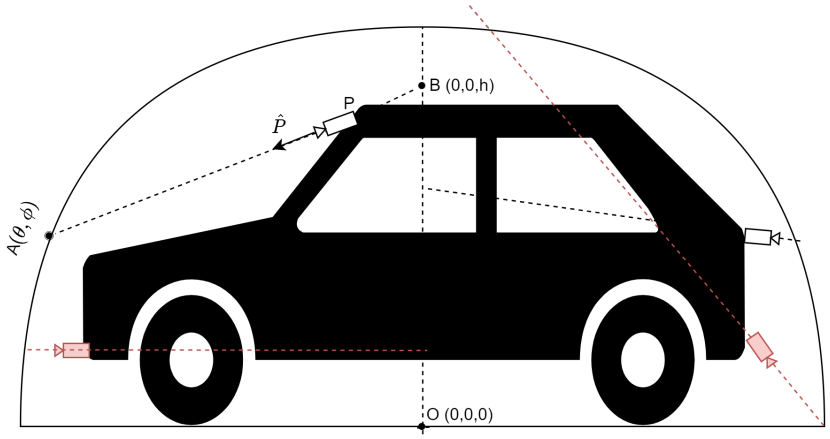


Figure 3.6: An illustration of the mapping of points sampled on the hemispherical surface to camera poses on the vehicle model's surface for the continuous CPO problem.

network pose optimization, a camera configuration is accepted if all n cameras have valid poses on the vehicle's surface.

As the problem involves non-linear steps, it is optimized using global black-box optimization methods. Black-box optimization methods work to achieve a balance between exploration and exploitation of the search space of an unknown objective function. These methods do not require computation of gradients, and therefore, are suitable for optimizing complex non-linear objective functions. The optimization methods work by defining a search space for the fitness function based on the bounds and constraints placed on the variables. At each iteration the search space is sampled and the fitness of the sample is calculated. The optimization functions use different strategies to go towards optima based on the fitness value at the current iteration and the global best fitness value encountered across previous iterations. Our work used a particle swarm optimization* method that uses fuzzy logic to balance exploration and exploitation, and a Bayesian Optimization (BO)† method which is known to provide a fast approach to optimize complex objective functions that are computationally expensive.

*https://github.com/aresio/fst-pso
†https://github.com/fmfn/BayesianOptimization

Chapter 4

Scientific Contributions

There two main scientific contributions made as part of this thesis. The first contribution is the development of a Binary Integer Programming-based method for multiple camera pose optimization using a multi-resolution approach with the cost function formulated at different resolution levels to maximize surrounding view coverage from vehicles, [Pub1, Pub2]. The second contribution is the development of a Mixed Integer Programming-based method for modelling camera pose and a specifically designed cost function to span both discrete and continuous domains with an aim to maximize overall coverage by the multiple camera system for realistic vehicle models, [Pub4].

4.1 BIP-based method for vehicle surround-view coverage

Binary Integer Programming is the commonly used problem formulation for camera placement optimization problems. The approach involves defining a space model for the problem and discretizing it by collecting samples from the space. The collected samples represent the sets of candidate camera poses and points in space that need to be covered by the camera. While the Integer programming method introduces a linear relaxation to the problem, BIP formulation adds a constraint requiring some variables to take a value of either one or zero. Despite the addition of the binary constraint, the BIP-based formulation is widely used for CPO problems as there exist exact and numerous heuristic optimization algorithms to solve the problem. Although, CPO problems have been studied well over the past couple of decades, they are largely limited to surveillance scenarios. Few works address the issue of optimal camera placement for vehicle surround view.

Through this contribution, we first introduced a BIP-based formulation for the specific case of CPO for vehicle surround-view coverage, [Pub1]. The challenge lies in the fact that firstly, the problem formulation needs to be done in 3D, while most surveillance applications simplify the problem to a projection into 2D space thereby, significantly reducing problem complexity.

Secondly, the sensitivity of orientation of the placed cameras is important which increases the sampling frequency in the dimension of camera orientation resulting in a larger optimization problem. Exact BIP-based optimization algorithms have a severe limitation in terms of computational resources and time. Although, heuristic algorithms address this concern, the accuracy may not always match with that of exact algorithms which are theoretically known to find the optimal solution.

To simplify the resource and time requirements of exact optimization algorithms, we proposed a novel heuristic algorithm, [Pub2], that works at multiple *resolution* levels by grouping the candidate cameras into clusters. The term *resolution* is used in similarity with image resolution as when candidate cameras are clustered, each camera pose represents a larger area on the vehicle's surface representing a *lower-resolution* image. The set of camera positions are clustered to represent the first (lowest) resolution and CPO is performed on the clusters. The selected clusters are unwrapped to the original resolution following which, they are clustered again or optimized without clustering depending on a predefined threshold that is system hardware dependent and set to avoid memory run-out errors. When CPO is performed on a subset of the original set of camera poses without any clustering, the solution obtained is the final solution of the optimization problem. This approach helps to reduce the time complexity of any optimization algorithm as at each resolution, the number of input decision variables is much lower. Additionally, experimental results showed that the proposed method slightly improves the coverage accuracy as well. The clustering method used for this heuristic algorithm has been published in [Pub3] as it improves results over the state-of-the-art supervoxel segmentation methods.

4.2 CPO in Continuous Domain

BIP-based problem is a subset selection problem as it requires all the variables to take only discrete variables and a set of candidate cameras and their coverage to be precomputed. Whereas, when part of the variables are allowed to take continuous values, the problem is known as Mixed Integer Programming problem. MIP problems also allow linear relaxations and there exist algorithms to solve the problems exactly to find the optimal solution. However, the complexity of finding an optimal solution for these problems is also $\mathscr{NP}$-hard and exact optimization algorithms are limited only to small sized problems, similar to BIP problems. The motivation to allow part of the variables to take continuous values lies in the fact that the quality of coverage by the optimized multi-camera network is dependent on the sampling frequency. When the sampling frequency is small, we get a poor approximation of the vehicle's surface but, high sampling frequency may result in a large number of variables thereby, increasing the complexity of the optimization algorithm. Modelling a problem with continuous variables is not easy as

defining coverage by the cameras requires complex geometrical calculations (surface volume intersection, etc.).

We proposed a method where the camera pose is defined in continuous space while, the control points are defined in discrete space by sampling the ground plane, [Pub4]. To model the camera positions in continuous space, an innovative solution is developed that uses black-box optimization techniques to solve the problem. These optimization methods do not require computation of gradients of the cost function and can balance exploration and exploitation of the search space. For the proposed method, during optimization, candidate cameras are sampled on a hemispherical surface around the vehicle's model which are mapped onto the vehicle's surface through ray-tracing. The direction of the ray defines the candidate camera's orientation while, the point of intersection of the ray with the vehicle's model defines it's position. A new camera coverage model based on the camera projection matrix, was also introduced as part of this work. This coverage model has the flexibility to model various types of cameras by simply changing the camera intrinsic parameters. Results show that modelling camera pose in continuous space achieves a significant improvement in total camera coverage in slightly higher time complexity when compared against the BIP model.

Chapter 5

Conclusion and Future Work

A new problem formulation for camera placement optimization for the specific case of vehicle surround-view was proposed. The problem was modelled as a Binary Integer Programming model with the objective as maximization of coverage of the surrounding area. Exact and heuristic algorithms were tested on the proposed camera placement optimization problem and the challenges in using exact algorithms for large simulations was highlighted. To address resource and time constraints from exact optimization algorithms, random sampling methods were used to reduce the size of input variables for the initial camera placement optimization problem. Highlighting the limitations of random sampling to reduce the number of decision variables in the optimization process, a new clustering-based approach was proposed. The clustering based approach works by grouping together candidate camera positions into sets of clusters depending on the spatial, textural and orientation similarities between the points on the vehicle's surface (or the candidate camera poses). Compared to random sampling-based selection of a subset of the candidate cameras, the clustering approach provides an improvement in overall coverage by the multi-camera network as important candidate cameras may be missed during random sampling while, they are preserved in the clustering-based approach. The proposed *multi-resolution* optimization approach reduced computational times by up to 60 times. The results for this proposed heuristic optimization method show that the method can also marginally improve coverage results as it can achieve *sub-voxel* accuracy for camera placement.

The clustering method used for the proposed multi-resolution optimization method was designed to emphasise the similarity between the surface normals of the points or voxels. For uniformly textured planar regions on a surface, the surface normals play an important role in distinguishing regional boundaries. As a result, the method used for the multi-resolution optimization approach was an improvement over state-of-the-art supervoxel clustering methods. The proposed method was tested on an open RGBD dataset with over 1400 point clouds and the results show that it produced most compact supervoxels and outperformed other methods on three of the four metrics. The proposed clustering method, however, requires longer computa-

tional time due to the iterative process of removing isolated cluster seeds and adding new ones in unclustered regions. A new CPO problem formulation using decision variables for camera poses in continuous space was proposed. The method follows a mixed integer programming-based approach where part of the decision variables are discrete while the rest are continuous. The continuous CPO problem was non-linear due to the presence of ray-tracing and mapping of coordinates from Cartesian to spherical coordinate space and vice versa. It was solved using global black-box optimization algorithms which do not need information about gradients of the objective function, including Bayesian Optimization and Particle Swarm Optimization. Results of the continuous CPO on more than a hundred 3D models showed that it performs significantly better than the discrete problem formulation at a marginally increased computational time.

The work presented in this thesis addresses some of the major challenges faced in CPO for vehicle surround view coverage but, there exists potential for future work. The work done continuous CPO can be extended to reduce the search space further. This can be achieved by identifying important or useful regions on the vehicle model's surface and limiting the range of the decision variables to only those regions. A smaller search space can increase the chance of finding camera placement solutions with higher quality. The camera models can be extended to include commonly used lenses such as, fish-eye lens. This can be achieved by modifying the intrinsic parameter matrix accordingly. The parameters can be modified to include lens distortion parameters for fish-eye or wide-angled lenses which may lead to realistic modelling of the problem. Additional optimization algorithms may be tested for the continuous CPO problem to increase accuracy and efficiency. As the introduced discrete CPO problems are linear and simple, it may benefit to model realistic cameras for that problem and compare the results against continuous CPO problem in detail. The camera matrix-based coverage model can be implemented for the discrete CPO problems for simplicity and generalizability of the problem of camera modelling. Lastly, although, the proposed supervoxel segmentation method produces compact supervoxels with better boundary adherence, it is of interest to decrease the computational time as high computational time requirement for supervoxels may be counter-productive to their use case. A better cluster seed initialization scheme may be used to circumvent the iterative process of cluster seeds' creation and deletion to improve performance.

Chapter 6

List of publications

Pub 1 V.A. Puligandla, S. Lončarić, ”Optimal Camera Placement To Visualize Surrounding View From Heavy Machinery”, in *2020 2nd Asia Pacific Information Technology Conference*, 2020 Jan 17, pp. 52–59.

Pub 2 V.A. Puligandla, S. Lončarić, ”A multiresolution approach for large real-world camera placement optimization problems”, *IEEE Access*, Vol. 10, 2022, pp. 61601-61616.

Pub 3 V.A. Puligandla, S. Lončarić, ”A Supervoxel Segmentation Method With Adaptive Centroid Initialization for Point Clouds”, *IEEE Access*, Vol. 10, 2022, pp. 98525-98534.

Pub 4 V.A. Puligandla, S. Lončarić, ”A Continuous Camera Placement Optimization Model For Surround View”, *IEEE Transactions on Intelligent Vehicles*, 2023, Jul 26, doi: 10.1109/TIV.2023.3299199

Chapter 7

Author's contribution to the publications

The results presented in this thesis result from the research work carried out during the period of 2019-2023 at the University of Zagreb Faculty of Electrical Engineering and Computing, Unska 3, HR-10000 Zagreb, Croatia, mostly as part of the ImmerSAFE research project funded under the European Union's (EU's) H2020-MSCA-ITN-2017 call, part of the Marie Sklodowska-Curie Actions-Innovative Training Networks (ITN) funding scheme under project 764951.

The thesis includes four publications written in collaboration with the coauthor of the published papers. The author's contribution to each paper consists of the text writing, software implementation, performing the required experiments, and results analysis and presentation.

[Pub1] In the paper titled **"Optimal Camera Placement To Visualize Surrounding View From Heavy Machinery"**, the author has proposed a problem formulation framework for camera placement optimization with a goal to maximize surrounding view coverage by multiple cameras placed a vehicle's surface with goal of achieving surround-view coverage around the vehicle. The problem od camera placement optimization was limited to surveillance scenarios, and the presented paper introduces this problem for vehicle surround-view coverage, whose challeneges require to be addressed due to the growing number of sensors being added to vehicles to achieve autonomous driving. The proposed methodology proposes optimal camera placement problem solving for the general case of any 3D models of vehicles.

[Pub2] In the paper titled **A multiresolution approach for large real-world camera placement optimization problems**, the author has addressed the limitations of exact optimization algorithms when applied to large simulated scenarios. Vehicle surround-view coverage requires a high degree of precision in the orientation of cameras which results in a large number of decision variables thereby, increasing the complexity of optimization algorithms. The proposed heuristic optimization algorithm works to limit the number of decision variables without affecting the coverage quality, and the method in-fact marginally improves the coverage quality.

[Pub3] In the paper titled, **A Supervoxel Segmentation Method With Adaptive Centroid Initialization for Point Clouds**, the author has presented a novel supervoxel segmentation

algorithm that emphasises the importance of point orientations when oversegmenting 3D RGBD point clouds. The method uses a novel cluster seed initialization scheme which results in the method out-performing the state-of-the-art in three out of four metrics. The method produces highly compact supervoxels with better boundary adherence which are desired qualities in a supervoxel segmentation algorithm.

[Pub4] In the paper titled, **A Continuous Camera Placement Optimization Model For Surround View**, the author has proposed a new problem formulation for camera placement optimization for vehicle surround-view coverage which models decision variables in the continuous space. Variables modelled in continuous space allow the camera poses to take any values between a range and doesn't require sampling of space to collect decision variables. This approach brings down the number of decision variables from thousands to tens, depending on the number of cameras whose poses need to be optimized. The method is modelled in a mixed integer non-linear programming formulation and is optimized using global optimization methods that do not require gradient information of the objective function. The proposed method also uses a novel camera coverage model based on the camera matrix.

Bibliography

[1]Buljeta, D., Vranješ, M., Mar četa, Z., Kovačević, J., "Surround view algorithm for parking assist system", in 2019 Zooming Innovation in Consumer Technologies Conference (ZINC). IEEE, 2019, str. 21–26.

[2]Appia, V., Hariyani, H., Sivasankaran, S., Liu, S., Chitnis, K., Mueller, M., Batur, U., Agarwa, G., "Surround view camera system for adas on ti's tdax socs", Texas Instruments Technical Note, Vol. 2, 2015.

[3]Hedi, A., Lon čarić, S., "A system for vehicle surround view", IFAC Proceedings Volumes, Vol. 45, No. 22, 2012, str. 120–125.

[4]Rosique, F., Navarro, P. J., Fernández, C., Padilla, A., "A systematic review of perception system and simulators for autonomous vehicles research", Sensors, Vol. 19, No. 3, 2019, str. 648.

[5]Häne, C., Heng, L., Lee, G. H., Fraundorfer, F., Furgale, P., Sattler, T., Pollefeys, M., "3d visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection", Image and Vision Computing, Vol. 68, 2017, str. 14–27.

[6]Shao, X., Liu, X., Zhang, L., Zhao, S., Shen, Y., Yang, Y., "Revisit surround-view camera system calibration", in 2019 IEEE International Conference on Multimedia and Expo (ICME). IEEE, 2019, str. 1486–1491.

[7]Chen, Y., Zhang, L., Shen, Y., Zhao, B. N., Zhou, Y., "Extrinsic self-calibration of the surround-view system: A weakly supervised approach", IEEE Transactions on Multimedia, 2022.

[8]Liang, Z., Chen, L., An, F., "3d vehicle surround view algorithm for embedded platform", in Journal of Physics: Conference Series, Vol. 2253, No. 1. IOP Publishing, 2022, str. 012026.

[9]Gasparyan, S., Vasilianov, G., “Real-time 3d surround view system for vehicle based on panoramic stitching image”, in International School on Neural Networks, Initiated by IIASS and EMFCSC. Springer, 2022, str. 85–93.

[10]Han, M. P., Monga, D., “Method and system for presenting panoramic surround view in vehicle”, uS Patent App. 14/585,682. Jun. 30 2016.

[11]Nobori, K., Ukita, N., Hagita, N., “A surround view image generation method with low distortion for vehicle camera systems using a composite projection”, in 2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA). IEEE, 2017, str. 386–389.

[12]Ostachowicz, W., Soman, R., Malinowski, P., “Optimization of sensor placement for structural health monitoring: A review”, Structural Health Monitoring, Vol. 18, No. 3, 2019, str. 963–988.

[13]An, H., Youn, B. D., Kim, H. S., “A methodology for sensor number and placement optimization for vibration-based damage detection of composite structures under model uncertainty”, Composite Structures, Vol. 279, 2022, str. 114863.

[14]Savkin, A. V., Huang, H., “Navigation of a uav network for optimal surveillance of a group of ground targets moving along a road”, IEEE Transactions on Intelligent Transportation Systems, Vol. 23, No. 7, 2021, str. 9281–9285.

[15]Koch, T., Körner, M., Fraundorfer, F., “Automatic and semantically-aware 3d uav flight planning for image-based 3d reconstruction”, Remote Sensing, Vol. 11, No. 13, 2019, str. 1550.

[16]Wang, X., Zhang, H., Fan, S., Gu, H., “Coverage control of sensor networks in iot based on rpso”, IEEE internet of things journal, Vol. 5, No. 5, 2018, str. 3521–3532.

[17]Altahir, A. A., Asirvadam, V. S., Sebastian, P., Hamid, N. H. B., Ahmed, E. F., “Optimizing visual sensors placement with risk maps using dynamic programming”, IEEE Sensors Journal, Vol. 22, No. 1, 2021, str. 393–404.

[18]Zhang, X., Chen, X., Alarcon-Herrera, J. L., Fang, Y., “3-d model-based multi-camera deployment: A recursive convex optimization approach”, IEEE/ASME Transactions on Mechatronics, Vol. 20, No. 6, 2015, str. 3157–3169.

[19]Mantini, P., Shah, S. K., “Camera placement optimization conditioned on human behavior and 3d geometry”, in International Conference on Computer Vision Theory and Applications, Vol. 4. SCITEPRESS, 2016, str. 225–235.

[20]Rahimian, P., Kearney, J. K., “Optimal camera placement for motion capture systems”, IEEE transactions on visualization and computer graphics, Vol. 23, No. 3, 2016, str. 1209–1221.

[21]Bogaerts, B., Sels, S., Vanlanduit, S., Penne, R., “Interactive camera network design using a virtual reality interface”, Sensors, Vol. 19, No. 5, 2019, str. 1003.

[22]Suresh, M. S., Narayanan, A., Menon, V., “Maximizing camera coverage in multicamera surveillance networks”, IEEE Sensors Journal, Vol. 20, No. 17, 2020, str. 10 170–10 178.

[23]Kim, J., Ham, Y., Chung, Y., Chi, S., “Systematic camera placement framework for operation-level visual monitoring on construction jobsites”, Journal of Construction Engineering and Management, Vol. 145, No. 4, 2019, str. 04019019.

[24]Angella, F., Reithler, L., Gallesio, F., “Optimal deployment of cameras for video surveillance systems”, in 2007 IEEE conference on advanced video and signal based surveillance. IEEE, 2007, str. 388–392.

[25]Indu, S., Srivastava, S., Sharma, V., “Optimal camera placement and orientation of a multi-camera system for self driving cars”, in Proceedings of the 2020 4th International Conference on Vision, Image and Signal Processing, 2020, str. 1–5.

[26]Dey, J., Taylor, W., Pasricha, S., “Vespa: A framework for optimizing heterogeneous sensor placement and orientation for autonomous vehicles”, IEEE Consumer Electronics Magazine, Vol. 10, No. 2, 2020, str. 16–26.

[27]Puligandla, V. A., Lon čarić, S., “Optimal camera placement to visualize surrounding view from heavy machinery”, in Proceedings of the 2020 2nd Asia Pacific Information Technology Conference, 2020, str. 52–59.

[28]Chvátal, V., “A combinatorial theorem in plane geometry”, Journal of Combinatorial Theory, Series B, Vol. 18, No. 1, 1975, str. 39–41.

[29]Hartley, R., Zisserman, A., Multiple View Geometry in Computer Vision, 2nd ed. Cambridge University Press, 2004.

[30]Machairas, V., Faessel, M., Cárdenas-Peña, D., Chabardes, T., Walter, T., Decenciere, E., “Waterpixels”, IEEE Transactions on Image Processing, Vol. 24, No. 11, 2015, str. 3707–3716.

[31]Levinshtein, A., Stere, A., Kutulakos, K. N., Fleet, D. J., Dickinson, S. J., Siddiqi, K., “Turbopixels: Fast superpixels using geometric flows”, IEEE transactions on pattern analysis and machine intelligence, Vol. 31, No. 12, 2009, str. 2290–2297.

[32]Van den Bergh, M., Boix, X., Roig, G., De Capitani, B., Van Gool, L., "Seeds: Superpixels extracted via energy-driven sampling", in Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VII 12. Springer, 2012, str. 13–26.

[33]Moore, A. P., Prince, S. J., Warrell, J., Mohammed, U., Jones, G., "Superpixel lattices", in 2008 IEEE conference on computer vision and pattern recognition. IEEE, 2008, str. 1–8.

[34]Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S., "Slic superpixels compared to state-of-the-art superpixel methods", IEEE transactions on pattern analysis and machine intelligence, Vol. 34, No. 11, 2012, str. 2274–2282.

[35]Veksler, O., Boykov, Y., Mehrani, P., "Superpixels and supervoxels in an energy optimization framework", in Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part V 11. Springer, 2010, str. 211–224.

[36]Weikersdorfer, D., Schick, A., Cremers, D., "Depth-adaptive supervoxels for rgb-d video segmentation", in 2013 IEEE International Conference on Image Processing. IEEE, 2013, str. 2708–2712.

[37]Gao, G., Lauri, M., Zhang, J., Frintrop, S., "Saliency-guided adaptive seeding for supervoxel segmentation", in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017, str. 4938–4943.

[38]Papon, J., Abramov, A., Schoeler, M., Worgotter, F., "Voxel cloud connectivity segmentation-supervoxels for point clouds", in Proceedings of the IEEE conference on computer vision and pattern recognition, 2013, str. 2027–2034.

[39]Lin, Y., Wang, C., Zhai, D., Li, W., Li, J., "Toward better boundary preserved supervoxel segmentation for 3d point clouds", ISPRS journal of photogrammetry and remote sensing, Vol. 143, 2018, str. 39–47.

[40]Sha, Z., Zhu, Q., Chen, Y., Wang, C., Nurunnabi, A., Li, J., "A boundary-enhanced supervoxel method for 3d point clouds", in IGARSS 2020-2020 IEEE International Geoscience and Remote Sensing Symposium. IEEE, 2020, str. 2771–2774.

[41]Ni, H., Niu, X., "Svla: A compact supervoxel segmentation method based on local allocation", Isprs journal of photogrammetry and remote sensing, Vol. 163, 2020, str. 300–311.

[42]Landrieu, L., Boussaha, M., “Point cloud oversegmentation with graph-structured deep metric learning”, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, str. 7440–7449.

[43]Hui, L., Yuan, J., Cheng, M., Xie, J., Zhang, X., Yang, J., “Superpoint network for point cloud oversegmentation”, in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, str. 5510–5519.

[44]Shreiner, D. *et al.*, OpenGL programming guide: the official guide to learning OpenGL, versions 3.0 and 3.1. Pearson Education, 2009.

[45]Schroeder, W. J., Avila, L. S., Hoffman, W., “Visualizing with vtk: a tutorial”, IEEE Computer graphics and applications, Vol. 20, No. 5, 2000, str. 20–27.

[46]Squillacote, A. H., Ahrens, J., Law, C., Geveci, B., Moreland, K., King, B., The paraview guide. Kitware Clifton Park, NY, 2007, Vol. 366.

[47]Tobler, R. F., Maierhofer, S., “A mesh data structure for rendering and subdivision”, computer science, 2006.

[48]Munshi, A., “The opencl specification”, in 2009 IEEE Hot Chips 21 Symposium (HCS). IEEE, 2009, str. 1–314.

[49]Lam, S. K., Pitrou, A., Seibert, S., “Numba: A llvm-based python jit compiler”, in Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, 2015, str. 1–6.

[50]Hörster, E., Lienhart, R., “On the optimal placement of multiple visual sensors”, in Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks, 2006, str. 111–120.

[51]Kritter, J., Brévilliers, M., Lepagnot, J., Idoumghar, L., “On the real-world applicability of state-of-the-art algorithms for the optimal camera placement problem”, in 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT). IEEE, 2019, str. 1103–1108.

[52]Murray, A. T., Kim, K., Davis, J. W., Machiraju, R., Parent, R., “Coverage optimization to support security monitoring”, Computers, environment and urban systems, Vol. 31, No. 2, 2007, str. 133–147.

[53]David, P., Idasiak, V., Kratz, F., “A sensor placement approach for the monitoring of indoor scenes”, in Smart Sensing and Context: Second European Conference, EuroSSC

2007, Kendal, England, October 23-25, 2007. Proceedings 2. Springer, 2007, str. 110–125.

[54]Horster, E., Lienhart, R., “Multi-camera networks: Concepts and applications”, 2009.

[55]Yabuta, K., Kitazawa, H., “Optimum camera placement considering camera specification for security monitoring”, in 2008 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2008, str. 2114–2117.

[56]Zhao, J., Sen-ching, S. C., “Optimal visual sensor planning”, in 2009 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2009, str. 165–168.

[57]Zhang, H., Xia, L., Tian, F., Wang, P., Cui, J., Tang, C., Deng, N., Ma, N., “An optimized placement algorithm for collaborative information processing at a wireless camera network”, in 2013 IEEE International Conference on Multimedia and Expo (ICME). IEEE, 2013, str. 1–6.

[58]Becker, E., Guerra-Filho, G., Makedon, F., “Automatic sensor placement in a 3d volume”, in Proceedings of the 2nd International Conference on PErvasive Technologies Related to Assistive Environments, 2009, str. 1–8.

[59]Malhotra, A., Singh, D., Dadlani, T., Morales, L. Y., “Optimizing camera placements for overlapped coverage with 3d camera projections”, in 2022 International Conference on Robotics and Automation (ICRA). IEEE, 2022, str. 5002–5009.

[60]Kim, T.-H., Park, T.-H., “Placement optimization of multiple lidar sensors for autonomous vehicles”, IEEE Transactions on Intelligent Transportation Systems, Vol. 21, No. 5, 2019, str. 2139–2145.

[61]Morsly, Y., Aouf, N., Djouadi, M. S., Richardson, M., “Particle swarm optimization inspired probability algorithm for optimal camera network placement”, IEEE Sensors Journal, Vol. 12, No. 5, 2011, str. 1402–1412.

[62]Erdem, U. M., Sclaroff, S., “Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements”, Computer Vision and Image Understanding, Vol. 103, No. 3, 2006, str. 156–169.

[63]Gupta, A., Pati, K. A., Subramanian, V. K., “A nsga-ii based approach for camera placement problem in large scale surveillance application”, in 2012 4th International Conference on Intelligent and Advanced Systems (ICIAS2012), Vol. 1. IEEE, 2012, str. 347–352.

[64]Mittal, A., Davis, L. S., “A general method for sensor planning in multi-sensor systems: Extension to random occlusion”, International journal of computer vision, Vol. 76, 2008, str. 31–52.

[65]Liu, J., Sridharan, S., Fookes, C., “Recent advances in camera planning for large area surveillance: A comprehensive review”, ACM Computing Surveys (CSUR), Vol. 49, No. 1, 2016, str. 1–37.

[66]Mavrinac, A., Chen, X., “Modeling coverage in camera networks: A survey”, International journal of computer vision, Vol. 101, 2013, str. 205–226.

[67]Boyd, S., Mattingley, J., “Branch and bound methods”, Notes for EE364b, Stanford University, 2007, str. 2006–07.

[68]Dantzig, G., Linear programming and extensions. Princeton university press, 1963.

[69]Land, A. H., Doig, A. G., An automatic method for solving discrete programming problems. Springer, 2010.

[70]Kritter, J., Brévilliers, M., Lepagnot, J., Idoumghar, L., “On the optimal placement of cameras for surveillance and the underlying set cover problem”, Applied Soft Computing, Vol. 74, 2019, str. 133–153.

[71]Mostafavi, S. A., Dehghan, M., “Optimal visual sensor placement for coverage based on target location profile”, Ad Hoc Networks, Vol. 9, No. 4, 2011, str. 528–541.

[72]Gonzalez-Barbosa, J.-J., Garcia-Ramirez, T., Salas, J., Hurtado-Ramos, J.-B. *et al.*, “Optimal camera placement for total coverage”, in 2009 IEEE International Conference on Robotics and Automation. IEEE, 2009, str. 844–848.

[73]Feige, U., “A threshold of ln n for approximating set cover”, Journal of the ACM (JACM), Vol. 45, No. 4, 1998, str. 634–652.

[74]Gandhi, R., Khuller, S., Srinivasan, A., “Approximation algorithms for partial covering problems”, Journal of Algorithms, Vol. 53, No. 1, 2004, str. 55–84.

[75]Slavík, P., “Improved performance of the greedy algorithm for partial cover”, Information Processing Letters, Vol. 64, No. 5, 1997, str. 251–254.

[76]Zhao, J., Yoshida, R., Cheung, S.-c. S., Haws, D., “Approximate techniques in solving optimal camera placement problems”, International Journal of Distributed Sensor Networks, Vol. 9, No. 11, 2013, str. 241913.

[77]Kennedy, J., Eberhart, R., “Particle swarm optimization”, in Proceedings of ICNN’95-international conference on neural networks, Vol. 4. IEEE, 1995, str. 1942–1948.

[78]Kennedy, J., Eberhart, R. C., “A discrete binary version of the particle swarm algorithm”, in 1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation, Vol. 5. IEEE, 1997, str. 4104–4108.

[79]Conci, N., Lizzi, L., “Camera placement using particle swarm optimization in visual surveillance applications”, in 2009 16th IEEE international conference on image processing (ICIP). IEEE, 2009, str. 3485–3488.

[80]Konda, K. R., Conci, N. *et al.*, “Global and local coverage maximization in multi-camera networks by stochastic optimization”, Infocommunications Journal, Vol. 5, No. 1, 2013, str. 1–8.

[81]Xu, Y.-C., Lei, B., Hendriks, E. A., “Camera network coverage improving by particle swarm optimization”, EURASIP Journal on Image and Video Processing, Vol. 2011, 2011, str. 1–10.

[82]Xu, Y.-C., Lei, B., Hendriks, E. A., “Constrained particle swarm algorithms for optimizing coverage of large-scale camera networks with mobile nodes”, Soft computing, Vol. 17, 2013, str. 1047–1057.

[83]Wang, X., Zhang, H., Gu, H., “Solving optimal camera placement problems in iot using lh-rpso”, IEEE Access, Vol. 8, 2019, str. 40 881–40 891.

[84]Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., Teller, E., “Equation of state calculations by fast computing machines”, The journal of chemical physics, Vol. 21, No. 6, 1953, str. 1087–1092.

[85]Gelfand, A. E., Smith, A. F., “Sampling-based approaches to calculating marginal densities”, Journal of the American statistical association, Vol. 85, No. 410, 1990, str. 398–409.

[86]Kirkpatrick, S., “Optimization by simulated annealing: Quantitative studies”, Journal of statistical physics, Vol. 34, 1984, str. 975–986.

[87]Debaque, B., Jedidi, R., Prevost, D., “Optimal video camera network deployment to support security monitoring”, in 2009 12th International Conference on Information Fusion. IEEE, 2009, str. 1730–1736.

[88]Liu, J., Fookes, C., Wark, T., Sridharan, S., “On the statistical determination of optimal camera configurations in large scale surveillance networks”, in Computer Vision–ECCV

2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part I 12. Springer, 2012, str. 44–57.

[89]Liu, J., Sridharan, S., Fookes, C., Wark, T., "Optimal camera planning under versatile user constraints in multi-camera image processing systems", IEEE Transactions on Image Processing, Vol. 23, No. 1, 2013, str. 171–184.

[90]Akbarzadeh, V., Gagne, C., Parizeau, M., Argany, M., Mostafavi, M. A., "Probabilistic sensing model for sensor placement optimization based on line-of-sight coverage", IEEE transactions on instrumentation and measurement, Vol. 62, No. 2, 2012, str. 293–303.

[91]Laurent, M., "A comparison of the sherali-adams, lovász-schrijver, and lasserre relaxations for 0–1 programming", Mathematics of Operations Research, Vol. 28, No. 3, 2003, str. 470–496.

[92]Vandenberghe, L., Boyd, S., "Semidefinite programming", SIAM review, Vol. 38, No. 1, 1996, str. 49–95.

[93]Galletly, J. E., "An overview of genetic algorithms", Kybernetes, Vol. 21, No. 6, 1992, str. 26–30.

[94]Holland, J. H., Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press, 1992.

[95]Feng, G., Liu, M., Wang, G., "Genetic algorithm based optimal placement of pir sensors for human motion localization", Optimization and Engineering, Vol. 15, 2014, str. 643–656.

[96]Indu, S., Chaudhury, S., Mittal, N. R., Bhattacharyya, A., "Optimal sensor placement for surveillance of large spaces", in 2009 Third ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC). IEEE, 2009, str. 1–8.

[97]Yao, Y., Chen, C.-H., Abidi, B., Page, D., Koschan, A., Abidi, M., "Sensor planning for automated and persistent object tracking with multiple cameras", in 2008 IEEE conference on computer vision and pattern recognition. IEEE, 2008, str. 1–8.

[98]Yao, Y., Chen, C.-H., Abidi, B., Page, D., Koschan, A., Abidi, M., "Can you see me now? sensor positioning for automated and persistent surveillance", IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), Vol. 40, No. 1, 2009, str. 101–115.

[99]van den Hengel, A., Hill, R., Ward, B., Cichowski, A., Detmold, H., Madden, C., Dick, A., Bastian, J., "Automatic camera placement for large scale surveillance networks", in 2009 Workshop on Applications of Computer Vision (WACV). IEEE, 2009, str. 1–6.

[100]Smith, N., Moehrle, N., Goesele, M., Heidrich, W., "Aerial path planning for urban scene reconstruction: A continuous optimization method and benchmark", ACM Transactions on Graphics (TOG), Vol. 37, 2018, str. 1 - 15.

[101]Kirchhof, N., "Optimal placement of multiple sensors for localization applications", in International Conference on Indoor Positioning and Indoor Navigation. IEEE, 2013, str. 1–10.

[102]Ismail, Z., Mustapha, S., Fakih, M. A., Tarhini, H., "Sensor placement optimization on complex and large metallic and composite structures", Structural Health Monitoring, Vol. 19, No. 1, 2020, str. 262–280.

[103]Chepuri, S. P., Leus, G., "Continuous sensor placement", IEEE signal processing letters, Vol. 22, No. 5, 2014, str. 544–548.

[104]Bianco, S., Tisato, F., "Sensor placement optimization in buildings", in Image Processing: Machine Vision Applications V, Vol. 8300. SPIE, 2012, str. 9–21.

[105]Guratzsch, R. F., Mahadevan, S., "Structural health monitoring sensor placement optimization under uncertainty", AIAA journal, Vol. 48, No. 7, 2010, str. 1281–1289.

Publications

Publication 1

V.A. Puligandla, S. Lončarić, ”Optimal Camera Placement To Visualize Surrounding View From Heavy Machinery”, in *2020 2nd Asia Pacific Information Technology Conference*, 2020 Jan 17, pp. 52–59.

Optimal Camera Placement To Visualize Surrounding View From Heavy Machinery

V. Anirudh Puligandla
apuligandla@fer.hr
University of Zagreb
Zagreb, Croatia

Sven Lončarić
sven.loncaric@fer.hr
University of Zagreb
Zagreb, Croatia

ABSTRACT

Computer vision-based advanced driver assistance systems (ADAS) increase safety of operations involving heavy machinery. ADAS systems using multiple cameras can be used for surround-view visualization of complex vehicles with blind spots. Such systems are also useful for autonomous vehicles. Multiple camera systems used to capture surrounding view of heavy machinery require complex design due to the complexity in size and shape of the vehicles. In this paper, we present a novel method for determining the optimal camera pose i.e. placement and orientation in three-dimensional space, given the shape of the vehicle, in order to maximize surrounding area coverage. The first method determines camera poses using a fixed pre-determined number of cameras, while the second method determines both camera poses and the number of cameras. The problem is modelled and solved using three different deterministic optimization algorithms: 1) single objective binary integer programming approach; 2) single objective greedy algorithm; and 3) bi-objective binary integer programming approach. The methods are validated using a set of realistic 3-D vehicle models. Experimental validation has been conducted to compare the proposed methods with respect to coverage quality and computation time metrics. The experimental results have demonstrated that the proposed methods provide accurate solutions to the camera pose and the number of camera optimization.

CCS CONCEPTS

• **Theory of Computation** → **Mathematical Optimization**; • **Computing Methodologies** → **Computer Vision**; • **Human Centered Computing** → **Visualization**.

KEYWORDS

optimal camera placement, binary integer programming, optimization, 3D visualization

1 INTRODUCTION

Increasing research interest in autonomous vehicles has led to the rise of surround view camera system solutions to display surrounding area of a vehicle to the driver with an aim to aid the driver while maneuvering and parking the vehicle. Several leading manufacturers have already launched products providing surround view of commercial vehicles, [3, 9, 23]. Consequently, research over the last few years focused on improving these vehicle surround vision systems starting from the camera system calibration stage, [18, 20, 35], to image/video processing, [18, 19, 44, 45], until the final displaying stage, [10, 17, 22]. Taking advantage of the large amounts of information generated by these surround vision systems, research has also covered applications such as parking lot detection, [16], and, pedestrian detection, [14]. Methods were further developed for 3D surround view systems, [11], with depth perception coming from stereo vision, [8], or, through the use of multi-modal sensing, [39]. The most interesting point to note here would be that all of the above mentioned methods are specific to small vehicles (cars).

Figure 1: Example showing complex shape and structure of various construction equipment. (source: Taycor Financial)

Recent research highlights the problems associated with the so called *blind spots* present on heavy machinery or construction equipment and the risk they pose to workers working nearby the equipment, [6, 37]. Such concerns show that surround view systems are required also for heavy machinery. However, designing a multi-camera system for vehicles with such complex shapes and structures is not as straightforward as for the case of cars (Figure 1). The motivation for the work presented in this article lies on the fact that multi-camera systems, to achieve surround vision for heavy machinery, can be

APIT 2020, January 17–19, 2020, Bali Island, Indonesia

ACM ISBN 978-1-4503-7685-3/20/01…$15.00
https://doi.org/10.1145/3379310.3379331

designed in a smart and cost efficient way by optimizing the number and locations of the cameras that are to be placed on the vehicle.

1.1 Related Work

The Optimal Camera Placement (OCP) problem has been thoroughly studied over the past decade, particularly for surveillance applications, [24, 26]. Apart from surveillance, OCP problem has found interest in various applications such as, wireless sensor network deployment, [4, 46], industrial monitoring, [47], motion capture systems, [38], human behaviour analysis, [28], agriculture, [12], and even online gaming, [2], to name a few.

Some literature [41] classifies the OCP problem into categories such as target-based, area-based and probing-based coverage models. The mention of OCP problem, here, refers to both area and target coverage models as a single category. Target coverage model can be considered as a subset or special case of the area coverage model because, although, both approach the problem with an objective to maximize coverage of a specific target or area, former might have additional constraints limiting the number of cameras or the total cost of the multi-camera system.

The OCP problem is almost always related with the Art Gallery Problem (AGP), [5]. Nevertheless, they also highlight the limitations in field of view of a camera as compared to that of security guards in the AGP. The first significant contribution towards exactly solving the OCP problem can be attributed to Erdem and Sclaroff, [7], and, Hörster and Lienhart, [21]. Hörster and Lienhart formulate the problem in a 2D grid using binary decision variables representing possible camera poses and target points that are to be covered. Their problem formulation results in selecting the best combination through an exhaustive search from a sufficiently large set. They defined multiple problem statements to minimize the cost of the multi-camera system or to maximize coverage of target points. Their approach of using a linear model based on Binary Integer Programming (BIP) formed the basis of many other approaches that were later presented, [24].

In [15], Gupta *et al.* solved the OCP problem using a multi-objective genetic algorithm. They use a BIP formulation to maximize multi-sensor system coverage based on different qualitative aspects. They mention the use of a search heuristic due to the computational complexity of looking for an exact solution. However, their method is limited to 2D. Citing the *"blind-area"* issue caused by 2D modelling of the OCP problem, Zhang *et al.* presented an extension to the BIP formulation into 3D, [46]. They model the surveillance scene into small regions and categorize them based on essentiality. Such modelling does not apply to the case of surround vision where the surrounding area of a vehicle is to be covered uniformly. Additionally, introduction of *"balance scheme"*, along with essential region and wireless connectivity constraints, further complicate the model.

Becker *et al.* proposed a voting scheme based algorithm for automatic sensor placement in a 3D volume, [4], but, this method is limited to optimization of only one sensor and does not apply to a multi-camera system. Kritter *et al.*, [25], explored the accuracy and applicability of deterministic approaches to solve the OCP problem and compare it with heuristic methods. They tested the methods on models of European cities for target surveillance and highlight real-world applicability of the OCP problem. Rebai *et al.*, [41], studied the applicability of bi-objective formulation by proposing three different exact optimization models. Ahn *et al.*, [1], proposed a two-phase algorithm to reduce the complexity of the problem. They divide the problem into low and high resolution phases, resulting in many sub-problems that are then optimized using a heuristic. The high computational complexity requirements of the exact BIP-based OCP problem resulted in the development of approximation algorithms that guarantee feasibility, but not, optimality, [48]. There are also some methods, [43, 47], that deviate from the usual 2D/3D grid based modelling by using continuous modelling for problem formulation.

To the best of our knowledge, no methods or results are known for multi-camera system deployment problem to achieve surround view for vehicles. This contribution is the first to consider such a problem. The OCP problem for surround vision differs from surveillance scenarios in the way the problem is modelled. Surveillance scenarios enforce certain restrictions on placement and orientation of cameras (such as height, viewing direction, target coverage, *etc.*), while, in the case of surround view, non-planar structure of the vehicles results in poses across 6 Degrees Of Freedom (DOF).

1.2 Binary Integer Programming

Our proposed problem formulation builds over the BIP method as mentioned in [21]. Our method is applied to 3D space, while, Hörster and Lienhart's problem is applied in 2D space. This implies that the constraints that apply to the optimization problem need to be extended to 3D space. Modelling the problem space by dividing it into a 3D grid of samples results in a set of binary variables. Since, the optimization method works on the set of binary variables, the resulting problem model remains similar for both 2D and 3D cases. However, binary variables in 3D space comprise of a larger set of discrete samples representing complex geometry, when compared to the case of 2D space. Constraints have to be enforced in additional dimensions, and, coupled with 3D space modelling, the problem, on the whole, is bigger in terms of computational complexity. Owing to these similarities and differences, the problem description of single objective BIP formulation has been omitted from this document, but, modelling the space in 3D is elaborated.

We also tested the Greedy algorithm as mentioned in [21] on our problem of surround view for heavy duty vehicles, while, extending it to the 3D case. As discrete problem formulation results only in a larger set of binary variables, the algorithm remains similar, and thus, has been omitted from this document. However, to adopt the method for our problem, the definition of *rank* has been slightly modified.

1.3 Contributions

Our interest in real-scenario applications will be communicated by, first, describing any realistic 3D model of an industrial heavy machine in a discrete volume (voxel grid) and extracting data points from the 3D model. We use binary decision variables to model possible camera locations and *background points* that need to be covered, and, formulate the problem using BIP based model. Two approaches are proposed; 1) single objective to maximize coverage, and; 2) bi-objective to minimize the number of required cameras and maximize coverage in a lexicographic fashion, [42]. The objectives

are solved using exact deterministic optimization algorithms, while, the single objective problem is also solved using a greedy algorithm. A simulation tool to collect data and to visualize the results has also been developed.

Our contribution differs from the approach presented in [21] *w.r.t.* the dimensions of the modelled area. It differs from [46] in the way the problem is modelled and formulated, and also, in the targeted application area. Lastly, it differs from [24] in the way the bi-objective problem formulation is approached.

1.4 Paper Organization

The rest of the document will guide the reader, firstly, through the formulation of the problem in section 2. Section 3 describes the three proposed methods in detail, while, experiments, evaluation and discussion are presented in section 4. The document concludes in section 5.

2 PROBLEM FORMULATION

The *problem set* or *space* comprises of a voxel grid. The distance between two adjacent points comprises an edge of each voxel and is of unit length. For the sake of simplicity, the corners of all the voxels are considered as 3D points for modelling the problem without any repetitive entries. These 3D points are then represented as binary decision variables to formulate the OCP problem. The following sub-sections will explain in detail, how *problem space* and the *camera's Field-Of-View* are defined.

2.1 Problem Statement

The OCP problem can be formulated in many ways, for *e.g.*, maximizing coverage, minimizing multi-camera system cost, minimizing number of cameras of a given type to achieve sufficient coverage, *etc.*. The following two problems are studied:

- **Problem 1:** Given the number of cameras, optimize their position and pose such that maximum coverage is achieved.
- **Problem 2:** Minimize the number of required cameras such that minimal defined coverage is achieved and optimize the positions and pose of these number of cameras so that coverage is maximized.

2.2 Modelling Camera's Field Of View

The camera's Field Of View (FOV) is defined in a simple way using a pyramidal structure. The camera position in 3D, $p(x, y, z)$, defines the origin or the top of the pyramid. A unit vector passing from camera's origin through the center of the pyramid's base, $\hat{\theta}(x, y, z)$, defines the orientation or *camera's look-at* direction. Additionally, horizontal and vertical FOV angles, α_h, α_v, define the length and breadth of the pyramidal FOV. Lastly, the height of the pyramid, z_f defines the far depth limit of the FOV for achieving minimal resolution. Together, these parameters form the five planes that describe the pyramid (Figure 2). Readers can refer to [29] for details on camera coverage modelling.

Unlike in surveillance scenarios, modelling a camera for the case of surround vision of vehicles requires 6DOF. This is because, in surveillance scenarios, the cameras are usually placed on walls limiting the primary orientation to only two axes (for *e.g.*, if we consider the height of the wall along y-axis, then, one can be sure that the camera will be facing along the $x - z$ plane). However, in the case of a vehicle the primary orientation could be along the $x - z$, or, $y - z$, or, $x - y$ planes. Hence, in such situations an additional vector, $\hat{\phi}(x, y, z)$, describing the *camera's up vector*, is required to prevent rotating the camera about a third axis, *i.e.*, *camera's look-at* direction.

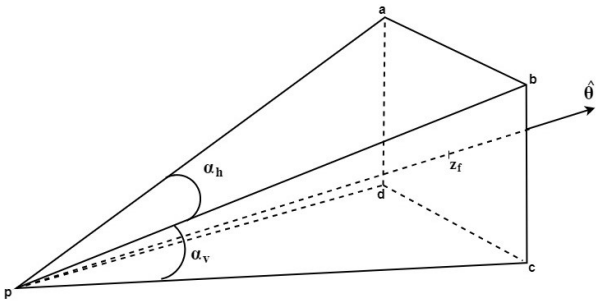


Figure 2: Camera's Field Of View.

Visibility analysis of the *background points* (*i.e.* to check whether a *background point* lies within the camera's FOV; details in section 2.4.4) is done by checking the point against the five planes that describe the camera. The 5 planes can be described with five points on the pyramid (camera origin and the four corners of the base of the FOV pyramid). As we already know the camera origin, the remaining four points can be calculated using linear geometry.

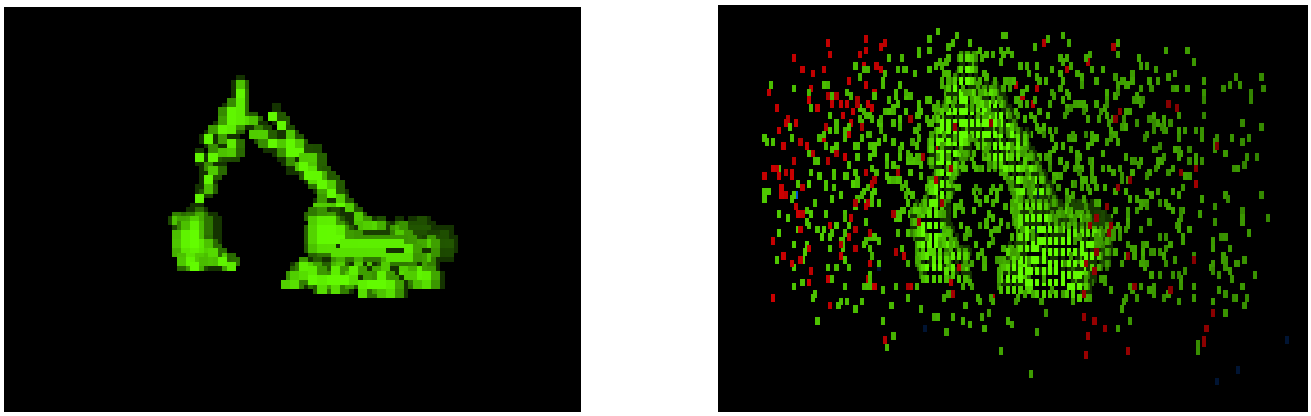
2.3 Modelling Space

The problem is modelled using a 3D grid of points. 3D models are, usually, represented as triangular meshes, [30], and saved in convenient formats in *wavefront object* files. Modelling space in a voxel grid has it's advantages over polygonal representation, [13]. Voxel representation enables a straightforward use of standard image processing algorithms. Firstly, a 3D model, openly available online in *wavefront object* files can be displayed in a voxel grid representation using any existing method that can convert a polygonal mesh into a voxel grid representation. For this purpose, we use an openly available tool developed by Patrick, [32, 36]. This object is then placed at the center of the voxel environment (*problem space*) (see Figure 3(a)).

Because, the aim of this work is to place cameras on the vehicle to obtain a 360° view, we collect all the voxels that represent object boundary and model them as binary decision variables. The *boundary voxels* represent a set $J \in \mathbb{R}^3$, comprised of 3D points, $j(x, y, z)$, where, $x, y, z \in \mathbb{Z}$. Each 3D point j further contains a set $\Phi \in \mathbb{R}^3$, representing possible orientations, $\phi(\hat{x}, \hat{y}, \hat{z})$, where, $\hat{x}, \hat{y}, \hat{z} \in \mathbb{R}$, the camera at location j can assume. The binary variables representing possible camera positions and poses can be defined as,

$$x_{j,\phi} = \begin{cases} 1 & \text{if a camera is placed at location} \\ & j \text{ with orientation } \phi \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Next step is to define a set $B \in \mathbb{R}^3$ consisting of 3D points $i(x, y, z)$, where, $x, y, z \in \mathbb{Z}$, to represent a set of points, surrounding the



(a) (b)

Figure 3: An example from the visualization tool developed by us describing the collection of data points. (a) Heavy Machine placed at the center of an empty voxel grid. (b) Background points representing a hollow spherical frustum around the dilated 3D model

vehicle, that are to be covered by the multi-camera system. These *background points* can be modelled using binary variables as,

$$b_i = \begin{cases} 1 & \text{if } \textit{background point} \text{ i is covered} \\ & \text{by at least one camera} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Having defined these binary variables we can now look into some data pre-processing requirements.

2.4 Data Pre-Processing

There are certain pre-processing steps that need to be taken to collect sets of points[42], $x_{j,\phi}$ and c_i, in a meaningful way. This sub-section will briefly explain some methods to collect and represent data before it can be passed to the optimization environment.

2.4.1 *Boundary Voxels.*

As, each voxel corner is represented either by '1' or '0' stating whether the point is occupied or not, simple binary-morphological operations can be applied to the volume by considering the 8-neighbourhood of each point.

We perform a pre-processing step of morphological dilation on the object to collect boundary voxels. Although, there exist efficient data structures to store voxel data that enable easy access to object boundaries, [31], implementing such data structures is out of the scope of this work. The idea behind morphological dilation is simple. Let us assume a set, $O_1 \in \mathbb{Z}$, containing all the voxels that define the object. The dilation operation will grow the object boundary by certain units of length depending on the size of the mask used (we use a $3 \times 3 \times 3$ grid as a mask for each of x, y, z dimensions), resulting in a new super-set of $O_1, O_2 \in \mathbb{Z}$. Now, a simple subtraction operation, $O_2 - O_1$, will represent the set, J, of *boundary voxels*.

2.4.2 *Camera orientations.*

The primary orientation, ϕ of the camera at each position j is represented by the normal, $n(\hat{x}, \hat{y}, \hat{z})$ of the uncovered face of the voxel at j. Since, each *boundary voxel* is, ideally, surrounded on five faces by adjacent voxels, the normal of the sixth face, facing outwards the object, is used to represent the primary orientation of the camera. The normals of all boundary voxels can be calculated using the marching cubes algorithm, [27, 33].

$$\begin{aligned} Rot_x &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \\ Rot_y &= \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \\ Rot_z &= \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{aligned} \tag{3}$$

Camera modelling for the case of surround view has an added Degree Of Freedom. This implies that the *camera's look-at* direction is not limited to two planes ($x - z$ or $y - z$, for *e.g.*). As we do not want to rotate the camera about its primary axis (*camera's look-at* direction), selection of two planes between the three available planes for rotating the camera, differentiates the problem formulation from the case of surveillance scenarios.

We select the axis with highest magnitude in the unit normal vector, n, as the primary axis of the orientation ϕ. The camera is then rotated about the other two axes at discrete steps of angles of rotation such that, the camera's FOV can cover 180° in both directions for all the orientations combined. As, the generated *boundary voxel* normals lie in the global plane, rotations about an axis are relatively simple to perform. Equation 3 describes the rotation matrices for rotations about the x, y, z axes. The unit normal vector n at each camera position j is multiplied by any two of these three matrices to obtain the possible orientations set, Φ, for that location j.

2.4.3 *Background Points.*

Following the standards defined by Ray *et al.*, [40], we collect empty voxels lying within a spherical cap of $12m$ radius from the center of the object. However, since we do not consider any points lying

below the ground plane, and, points lying above the height of the object, the spatial arrangement of *backgroundpoints* represents a hollow spherical segment or frustum (Figure 3(b)).

2.4.4 Visibility analysis.

A database of *background points* that are visible by a camera placed at position j with an orientation ϕ is required for formulation of optimization problem. It is constructed as a visibility matrix for the combined set J, Φ, B, with each entry in the matrix representing a binary variable, $g_{j,\phi,i}$. It is defined as follows,

$$g_{j,\phi,i} = \begin{cases} 1 & \text{if } \textit{background point } i \text{ is} \\ & \text{covered by a camera placed at} \\ & \text{position } j \text{ with orientation } \phi \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Constructing visibility matrix is computationally expensive, as, it involves a large number of geometrical calculations. It requires comparing the distance of each *background point* with each of the five planes that form the camera's FOV for all the possible camera pose combinations. This step is implemented on the GPU to achieve significant improvement in calculation time. However, details of this OpenCL, [34], implementation are omitted from this document due to space constraints.

3 METHODOLOGY

We propose different methods to solve the OCP problem for surround vision. The selection of objective function and optimization is, primarily, based on the BIP model proposed in [21]. We adopt their objective function and constraints, while, applying them on 3D space instead of a 2D plane. The problem is explained along with the optimization algorithms, for the sake of completeness.

We explore optimization of a single objective function, as well as, a bi-objective problem formulation. Optimization of single objective problem is studied using both, an exact algorithm, and, a search heuristic. While, the bi-objective problem is optimized using an exact algorithm. Although, Rebai *et al.*, [41], proposed multiple bi-objective optimization problems for surveillance scenarios, our problem is solved in a lexicographic fashion.

3.1 Single Objective Approach

First, we need to recall the binary decision variables defined in equations 1, 2 and 4. As our objective is to maximize coverage for a given number of cameras, the objective can be defined as maximization of sum of all *background points*,

$$max \sum_{i} b_i \tag{5}$$

Points in 3D space are represented using binary decision variables and enforced constraints largely remain the same as proposed in [21]. However, we noticed during experimental analysis that equations 6,7 and 8 enforce stricter constraints than the ones proposed in [21]. They result in reducing the overall computation time for solving the model. The rest of the constraints are omitted from this document to avoid repetition.

$$v_{\phi,j,i} \leq x_{j,\phi} \tag{6}$$

$$v_{\phi,j,i} \leq b_i \tag{7}$$

$$b_i + x_{j,\phi} \leq v_{\phi,j,i} + 1 \tag{8}$$

As, the maximization procedure has no bounds on the number of cameras to be placed, it might end up placing cameras at all the possible locations to cover all the available *background points*. Thus, this approach of using a single objective function to maximize surrounding area coverage requires a pre-defined number of cameras that are to be placed on the vehicle that is provided by the user.

3.2 Single Objective Heuristic Approach

The greedy algorithm was implemented in this context, in the same way it is described in [21]. However, we enforce different stopping criteria. Since, the objective is to maximize coverage given a number of cameras, the algorithm will be stopped when all N cameras are placed. The greedy algorithm is a constructive heuristic. But, as it is a deterministic algorithm, it has been included in this work.

The visibility matrix is required also for greedy algorithm. Once we have the visibility matrix, we need to compute the rank for each possible camera location and pose combination. The rank of a camera placed at position, j, with an orientation, ϕ, is nothing but the number of *background points* covered by it. So, if we define a set c_{cov} consisting of all the *background points* that are covered by a camera, $x_{j,\phi}$, then the rank, r, can be given as,

$$r_{j,\phi} = \sum c_{cov} \tag{9}$$

The greedy algorithm, at each iteration, selects the camera position-pose combination, $x_{j,\phi}$, with the highest rank. This procedure is repeated sequentially for each camera until the required number of, N, cameras are placed.

3.3 Bi-objective Approach

A bi-objective optimization approach is proposed in this section. The idea behind the bi-objective formulation is that, user interference to specify a number of required cameras is not required. The problem will consist of the following two objective functions,

$$\begin{aligned} min \sum_{\phi,j} x_{\phi,j} \\ max \sum_{i} b_i \end{aligned} \tag{10}$$

The first objective is to minimize the sum of all the binary variables representing possible camera poses. While, the second objective is to maximize coverage as in the case of single-objective formulation. Constraints used for the single objective formulation are re-used for the bi-objective problem formulation together with new constraints to be able to solve both objective functions simultaneously. The following equations will be discussed under this section for the sake of completeness in explaining this novel approach.

Visibility of a *background point* by a camera placed at each possible camera position for each given camera orientation can be verified using the binary variable $g_{\phi,j,i}$ in the form of the following constraint,

$$b_i \cdot \left(\sum_{j,\phi} x_{j,\phi} \cdot g_{\phi,j,i} - 1 \right) \geq 0 \tag{11}$$

Constraint in equation 12 ensures that at most one camera orientation is selected for a selected camera position j.

$$\sum_{\phi} x_{j,\phi} \leq 1 \tag{12}$$

The minimization procedure in the first objective has no lower bound, hence, if additional constraints are not enforced, it might end up placing no camera at all. Therefore, a new constraint has to be added to set a lower bound on the required number of cameras to be placed. We can define a constraint based on the minimal required percentage of coverage as follows,

$$\sum_{i} b_i \geq p \tag{13}$$

Where, p is the minimal required points that need to be covered, and, it can be expressed as a fraction of the total number of *background points* that need to be covered (a required coverage of 95% would translate as $p = 0.95 \cdot I$, where, I, is the total number of background points). Objective functions in 10 together with constraints 11, 12, 13, 6, 7 and 8 complete the bi-objective formulation for the general case of camera pose optimization for heavy machinery surround view.

The two objective functions are solved in a lexicographic fashion, [42]. That means that the two objective functions are solved sequentially based on a priority assigned to each of the objective functions. We approach the problem by first optimizing the number of required cameras, followed by, optimizing the coverage. the two objectives are solved in the same environment which means that all the constraints apply for both the problems, and, the solution from the minimizing problem can be used as a starting point for maximization problem. One advantage of approaching the problem in a lexicographic fashion is that no additional parameter is required to have a trade-off between the two objective functions. This method guarantees that both the objectives are optimally solved.

4 EXPERIMENTAL ANALYSIS

Experiments were conducted on a simulation tool that was developed using Qt creator and c++ programming language. The exact algorithms were run on a cluster of CPUs together amounting to 24 cores and 125GB of RAM.

To run tests on realistic scenarios, a 3D model of the desired industrial vehicle is fit into a $32 \times 32 \times 32$ voxel grid. The dimensions of a real vehicle of that type are then mapped to voxel grid by comparing with the largest of three dimensions occupied in the cubic voxel grid. This procedure roughly describes the length of one voxel in metric units. The camera parameters are also defined in number of voxels based on this mapping. This voxelized object is then placed at the center of a $100 \times 35 \times 100$ empty voxel grid with only the object representing occupied voxels.

The camera parameters α_v, α_h and z_f are considered equal to $77^0, 77^0$ and 40 voxels, respectively, for all cases. Camera at each possible position is rotated horizontally and vertically at steps of $\pm 25.75^0$ such that these 9 orientations together cover a hemisphere, facing outwards. One of the major drawbacks of not using search heuristics lies in memory requirements. To be able to solve the problem using reasonable amount of RAM, about 1% of the collected *background points* and 5% of existing *boundary voxels* are randomly selected during optimization. No other conditions are placed on environment modelling to maintain generality of the problem.

The three proposed methods were tested on 3D models of multiple heavy machinery that are primarily used at construction sites. The algorithms were tested under criteria guaranteeing optimal solution, as well as, with a trade-off between optimality and feasibility. To ensure a feasible solution within reasonable computational times, a gap of 5% from the best bound integer was allowed in the solutions, for some tests. The methods were tested on simulated models of heavy machinery, namely, bulldozer, excavator, lift truck, mining truck and wheel tractor scraper. These particular machinery were selected based on their size and shape complexity.

4.1 Results & Discussion

Random selection of *Background points* and *boundary voxels* results in different number of variables for different tests for the same model. This results in a different value of the objective for each optimization. Hence, to maintain uniformity, the coverage value is expressed in terms of percentages of the total area covered rather than the exact objective value. However, some tests were performed with a pseudo-random selection of the decision variables resulting in the selection of the same set of *background points* and *boundary voxels* for subsequent tests, so that, the three methods can be compared against each other for a given 3D model of the heavy machine.

Table 1 shows the percentage of coverage achieved by the single objective optimization for the considered simulations of heavy machinery. The second column mentions computation time in seconds amounting for the complete process of root node relaxation and the *branch and cut* process. The single objective problems were optimized for all the considered models for a fixed number of cameras. The required number of cameras was set to five for all cases for both BIP based approach, as well as, for the greedy approach.

The percentage of coverage matches with that in Table 3 only for the case of bulldozer and tractor scraper models (Table 1). Although, the methods are deterministic, the minor mismatch in resulting coverage between the two cases of single and bi-objective formulations can be attributed to the random selection of *boundary voxels*.

Table 1: Table showing surrounding area coverage and associated computation time for different heavy machinery for the case of single objective optimization

Heavy Machinery	**C**overage (%)	**T**ime (s)
bulldozer	95.60	4635
excavator	91.78	1654
lift crane	91.69	5009
mining truck	91.73	7334
tractor scraper	95.62	2874

There is significant increase in computation time for lift crane model for the case of single objective problem (Tables 1 and 3). Theoretically, bi-objective problem should require more computation time as it optimizes two objective functions simultaneously. Since, the optimal number of cameras for achieving at least 95% coverage for the case of lift crane is *six*, it took longer to achieve maximum coverage when optimized with only *five* cameras.

Table 2: Table showing surrounding area coverage and associated computation time for different heavy machinery for the case of single objective optimization using greedy heuristic

Heavy Machinery	**C**overage (%)	**T**ime (s)
bulldozer	91.43	~3
excavator	85.70	~2
lift crane	84.03	~2
mining truck	83.34	~3
tractor scraper	84.65	~3

Table 3: Table showing number of cameras placed, surrounding area coverage and associated computation time for different heavy machinery for the case of bi-objective optimization

Heavy Machinery	**no. of cameras**	**Coverage (%)**	**Time (s)**
bulldozer	5	96.78	15,035
excavator	6	95.52	12,331
lift crane	6	96.49	1,209
mining truck	-	-	-
tractor scraper	5	94.94	2,314

Table 2 shows the percentage of coverage achieved by the single objective optimization using greedy heuristic for 3D models of different heavy machinery. Greedy method could provide a solution in insignificant time when compared to the other methods. Greedy method has two major time consuming steps; 1) calculation of visibility matrix at each iteration; 2) sorting of the rank matrix. Speed up achieved by the Greedy method can be attributed to GPU acceleration implemented for calculating the visibility matrix. Thus, resulting computational complexity depends on sorting the rank matrix, which is trivial for the considered number of variables. However, Greedy algorithm places the cameras one at a time by picking the best available location and orientation combination at each iteration without looking at the combinations of given number of cameras. Hence, Greedy method provides sub-optimal solutions when compared with the exact methods.

Table 3 shows the results for the bi-objective model. These results represent feasible solutions that have a gap of about 6% from the best possible integer value. Lift crane and tractor scraper have comparatively smaller sizes, resulting in fewer *boundary voxels*. Similarly, mining truck has much larger surface area with a complex body structure resulting in a very high number of *boundary voxels*. The bi-objective problem for the case of mining truck could not be solved as the computer ran out of memory due to high number of *boundary voxels*. Therefore, it can be stated that BIP based optimization is affected by the number of considered variables.

Table 4: Table showing number of cameras placed, surrounding area coverage and associated computation time for bulldozer model with emphasis on optimality

Problem	**no. of cameras**	**Coverage (%)**	**Time (s)**
Single Objective	5	97.61	5,554
Bi-objective	5	97.61	70,957

The single objective BIP based problem and the bi-objective problem were solved with an emphasis on optimality for the bulldozer model. No gap was allowed for these tests, thus, resulting in an optimal solution for the given environment. The results are shown in Table 4. It is evident from the results that emphasis on optimality introduces large overhead on required computational time. Hence, it can be argued that significant trade-off between solution quality and computational time can be achieved by stopping the optimization process as soon as a feasible solution within the permissible gap is obtained.

Multiple tests were executed to check if random selection has any effect on the quality of the resulting solution. Five tests with different subsets of randomly selected variables were executed for the bi-objective problem on the bulldozer and excavator models. Table 5 shows these results with the last column mentioning the best coverage value obtained in the five tests. It was observed in these tests that there is not any significant standard deviation in the coverage values.

Table 5: Table showing solution values averaged over 5 tests for different heavy machinery using bi-objective optimization problem

Problem	**Coverage (%)**	**Time (s)**	**Best (%)**
bulldozer	96.18	12441	96.78
excavator	95.87	15139	96.15

5 CONCLUSION

Optimal Camera Placement has been, for the first time, implemented in the context of heavy machinery. Multiple cameras were successfully placed on the heavy machines to achieve surround view vision, while, minimizing the required number of cameras. Three novel methods were studied for optimally placing cameras on five different heavy industrial vehicle (heavy machinery) models. A complete solution including problem modelling and optimization stages, has been proposed.

The three proposed approaches were compared on the basis of surrounding area coverage quality and computational time, while, specifying trade-offs between optimality and feasibility. A visualization tool has also been developed to collect data and to visualize optimized solutions.

5.1 Future Work

The motivation for this work lies in the fact that large blind spots exist for such heavy machinery that can be a threat to human life in vicinity of the machinery. Some weight can be given to the *background points* lying in these blind spot regions so as to effectively cover those particular points to minimize the risk associated with it. Set-cover based optimization algorithms could also be tested for the case of surround view visualization for heavy machinery.

6 ACKNOWLEDGEMENTS

This work is supported by the ImmerSAFE (Project number 764951) project funded under the EU's H2020-MSCA-ITN-2017 call and is

part of the Marie Sklodowska-Curie Actions - Innovative Training Networks (ITN) funding scheme.

7 REFERENCES

[1] Jun-Woo Ahn, Tai-Woo Chang, Sung-Hee Lee, and Yong Won Seo. 2016. Two-phase algorithm for optimal camera placement. *Scientific Programming* 2016 (2016).

[2] Ian F Akyildiz, Tommaso Melodia, and Kaushik R Chowdhury. 2007. A survey on wireless multimedia sensor networks. *Computer networks* 51, 4 (2007), 921–960.

[3] Jon H Bechtel, Joseph S Stam, Eric R Fossum, and Sabrina E Kemeny. 2009. Vehicle vision system. US Patent 7,567,291.

[4] Eric Becker, Gutemberg Guerra-Filho, and Fillia Makedon. 2009. Automatic sensor placement in a 3D volume. In *Proceedings of the 2nd International Conference on PErvasive Technologies Related to Assistive Environments*. ACM, 36.

[5] Vasek Chvatal. 1975. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory, Series B* 18, 1 (1975), 39–41.

[6] Par Degerman, Joseph Ah-King, T Nystroem, Marc-Michael Meinecke, and Simon Steinmeyer. 2012. Targeting lane-change accidents for heavy vehicles. *VDI-Berichte* 2166 (2012).

[7] Uğur Murat Erdem and Stan Sclaroff. 2006. Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements. *Computer Vision and Image Understanding* 103, 3 (2006), 156–169.

[8] Jose Esparza, Michael Helmle, and Bernd Jähne. 2014. Towards surround stereo vision: Analysis of a new surround view camera configuration for driving assistance applications. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 1493–1495.

[9] Thomas Focke, Henning Von Zitzewitz, and Thomas Engelberg. 2014. Method and device for determining processed image data about a surround field of a vehicle. US Patent 8,712,103.

[10] Tarak Gandhi and Mohan M Trivedi. 2006. Vehicle surround capture: Survey of techniques and a novel omni-video-based approach for dynamic panoramic surround maps. (2006).

[11] Yi Gao, Chunyu Lin, Yao Zhao, Xin Wang, Shikui Wei, and Qi Huang. 2017. 3-D surround view for advanced driver assistance systems. *IEEE Transactions on Intelligent Transportation Systems* 19, 1 (2017), 320–328.

[12] Antonio-Javier Garcia-Sanchez, Felipe Garcia-Sanchez, and Joan Garcia-Haro. 2011. Wireless sensor network deployment for integrating video-surveillance and data-monitoring in precision agriculture over distributed crops. *Computers and Electronics in Agriculture* 75, 2 (2011), 288–303.

[13] Scott Gebhardt, Eliezer Payzer, Leo Salemann, A Fettinger, E Rotenberg, and C Seher. 2009. Polygons, point-clouds and voxels: A comparison of high-fidelity terrain representations. In *Simulation Interoperability Workshop and Special Workshop on Reuse of Environmental Data for Simulation—Processes, Standards, and Lessons Learned*.

[14] Markus Gressmann, Günther Palm, and Otto Löhlein. 2011. Surround view pedestrian detection using heterogeneous classifier cascades. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 1317–1324.

[15] Ankit Gupta, Kumar Ashis Pati, and Venkatesh K Subramanian. 2012. A NSGA-II based approach for camera placement problem in large scale surveillance application. In *2012 4th International Conference on Intelligent and Advanced Systems (ICIAS2012)*, Vol. 1. IEEE, 347–352.

[16] Kazukuni Hamada, Zhencheng Hu, Mengyang Fan, and Hui Chen. 2015. Surround view based parking lot detection and tracking. In *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1106–1111.

[17] Maung P Han and Dhruv Monga. 2016. Method and system for presenting panoramic surround view in vehicle. US Patent App. 14/585,682.

[18] Christian Häne, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, Paul Furgale, Torsten Sattler, and Marc Pollefeys. 2017. 3D visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection. *Image and Vision Computing* 68 (2017), 14–27.

[19] Adam Hedi and Sven Lončarić. 2012. A system for vehicle surround view. *IFAC Proceedings Volumes* 45, 22 (2012), 120–125.

[20] Lionel Heng, Gim Hee Lee, and Marc Pollefeys. 2015. Self-calibration and visual slam with a multi-camera system on a micro aerial vehicle. *Autonomous robots* 39, 3 (2015), 259–277.

[21] Eva Hörster and Rainer Lienhart. 2006. On the optimal placement of multiple visual sensors. In *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*. ACM, 111–120.

[22] Kohsia S Huang, Mohan M Trivedi, and Tarak Gandhi. 2003. Driver's view and vehicle surround estimation using omnidirectional video stream. In *IEEE IV2003 Intelligent Vehicles Symposium. Proceedings (Cat. No. 03TH8683)*. IEEE, 444–449.

[23] Klaus Huebner, Koba Natroshvili, Johannes Quast, and Kay-Ulrich Scholl. 2017. Vehicle surround view system. US Patent 9,679,359.

[24] Julien Kritter, Mathieu Brévilliers, Julien Lepagnot, and Lhassane Idoumghar. 2019. On the optimal placement of cameras for surveillance and the underlying set cover problem. *Applied Soft Computing* 74 (2019), 133–153.

[25] Julien Kritter, Mathieu Brévilliers, Julien Lepagnot, and Lhassane Idoumghar. 2019. On the real-world applicability of state-of-the-art algorithms for the optimal camera placement problem. In *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, 1103–1108.

[26] Junbin Liu, Sridha Sridharan, and Clinton Fookes. 2016. Recent advances in camera planning for large area surveillance: A comprehensive review. *ACM Computing Surveys (CSUR)* 49, 1 (2016), 6.

[27] William E Lorensen and Harvey E Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. In *ACM siggraph computer graphics*, Vol. 21. ACM, 163–169.

[28] Pranav Mantini and Shishir K Shah. 2016. Camera Placement Optimization Conditioned on Human Behavior and 3D Geometry.. In *VISIGRAPP (3: VISAPP)*. 227–237.

[29] Aaron Mavrinac and Xiang Chen. 2013. Modeling coverage in camera networks: A survey. *International journal of computer vision* 101, 1 (2013), 205–226.

[30] Kenton McHenry and Peter Bajcsy. 2008. An overview of 3d data content, file formats and viewers. *National Center for Supercomputing Applications* 1205 (2008), 22.

[31] Donald Meagher. 1982. Geometric modeling using octree encoding. *Computer graphics and image processing* 19, 2 (1982), 129–147.

[32] Patrick Min. 2004 - 2019. binvox. `http://www.patrickmin.com/binvox` or `https://www.google.com/search?q=binvox`. Accessed: yyyy-mm-dd.

[33] Claudio Montani, Riccardo Scateni, and Roberto Scopigno. 1994. Discretized marching cubes. In *Proceedings of the conference on Visualization'94*. IEEE Computer Society Press, 281–287.

[34] Aaftab Munshi. 2009. The opencl specification. In *2009 IEEE Hot Chips 21 Symposium (HCS)*. IEEE, 1–314.

[35] Koba Natroshvili and Bernd Gassmann. 2012. Surround view system camera automatic calibration. US Patent App. 13/466,743.

[36] Fakir S. Nooruddin and Greg Turk. 2003. Simplification and Repair of Polygonal Models Using Volumetric Techniques. *IEEE Transactions on Visualization and Computer Graphics* 9, 2 (2003), 191–205.

[37] Petr Pokorny, Jerome Drescher, Kelly Pitera, and Thomas Jonsson. 2017. Accidents between freight vehicles and bicycles, with a focus on urban areas. *Transportation research procedia* 25 (2017), 999–1007.

[38] Pooya Rahimian and Joseph K Kearney. 2016. Optimal camera placement for motion capture systems. *IEEE transactions on visualization and computer graphics* 23, 3 (2016), 1209–1221.

[39] Akshay Rangesh, Kevan Yuen, Ravi Kumar Satzoda, Rakesh Nattoji Rajaram, Pujitha Gunaratne, and Mohan M Trivedi. 2017. A multimodal, full-surround vehicular testbed for naturalistic studies and benchmarking: Design, calibration and deployment. *arXiv preprint arXiv:1709.07502* (2017).

[40] Soumitry J Ray and Jochen Teizer. 2013. Computing 3D blind spots of construction equipment: Implementation and evaluation of an automated measurement and visualization method utilizing range point cloud data. *Automation in Construction* 36 (2013), 95–107.

[41] Maher Rebai, Matthieu le Berre, Faicel Hnaien, and Hichem Snoussi. 2016. Exact biobjective optimization methods for camera coverage problem in three-dimensional areas. *IEEE Sensors Journal* 16, 9 (2016), 3323–3331.

[42] Mark J Rentmeesters, Wei K Tsai, and Kwei-Jay Lin. 1996. A theory of lexicographic multi-criteria optimization. In *Proceedings of ICECCS'96: 2nd IEEE International Conference on Engineering of Complex Computer Systems (held jointly with 6th CSESAW and 4th IEEE RTAW)*. IEEE, 76–79.

[43] Avital Avigad Steinitz. 2012. *Optimal camera placement*. Ph.D. Dissertation. Citeseer.

[44] Mengmeng Yu and Guanglin Ma. 2014. 360 surround view system with parking guidance. *SAE International Journal of Commercial Vehicles* 7, 2014-01-0157 (2014), 19–24.

[45] Buyue Zhang, Vikram Appia, Ibrahim Pekkucuksen, Yucheng Liu, Aziz Umit Batur, Pavan Shastry, Stanley Liu, Shiju Sivasankaran, and Kedar Chitnis. 2014. A surround view camera solution for embedded systems. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 662–667.

[46] Hongguang Zhang, Lingnan Xia, Fei Tian, Peng Wang, Jianzhu Cui, Chao Tang, Nana Deng, and Na Ma. 2013. An optimized placement algorithm for collaborative information processing at a wireless camera network. In *2013 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 1–6.

[47] Xuebo Zhang, Xiang Chen, Jose Luis Alarcon-Herrera, and Yongchun Fang. 2015. 3-D model-based multi-camera deployment: A recursive convex optimization approach. *IEEE/ASME Transactions on Mechatronics* 20, 6 (2015), 3157–3169.

[48] Jian Zhao, Ruriko Yoshida, Sen-ching Samson Cheung, and David Haws. 2013. Approximate techniques in solving optimal camera placement problems. *International Journal of Distributed Sensor Networks* 9, 11 (2013), 241913.

Publication 2

V.A. Puligandla, S. Lončarić, ”A multiresolution approach for large real-world camera placement optimization problems”, *IEEE Access*, Vol. 10, 2022, pp. 61601-61616.

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 8, 2022, accepted May 12, 2022, date of publication May 23, 2022, date of current version June 15, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3176817

A Multiresolution Approach for Large Real-World Camera Placement Optimization Problems

V. ANIRUDH PULIGANDLA, (Member, IEEE), AND SVEN LONČARIĆ, (Senior Member, IEEE)
Image Processing Group, Department of Electronic Systems and Information Processing, Faculty of Electrical Engineering and Computing, University of Zagreb, 10000 Zagreb, Croatia

Corresponding author: V. Anirudh Puligandla (apuligandla@fer.hr)

This work was supported by the ImmerSAFE Project under the European Union's (EU's) H2020-MSCA-ITN-2017 Call and is part of the Marie Sklodowska-Curie Actions—Innovative Training Networks (ITN) Funding Scheme under Project 764951.

ABSTRACT There have been numerous attempts at solving the optimal camera placement problem across multiple applications. Exact linear programming-based, as well as, heuristic combinatorial optimization methods were shown to be successful in providing optimal or near-optimal solutions to this problem. Working over a discrete space model is the general practice when solving the camera placement problem. However, discretized environments often limit the methods' usage only to small-scale datasets due to resource and time constraints that grow exponentially with the number of 3D points collected from the discrete space. We propose a multi-resolution approach that enables the usage of existing optimization algorithms on large real-world problems modelled using high resolution 3D grids. Our method works by grouping together the given discrete set of possible camera locations into clusters of points, multiple times, resulting in multiple resolution levels. Camera placement optimization is repeated for all resolution levels while propagating the optimized solution from low to high resolutions. Our experiments on both simulated and real data with grids of varying sizes show that using our multi-resolution approach, existing camera placement optimization methods can be used even on high resolution grids consisting of hundreds of thousands of points. Our results also show that the strategy of grouping points together by exploiting underlying 3D geometry to optimize camera poses is not only significantly faster than optimizing on the entire set of samples but, it also provides better camera coverage.

INDEX TERMS Image decomposition, integer linear programming, heuristic algorithms, pareto optimization.

I. INTRODUCTION

Advanced Driver Assistance Systems (ADAS) that provide surrounding view or top view from vehicles using multiple in-vehicle cameras have recently found lot of interest and applications in the vehicle industry, [1], [2]. Multiple camera systems for vehicle surround-view generally use four wide-angle or fish-eye lens cameras placed on the vehicle to generate new perspectives from the obtained images. For small vehicles, like cars, it is intuitive to place four cameras on four sides of the vehicle. Assembling a multi-camera surround-view system for larger vehicles, such as, construction equipment, is complicated due to large sizes and irregular shapes of these heavy machines. Calibration of multiple cameras is a complex task. It is known that vehicle surround view systems rely on camera calibration, [3], to estimate the cameras' poses with high accuracy. An optimal camera placement setup proposed using, high-resolution, realistic 3D vehicle models can aid in camera calibration by providing an initial accurate estimate of cameras poses in real-world.

Camera placement optimization or optimal camera placement (OCP) is a decades old problem. OCP problems have been applied to a wide range of applications such as, video surveillance, [4], 3D reconstruction of objects, [5], human behaviour monitoring and motion capture systems, [6], [7], OCP with VR interface, [8], and so on. Problem formulation frameworks range from simple single objective, minimization or maximization problems, [9], to more complex, multi-objective or non-linear problems, [10]. Despite the variety, one thing common to all of them is that the modelled space is discretized. It is an established practice in OCP

The associate editor coordinating the review of this manuscript and approving it for publication was Diego Oliva.

problems to discretize the space and use combinatorial optimization algorithms to find the optimal locations for placing cameras. For example, for surveillance applications, points are sampled randomly or at equal intervals over a floor plan. While it is true that continuous space model would provide better accuracy, continuous models are not used due to their complexity (e.g., estimating camera coverage requires calculating intersection between polygons or volumes). Using sets of points for optimization is not an efficient process when compared with continuous space model, but it has shown to be a successful approach for all the existing OCP problems across various applications. A large number of sampled points are required for an acceptable approximation of the continuous space. However, a greater number of input points imply a larger solution search space for the optimization algorithms resulting in an increase in the time required for finding the solution to the OCP problem.

This work focuses on the use-case scenario of finding optimal locations and directions of multiple cameras that need to be placed on a vehicle to achieve surround-view vision. Vehicle surround-view camera systems produce surround-view outputs by combining the video streams of individual cameras. Slight variation in a camera's orientation may result in serious artefacts in the surround-view output. Therefore, it is necessary to ensure that the OCP problem is applied on high-resolution, realistic 3D models of vehicles so that the resulting optimized camera poses precisely adhere to the real vehicle's structure and surface boundaries. To show applicability in real-world scenarios, we use high-detailed realistic 3D models of vehicles in a space modelled using high-resolution 3D grids where each voxel represents few millimeters on the real vehicle's surface. While OCP problems used in this work consist of tens of thousands of decision variables, the widely used branch-and-bound optimization algorithm (a method that guarantees the global optimal solution), [11], cannot handle more than a few hundred variables due to limitations on the amount of required resources, rendering it impractical for most real-world OCP problems.

To overcome the limitations on the amount of resources and time required by the optimization process, we propose a new multi-resolution (MR) approach that works on only a small subset of the input points at one time, thus, reducing the size of the solution search space for any given optimization algorithm. Our method is an iterative process that works, at each iteration, by grouping the input points into a given number of clusters and optimizing camera poses on the clusters of points. At the end of an iteration, only the selected clusters of points (i.e., solution obtained from optimization) are propagated as input points for optimization in the next iteration, while the rest of the points are discarded. Our proposed multi-resolution representation can be compared to the image representations used in different ''multi-scale'' image processing methods. We choose the term ''resolution'' because, when a number, N, of camera positions are grouped into a number, K, of clusters, where $K < N$, the cluster center points, computed as the mean of all the points belonging to the cluster, are represented as the new set of camera positions for an iteration. This implies that the vehicle's surface is represented by K number of points instead of N resulting in a ''low-resolution'' description of the vehicle's 3D surface. But, as the algorithm progresses through iterations, fewer number of points get grouped together into clusters, resulting in a ''high-resolution'' representation of the vehicle's surface.

Primary advantage of our proposed MR method comes from low-resolution descriptions as they reduce the size of the search space for any optimization method irrespective of the choice of the algorithm. This quality enables us to use branch and bound-optimization methods on large real-world models without encountering the resource constraints that are integral to those methods. While it may be assumed that working with only a subset of the points may result in lower coverage accuracy, our results show that, in fact, our approach improves the coverage accuracy that is obtained when camera poses are optimized on the complete set of sampled points. We believe this improvement arises due to two factors; 1) by clustering the sampled points based on their 3D position and surface orientation, we can effectively capture the geometrical features of the vehicle's surface, thereby identifying important regions on the vehicle for camera placement; 2) by representing the clusters of points by the mean position, we can achieve sub-voxel accuracy on the vehicle's model, thereby adding some degree of continuity to an otherwise discrete optimization problem.

Camera placement optimization on the entire set of samples without clustering will be called as single resolution (SR) optimization in the remainder of this document. The results from SR optimization on the same data are used to establish the efficacy of our proposed MR method. It is important to note that all discrete optimization problems used in different OCP applications require a ''look-up'' table describing the coverage of every point in the dataset. This look-up table, known as *visibility matrix*, is computed beforehand as a pre-processing step. In fact, this step is expensive in terms of time required as it involves millions of geometrical calculations. Existing literature in this field overlooks this problem as it considered to be a pre-processing step, where the visibility matrix can be computed once per dataset and stored in a file, for example. However, our experiments show that, for large real-world data, computing this matrix is impractical as it takes several hours when the number of sampled points range in tens of thousands. Our results show that by reducing the number of input camera points, the MR method significantly reduces the time required for the overall optimization process (including the time required for the pre-processing step of visibility calculations) while improving the camera coverage quality.

The rest of the document is organized as follows: Section II provides an overview of prior relevant research, Section III describes the optimal camera placement problem for vehicle surround-view, Section IV describes the multi-resolution

method and the results and conclusions are discussed in SectionsVandVI, respectively.

II. BACKGROUND WORK

The origins of optimal camera placement trace back to the art gallery problem, [12]. Work done by Hörster and Lienhart, [9], and by Erdem and Sclaroff, [13], can be considered pioneering in the field of optimal camera placement for indoor surveillance scenarios. Given an indoor setting with a floor plan, their work aims to optimally place a certain number of cameras in designated regions to cover as much of the floor as possible. They proposed multiple problem statements, such as, to minimize the cost of the multi-camera network, maximize coverage given a desired number of cameras, etc. They use binary decision variables to model possible camera locations and the points that are to be covered by the placed cameras (named *control points*). They use the branch-and-bound method which is classified as non-heuristic as it can provide provable bounds around the optimal solution, [11]. The proposed linear programming framework is aptly named binary integer programming (BIP), and has been proven, over the past decade, to be the most accurate model, although, expensive in terms of time and resource consumption.

Most BIP-based formulations are $\mathcal{NP}$-complete. Branch-and-bound algorithms, due to this reason, often require impractical amounts of time and resources to find exact optimal solutions for even small datasets. To circumvent this limitation, numerous approximate optimization methods have been proposed. Despite being prone to local optima, approximate methods find widespread interest in research and practical applications due to lower algorithmic complexity. Hörster and Lienhart, [9], presented an adaptive greedy algorithm that works by creating a rank-matrix for all the possible camera locations. The rank is calculated as the total number of control points covered by each camera pose. The algorithm iteratively picks a camera pose with the highest rank until the required number of cameras are placed. It is an adaptive greedy heuristic because once a camera is selected, that position and the covered control points are removed and only the remaining points are considered for the next camera. Such a feature, however, removes the combinatorial aspect of the OCP problem, but enables it to provide a feasible solution in significantly less time than branch-and-bound-based optimization. Due to their lower complexity, Greedy algorithms are often used for an upper bound on the solution or as a solution initialization step in a more complex algorithm, [14].

The authors in [15] proposed a search heuristic method called particle swarm optimization (PSO). The method works by initializing a number of particles spread randomly across the search space. The particles are compared in terms of the objective function value and the best one is picked as the solution to the OCP problem. In the category of genetic algorithms, Gupta *et al.*, proposed an algorithm that works in a similar fashion to the genes present in the human body. It starts with an initial solution and makes mutations and crossovers of the initialized solution while keeping a record of the objective function values of each candidate solution. The candidate with the best objective function value is selected as the final solution after a certain number of mutations. Evolutionary algorithms like PSO have been extensively studied in camera placement optimization problems. Wang *et al.* proposed two variants of PSO algorithm in [16] and [17], respectively. While the core algorithm remains same as the standard PSO, in RPSO, they use a weighted particle re-sampling scheme to maintain diversity in the particle population. Similarly, in LH-RPSO, [17], they replace the re-sampling strategy of RPSO with a new *Latin Hypercube*-based particle sampling strategy. Both their methods are claimed to improve solutions obtained by the standard PSO algorithm. Some other approximate optimization techniques, including probabilistic search space sampling techniques and evolutionary algorithms are detailed in [18]–[20].

All the approximate optimization methods try to uniformly search the solution space without having to go through all the combinations. Due to this reason, they are faster than exact BIP-based optimization. But, unlike BIP-based method, they fail to find the global optimal solution and often end up in local optima. In, [21], citing the difficulties in solving real-world OCP problems due to its $\mathcal{NP}$-hard characteristic, Ahn *et al.* proposed a *two-phase* algorithm as an approximate optimization algorithm. Their method is similar to the method we propose in this work, as both try to solve a simplified OCP problem at lower image resolutions first. However, their method works only in two resolutions and is proposed only for two-dimensional data. Additionally, they use different optimization algorithms for the two resolution levels. In contrast, we propose a general method that is applicable to 3D data of any kind and size. The only requirement for our method is that the discretized possible camera locations have an associated camera view-direction vector.

Despite the vast amount of literature in this field, their applicability to real world OCP problems is challenging. One of the algorithms presented by Kritter *et al.* [14] is a *row weighting local search* (RWLS) algorithm that was originally proposed by Gao *et al.*, [22], for the general *unicost set covering problem*. The RWLS optimization method was recently explored in the context of OCP in [23]. Kritter *et al.* mentioned in [24] that RWLS was not yet studied in OCP scenarios. They proved the method's superiority in [14] where they studied the OCP problem for large 3D models of European cities. However, citing the method's complexity, they mention that their models were sampled sparsely at large intervals to obtain a feasible solution in reasonable amount of time. In contrast, we test our method on models with tens of thousands of sampled points. Apart from these application-specific methods, some surveys give a comprehensive overview on various modelling and optimization strategies for the OCP problem, [24], [25].

There also exist some bi-objective methods that aim to maximize coverage while simultaneously minimizing the cost of the multi-camera system, [26]. Although, triangular or pyramidal camera field-of-view models are prevalent, there exist many other types of camera coverage models in usage, [27]. Finally, it is to be noted that the choice of coverage quality metric is also important to achieve reliable results, [28].

III. OCP FOR VEHICLE SURROUND-VIEW

The optimal camera placement problem includes few steps such as: discretizing the space, defining binary decision variables, defining an objective function (e.g., maximizing camera coverage), and finding the optimal value of the objective function while enforcing certain constraints on values taken by the variables. Our proposed formulation of the problem of OCP for vehicle surround-view is detailed in the following sub-sections. Our problem formulation is like the BIP-based framework proposed in [9]. Some modifications were made to the problem to allow greater degrees of freedom for camera poses and extension to three-dimensional space. Because the camera view directions have an additional degree of freedom, these modifications only increase the complexity of defining the variables, whereas, the objective function, constraints and the optimization procedure remain unchanged.

A. MODELLING SPACE

We model the space as a volume using a structured grid. Structured grids are a collection of points forming a symmetric 3D grid, [29]. These points usually represent the centroids of encompassing voxels. Such a representation is discrete in nature, hence, there is no further need to sample the space for discrete data. The space for the optimization problem is defined within the volume. To keep the simulations realistic, we collected 3D models of different heavy machines used in the construction industry, that are freely available on crowd-sourced internet platforms, such as, [30]. The polygonal models were voxelized into a structured grid using an open source tool called *binvox*, [31], [32].

The linear programming formulation for the optimal camera placement problem requires definition of two discrete sets of variables: (1) possible camera poses, and (b) points that are viewed by the placed cameras (commonly known as *control points*). In the problem of OCP for vehicle surround-view, the voxelized vehicle model is placed at the center of the volume, with the slice of voxels at $y = 0$ representing the ground plane. The control points are defined around the vehicle on a spherical cap surface, [33], with a radius of 12 meters from the center of the vehicle. This value of radius was chosen as it gives a level of visibility that is required for surround-view under safety critical conditions, [34]. Fig. 1 shows an example visualization of the modelled space with a voxelized bulldozer vehicle model placed at the center of the volume. The vehicle is shown in red and the control points are shown in blue.

For simplicity, we consider voxels on the entire surface of the vehicle as possible locations where cameras can be placed. To extract the vehicle's surface, first, a morphological dilation operation is performed on the voxels representing the vehicle to grow its boundaries by one unit. Then, the original volume is subtracted from the dilated volume, leaving behind a one-voxel thick boundary on the vehicle's surface. These boundary voxels are defined as possible locations for camera placement, $x_i \ \forall i \in N$, where N is the number of boundary voxels. Each position x_i is associated with a camera view direction vector, $\hat{x_i}$, computed using the marching cubes algorithm, [35], on the vehicle's surface. Each camera is rotated in the pitch and yaw directions about the primary view direction, at equal steps up to 90^o on either side.

If there are a total of Φ rotations for each camera, then the set of all possible camera poses can be written as,

$$x_{i\phi} = \begin{cases} 1 & \text{if a camera is placed at location} \\ & i \text{ with orientation } \phi \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where, $\phi = 1, \ldots, \Phi$. The spherical cap around the vehicle is defined by three parameters: radii of the top and bottom circular segments and the height of the spherical cap. As a spherical cap is defined in continuous space, all the voxels that lie within 0.5 units of the surface of the spherical cap are set as control points. The control points are given as,

$$c_j = \begin{cases} 1 & \text{if } \textit{control point} \text{ j is covered} \\ & \text{by at least one camera} \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

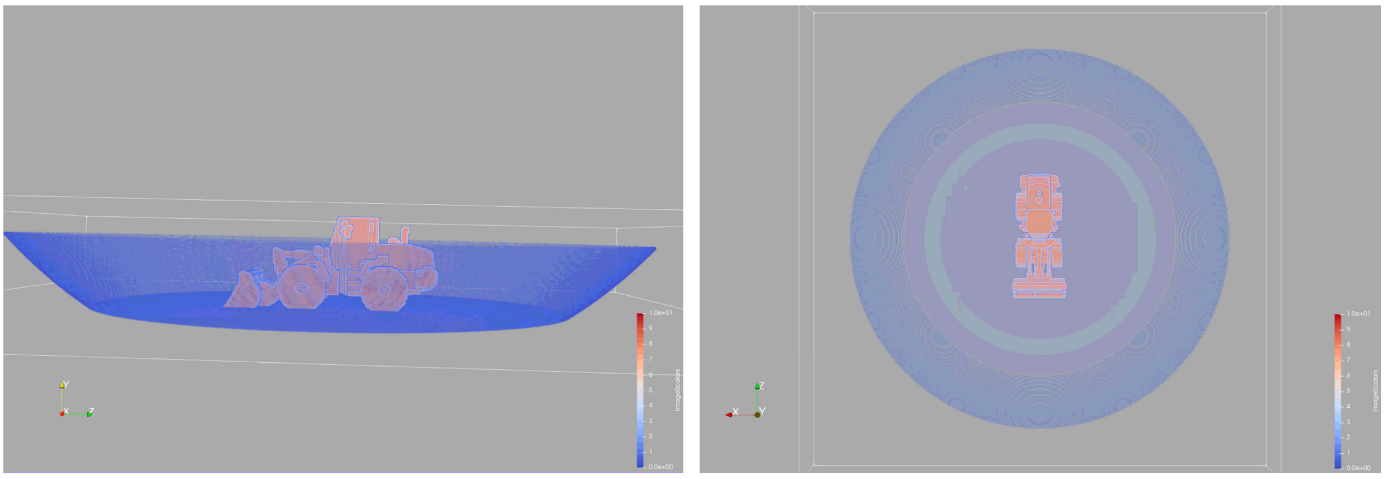
for all $j = 1 : J$, where J is the total number of voxels on the spherical cap surface. The objective function to maximize multi-camera coverage by optimally placing a pre-defined number of cameras, n, is given as,

$$max \sum_j c_j. \tag{3}$$

The poses for these n cameras are selected from the set of all possible camera poses $x_{i\phi}$, such that they maximize the sum of control points c_j that are covered by the n cameras.

B. MODELLING CAMERA's FoV

The camera's field-of-view (FoV) has been previously defined in numerous ways, in both two and three dimensions, for various optimal camera placement problems. Depending on the application, a camera's FoV can have multiple coverage criteria, such as, visibility/occlusion, resolution, etc., [27]. Moreover, some applications define multiple types of cameras (for e.g., wide-angle lens, fish-eye lens, etc.). For simplicity, we define only one type of camera for our optimization problem with only one criteria for visibility/occlusion. Defining and handling multiple types of cameras and criteria is a simple and scalable process, but



(a) Side View (b) Top View

FIGURE 1. Example visualization of modelled space with bulldozer vehicle model. The voxels in red represent the set of possible camera locations, whereas, voxels in blue show the control points.

with a downside of increased complexity of the optimization problem. For our problem, we adopt the commonly used pyramidal FoV model in three dimensions. The pyramidal FoV is described using three parameters, namely, depth of the view frustum, z_f, horizontal field of view angle, α_h and vertical field of view angle α_v. Additionally, the origin p and view direction vector $\hat{\theta}$ associate the FoV pyramid to possible camera poses $x_{i\phi}$.

C. VISIBILITY CHECKS

It is necessary to define an additional variable $g_{i\phi j}$ specifying if a control point c_j is covered by a camera pose $x_{i\phi}$ or not. It can be precomputed for all camera poses $x_{i\phi}$ using simple geometrical calculations and stored in a two-dimensional visibility matrix. The variable is described as,

$$g_{i\phi j} = \begin{cases} 1 & \text{if } \textit{control point } j \text{ is} \\ & \text{covered by a camera placed at} \\ & \text{position } i \text{ with orientation } \phi \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Visibility of a control point c_j by a camera at $x_{i\phi}$ is determined by using point in plane calculations. If a control point lies inside all the five planes of the camera's FoV pyramid, then the corresponding entry in the visibility matrix is marked as 1. This pre-processing step can be expensive when the set X consists of tens of thousands of points. For N camera positions, J control points and Φ orientations for each camera position, the time complexity of our visibility algorithm is $\mathcal{O}(N\Phi J)$. This complexity can be considered to be simpler when compared to other state-of-the-art methods as we do not consider any static or dynamic occlusions (except for self-occlusion) in our model. However, without parallel computations, the time required for these calculations may be impractical for real-world OCP scenarios. As the geometrical calculations are mutually independent, to speed up the process of creating the visibility matrix, we perform these calculations parallel on the GPU using OpenCL.

In our experiments, the largest model has over 14 million points in the initial set $x_{i\phi}$, and over 1500 sampled control points. As it is not possible to construct a matrix of this size, we introduce two additional conditions to filter out camera poses that do not make any significant contribution to the solution. The first condition is used to remove cameras whose FoV is occluded by the vehicle itself. A threshold value, cov_{self}, defines a tolerance value on the number of vehicle points that are allowed to fall within a camera's FoV. All camera poses with occlusion more than the threshold are not included in the final visibility matrix. Additionally, camera poses that do not cover any control points (e.g., cameras pointing directly up) do not contribute to the solution. Similarly, cameras covering too few control points also do not contribute to the solution of a coverage maximization problem with fixed pre-defined number of cameras. A pre-set parameter cov_{min}, defines a lower bound on the number of control points a camera needs to cover. All cameras that cover less than cov_{min} control points are not included in the final visibility matrix.

D. INTEGER PROGRAMMING MODEL

Having defined the binary variables and the objective function, we need to enforce certain constraints that describe coverage or place some restrictions on camera placement. First, a constraint that expresses coverage defining variable, $g_{i\phi j}$, in terms of other variables defined in (1) and (2), is given as,

$$c_j \cdot \left(\sum_{i\phi} x_{i\phi} \cdot g_{i\phi j} - 1 \right) \geq 0. \tag{5}$$

The above inequality is non-linear as it involves the product of two binary variables. It can be linearized by adding a binary variable, $v_{i\phi j}$, representing the product, $c_j \cdot x_{j\phi}$, and

the following two constraints,

$$c_j + x_{i\phi} \geq 2 \cdot v_{i\phi j} \tag{6}$$
$$c_j + x_{i\phi} - 1 \leq v_{i\phi j}. \tag{7}$$

The following constraint is required to ensure that n camera poses are selected,

$$\sum_{i\phi} x_{i\phi} = n. \tag{8}$$

Lastly, to ensure that one possible camera location does not have more than one camera placed, the following constraint needs to be added,

$$\sum_{\phi} x_{i\phi} \leq 1. \tag{9}$$

The integer programming model for the optimal camera placement problem for vehicle surround-view coverage is defined as,

$$\text{maximize} \quad \sum_{j} c_j$$
$$\text{subject to (5), (6), (7), (8) \& (9).} \tag{10}$$

IV. MULTI-RESOLUTION METHOD

All approximate optimization methods used for optimal camera placement problems build on the core idea of effectively exploring only a subset of the solution search space without having to search all of it for the global optimal solution. Our proposed multi-resolution method, on the other hand, is built to explore only the important regions in the modelled space, therefore, reducing the size of search space before even the optimization procedure begins. In fact, with fewer number of points in the set of possible camera positions, x_i, our method can exponentially decrease the number of calculations required in the pre-processing step of visibility checks. The idea behind the multi-resolution method is to group the initial set of possible camera placement locations, x_i, into K clusters of points, and optimize for n camera poses while considering the cluster centroids as the new set of possible camera locations. The clustering step reduces the size of x_i from N to K, where $K \ll N$, therefore decreasing the total number of combinations of possible solutions (i.e., the solution search space). The two steps of clustering and optimization are repeated iteratively, while propagating the solution from one iteration to the next. It means that only the camera points comprising the n clusters of points, selected in this iteration as the optimal camera poses, are passed onto the next iteration, while the rest of the camera points are discarded. The iterative process stops when the number of camera points comprising the selected n clusters, in a certain iteration, is less than a user-defined limit, l. At this stage, camera poses are optimized, for one last time, without any further clustering of the input set of points, x_i. Figure 2 shows an illustration of the multi-resolution method, and, the next paragraph explains, through an example, the iterative process and the reasoning behind naming the method as *multi-resolution*.

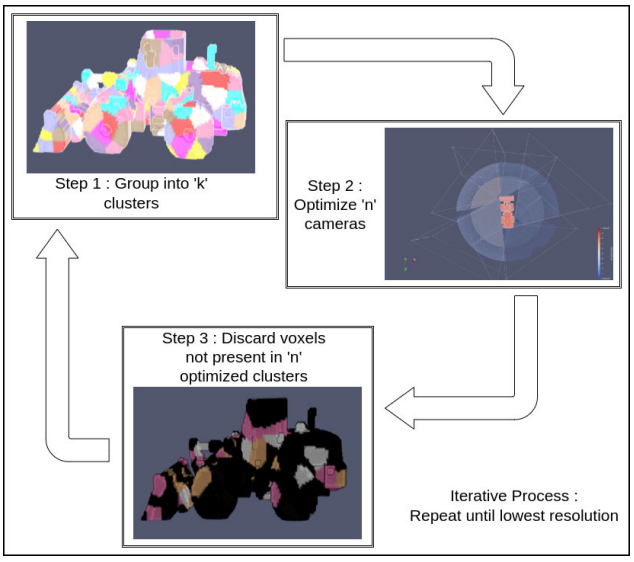


FIGURE 2. An illustration of the multi-resolution optimization method showing an example situation of placing two cameras on the data points. At each resolution the selected camera points are highlighted in solid color. Only the sets of points shown in solid color propagate to the subsequent resolution level, with the rest being discarded.

Consider a 3D model of a bulldozer vehicle. A standard bulldozer vehicle would measure 4.5 meters in its longest dimension (length). When it's 3D model is voxelized into a cube of length 128 voxels, the longest dimension is fit exactly into 128 voxels. It implies that each voxel in the voxelized 3D model would measure $\frac{4.5}{128} \approx 35mm$, i.e., $r = 35mm$ is the 3D image's resolution. Assuming the bulldozer model consists of $N = 11,000$ voxels representing the surface of the vehicle and that the voxel surface is uniformly one voxel thick, the surface area can be calculated as $N \times r^2$. Now, if the surface voxels are grouped together into $K = 100$ clusters, assuming that all the clusters are uniform in shape and size, the resolution of the resulting clustered image will be $\sqrt{\frac{N \times r^2}{K}} \approx 367mm$. Because the vehicle's surface is represented by larger voxels than the original image, we call the clustered data as low-resolution image. Say we want to optimize poses of $n = 5$ cameras. Given that the approximate size of each cluster in this case is 110 voxels, we will have $N = 550$ points, belonging to the 5 clusters that were selected as the solution in this iteration, that will be passed as input to the next iteration. Assuming $l = 200$, the input points will be clustered again into $K = 100$ clusters because $550 > l$. This results in a resolution of $82mm$ for this iteration. It is to be noted that this resolution is higher than previous iteration but it is still lower than the original resolution of $35mm$. The clustering and optimization steps are repeated until when N becomes lower than l. When $N < l$, the input points are not clustered any further and the set x_i for optimization at this stage is a small subset of the original set of points with the resolution, $r = 35mm$. Because the resolution of the last iteration is same as the resolution of the original voxelized

model, we call it as the *highest* resolution, while, the first iteration is called *lowest* resolution.

A. CLUSTERING BASED ON POINT ORIENTATION

3D image clustering is an extensively studied topic in the field of computer vision. Well-known, unsupervised clustering methods such as the K-means and the Gaussian mixture model algorithms, [36], [37], are proven to be efficient on general 2D/3D point data. There also exists a vast variety of application specific clustering methods, such as, for 3D point cloud data, [38], [39], 2D images, [40], 3D image (RGB+D) data, [41], etc. The data we use here for OCP is like 3D point clouds, i.e., each camera position is represented using 3D spatial coordinates along with an associated 3D vector representing the camera's view direction. However, to our knowledge, clustering methods tailored specifically for OCP problems do not exist. We propose a new clustering method adapted from the *SLIC* algorithm proposed by Achanta *et al.*, [42].

Our clustering algorithm is an iterative process, where all the points are assigned to a set of cluster centers, while at each iteration, new clusters centers are added or existing ones are removed depending on the number of outlier points. The input to the algorithm is a set of N 6D vectors, $a_i = [x_i\ y_i\ z_i\ u_i\ v_i\ w_i]^T$, that are a merger of the set of possible camera positions, $P_i = (x_i, y_i, z_i)$, and their associated primary orientation vectors, $\hat{P}_i = (u_i, v_i, w_i)$. It is to be noted that clustering is done based only on the primary orientations of the camera points. At this stage, the rotations, Φ, for the camera positions do not yet come into play as those rotations about the primary view direction vector are computed after clustering, and before optimization, to form the set, $x_{i\phi}$, of possible camera poses. The input set of points are grouped together into a set of K clusters, where the value of K is chosen by the user and passed as input to the algorithm. Initially, a set of clusters seeds, $C_k = [x_k\ y_k\ z_k\ u_k\ v_k\ w_k]^T$, $k = 1, \ldots, K$ are selected from the input set of points, a_i. The output of the algorithm is a *labels* vector of values k for each point a_i indicating the cluster it belongs to, and a set of cluster centers C_k that are estimated as the average of all the points a_i belonging to each cluster k. The clustering process can be broadly categorized into three steps: 1) initialization 2) assignment and 3) update.

The initial seeds are selected through a strategy that exploits the orientations associated with the camera positions. Assuming that all the clusters are uniform in size, the approximate size of each cluster is given as, $c_{size} = \frac{N}{K}$. Under another assumption that the clusters are uniform in shape, the length of each cluster can be given as, $S = \sqrt[3]{\frac{N}{K}}$. The input points a_i are first voxelized using the *voxel grid* filter provided as part of the Point Cloud Library API, [43], with S as the length of the resulting voxels. The voxel grid filter essentially downsamples the points using a 3D grid, where the centers of each cell represent the mean of all the points that fall within that cell. Selecting a length of S for the voxel grid ensures that the resulting points are all spaced at least S units apart. Downsampling with a voxel grid results in a number of points that is usually larger than the number of clusters, K. We randomly select K points, from the downsampled set of points, to represent the initial set of clusters centers, C_k.

In the assignment step, the points a_i are, each, assigned a label k corresponding to the cluster center C_k that it belongs to, based on a distance metric D. D is computed as a combination of the Euclidean distances between the position and the normal vectors of a point a_i and a cluster center C_k. If the Euclidean distance between the position vectors of a camera point and a cluster center is given as, $d_s = \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2 + (z_i - z_k)^2}$, and the Euclidean distance between their normal vectors is given as, $d_n = \sqrt{(u_i - u_k)^2 + (v_i - v_k)^2 + (w_i - w_k)^2}$, then the distance metric is written as,

$$D = \sqrt{d_n^2 + \frac{m}{S} \cdot d_s^2}, \textit{Changes} \tag{11}$$

where, m, is a user-defined parameter that acts as a relative weight between d_s and d_n. The spatial distance has to be normalized by the search radius S because, d_n is usually under 1 as it is computed using unit normal vectors, whereas, d_s varies from model to model. During assignment of points to clusters, only the points lying within a radius of S units from a cluster center are searched and compared using the distance metric. It implies that the farthest point assigned to a cluster center cannot be more than S units away from it, spatially. Therefore, normalizing by S brings the value of d_s similar to that of d_n.

We go through each cluster center sequentially, search a spherical neighbourhood of radius S around it, and assign the encountered points to the cluster center if it is the smallest distance, D, the point has seen so far. There can be some overlap between different clusters as the search radius around a cluster center is same as the distance between two cluster centers (recollect that both are equal to S). Due to the overlap, it might happen that some points initially assigned to one cluster may get reassigned to another. This, however, provides some flexibility to the size of clusters, allowing some clusters to be larger or smaller to better fit the vehicle's 3D structure. Once the neighbourhoods of all the initialized cluster centers are checked, the outlying points that have not been assigned to any of the cluster centers are collected (Let us call the count of outlying points as N_{ol}). At the same time, the size of each cluster, i.e., the number of points assigned to each cluster center are also counted. The number of unassigned points together with sizes of exiting clusters give us an estimate on how many cluster centers to be added or removed. This feature makes the clustering process dynamic in nature, helping to accurately cover the vehicle's entire surface.

In the update step, first the cluster centers that do not have any assigned points are removed. If K_r cluster centers are removed, the remaining $K - K_r$ cluster centers, C_k, are updated as the mean of all the points, a_i, assigned to the each cluster center. An estimate of cluster centers that need to be added is obtained by diving the number of

outlying points by the, previously calculated, approximate cluster size, c_{size}, i.e., $c_{add} = \frac{N_{ol}}{c_{size}}$. The set of outlying points is processed in the same way as the initialization step, to initialize c_{add} number of new cluster centers. This process of adding and/or removing cluster centers changes the number of clusters (say, the final number of clusters after completion is K') that was initially selected by the user. The approximate cluster size, calculated previously from the user-inputted number of clusters, K, however, remains constant throughout the clustering, making c_{size} (or S) as the primary parameter that controls the clustering process. Therefore, the two parameters, K and S, are interchangeable as input to the clustering algorithm. We run the clustering process with k as input.

The assignment and update steps are repeated iteratively until, either the number of outlying points changes by less than 10% from the previous iteration, or if there are no more cluster centers required to be added. For example, when $N_{ol} < c_{size}$, c_{add}, as an integer division, becomes zero, implying that no additional cluster centers can be initialized. Our experiments show that the algorithm usually runs until there is no possibility of adding any more clusters. Hence, when the stopping criteria is satisfied, the existing cluster centers are updated as the mean of all the points assigned to the cluster, and the outlier points (if any) are checked against all the cluster centers using the distance metric, D, and assigned to the closest cluster center, as a last step. At the end, the total number of clusters may be different than the value of K initially chosen by the user, but the dynamic process ensures that the surface of the vehicle is effectively captured by uniform clusters. The complete algorithm is detailed in Algorithm1. The clustering algorithm has a time complexity of $\mathcal{O}(N)$.

Algorithm 1: Clustering Based on Point Orientation

Input: K,$a_i = [x_i\ y_i\ z_i\ u_i\ v_i\ w_i]^T\ \forall\ i = 1 : N$;
Result: $labels_i = k\ \forall\ i = 1 : N$, $C_k\ \forall\ k = 1 : K'$
$S = \sqrt[3]{\frac{N}{K}}$;
Initialize: $C_k = [x_k\ y_k\ z_k\ u_k\ v_k\ w_k]^T\ \forall\ k = 1 : K$;
$D_i = \inf$, $labels_i = -1\ \forall\ i = 1 : N$;
$K' = K$, $t = 0$, $N_{ol}(t) = N$, $c_{size} = \frac{N}{K}$;
while *(1)* **do**
  for $k = 1$ **to** K' **do**
    for a_i *in neighbourhood of radius* S **do**
      D = D(a_i, C_k) as in equation 11;
      if $D < D_i$ **then**
        $D_i = D$;
        $labels_i = k$;
      end
    end
  end
  outliers = collect points with label == -1;
  $N_{ol}(t+1)$ = size(outliers);
  remove all centers with $size(C_k) < 1$;
  $c_{add} = \frac{N_{ol}(t+1)}{c_{size}}$;
  if $(\frac{N_{ol}(t) - N_{ol}(t+1)}{N_{ol}(t)} < 0.1)\ ||\ (c_{add} < 1)$ **then**
    break;
  end
  Re-estimate C_k as mean of all a_i with $labels_i == k$;
  Initialize c_{add} clusters and append to C_k;
  $K' = K$ after adding and/or removing clusters;
  $t = t + 1$;
end
Force assign outlying points to nearest cluster;
Re-estimate C_k as mean of all a_i with $labels_i == k$;

B. ALGORITHM

The initially collected set of voxels representing the vehicle's surface, P_i, and their primary orientation vectors, $\hat{P}_i$ (together represented as a set of 6D points, a_i), and the set of control points, $b_j = [x_j, y_j, z_j]^T$ are passed as input to the multi-resolution (MR) algorithm. Additionally, the required number of cameras to be optimally placed, n, is also passed as input to the algorithm. As illustrated at the beginning of this section, the MR method involves two steps: 1) Clustering of input points, and; 2) Optimizing for camera poses considering the cluster centers as the new set of possible camera locations. These two steps are repeated iteratively until the total number of variables in the set a_i is more than the pre-defined limit, l. The algorithm starts with a loop where the two steps of clustering and optimization are repeated.

Initially, the set of points a_i are clustered into K' clusters. Following this step, the set of K' cluster centers, C_k, represent the new set of possible camera locations, a_i, thus, decreasing the size of input from N to K'. These points are then rotated, about the associated primary direction vector (remember that $C_k[x_k\ y_k\ z_k\ u_k\ v_k\ w_k]^T$ is 6D vector representing both position and orientation) to form the set of variables $x_{i\phi}$. Together with Φ rotations in the four directions, the resulting set of possible camera poses consists of $((\Phi + 1) \times K')$ number of points (instead of $(\Phi + 1) \times N$ points when considered without clustering). Followed by the rotation step, visibility checks are performed on the sets of variables, $x_{i\phi}$ and c_j, to create the visibility matrix, as detailed in SectionIII-C. Post the visibility checks step, the accepted camera poses (the exact number depends on parameters cov_{self} and cov_{min} and varies from model to model) comprise only a small subset of the original set of points, as most of the possible camera poses that do not fulfill visibility criteria are discarded. Binary variables, $x_{k\phi}$ and c_j, are then initialized over the accepted camera poses and the complete set of control points, and passed as input to the user-chosen optimization method, along with the binary variables $g_{k\phi j}$ that represent the visibility matrix.

This algorithm can work with any optimization method, as all discrete combinatorial optimization methods take the same input (integer decision variables, set of constraints and a pre-computed visibility matrix) and produce the same output (set of optimal camera poses). The optimization process

then selects n (where n is a number chosen by the user representing the number of cameras to be optimally placed) optimal camera poses from the input set, by optimizing the objective function while enforcing the constraints. After completion of the optimization process, the obtained solution is processed for the next iteration. The points belonging to the n optimally selected clusters are extracted to represent the new set of possible camera locations a_i. The rest of the points belonging to the remaining $K' - n$ clusters are discarded. If the number of points, N, in the new set a_i are more than l, then the steps of clustering, creating rotated direction vectors, visibility checks, and optimization, are repeated. Otherwise, the program exits the loop and optimization is done on the remaining set of points a_i, for one last time, without clustering. As these points are not clustered any further, this step of final optimization is called the *highest resolution* level. It might sometimes happen that the solution obtained at the highest resolution is worse than a solution obtained in one of the previous iterations. Therefore, the solution at each iteration is saved and the best of all solutions is presented as the final result of the MR algorithm. The output of the algorithm is a set of n 6D vectors, $camSol_n = [x_n\ y_n\ z_n\ u_n\ v_n\ w_n]^T$ representing the poses of the optimally placed cameras.

The multi-resolution optimization algorithm is detailed in Algorithm2. Initialization of parameters m, l, Φ, cov_{min} and cov_{self} is discussed later in Section V-A. In the algorithm, the functions *cluster()*, *rotate()*, *visibilityChecks()* and *optimize()* each perform a specific task as implied by their names, such as; the function *cluster()* performs the clustering operation as described in Algorithm 1; the *rotate()* function produces a set of Φ rotated direction vectors about each camera pose's primary orientation, as described in Section III-A; *visibilityChecks()* performs all the necessary calculations described in Section III-C and sets up the visibility matrix, and; function *optimize* passes all the variables and the parameter n to the user-selected optimization method (such as, branch-and-bound, greedy heuristic, etc.) to obtain the optimal camera poses. In the *rotate()* function, the notation "$\hat{}$" refers to the direction vector ($\hat{P} = [u, v, w]^T$) part of the 6D variable. Lastly, the *optimize()* function in the loop returns a vector of labels, k_n, of the n solution clusters. k_n is shown in the while loop, whereas it is omitted from the call to *optimize()* function in the *highest resolution* because, the input points at this stage correspond to points from the original voxel image without any clustering. By saving the clusters' labels, we can easily track and retrieve all the camera points that belong to those clusters.

The time complexity of our proposed MR algorithm depends on three factors, i.e., the individual complexities of clustering, visibility checks and optimization steps. Time required for the clustering step, with a linear complexity depending on the number of input points in set x_i, varies from one resolution level to another. It requires highest computational time for the lowest resolution level when the size x_i is equal to N, whereas, the computational time for subsequent resolution levels decreases with each level as the number of points in input set x_i at a resolution level $t + 1$ is always less than the number of points in x_i at level t. The complexity of visibility checks step is given as $\mathcal{O}(K'\Phi J)$, as after the clustering step, the MR method works with only K' points. This is significantly lower than the corresponding complexity for SR method ($\mathcal{O}(N\Phi J)$) as $K' \ll N$. The complexity of the optimization step depends on the chosen optimization algorithm. For example, the linear programming based-branch-and-bound algorithm has an exponential complexity whereas the particle swarm optimization method has a linear complexity in the number of particles. The number of points passed as input to the optimization algorithm, after visibility checks operation, represent only a small subset of the points in $x_{i\phi}$. Consider the number of input points after visibility checks for to be N' for SR method and N'' for MR method. Then for the simplest of all cases, if we assume that all the chosen optimization methods have linear time complexity, the complexity of MR method can be given as, $\mathcal{O}(N'')$, which is again significantly lower than the complexity for SR method ($\mathcal{O}(N')$) because, N'' is a subset of the set with $K' \times \Phi$ points whereas, N' is a subset of the much larger set with $N \times \Phi$ points. If the values of parameters cov_{min} and cov_{self} are kept same for MR and SR methods, then N'' will always be much lesser than N'.

Algorithm 2: Multi-Resolution Optimization

Input: $a_i = [x_i\ y_i\ z_i\ u_i\ v_i\ w_i]^T\ \forall\ i = 1 : N$,
$c_j = [x_j, y_j, z_j]^T\ \forall\ j = 1 : J$, K, n;
Result: $camSol_n = [x_n\ y_n\ z_n\ u_n\ v_n\ w_n]^T\ \forall\ 1 : n$
Initialize parameters m, l, Φ, cov_{min} and cov_{self};
$cov_{best} = 0$;
/* lower resolutions */
while $N > l$ **do**
  $[C_k,\ labels_i] = \text{cluster}(a_i, K)$;
  $C_{k\phi} = \text{rotate}(\hat{C}_k)$;
  $x_{k\phi} = \text{VisibilityChecks}(C_{k\phi}, c_j)$;
  $[sol_n, k_n, cov] = \text{optimize}(x_{k\phi}, c_j, n)$;
  if $cov > cov_{best}$ **then**
    $camSol_n = sol_n$;
    $cov_{best} = cov$;
  end
  a_i = all points with $labels_i = k\ \forall\ k = k_n$;
  $N = size(a_i)$;
end
/* higest resolution */
$a_{i\phi} = \text{rotate}(\hat{a}_i)$;
$x_{i\phi} = \text{VisibilityChecks}(a_{i\phi}, c_j)$;
$[sol_n, cov] = \text{optimize}(x_{i\phi}, c_j, n)$;
if $cov > cov_{best}$ **then**
  $camSol_n = sol_n$;
  $cov_{best} = cov$;
end

V. RESULTS

Experiments were conducted on synthetic data as well as on 3D models of real vehicles used in the construction industry. The need for synthetic data arises due to large sizes of high detail 3D models of real construction equipment. The main problem arises with the binary integer programming formulation-based branch-and-bound optimization algorithm that has high memory or time requirements. Due to this shortcoming OCP environments with large sizes cannot be handled using the available resources. Therefore, we created simulated data using simple geometrical structures, roughly resembling vehicles. As, this data has small number of samples, we present a detailed comparison of all the considered optimization methods. However, tests on real data further validate the efficacy of our method. We created four models resembling a car, a van, a truck and a bus in four different sizes, S, M, L and H. All the models in the four size categories have about, 75, 200, 650 and 1300 voxels, respectively (or possible camera locations). The number of control points remains same for the four vehicle models at about 320, 530, 1650 and 3360, control points, for the four sizes, respectively. In total, the simulated data consists of 16 instances. Similarly, there are 16 instances of real data, i.e, four vehicle models, namely, bulldozer, JCB, mining truck and tractor scraper, in four sizes labelled as 32, 64, 128, and 256. The real data was obtained as 3D polygon models which were voxelized into cubes of lengths 32, 64, 128 and 256 voxels. The JCB vehicle model, being the smallest of the four has $\sim$ 2300 voxels on the surface of the model (possible camera locations) when voxelized into a cube of 32 voxels, and $\sim$ 115000 camera locations when voxelized into a cube of 256 voxels. Whereas, the mining truck vehicle model, being the largest of the four has $\sim$ 6600 and $\sim$ 360000 surface voxels when voxelized into cubes of 32 and 256 voxels, respectively. The tractor scraper and bulldozer models have similar size with $\sim$ 170000 surface voxels at *size-256*. It is to be noted that for every instance, the total number of variables, for each model, before visibility checks is given by the number of boundary voxels multiplied by $\Phi = 97$. This implies that for the largest model (Mining truck-256), the total number of input variables before visibility checks is $\sim$ 35 million. The polygon and voxelized models of real vehicles have been made publicly available.[1]

On simulated data, for each of the 16 instances, we test five optimization methods using the single resolution (SR) and multi-resolution (MR) optimization strategies. To recollect, SR is when we optimize for camera poses on all the camera locations (voxels) that are part of the vehicle's surface, without any clustering or sub-sampling. Whereas MR is when we cluster the sets of vehicle's surface voxels multiple times and optimize camera poses on the set of cluster centers. To validate our proposed MR optimization strategy, using C++ programming language, we implemented the Greedy Heurisitc (GH) proposed in [9], Metropolis Sampling (MS) proposed in [19], row weighting local search heuristic (RWLS) algorithm proposed in [22] and the LH-RPSO algorithm proposed in [17]. The LP-based branch-and-bound algorithm (LP) provided by CPLEX, [44], is the fifth optimization algorithm that we used for validation. As the branch-and-bound algorithm is known to provide provable bounds on the optimal solution, it is a good choice to compare against the results from approximate algorithms on the data that we used for this work. However, on the 16 instances of real data, we chose to test only one of the two evolutionary algorithms (MS and LH-RPSO). In consequence, we dropped MS method as coverage results on simulated data showed that LH-RPSO is better and consistent than the MS method. Control point coverage for simulated data and real data are shown in Table 1 and Table 3, respectively. Whereas, the total optimization times for tests on simulated and real data are presented in Tables 2 and 4, respectively. Figure 3 shows an example visualization of the bulldozer, JCB and mining truck vehicle models at *size-256*. The second row in the figure shows visualizations of three OCP solutions. The next two sub-sections detail the values for all the parameters and some modifications to SOTA algorithms, respectively. Detailed analysis of the results is presented in Section V-C.

A. PARAMETERS

The value of the limit is decided based on the system's hardware, i.e., the number of variables and constraints the LP-based branch and bound algorithm can handle without running into the *out-of-memory* error. All the experiments were run on an Intel Core i7-8700K CPU with 12 processing cores and 32GB of RAM. Parallel processing was done using the 12 processing cores of the same CPU. Given our system's hardware, and through trial-and-error, we chose a value of $l = 200$ for all our experiments. In all our experiments spatial distance was given twice the importance as compared against d_n, therefore, we chose $m = 2.0$ for the clustering process. The primary direction vector at each possible camera location was rotated at steps of 3.75° in each direction, producing a total of 96 rotations, i.e., for each possible camera position, $\Phi = 97$, including the primary camera orientation. $cov_{self} = 0$, for all experiments ensured that camera poses occluded by the vehicle were excluded from the visibility matrix. In all cases, we optimized to place five cameras, i.e., $n = 5$ for all experiments.

Choosing a uniform value for cov_{min} is tricky as, a small value may allow too many variables in the visibility matrix, leading to a possible *out-of-memory* error when using the branch-and-bound algorithm, whereas, a lager value may lead to too few variables in the visibility matrix, thereby removing the combinatorial aspect of the optimization problem. To maintain uniformity across all the experiments, we chose a value for cov_{min} in such a way that an accepted camera pose would cover at least 70% of an expected number of control points for one camera, under ideal circumstances.

[1] https://github.com/AnirudhPuligandla/multi-resolution-optimal-camera-placement.git

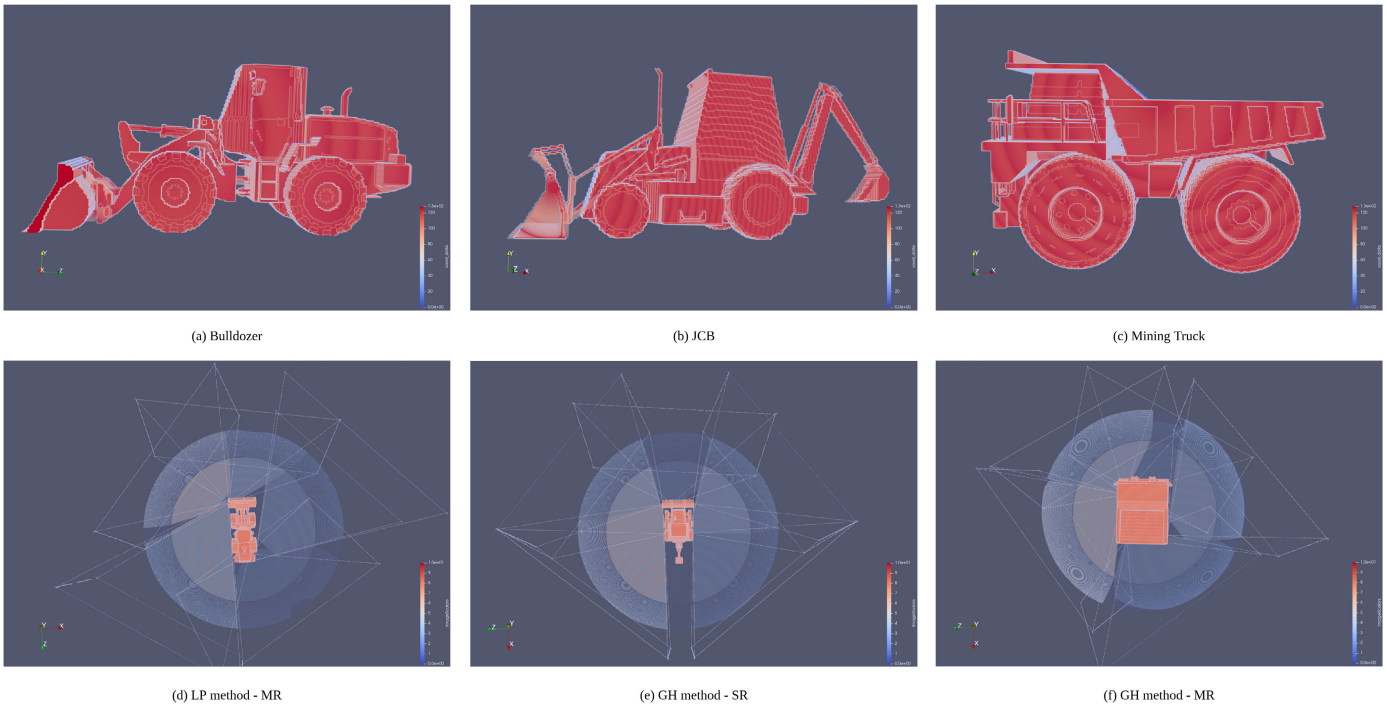


FIGURE 3. Figure showing 3D models of (a) Bulldozer, (b) JCB, and, (c) Mining truck, heavy vehicles, voxelized into cube of length 256 voxels. The bottom row shows visualizations of OCP results with five optimally placed cameras, obtained using (d) LP method at multiple resolutions for bulldozer model, (e) GH method at single resolution for JCB model, and, (f) GH method at multiple resolutions for mining truck model.

For example, if a space model of a particular size category contains J number of control points, the expected coverage for one camera in the ideal scenario will be $\frac{J}{n}$. A value $cov_{min} = 0.7 \times \frac{J}{5}$ was set which remains constant across all vehicle models for one size category. The maximum time limit for each test, i.e., the total time for the individual steps of clustering, visibility checks and optimization, was set at 48 hours.

Considering the larger sizes of real data, the values of parameters of individual optimization algorithms were different for simulated and real data. Parameter $l = 50$ was set for simulated data as the smallest model has only 75 camera poses. We set the same value of l for instances of simulated data to maintain uniformity. The LP method provided by CPLEX has multiple parameters that decide on optimality, feasibility, choice of optimization algorithm, etc. For tests on both real and simulated data, we chose to emphasise on feasibility over optimality and allowed a tolerance of upto 6% in the gap of the feasible solution from the optimal solution. The GH method does not have any parameters. The MS algorithm was run until either all the available camera poses were visited, or, until a number of iterations equal to the available camera poses were completed. The RWLS algorithm was run for 250 iterations on simulated data and for 500 iterations on real data. The LH-RPSO algorithm was run with 30 particles for both data, and, for 1000 iterations on real data and 500 iterations on simulated data. These values were chosen as a trade-off between coverage quality and time required for optimization. All these values ensure that optimization can be completed in reasonable amount of time without noticeably compromising on coverage. As the parameter values are kept same for both MR and SR methods, it allows for a fair comparison between the two optimization strategies.

B. IMPROVEMENTS TO SOTA METHODS

For the RWLS method, it is required to compute a matrix describing neighbourhood relationships between all the camera poses. A camera pose is considered a neighbour to another if at least one control point is covered by both the cameras. For an instance with α camera poses, the time complexity for this step is given as, $\mathcal{O}(\alpha^2)$. Although this is lower than the complexity of our visibility checks algorithm, this step takes as much time, particularly for the SR method, as in most cases the number of camera poses selected after visibility checks is higher than the number of control points. We perform these calculations parallel on our multi-core CPU using OpenCL kernel code. Similarly, the GH method requires re-computation of the visibility matrix after each iteration. As we already perform parallel computations for visibility checks, we re-use the same OpenCL kernel code for the GH method to achieve an overall speedup.

C. ANALYSIS

In the presented tables, each row shows results for one instance, with alternate columns presenting the results for SR and MR methods, for the selected four or five optimization methods. For each row in Tables 1 and 3, one entry is highlighted in bold to represent the best obtained control point coverage for that instance. Some entries in the tables

TABLE 1. Table showing objective function value (in % of control points covered), obtained from the five optimization algorithms (LP, GH, MS, RWLS and LH-RPSO) using both MR and SR methods for the four simulated vehicle models in four size categories (S,M,L,H) for each model.

Vehicle	Size	LP		GH		MS		RWLS		LH-RPSO	
		SR	MR	SR	MR	SR	MR	SR	MR	SR	MR
Car	S	**84.2**	**84.2**	83.0	83.0	81.0	80.8	83.5	83.3	82.9	83.9
	M	86.2	**86.6**	85.3	85.5	81.4	81.2	84.9	86.2	84.1	85.9
	L	–	**86.9**	84.3	84.3	80.6	81.6	85.6	85.3	84.8	83.3
	H	–	85.2	81.5	84.8	79.1	82.5	85.8	**87.3**	86.2	86.7
Van	S	82.2	83.0	82.9	82.8	79.9	79.6	83.3	84.2	**84.5**	**84.5**
	M	85.6	**86.3**	85.5	84.4	78.5	79.6	85.3	85.7	85.5	85.9
	L	–	**86.7**	84.3	86.2	79.5	81.2	85.4	84.2	84.1	84.3
	H	–	86.4	81.5	86.4	80.4	82.0	85.7	**87.6**	84.0	84.6
Truck	S	**84.2**	83.6	82.9	82.9	78.0	79.3	82.9	83.0	83.2	83.2
	M	**85.7**	**85.7**	85.4	85.4	79.8	81.3	84.9	85.3	85.5	85.5
	L	–	**87.7**	84.3	87.2	81.7	80.6	84.3	85.8	85.1	85.7
	H	–	**88.2**	81.5	87.6	80.1	85.7	85.7	87.5	85.1	85.0
Bus	S	84.2	84.2	83.0	83.9	78.7	80.8	83.0	**84.5**	84.2	83.7
	M	85.6	87.6	85.5	**88.3**	77.4	80.5	85.1	86.2	85.1	84.9
	L	–	89.7	84.3	**89.9**	80.8	81.5	85.5	88.4	84.2	87.5
	H	–	**88.0**	81.4	**88.0**	82.1	82.4	85.1	85.7	84.2	83.4

TABLE 2. Table showing total optimization time (in seconds) taken by the five optimization algorithms (LP, GH, MS, RWLS and LH-RPSO) using both MR and SR methods for the four simulated vehicle models in four size categories (S,M,L,H) for each model.

Vehicle	Size	LP		GH		MS		RWLS		LH-RPSO	
		SR	MR	SR	MR	SR	MR	SR	MR	SR	MR
Car	S	61.52	81.51	1.24	1.47	1.18	1.24	9.88	8.95	4.71	9.27
	M	309.91	178.65	2.84	1.92	2.42	1.53	51.81	16.73	8.91	14.64
	L	–	1113.77	11.61	4.11	15.04	3.73	526.99	83.96	36.76	70.77
	H	–	2629.39	36.24	4.77	57.08	4.98	2174.6	75.47	73.16	145.21
Van	S	75.95	63.35	3.20	6.13	1.61	1.34	12.8	9.64	5.53	9.13
	M	568.84	171.62	3.04	1.83	2.85	1.75	56.86	19.04	9.25	14.44
	L	–	1625.48	12.96	3.52	15.44	3.53	548.15	57.92	32.57	70.08
	H	–	3358.98	41.98	7.39	52.99	9.04	2215.73	254.75	78.27	150.48
Truck	S	69.47	58.36	1.37	1.55	1.19	1.32	11.01	7.85	5.38	9.26
	M	315.63	235.84	2.65	2.06	2.57	1.74	52.75	20.69	8.78	14.84
	L	–	1036.62	11.57	3.3	13.44	3.42	508.40	86.88	31.18	47.49
	H	–	2521.24	36.08	7.01	45.96	7.28	2124.72	166.26	70.97	148.62
Bus	S	68.9	95.36	1.59	1.5	1.44	1.39	12.17	9.02	5.30	8.81
	M	433.13	272.03	3.32	1.82	3.28	1.69	61.98	22.17	9.55	14.86
	L	–	821.96	15.71	3.6	17.29	3.47	585.91	60.27	33.71	47.14
	H	–	2490.46	50.74	5.48	60.34	6.78	2166.97	96.59	85.07	145.15

are marked as '–' because, they could not be completed either because of the *out-of-memory* error or because, the entire optimization process could not be completed in under 48 hours. In the case of simulated data, experiments on any of the four vehicle models for 'L' and 'H' sizes could not be completed using LP method as the branch-and-bound algorithm from CPLEX ran out of available RAM due to high number of variables and constraints. From our trials, we observed that with our available RAM the LP method can be run till completion, only when the total number of variables, i.e., the number of camera poses selected after visibility checks plus the number of control points, is less than $\sim$ 3000. Similarly, for real data we omitted the entire column for SR method using LP optimization because, none of those tests were completed as all the sixteen instances had more than 3000 variables. This shows that using the MR strategy we can use LP optimization on large data where it was not possible otherwise.

From Tables1and3, we can see that optimization process using LP method was completed for only eight of the thirty two instances when using SR method. Whereas, with MR method, optimization using LP method was completed for thirty one out of the thirty-two instances. The only exception is the *tractor scraper 256* model where the total number of variables exceeded 3000. Although this optimization can be completed by lowering the value of parameter l, we did not do that to maintain uniformity in parameter values across all tests. By lowering the value of l we can set an upper bound on the number variables that are passed to LP optimization to thereby, avoid the *out-of-memory* error. It is to be noted that changing the value of l will also change the number of resolution levels, i.e., the number of iterations of the MR method. We believe that a lower or higher number of iterations changes the solution at that iteration but, it does not have any effect on the overall best solution obtained using the MR method.

From Table1, it can be seen that for simulated data, MR strategy provides the best control point coverage for twelve out of the sixteen instances. For three instances, both MR and SR methods provided the same best coverage while the SR method provided the best coverage in only one instance. These results show that the MR strategy can provide either the same or better coverage in 93% of the cases. Table 2 shows that, in fact, the MR method needs less computational time than state-of-the-art methods to provide better control point coverage. The same quality is re-iterated in the results on real data (see Table 3) where, the MR method provides the same or better control point coverage in 81% of the cases (13 out of 16 instances). Overall, the best coverage was obtained when using the LP optimization method in 18 out of the 32 instances, i.e., in more than half of the cases. This shows the importance of complex methods which can guarantee global optimal solution or provide provable bounds around the global optimum. Moreover, in some cases, where other approximate optimization methods provide better control point coverage, the LP method may also result in a better solution if allowed to run until it finds the optimal solution albeit at the cost of increased computational time.

If we observe on a case to case basis, including results for all the instances from all the optimization methods, we can see that the MR method provided better coverage accuracy than the SR method in 55 out of the total 80 cases (68.75%) on simulated data whereas, the SR method provides better coverage accuracy in only 15 cases (18.75%) while the results from both SR and MR methods are same in the remaining 10 cases (12.5%). In the cases where the result from MR method is better than SR method (excluding the cases where the SR method does not have a result), the average gap between the two corresponding solutions is 1.86%. In cases, where the SR method provided better coverage, the result from MR method is worse by an average of 0.56%. Similarly, in results on real data, the MR method provided better coverage than SR method in 54 out of a total 64 cases with an average gap of 0.82% while it provided the same result as SR method in one case, and, lower coverage in 9 cases with an average gap of -0.86%. This implies that the MR method provided either the same or better coverage than the SR method in at least 80% of all the experiments on both simulated and real data. Moreover, in some cases when it provided lower coverage than SR method, the percentage error between the two solutions is less than 1% in all cases. On the whole, looking at these results, we can say that the MR method provides a solution that is within -1% to $+2\%$ of the solution from SR optimization in significantly less computational time.

From Tables 1 and 3, it may be observed that camera coverage decreases with the increasing number of voxels. While it is intuitive to believe that camera coverage should increase with the number of voxels, we would like to emphasise that there are various parameters that influence the overall coverage for OCP problems. When the size of the vehicle model increases, the camera view frustum parameters also change, so do the number of control points and their sampling rates. Therefore, we emphasise that the coverage values are to be compared per instance (per row in the tables) and between different optimization methods but, not between different sizes or vehicle models. Although the coverage for the 32 size and 256 size for any model are similar, the vehicle models at 32 size have severely distorted shapes and low detail. At least the 128-size model is required for a good approximation of the real-world vehicles. For the 32 and 64 sizes, we sample the set of control points at one in every 20 points. This sampling frequency, however, results in a substantial number of control points for the 128 and 256 sizes that is more than the limit of 3000. To test the LP method, we decreased the control point sampling frequency for 128 and 256 sizes down to one in a hundred points and one in 250 points, respectively. As we can see from coverage values for 128 and 256 sizes, sampling lesser number of control points does not have any impact on the result if control points are sampled uniformly from the original set.

TABLE 3. Table showing objective function value (in % of control points covered), obtained from the four optimization algorithms (LP, GH, RWLS and LH-RPSO) using both MR and SR methods for the four real vehicle models in four sizes (32, 64, 128, 256) for each vehicle model.

Vehicle	Size	LP	GH		RWLS		LH-RPSO	
		MR	SR	MR	SR	MR	SR	MR
Bulldozer	32	92.2	90.1	92.2	**93.1**	92.0	87.4	85.6
	64	92.9	90.9	**93.2**	92.8	92.6	88.8	90.8
	128	**93.9**	88.9	90.8	93.2	93.5	87.1	88.3
	256	92.6	–	**93.0**	–	91.9	–	88.3
JCB	32	**93.6**	90.9	91.8	**93.6**	92.4	86.7	90.0
	64	**93.7**	90.6	91.2	**93.7**	92.9	84.4	85.4
	128	**93.3**	88.8	90.6	**93.3**	92.7	85.8	90.9
	256	**92.8**	–	90.2	–	91.3	–	89.6
Mining Truck	32	88.4	85.2	87.1	**89.6**	89.4	85.1	87.6
	64	89.4	88.6	88.6	**91.5**	90.9	87.0	88.7
	128	**90.0**	89.4	89.5	–	89.7	87.3	87.7
	256	**90.3**	–	88.0	–	89.9	–	86.9
Tractor Scraper	32	90.5	89.3	90.8	91.8	**92.2**	88.4	90.1
	64	89.7	89.1	89.5	91.9	**92.0**	89.9	88.6
	128	**91.8**	89.8	90.2	91.2	91.4	87.9	88.4
	256	–	–	88.9	–	**91.2**	–	86.5

As mentioned previously, the time complexity of MR method is lower as, firstly, the size of input to the pre-processing step of visibility checks is lower when compared to SR method. The smaller input size to visibility checks step implies that the size of input to optimization algorithm is also, low. Therefore, even though we use the same optimization methods, the overall computational time required for pre-processing and optimization steps is lower for MR method by several factors. the clustering step has low complexity when compared against the steps of visibility checks and optimization. For simulated data, the time required for clustering varied from $1 - 10ms$ for 'S' size category to $\sim$ $500ms$ for 'H' size category. For the mining truck vehicle model in the highest size category (256), computational time for the clustering step was $\sim$ $10s$. This is only a fraction of the total computational time that includes times for clustering, visibility checks and optimization (see the corresponding entry in Table 4). From

TABLE 4. Table showing total optimization time (in seconds) taken by the four optimization algorithms (LP, GH, RWLS and LH-RPSO) using both MR and SR methods for the four real vehicle models in four sizes (32, 64, 128, 256) for each vehicle model.

Vehicle	Size	LP	GH		RWLS		LH-RPSO	
		MR	SR	MR	SR	MR	SR	MR
Bulldozer	32	138.95	42.03	5.56	1035.90	67.79	45.06	21.80
	64	834.49	584.73	16.45	11273.17	227.34	573.84	140.71
	128	1426.68	13410.00	163.46	47665.15	469.09	13116.02	239.98
	256	3655.41	–	1797.59	–	2133.70	–	1778.98
JCB	32	246.09	36.44	4.64	736.65	71.88	39.00	19.02
	64	1806.86	469.25	20.99	8735.27	307.37	450.68	119.18
	128	1753.34	8300.07	142.37	35591.02	601.00	7934.27	200.45
	256	3869.25	–	1388.99	–	2162.99	–	1702.46
Mining Truck	32	79.28	229.89	11.30	576.52	38.25	223.29	30.7
	64	736.89	4804.36	56.38	9711.18	200.69	4469.13	172.09
	128	1538.45	1.42 e+05	849.32	–	1225.67	1.19 e+05	1015.02
	256	14347.50	–	11084.80	–	12079.00	–	11440.37
Tractor Scraper	32	233.91	30.81	3.89	631.55	70.51	29.57	17.36
	64	1668.14	613.31	18.74	9573.47	374.17	600.47	117.49
	128	834.01	12487.27	120.64	26889.26	439.61	12435.59	217.91
	256	–	–	1811.24	–	2463.68	–	1879.59

Tables2 and 4 we can see that the total computational time for MR method is always less than the corresponding entry for SR method. The only exceptions are the times shown for LH-RPSO method for all size categories and all methods for 'S' size in Table 2. For 'S' size category, for example, the car model has 75 camera poses. By clustering them into 20 clusters we do not gain any noticeable speedup. In fact, when using MR method by optimizing twice (i.e., at two resolutions), computational time only increases as the complexity of visibility checks step or optimization step does not change noticeably when the number of variables is so low. Computational time for most of our selected optimization algorithms increases exponentially with an increase in the number of input variables. Therefore the gap between computational times, for SR and MR methods, becomes visible with increasing size of the OCP instance.

Only the LH-RPSO method has constant complexity as it depends on the number of particles rather than the number of input variables. Therefore, in all cases for simulated data, the times shown for LH-RPSO method using MR optimization is higher than the corresponding entry for SR method, primarily due to optimizing for camera poses multiple times. The results for LH-RPSO method are however, different on real data. From the last two columns in Table 4, we can see that LH-RPSO (MR) method is 1.7 times (tractor scraper-32) to 117 times (mining truck-128) faster than LH-RPSO (SR). Despite doubling or tripling (depending on the number of resolutions) the time required for optimization, the MR method when using LH-RPSO takes overall less time computational time on real data. This is because of the significant amount of time saved during the visibility checks step. High complexity of the pre-processing step, when the size of input is large, overshadows the computational complexity of the optimization step, particularly when using GH or LH-RPSO optimization methods. LP and RWLS are however, complex methods, and owing to this complexity, they can produce better coverage when compared to GH or LH-RPSO optimization methods. SR Optimization for 256 size category on real data could not be completed as the visibility checks step took more than 48 hours for all the vehicle models. Optimizations using RWLS method on real data for 128 size category, also were not completed as the total time for visibility checks and optimization took more than 48 hours. Therefore, for the remaining 24 instances, i.e., all instances excluding 'S' size category in simulated data and 256-size category on real data, the MR method is at least 1.3 times (Truck-M (GH)) and up to 167 times (Mining Truck-128 (GH)) faster than the SR method.

TABLE 5. Table showing computational time (in seconds) for the individual steps of pre-processing and optimization for the tractor scraper model using SR optimization and the bulldozer model using MR optimization. The shown values are for the 32 and 128 size categories.

Instance	pre-processing	optimization		
		GH	RWLS	LH-RPSO
scraper-32(SR)	22.33	8.48	609.22	7.27
scraper-128(SR)	12422.20	65.07	14467.06	13.39
bulldozer-32(MR)	0.75	4.81	67.04	21.05
bulldozer-128(MR)	140.05	23.41	329.04	99.75

Table5 shows the time taken for the individual steps of visibility checks and optimization. In the table, we show the computational times for the tractor scraper-32 and tractor scraper-128 instances using SR optimization strategy and bulldozer-32 and bulldozer-128 instances using MR method, only as an example to highlight the share of computational times of the pre-processing and optimization steps in the total time. These trends, however, apply across all instances with minor variations in the actual times as per the size category and the number of variables contained in the models. We did not present individual times for all instances and methods to keep the results concise, and also because, our aim is to highlight gain in total computational times, i.e., time taken for all steps including pre-processing, optimization and clustering steps. The time required for the pre-processing step of visibility checks at 128 size category is over

500 times more than that at 32 size category when using SR optimization, whereas, it is only 185 times when using MR optimization. Similarly, the required time for optimization using RWLS method at 128 size category is 23.7 times more than the time at 32 size category when using SR method, whereas, the increase is only 4.9 times when using MR method. Finally, it can be said that by optimizing for camera poses on smaller subsets of data at multiple resolution levels, the overall computational time can be reduced significantly. Moreover, as the MR method produces coverage within an average gap of $[-1, 2]\%$ of the coverage obtained with SR method, it can be said that our proposed clustering method provides a good approximation of the vehicle's surface.

Although it might be interesting to run t-tests on the computational times to see if the difference in times are statistically different, we did not perform those tests as it is evident from Tables2and4that the computational times for MR method are significantly less than those of SR method. With the help of MR method, optimization was completed in reasonable time on large instances where the SR method took more than 48 hours, or, where the LP method could not be used due to a large number of input variables. However, the MR method also has a limitation when using the LP method. Optimization for tractor scraper-256 (MR) using LP optimization was not completed because, at the last resolution, the number of variables after visibility checks was high, leading to the *out-of-memory* error. While the tests on the tractor scraper model were run with $K = 110$, the optimization on tractor scraper-256 (MR) using LP method was run until completion when the parameters were altered as, $l = 100$ and $K = 85$ to obtain a coverage of 88.5%. Looking at the results from other optimization methods on this instance, it can be said that a lower value of K degraded the result. As this optimization could not be completed on this instance, it may be that this is the limit beyond which the LP optimization cannot be used even with MR strategy.

Moreover, this brings us to the importance of the role played by parameter K in the MR method. For the experiments on real data, we selected the optimal value of K for each vehicle model by running the GH method for $K = 90, 95, \ldots, 105, 110$ and picking K that provided best coverage. On simulated data, we set $K = 30$, for all size categories of car, van and truck models and $K = 50$ for the bus model. In the case of real data, $K = 95$ produced the best coverage for bulldozer and JCB models while $K = 110$ produced best coverage for mining truck and tractor scraper models. We run GH optimization using MR method on the tractor scraper-64 instance with the values of K from 90 to 110 at steps of one. The mean control point coverage obtained in these tests was 89.3 with a variance of 0.98. In general, the optimal value K varies from model to model and there does not exist a general strategy to select an optimal value of K. The user must manually choose a value K that is appropriate to the data. However, it is possible that a small perturbation around the chosen value of K may produce better results.

VI. CONCLUSION AND FUTURE WORK

We proposed a new clustering-based multi-resolution optimization method for the optimal camera placement problem for vehicle surround-view. It was shown that with our proposed method the OCP problem can be solved for large real world vehicle models in significantly less time. Moreover, with the MR method, linear programming-based branch-and-bound method can be used for large data where otherwise, SR optimization strategy does not work due to the optimization method's high resource requirements. Results from experiments on eight simulated and real vehicle models of various sizes show that MR method produces the same or even better camera coverage than the SR method, in only a fraction of the time. Results show that our proposed method is over 150 times faster than state-of-the-art. The fact that the coverage values obtained using MR method lie within $1 - 2\%$ difference of the coverage values obtained with SR method, shows that our proposed clustering method effectively captures the 3D geometry of the vehicle's surface. With the help of clustering into different resolution levels, we can decrease size of the input and the number of variables, thereby, decreasing the computational times of the pre-processing step, as well as, the optimization step by a big factor.

While, the MR method extends the applicability of LP optimization method to large data, it still faces limitations, when the number of clusters is kept high. It will be interesting to study other clustering strategies to see if this problem can be circumvented. We test the method only for the use-case scenario of OCP for vehicle surround-view. The method must be studied on other types of data (floor plan surveillance, for example) to examine the generality of MR optimization strategy. Establishing a relationship between the number of clusters, K, and the overall camera coverage may also help to generalize the method for general applications.

REFERENCES

[1] D. Buljeta, M. Vranjes, Z. Marceta, and J. Kovacevic, "Surround view algorithm for parking assist system," in *Proc. Zooming Innov. Consum. Technol. Conf. (ZINC)*, May 2019, pp. 21–26.

[2] V. Appia, H. Hariyani, S. Sivasankaran, S. Liu, K. Chitnis, M. Mueller, U. Batur, and G. Agarwa, "Surround view camera system for ADAS on TI's TDAx SoCs," White Paper, 2015.

[3] A. Hedi and S. Lončarić, "A system for vehicle surround view," *IFAC Proc. Volumes*, vol. 45, no. 22, pp. 120–125, 2012.

[4] F. Angella, L. Reithler, and F. Gallesio, "Optimal deployment of cameras for video surveillance systems," in *Proc. IEEE Conf. Adv. Video Signal Based Surveill.*, Sep. 2007, pp. 388–392.

[5] X. Zhang, X. Chen, J. L. Alarcon-Herrera, and Y. Fang, "3-D model-based multi-camera deployment: A recursive convex optimization approach," *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 6, pp. 3157–3169, Dec. 2015.

[6] P. Mantini and S. K. Shah, "Camera placement optimization conditioned on human behavior and 3D geometry," in *Proc. 11th Joint Conf. Comput. Vis., Imag. Comput. Graph. Theory Appl.*, 2016, pp. 227–237.

[7] P. Rahimian and J. K. Kearney, "Optimal camera placement for motion capture systems," *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 3, pp. 1209–1221, Mar. 2016.

[8] B. Bogaerts, S. Sels, S. Vanlanduit, and R. Penne, "Interactive camera network design using a virtual reality interface," *Sensors*, vol. 19, no. 5, p. 1003, Feb. 2019.

[9] E. Hörster and R. Lienhart, "On the optimal placement of multiple visual sensors," in *Proc. 4th ACM Int. Workshop Video Surveill. Sensor Netw.*, 2006, pp. 111–120.
[10] N. Kirchhof, "Optimal placement of multiple sensors for localization applications," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat.*, Oct. 2013, pp. 1–10.
[11] S. Boyd and J. Mattingley, "Branch and bound methods," Stanford Univ., Stanford, CA, USA, Tech. Rep., EE364b, 2007, pp. 2006–2007.
[12] F. Hoffmann, "On the rectilinear art gallery problem," in *International Colloquium on Automata, Languages, and Programming*. Cham, Switzerland: Springer, 1990, pp. 717–728.
[13] U. M. Erdem and S. Sclaroff, "Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements," *Comput. Vis. Image Understand.*, vol. 103, no. 3, pp. 156–169, 2006.
[14] J. Kritter, M. Brevilliers, J. Lepagnot, and L. Idoumghar, "On the real-world applicability of state-of-the-art algorithms for the optimal camera placement problem," in *Proc. 6th Int. Conf. Control, Decis. Inf. Technol. (CoDIT)*, Apr. 2019, pp. 1103–1108.
[15] Y. Morsly, N. Aouf, M. S. Djouadi, and M. Richardson, "Particle swarm optimization inspired probability algorithm for optimal camera network placement," *IEEE Sensors J.*, vol. 12, no. 5, pp. 1402–1412, May 2011.
[16] X. Wang, H. Zhang, S. Fan, and H. Gu, "Coverage control of sensor networks in IoT based on RPSO," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3521–3532, Apr. 2018.
[17] X. Wang, H. Zhang, and H. Gu, "Solving optimal camera placement problems in IoT using LH-RPSO," *IEEE Access*, vol. 8, pp. 40881–40891, 2019.
[18] P. Liu, Q. Hu, K. Jin, G. Yu, and Z. Tang, "Toward the energy-saving optimization of WLAN deployment in real 3-D environment: A hybrid swarm intelligent method," *IEEE Syst. J.*, early access, Apr. 5, 2021, doi: 10.1109/JSYST.2021.3065434.
[19] J. Zhao, R. Yoshida, S.-C.-S. Cheung, and D. Haws, "Approximate techniques in solving optimal camera placement problems," *Int. J. Distrib. Sensor Netw.*, vol. 9, no. 11, Nov. 2013, Art. no. 241913.
[20] M. Brévilliers, J. Lepagnot, L. Idoumghar, M. Rebai, and J. Kritter, "Hybrid differential evolution algorithms for the optimal camera placement problem," *J. Syst. Inf. Technol.*, vol. 20, no. 4, pp. 446–467, Nov. 2018.
[21] J.-W. Ahn, T.-W. Chang, S.-H. Lee, and Y. W. Seo, "Two-phase algorithm for optimal camera placement," *Sci. Program.*, vol. 2016, pp. 1–16, Sep. 2016.
[22] C. Gao, X. Yao, T. Weise, and J. Li, "An efficient local search heuristic with row weighting for the unicost set covering problem," *Eur. J. Oper. Res.*, vol. 246, no. 3, pp. 750–761, Nov. 2015.
[23] W. Lin, F. Ma, Z. Su, Q. Zhang, C. Li, and Z. Lü, "Weighting-based parallel local search for optimal camera placement and unicost set covering," in *Proc. Genetic Evol. Comput. Conf. Companion*, Jul. 2020, pp. 3–4.
[24] J. Kritter, M. Brévilliers, J. Lepagnot, and L. Idoumghar, "On the optimal placement of cameras for surveillance and the underlying set cover problem," *Appl. Soft Comput.*, vol. 74, pp. 133–153, Jan. 2019.
[25] J. Liu, S. Sridharan, and C. Fookes, "Recent advances in camera planning for large area surveillance: A comprehensive review," *ACM Comput. Surveys*, vol. 49, no. 1, pp. 1–37, Jul. 2016.
[26] M. Rebai, M. L. Berre, F. Hnaien, and H. Snoussi, "Exact biobjective optimization methods for camera coverage problem in three-dimensional areas," *IEEE Sensors J.*, vol. 16, no. 9, pp. 3323–3331, May 2016.
[27] A. Mavrinac and X. Chen, "Modeling coverage in camera networks: A survey," *Int. J. Comput. Vis.*, vol. 101, no. 1, pp. 205–226, 2013.
[28] A. Mavrinac, X. Chen, and Y. Tan, "Coverage quality and smoothness criteria for online view selection in a multi-camera network," *ACM Trans. Sensor Netw.*, vol. 10, no. 2, pp. 1–19, Jan. 2014.
[29] J. F. Thompson, Z. U. Warsi, and C. W. Mastin, *Numerical Grid Generation: Foundations and Applications*. Amsterdam, The Netherlands: Elsevier, 1985.
[30] C. Trader. *CGtrader*. Accessed: Aug. 10, 2021. [Online]. Available: https://www.cgtrader.com/
[31] F. S. Nooruddin and G. Turk, "Simplification and repair of polygonal models using volumetric techniques," *IEEE Trans. Vis. Comput. Graph.*, vol. 9, no. 2, pp. 191–205, Apr./Jun. 2003.
[32] P. Min. *BinVox*. Accessed: Feb. 5, 2021. [Online]. Available: http://www.patrickmin.com/binvox and https://www.google.com/search?q=binvox
[33] J. W. Harris and H. Stöcker, *Handbook of Mathematics and Computational Science*. Cham, Switzerland: Springer, 1998.
[34] S. J. Ray and J. Teizer, "Computing 3D blind spots of construction equipment: Implementation and evaluation of an automated measurement and visualization method utilizing range point cloud data," *Autom. Construct.*, vol. 36, pp. 95–107, Dec. 2013.
[35] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *ACM SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 163–169, Jul. 1987.
[36] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern Recognit.*, vol. 36, no. 2, pp. 451–461, Feb. 2003.
[37] T. Hastie, R. Tibshirani, and J. Friedman, "The elements of statistical learning: Data mining, inference, and prediction, 20 springer series in statistics,".
[38] K. Klasing, D. Wollherr, and M. Buss, "A clustering method for efficient segmentation of 3D laser data," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2008, pp. 4043–4048.
[39] H. Kisner and U. Thomas, "Segmentation of 3D point clouds using a new spectral clustering algorithm without a-priori knowledge," in *Proc. 13th Int. Joint Conf. Comput. Vis., Imag. Comput. Graph. Theory Appl.*, 2018, pp. 315–322.
[40] P. Scheunders, "A comparison of clustering algorithms applied to color image quantization," *Pattern Recognit. Lett.*, vol. 18, nos. 11–13, pp. 1379–1384, Nov. 1997.
[41] J. Papon, A. Abramov, M. Schoeler, and F. Worgotter, "Voxel cloud connectivity segmentation–supervoxels for point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2027–2034.
[42] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, May 2012.
[43] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 1–4.
[44] C. U. Manual, "Ibm ilog cplex optimization studio," *Version*, vol. 12, pp. 1987–2018, May 1987.

V. ANIRUDH PULIGANDLA (Member, IEEE) was born in Hyderabad, Telangana, India, in 1992. He received the B.Tech. degree in electronics and communications engineering from Amity University, Jaipur, Rajasthan, India, in 2014, the B.Sc. degree in computer vision and robotics from the University of Burgundy, France, in 2016, and the M.Sc. degree in computer vision and robotics from the University of Burgundy, the University of Girona, Spain, and Heriot Watt University, U.K., in 2018, as part of the Erasmus Mundus Joint Master's Degree Program. He is currently pursuing the Ph.D. degree with the University of Zagreb, Zagreb, Croatia, under the Marie-Curie Actions ITN Fellowship. His research interests include discrete and continuous optimization, signal and image processing, and 3D reconstruction from multiple camera systems using multi-view stereo.

SVEN LONČARIĆ (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Cincinnati, OH, USA, in 1994, as a Fulbright Scholar. From 2001 to 2003, he was an Assistant Professor at the New Jersey Institute of Technology, USA. He is currently a Professor of electrical engineering and computer science at the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia. He is the Director of the Center for Computer Vision, University of Zagreb; and the Head of the Image Processing Group. He is the Co-Director of the Center of Excellence in Data Science and Cooperative Systems. He was the principal investigator on a number of research and development projects. He has coauthored more than 250 publications in scientific journals and conferences. His research interests include image processing and computer vision. He is a member of the Croatian Academy of Technical Sciences. He has received several awards for his scientific and professional work. He was the Chair of the IEEE Croatia Section.

Publication 3

V.A. Puligandla, S. Lončarić, ”A Supervoxel Segmentation Method With Adaptive Centroid Initialization for Point Clouds”, *IEEE Access*, Vol. 10, 2022, pp. 98525-98534.

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 30 June 2022, accepted 6 September 2022, date of publication 12 September 2022, date of current version 22 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3206387

RESEARCH ARTICLE

A Supervoxel Segmentation Method With Adaptive Centroid Initialization for Point Clouds

V. ANIRUDH PULIGANDLA, (Member, IEEE), AND SVEN LONČARIĆ, (Senior Member, IEEE)
Image Processing Group, Department of Electronic Systems and Information Processing, Faculty of Electrical Engineering and Computing, University of Zagreb, 10000 Zagreb, Croatia

Corresponding author: V. Anirudh Puligandla (apuligandla@fer.hr)

This work was supported in part by the Immersive Visualization for Safety Critical Environments (ImmerSAFE) Project funded through the EU's H2020-MSCA-ITN-2017 Call under Project 764951, and in part by the Marie Sklodowska-Curie Actions–Innovative Training Networks (ITN) Funding Scheme.

ABSTRACT Supervoxels find applications as a pre-processing step in many image processing problems due to their ability to present a regional representation of points by correlating them into a set of clusters. Besides reducing the overall computational time for subsequent algorithms, the desirable properties in supervoxels are adherence to object boundaries and compactness. Existing supervoxel segmentation methods define the size of a supervoxel based on a user inputted resolution value. A fixed resolution results in poor performance in point clouds with non-uniform density. Whereas, other methods, in their quest for better boundary adherence, produce supervoxels with irregular shapes and elongated boundaries. In this article, we propose a new supervoxel segmentation method, based on k-means algorithm, with dynamic cluster seed initialization to ensure uniform distribution of cluster seeds in point clouds with variable densities. We also propose a new cluster seed initialization strategy, based on histogram binning of surface normals, for better boundary adherence. Our algorithm is parameter-free and gives equal importance to the color, spatial location and orientation of the points resulting in compact supervoxels with tight boundaries. We test the efficacy of our algorithm on a publicly available point cloud dataset consisting of 1449 pairs of indoor RGB-D images, i.e., color (RGB) images coupled with depth information (D) mapped per pixel. Results are compared against three state-of-the-art algorithms based on four quality metrics. Results show that our method provides significant improvement over other methods in the undersegmentation error and compactness metrics and, performs equally well in the boundary recall and contour density metrics.

INDEX TERMS Clustering methods, supervoxels, over-segmentation, point clouds.

I. INTRODUCTION

Like superpixels in 2D images, supervoxels are a collection of 3D points or pixels of a 3D image that are grouped together based on closeness between their spatial location and other textural features. For this work, we define supervoxels as disjoint clusters of points in a point cloud. Supervoxels represent regions in a point cloud that share common features, such as spatial location, color, and orientation. Subsequent computationally intensive image processing algorithms work on supervoxels instead of individual points or pixels to save computational time. Supervoxels find applications in various fields, such as point cloud segmentation and classification, [1], 3D semantic segmentation of point clouds, [2], [3], medical imaging, [4], [5], object detection, [6] and saliency detection, [7], to name a few. Despite so many applications, there is few literature that deals with clustering methods tailored for point clouds.

The desirable properties in a supervoxel include: (1) boundary adherence, i.e., a supervoxel should preserve object boundaries and should overlap with only one object and not cross over the boundaries, (2) compactness, i.e., supervoxels should have a regular shape and should not have elongated and arbitrary boundaries, and (3) efficiency, i.e., they should be computed fast enough not to decrease the efficiency of subsequent algorithms that use supervoxels.

The associate editor coordinating the review of this manuscript and approving it for publication was Joewono Widjaja.

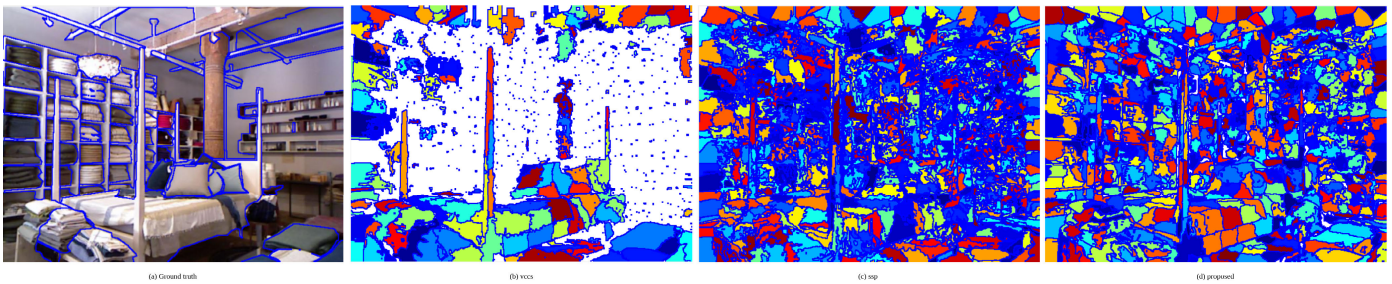


FIGURE 1. An example showing deficiencies in existing supervoxel segmentation methods and improvements in those aspects achieved by our proposed method.

Existing supervoxel segmentation methods fall short on some of these desirable aspects, especially, compactness. Fig. 1 shows an example of some of the shortcomings of existing methods. Fig. 1 shows a point cloud of a complex scene with many objects and high depth of field. Methods that rely on a constant user-inputted supervoxel resolution value fail when the point cloud density or depth varies steeply (see Fig. 1(b)), while other methods that adapt well to variable point density and can provide good boundary adherence for general point clouds, produce irregularly shaped supervoxels (see Fig. 1(c)). Non-compact supervoxels introduce spatial discontinuity and are not a desirable property in good supervoxels. Moreover, the artificially elongated boundaries may influence the values of comparison metrics to show more accuracy but do not look visually appealing.

In this work, we propose a new clustering method for colored point clouds. Our method is based on the k-means algorithm, like the Simple Linear Iterative Clustering (SLIC) algorithm by Achanta *et al.* [8]. The distance metric used in our method gives equal importance to color, points' spatial position and their orientation and is thus, free from implicit parameters. This property produces compact clusters with regular shapes. To maintain sensitivity to varying point density, we introduce a cluster seed re-initialization strategy to dynamically remove cluster seeds with very few assigned points or to add additional cluster seeds in regions with considerable number of unlabeled points. For better boundary adherence, we introduce a new strategy to first create a histogram of all points based on their surface normals and initialize cluster seeds according to the histogram bins. This strategy allows us to create cluster seeds in small, isolated regions or objects that may be missed otherwise when cluster seeds are distributed uniformly across the spatial extent. We tested the efficacy of our method on the publicly available NYU Depth V2 dataset, [9], and compared it against three state-of-the-art supervoxel segmentation methods based on four evaluation metrics.

Results show that our method performs best in terms of undersegmentation error and compactness metrics. While the performance of our method in the boundary recall metric is comparable to existing methods, we show that our method produces compact supervoxels with fine boundaries which make it look visually appealing than other methods. Although dynamic cluster seed initialization introduces additional computational overhead, the gained accuracy in terms of boundary adherence and compactness can be beneficial for algorithms that are not seriously restricted in time. We previously introduced this method in [10] where we used it to cluster a set of 3D points with surface normals representing camera poses on the surface of a vehicle's 3D model. It was introduced as a pre-processing step to reduce the input complexity of an optimal camera placement (OCP) problem for vehicle surround vision. The method showed promising results for the OCP problem by significantly reducing the overall computational time (up to 160 times). However, the supervoxel method was not analyzed as it was used as a pre-processing step. In this article, we present a detailed analysis of the method on colored point clouds to compare its efficacy against state-of-the-art supervoxel segmentation methods. The rest of the document is organized as follows: Section II details relevant literature, Section III details our proposed clustering method and the results are discussed in Section IV.

II. BACKGROUND WORK

Superpixels are 2D versions of supervoxels. While superpixels are extensively studied in the field of image processing, [8], [11], [12], [13], [14], supervoxels have not been studied enough despite their requirement due to recent advances in 3D image analysis. In the beginning, video sequences or stacks of 2D images collected over time were considered as 3D images. Therefore, the first 3D extensions of superpixel methods were tailored to deal with stacks of images, with time being the third dimension. [8], [14], [15] are some of the first supervoxels methods that extended their work to video sequences. Moore *et al.* [14] produced over-segmentation on videos by iteratively partitioning pixels into clusters by horizontal and vertical cutting in 3D grid. Achanta *et al.* [8] proposed an efficient and widely successful approach based on the k-means algorithm. In their method, they distribute cluster seeds uniformly across a 2D or 3D grid, search a local neighbourhood around each cluster seed and assign points to the closest cluster center based on a distance metric that relates pixels to a cluster center using position and color information. The primary idea behind the clustering method we propose here is based on this method.

In [16], Veksler *et al.*, proposed another supervoxel method for videos where they formulate it as an energy minimization problem and solve it using graph cuts. [17], [18], [19] are some of the pioneering works on supervoxel segmentation for RGB-D images. In [17], the authors extended their previous work on depth-adaptive superpixels to RGB-D videos. They used color and point normal information to construct a graph of spatio-temporal supervoxels and used spectral graph clustering to partition the graph into spatio-temporal segments. Gao *et al.* [18], proposed a new cluster seed initialization scheme for dense cluster seed initialization in salient regions of the image. Their motivation for adaptive cluster seed initialization is like ours and such a strategy works well to improve overall accuracy by producing clusters with non-uniform sizes and densities. Zhou *et al.* [20], used hierarchical edge weighted Voronoi tessellation to propose a multi-scale supervoxel algorithm that gradually constructs supervoxels at higher levels based on the supervoxels constructed at lower levels. The method that we propose here works on point clouds in 3D space. It is like other methods for RGB-D data only in the sense that we use mapped RGB-D to construct point clouds.

Papon *et al.* [21] proposed one of the first supervoxel segmentation methods (vccs) tailored for point clouds. They first voxelate the point cloud and cluster them based on voxel adjacency. They initialize cluster seeds uniformly across the voxelated point cloud and use the same distance metric as SLIC, [8]. They use voxel adjacency graphs to iteratively add neighbors to cluster seeds until all the voxels are assigned a label. Their method is simple and fast but, the voxel resolution parameter fails to adapt well to point clouds with variable density. Also, voxelization produces an approximation of the underlying points, thereby decreasing the quality of the method's boundary adherence. Our proposed method is like vccs in some aspects, but the primary difference is that our method works directly on the points. Lin *et al.* [22], more recently proposed a new supervoxel segmentation method for point clouds while citing the limitations of vccs. They formulate it as a subset selection problem based on an energy function that can be optimized to find optimal subsets. Their method does not require initialization of cluster seeds and is claimed to produce supervoxels with non-uniform resolution to better adapt to boundaries. Their method, however, produces irregularly shaped supervoxels with arbitrarily elongated boundaries (see Fig. 1(c)).

More recently, supervoxel methods for point clouds have garnered increased research interest. In [23], the authors propose modified versions of the vccs algorithm that are better suited for point clouds. In the method, they gather point neighbours without voxelization and combine neighbours computed by different methods to create supervoxels directly on the point cloud. Dong *et al.* [24] proposed a method that is capable of GPU acceleration. They divide the algorithm into two stages, where they produce an initial segmentation based on energy functions in the first stage and improve the result by minimizing segmentation energy in the second stage. Ni and Niu [25], proposed a new supervoxel segmentation method based on local allocation. They propose a novel cost function for preserving boundaries which is claimed to achieve satisfactory results through local minimization enforcement. Lastly, [26], [27] use deep learning methods for the learning geometrical features of point clouds and produce supervoxel segmentation. However, none of these recent methods emphasise on the compactness of supervoxels.

III. SUPERVOXEL SEGMENTATION FOR POINT CLOUDS

Our clustering algorithm is an iterative process like *k-means* algorithm. It is dynamic in nature as at each iteration, we allow for new clusters centers to be added and/or existing ones to be removed depending on the number of outlier points and the clusters' sizes. The algorithm works on a set of N points, $\mathcal{P} = \{p_1, \ldots, p_N\}$, where each point is represented by its position in 3D, (x, y, z), color in RGB space, (r, g, b), and a unit surface normal vector, (u, v, w), as,

$$p_i = [x_i\ y_i\ z_i\ r_i\ g_i\ b_i\ u_i\ v_i\ w_i]^T\ \forall i = 1 : N. \tag{1}$$

The goal is to group them into K disjoint subsets $\mathcal{S} = \{S_1, \ldots, S_K\}$, where each subset, S_k represents a supervoxel with the label k. At the end of the clustering process, it is expected that every point in $\mathcal{P}$ is assigned a label $k \in [0, \ldots, K]$, depending on which supervoxel the point belongs to. Each supervoxel is represented by its centroid, C_k, that is a 9D vector calculated as the mean of all points assigned to it. The points are assigned to a supervoxel based on a similarity metric, D, calculated as the Euclidean distance between a point p_i and a cluster center C_k. Supervoxel segmentation algorithms have individual strategies to tackle outlying points. At the end of all iterations of our algorithm, we assign a label $k = 0$ to the outlier points to mark them as *unlabeled*.

The algorithm requires one input parameter, i.e., the number of supervoxels, K. Our proposed algorithm differs from VCCS, [21], in three aspects: (1) for cluster center initialization, instead of uniformly sampling the point cloud we exploit the surface geometry to identify important regions in the point cloud, (2) instead of projecting the points into *lab* color space, we propose a method to use the similarity metric in the RGB color space, and (3) we allow to add or remove supervoxels dynamically to ensure that the entire point cloud is covered by the over-segmentation. The following sub-sections detail the individual steps of the algorithm, i.e., cluster center initialization in Section III-A and, assignment and update steps in Section III-B.

A. INITIALIZATION

We propose a novel approach to select initial cluster seeds based on points' orientation while ensuring that they are not initialized close to one another. Like the strategy used in [8], we assume that supervoxels are regular in shape and estimate the sidelength of each supervoxel as $S = \sqrt{\frac{N}{K}}$. Two cluster centers placed close to one another will have a significant overlap in their search spaces. This results in competition for

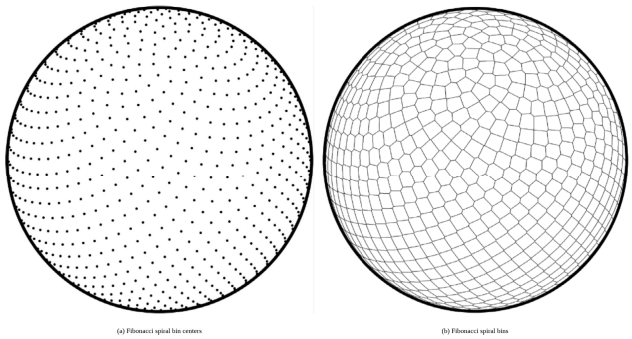


FIGURE 2. Fibonacci spiral bins on a unit sphere, [28].

the same set of points in every iteration and the algorithm may never converge. Therefore, we first voxelate the point cloud with a voxel sidelength of S and select K voxels as initial cluster seeds. In the process of voxelation, a uniform grid is placed over the point cloud and the value of each grid cell (voxel) is given as the average of all the points that lie within that cell. Points from cloud that are spatially closest to the K selected voxel centers are chosen as the initial cluster seeds. While methods like vccs, [21], select all the voxels in the voxelized point cloud as initial cluster seeds, we propose a new point orientation-based histogram binning to select only K important points as seeds from all the voxels. This strategy ensures that we do not overfit the data and create only K supervoxels as specified by the user.

To identify geometrically important regions, we construct a histogram of the voxels' surface normals using the Fibonacci spiral binning technique as described in [28]. Fibonacci spiral binning works by creating a Fibonacci spiral on the sphere from the north to the south pole with each bin location placed at equal increments along the spiral. Fig. 2(a) shows an illustration of bin centers initialized along the Fibonacci spiral on a sphere. An illustration of Fibonacci bins is shown in Fig. 2(b). Authors in [28] argue that Fibonacci spiral produces uniformly distributed bins around the sphere when compared with other binning techniques (e.g., equiangle grid). This binning technique requires that an odd number of bins must be created to have an equal number of bins in the two hemispheres. If we want to create $b_n = K$ number of bins, then the bin centers are given in spherical coordinates as,

$$B_{[\theta,\phi]}(d) = \left[sin^{-1}\left(\frac{2d}{b_n}\right) + \frac{\pi}{2}, \frac{2\pi}{\tau} mod(d, \tau)\right], \qquad (2)$$

where, θ and ϕ are the azimuthal and polar angles, respectively, $\tau = \frac{1+\sqrt{5}}{2}$ is the golden ratio, and $d \in \frac{1-b_n}{2}, \ldots, \frac{b_n-1}{2}$ is an integer used to represent the bin centers. The unit surface normals of all the voxels from the voxelated point cloud are first projected into spherical coordinates and then assigned to closest bin center by either a brute force approach or by a faster implementation as proposed in [28]. We use the implementation proposed in [28] as it is faster than the brute force approach especially when K is large.

For indoor scenes taken by one still camera (like the data used here), the surface normals lie within only one hemisphere as the normals point towards the camera. Therefore, for better binning accuracy in indoor point clouds, we create $b_n = 2K$ number of bins. After creating the histogram of surface normals, all the bin centers without any assigned normals are deleted and one voxel from each of the remaining bins (say we have b'_n bins with at least one assigned normal) is selected as a cluster seed. The remaining $K - b'_n$ cluster seeds are initialized at equal intervals in the remaining voxels across all bins. Through this strategy, we give more importance to the geometry of the scene than to the spatial distribution of points. Identifying important regions through binning of normals allows for greater representation of small distinct objects, while at the same time, there is a higher chance that large objects (e.g., walls) get multiple cluster seeds.

B. ALGORITHM

In the assignment step, the points p_i are assigned to the cluster center C_k, based on a distance metric D. D is computed as a combination of the Euclidean distances between the position, color, and normal vectors of a point p_i and a cluster center C_k. For a given point p_i and a cluster center C_k, $d_s = \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2 + (z_i - z_k)^2}$ is the distance between position vectors and $d_n = \sqrt{(u_i - u_k)^2 + (v_i - v_k)^2 + (w_i - w_k)^2}$ is the distance between normal vectors. In [29], the authors propose a novel low-cost approximation for calculating the distance between two colors in RGB space directly. They cite subjective experiments to claim that their proposed formulation overcomes limitations of LUV color space. Moreover, as general datasets have color information given in RGB space, computations to convert from RGB space to LUV space can be avoided through this non-linear distance metric. We propose to calculate the color distance between a point and a cluster center as $d_c = \sqrt{f_r \cdot (r_i - r_k)^2 + f_g \cdot (g_i - g_k)^2 + f_b \cdot (b_i - b_k)^2}$, where, $f_r = 2 + \frac{r_m}{256}, f_g = 4$, and $f_b = 2 + \frac{255 - r_m}{256}$ are weights for the respective colors, and $r_m = \frac{r_i + r_k}{2}$ is the mean of red color. The distance metric for comparing two points is then given as,

$$D = \sqrt{d_n^2 + \frac{1}{n_s} \cdot d_s^2 + \frac{1}{n_c} \cdot d_c^2}, \qquad (3)$$

where, n_s and n_c are normalization factors for spatial and color distances, respectively. Only the points lying inside a spherical neighbourhood of radius S are processed for each cluster center. This implies that a point can be at the most S units from its cluster center. Therefore, we set $n_s = S$. Similarly, as the range of color values range in [0, 256], we set $n_c = 256$. d_n does not require normalization as a point's orientation is given by unit surface normals.

The algorithm starts by going through each cluster center sequentially and searching a spherical neighbourhood of radius S around it. The encountered points are assigned to the cluster center if it is the smallest distance, D, the point

has seen so far. After this operation, the outlying points (unassigned points) are collected (Let us call the count of outlying points as N_{ol}). As these point clouds are not dense enough to be considered as volumes, we can ignore the depth dimension of the points and assume that the approximate size of each supervoxel to be $c_{size} = \frac{N}{K}$. This estimated size of a supervoxel is used later in the update step of the algorithm to estimate the number of cluster centers to be added. By keeping c_{size} constant throughout the algorithm, we can keep the final number of supervoxels close to the user-specified value K.

In the update step all the cluster centers with the number of assigned points less than 10% of c_{size} are removed. If K_r cluster centers are removed, the remaining $K - K_r$ cluster centers are updated as the mean of all the points, p_i, assigned to each. After updating the cluster centers, additional cluster centers are initialized in the set of unassigned points. A new point cloud is created from the set of unassigned points and new clusters are initialized in it following the same procedure as described in SectionIII-A. The number of cluster centers that need to be initialized from the unassigned points is given as $c_{add} = \frac{N_{ol}}{c_{size}}$. The process of dynamically adding and/or removing cluster centers results in different number of clusters (say K') from the user selected value K. The actual value of the difference $K' - K$ depends on the spatial distribution of the points. However, the process of dynamic cluster seed initialization has two advantages: (1) small but geometrically distinct regions get their own cluster seeds, and (2) outlier points do not get incorrectly assigned to any clusters.

The assignment and update steps are repeated iteratively until, either the number of outliying points changes by less than 10% from the previous iteration, or if there are no more cluster centers required to be added. For example, when $N_{ol} < c_{size}$, c_{add}, as an integer division, becomes zero, implying that no additional cluster centers can be initialized. Our experiments show that the algorithm usually runs until there is no possibility of adding any more clusters. Hence, when the stopping criteria is satisfied, it is only the outlier points that remain unassigned. As a last step, the K' cluster centers are updated as the mean of all points assigned to each of them. By leaving the outlier points as unassigned, our algorithm achieves better accuracy and object boundary adherence as most datasets consist of a category of unlabeled points. The complete algorithm is detailed in Algorithm 1. The clustering algorithm has a time complexity of $\mathcal{O}(N)$.

Algorithm 1 Clustering Based on Point Orientation

Input: K,$p_i = [x_i\ y_i\ z_i\ r_i\ g_i\ b_i\ u_i\ v_i\ w_i]^T\ \forall\ i = 1 : N$;
Result: $labels_i\ \forall\ i = 1 : N$, $C_k\ \forall\ k = 1 : K'$
$S = \sqrt{\frac{N}{K}}$;
Initialize: $C_k = [x_k\ y_k\ z_k\ r_k\ g_k\ b_k\ u_k\ v_k\ w_k]^T\ \forall$
$k = 1 : K$;
$D_i = \inf$, $labels_i = -1\ \forall\ i = 1 : N$;
$K' = K$, $t = 0$, $N_{ol}(t) = N$, $c_{size} = \frac{N}{K}$;
while *(1)* **do**
  for $k = 1$ **to** K' **do**
    for p_i *in neighbourhood of radius* S **do**
      D = D(p_i, C_k) as in equation 3;
      if $D < D_i$ **then**
        $D_i = D$;
        $labels_i = k$;
      end
    end
  end
  outliers = collect points with label == -1;
  $N_{ol}(t+1)$ = size(outliers);
  remove all centers with $size(C_k) < 0.1 \times c_{size}$;
  Re-estimate C_k as mean of all p_i with $labels_i == k$;
  $c_{add} = \frac{N_{ol}(t+1)}{c_{size}}$;
  if $(\frac{N_{ol}(t) - N_{ol}(t+1)}{N_{ol}(t)} < 0.1)\ ||\ (c_{add} < 1)$ **then**
    break;
  end
  Initialize c_{add} clusters and append to C_k;
  $K' = K$ after adding and/or removing clusters;
  $t = t + 1$;
end

IV. RESULTS

A. PARAMETERS AND METRICS

We test the proposed clustering method's efficacy on the openly available NYU-V2 Depth dataset, [9]. The dataset consists of 1449 densely labelled pairs of aligned RGB and depth images. The aligned depth information was mapped to corresponding pixels to obtain 3D point clouds with color information in RGB space. The proposed method is compared against three state-of-the-art clustering algorithms: (1) the original voxel cloud connectivity segmentation (vccs) method that works on voxelated point clouds, [21], (2) a supervoxel segmentation method framed as a subset selection problem (ssp), [22], and (3) a K-nearest-neighbours version of vccs method (vccs-knn) that works directly on the point clouds without voxelation, provided by the authors in [22]. The vccs method is available as part of PCL (Point Cloud Library), [30], and it was tested using the default parameter settings. Voxel resolution for VCCS method was set at 0.1m for all experiments. The ssp and vccs-knn methods were tested using their openly available source code[1] with the parameters for both methods set to the values as proposed in their article, [22]. As the methods are tested on the same dataset, the default parameters must be set to produce the best results. While the vccs, ssp and vccs-knn methods implicitly compute the surface normals for the point clouds, for our method they were computed using the standard nearest-neighbours-based method provided by PCL with number of neighbours equal to 30. Our method requires only one input parameter, i.e., the number of clusters K. All the experiments

[1] https://github.com/yblin/Supervoxel-for-3D-point-clouds.git

were run on a computer with an Intel Core i7-8700K CPU and 16GB of RAM. Our method is coded in C++ programming language.

We compare the performance of the algorithms using four evaluation metrics. For evaluations, we first represent the labelled point clouds as 2D labeled images and compare them against the labeled 2D ground truth images. Boundary recall (R) measures the fraction of ground truth boundaries that fall within a distance ϵ of at least one estimated supervoxel boundary. We follow the definition of boundary recall as proposed in [31]. Given a ground truth boundary image G and an estimated boundary image B, R is computed as the fraction of true positives (TP) and the sum of true positives and true negatives (TN), i.e., $R = \frac{TP}{TP+TN}$, where TP are defined as the number of boundary pixels in G for whose exist a boundary pixel in B in range ϵ, and TN as the number of boundary pixels in G for whose do not exist a boundary pixel in B in range ϵ. In this article, we use $\epsilon = 2$ pixels. An R value of 1 reflects best performance indicating the methods precision in identifying object boundaries whereas, a value of 0 reflects otherwise. The second metric we use is the undersegmentation error (UE). According to [31], UE measure to what extent estimated segmentation boundaries cross-over the ground truth boundaries. Generally, the number of pixels of a segment that cross over the boundary are measured. However, this method imposes a high penalty on large supervoxels with only a small overlap with the ground truth segment. To avoid this, In [31], they propose a new method where the smaller value of either the region that crosses over the boundary or the region that lies within the segment is counted depending on whichever is smaller. It is defined as,

$$UE = \frac{1}{N}\left[\sum_{S\in GT}\left(\sum_{P:P\cap S\neq 0} min(P_{in}, P_{out})\right)\right] \tag{4}$$

where, S are the ground truth (GT) segments, P are the estimated segments, N is the total number of pixels, P_{in} is the part of the estimated segment that lies within S and P_{out} is the part of the estimated segment that crosses over the ground truth segment's boundary. A UE value of 0 implies that the method has best adherence to object or segment boundaries whereas, UE = 1 indicates otherwise.

The third metric we use is the compactness (C) of the supervoxels. In mathematics, compactness of a shape is commonly measured through the isoperimetric quotient which compares the area of a shape to the area of a circle with the same perimeter as this shape, [32]. If A_P is the area and L_P is the perimeter of a superpixel (supervoxel projected in 2D), P, then the radius of a circle with the same perimeter as P is given as, $r = \frac{L_P}{2\pi}$. If A_S is the area of the circle with radius r, then the isoperimetric quotient is given as,

$$Q_P = \frac{A_P}{A_S} = \frac{4\pi A_P}{L_P^2}. \tag{5}$$

Therefore, if I is the set of all segments in a segmented image, then the compactness measure is given as,

$$C = \sum_{P\in I} Q_P \cdot \frac{|P|}{N} \tag{6}$$

where, $|P|$ is the size of the segment and N is the total number of pixels in the image (or points in the point cloud). $C = 1$ implies that the estimated segments are perfect circles whereas, $C = 0$ implies that the segments have highly irregular and non-convex shapes. Lastly, we also compare the algorithms based on the contour density (CD) metric, [11]. CD measures the fraction of boundary pixels in the segmentation image. Given a set of boundary pixels, B, of an estimated segmentation contour density is defined as, CD= $\frac{|B|}{2N}$, where N is the total number of pixels. The fraction is divided by two because computation of segment boundaries produces edges that are two-pixels wide. The contour density metric also indicates regularity of the boundaries as higher values of CD mean that there exist more number of boundary pixels for the same number of supervoxels. Higher values of CD indicate that the object boundaries are irregular and elongated.

B. EVALUATION

All the above-mentioned algorithms were tested on the NYU V2 Depth dataset and compared using the above-mentioned metrics. While supervoxel resolution as a measure has geometrical significance, we believe that the number of clusters is easier to interpret for a general user. The complexities of all algorithms is given in terms of the input size. As a result, a user can have better control on estimating the complexity of subsequent algorithms which would be used on the segmented point cloud when there is a direct control on the number of supervoxels that need to be produced in a point cloud over-segmentation. Moreover, it is important to note that each algorithm produces a different number of supervoxels for a point cloud at any given supervoxel resolution. In all cases, ssp method produces the highest number of supervoxels between all the algorithms for any given resolution. It is well known that as the number of supervoxels increases the performance in terms of metrics also increases. Due to this reason, as one method produces more supervoxels than another for the same segmentation, comparison by the supervoxel resolution becomes an unfair comparison for methods that produce fewer number of supervoxels. Therefore, we choose to compare results based on the number of supervoxels produced.

We tested the vccs, ssp and vccs-knn algorithms at supervoxel resolutions (in meters) 0.10, 0.11, 0.12, 0.13, 0.14, 0.16, 0.18, 0.20, 0.25 and 0.35. To keep the number of supervoxels produced by our method in similar range as other methods, we set the output number from vccs method as the input number of clusters for our method. Although the number of clusters produced by our method is dynamic, it is usually within a range of ± 100 clusters from the user-chosen number K. To maintain uniformity in comparison, for each experiment, we round the output number of clusters to the

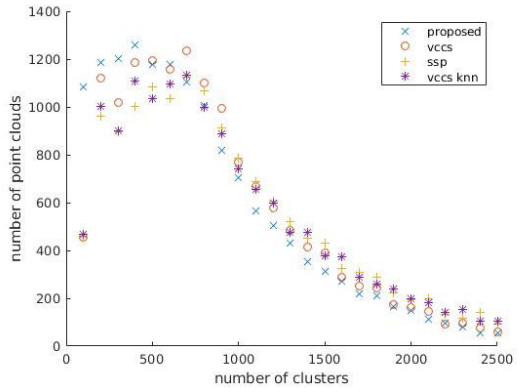


FIGURE 3. Histogram distribution of number of point clouds for which results were obtained, binned according to the number of clusters.

nearest 100 and order the results according to these multiples. We believe that this strategy allows for a fair comparison as results from some point clouds may get rounded to higher multiple of hundred while a similar number of point cloud results may get rounded to a lower multiple of hundred. As the number of supervoxels found varies with the method, density, or spatial spread of the point cloud, and the supervoxel resolution, it is not possible to have a control on how many point clouds produce results for a given number of clusters. A histogram of the number of point clouds, ordered by the cluster entries for which results were obtained, is shown in Fig.3. We present results only when there exist at least 30 point clouds contributing to the results for a given number of clusters entry.

Quantitative results of the four metrics obtained from testing the four methods on the entire 1449 image pairs of the dataset are shown in Fig.4. Overall, our proposed method performs best in the undersegmentation error and compactness metrics while the ssp and vccs methods perform best in the boundary recall and contour density metrics, respectively. The ssp method and our proposed clustering method perform similarly well in R and UE metrics. With a relative difference of 0.0086 between both the methods' overall mean R for all the tests, the ssp method performs marginally better than our method in the boundary recall metric. Whereas, in the UE metric, our method is relatively 3.17% better than ssp method. Moreover, a student's t-test on the results from our method and ssp shows that the means of UE of our method are significantly better than those of ssp, with a $p - value = 1.32e - 6$ and a $t - value = 6.38$ in the confidence interval $CI = [0.0070, 0.0136]$. As evident from Fig. 4(a), both these methods show significantly better boundary adherence (significantly better performance in both R and UE metrics) than both the variants of vccs method. The major advantage of our method lies in the regularly shaped supervoxels that it produces. The plot in Fig. 4(c) shows that our method significantly outperforms the three other methods in terms of producing most regularly shaped and compact supervoxels. The ssp method performs worst in this metric as their method produces supervoxels with highly irregular boundaries. This quality of ssp method is also reflected in the CD metric (see Fig. 4), where ssp method performs worst out of all the methods. Large values of CD for ssp method, or a large number of boundary pixels for a given number of supervoxels, reflects the irregularly shaped supervoxels produced by it.

It is to be noted that a higher number of boundary pixels may result in artificially inflated values of R as there is a greater chance that a segmentation boundary lies in close distance to a given ground truth boundary. Therefore, it is possible that the ssp method may not produce results that are visually as appealing as reflected by their quantitative analysis values. The same can be verified from Fig. 5, where it can be seen that although the supervoxels produced by ssp method agree to object boundaries to an extent, the resulting supervoxels are irregular in shape. Irregularly shaped supervoxels are undesirable for subsequent applications as it introduces spatial discontinuity within the segmentation. With the least CD values of all methods, the vccs method can be expected to produce the most compact supervoxels. Although it produces more regular shaped supervoxels than ssp and vccs-knn methods, our method outperforms it because the vccs method fails on noisy point clouds or on point clouds with low spatial density (or high variation in depth). The same can be verified visually from the second row of Fig. 5(b), where there exist empty regions (seen in white) and small isolated supervoxels in the segmentation. While missing regions or small and isolated supervoxels do not contribute to the CD, they impose a serious penalty on the compactness metric, thereby resulting in lower values of C for vccs method.

Fig. 5 shows visualizations of segmentations from the four methods on some example point clouds. In general, our method produces visually appealing segmentations with compact and regularly shaped supervoxels that addhere to object boundaries well. An exception where our method fails to produce compact supervoxels can be seen in the second row of Fig. 5. The mesh doors at vicinity of the viewpoint act as noise in that region of the point cloud as they appear as scattered points at a different depth than the background, resulting in irregular supervoxels. The superior compactness of supervoxels produced by our method is visible in the fourth row of Fig. 5. The highlighted region consists of planar regions with very few objects. Yet, all methods except ours produced supervoxels with arbitrary shapes in that region. The shown segmentations are for 500, 700, 2000 and 1100 supervoxels for the first to fourth rows, respectively. The result for vccs-knn method in the third row of the figure consists of only 800 supervoxels as the method produced only those many at a resolution of 0.1m, while the rest of the methods produced about 2000 supervoxels. Finally, it can be said that this figure presents an accurate visual reiteration of the results shown as part of quantitative analysis. Visualizations show that the vccs method fails in the case of point clouds with complex scenes and when the point clouds

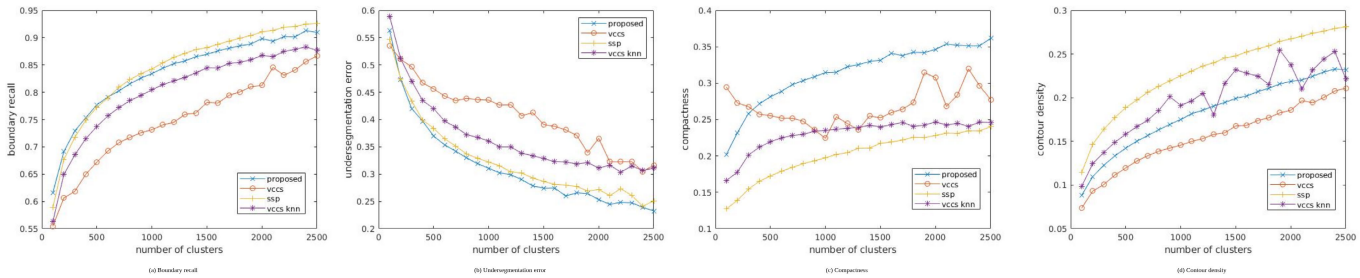


FIGURE 4. Quantitative analysis of the four methods on NYU Depth V2 dataset.

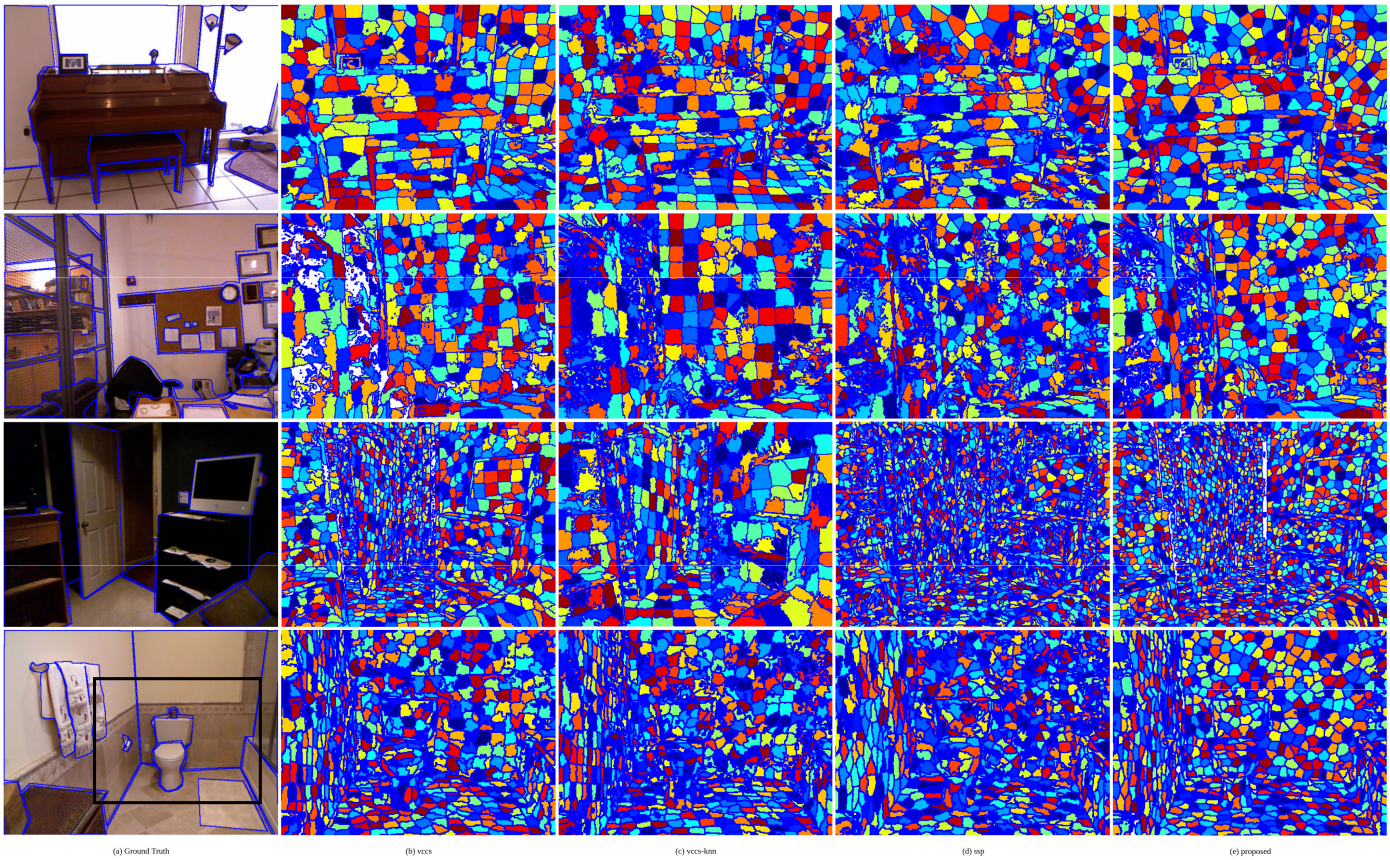


FIGURE 5. Visual representation of segmentation results on NYU Depth V2 dataset.

have varying depth and points density. The vccs-knn method produces supervoxels with irregular shapes, while the ssp method also produces irregularly shaped suypervoxels with elongated boundaries but with greater accuracy. Our method produces compact supervoxels with clear and tight boundaries that addhere well to the scene, for a range of supervoxel resolutions (100 supervoxels to over 2000 supervoxels) with some exceptions as shown for the point cloud in the second row of 5.

A comparison of the total running times by vccs, ssp and our method are shown in Fig. 6. In terms computational time, the vccs method proves its simplicity and robustness as it performs segmentation in about 1s. While all methods have constant complexity, our method, with a complexity of $\mathcal{O}(N)$, requires significantly longer computational time when compared against the other methods. This is expected as our algorithm involves the assignment step for all the points in every iteration. The step of collecting unassigned points and initializing and/or removing cluster centers adds additional computational overhead. However, our proposed initialization procedure based on Fibonacci binning produces accurate initial seeds. The algorithm typically converged in $6-8$ iterations in all cases. While the overall complexity of our algorithm seems big when compared to other methods, the gained accuracy and quality of segmentation may be beneficial to applications that do not have serious limitations

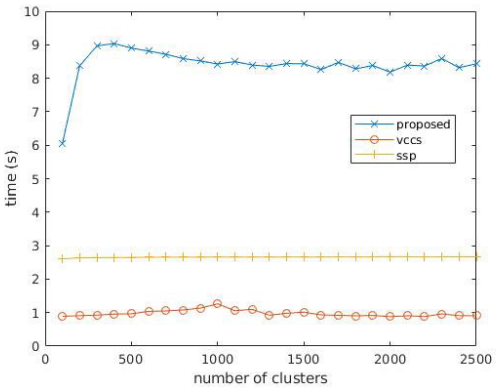


FIGURE 6. Time taken for segmentation by the vccs, ssp and our proposed methods on NYU Depth V2 dataset.

on computational time but, which may benefit from accurate and compact supervoxel segmentation. Performing the assignment step only on the points at supervoxel boundaries or maintaining a database of initially unassigned points and working only on that set of points iteratively, may benefit the algorithm in terms of speed.

V. CONCLUSION AND FUTURE WORK

We proposed a new supervoxel segmentation method with dynamic cluster seed initialization. Our method inherits the advantages of the k-means algorithm. Coupled with a novel cluster seed initialization strategy based on Fibonacci binning of surface normals, the method achieves superior boundary adherence when compared against existing state-of-the-art methods. As the algorithm is parameter-free, it gives equal importance to the spatial location, color, and surface normals of all points to produce regularly shaped compact supervoxels with tight boundaries. Quantitative analysis using four metrics on the publicly available NYU Depth V2 dataset shows that our method performs equally good as the ssp and vccs methods in the boundary recall and contour density metrics, respectively. Our proposed method shows significantly better performance than all other methods in the undersegmentation error and compactness metrics. Visual representations of segmentation results on some of the point clouds show that our method produces visually appealing supervoxels with a high degree of compactness that adhere well to object boundaries.

However, the added accuracy of our proposed algorithm comes at the cost of increased complexity. For future work, we propose to explore the possibility of reducing the overall computational time of our algorithm. Calculating the distance metric in the assignment step for only the points at supervoxel boundaries and maintaining a database of unassigned points and working iteratively only on that set of points can be approaches to improve the time complexity of our algorithm. Another improvement to decrease the number of calculations per iteration could be to stop the assignment step after meeting a certain criterion. An example criterion could be to track the change in cluster seeds based on the $\mathcal{L}^1$-norm between cluster centers at current and previous iterations and stop the assignment step when the norm becomes lower than threshold.

REFERENCES

[1] Y. Xu, Z. Ye, W. Yao, R. Huang, X. Tong, L. Hoegner, and U. Stilla, "Classification of LiDAR point clouds using supervoxel-based detrended feature and perception-weighted graphical model," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 72–88, 2020.

[2] H. Luo, K. Khoshelham, L. Fang, and C. Chen, "Unsupervised scene adaptation for semantic segmentation of urban mobile laser scanning point clouds," *ISPRS J. Photogramm. Remote Sens.*, vol. 169, pp. 253–267, Nov. 2020.

[3] S.-S. Huang, Z.-Y. Ma, T.-J. Mu, H. Fu, and S.-M. Hu, "Supervoxel convolution for online 3D semantic segmentation," *ACM Trans. Graph.*, vol. 40, no. 3, pp. 1–15, Jun. 2021.

[4] J. Huo, J. Wu, J. Cao, and G. Wang, "Supervoxel based method for multi-atlas segmentation of brain MR images," *NeuroImage*, vol. 175, pp. 201–214, Jul. 2018.

[5] S. Hansen, S. Kuttner, M. Kampffmeyer, T.-V. Markussen, R. Sundset, S. K. Øen, L. Eikenes, and R. Jenssen, "Unsupervised supervoxel-based lung tumor segmentation across patient scans in hybrid PET/MRI," *Expert Syst. Appl.*, vol. 167, Apr. 2021, Art. no. 114244.

[6] H. Guan, Y. Yu, J. Li, and P. Liu, "Pole-like road object detection in mobile LiDAR data via supervoxel and bag-of-contextual-visual-words representation," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 4, pp. 520–524, Apr. 2016.

[7] J.-S. Yun and J.-Y. Sim, "Supervoxel-based saliency detection for large-scale colored 3D point clouds," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 4062–4066.

[8] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2011.

[9] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. ECCV*, 2012, pp. 746–760.

[10] V. A. Puligandla and S. Lončarić, "A multiresolution approach for large real-world camera placement optimization problems," *IEEE Access*, vol. 10, pp. 61601–61616, 2022.

[11] V. Machairas, M. Faessel, D. Cárdenas-Peña, T. Chabardes, T. Walter, and E. Decenciére, "Waterpixels," *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 3707–3716, Nov. 2015.

[12] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi, "TurboPixels: Fast superpixels using geometric flows," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2290–2297, Dec. 2009.

[13] M. Van den Bergh, X. Boix, G. Roig, B. D. Capitani, and L. Van Gool, "Seeds: Superpixels extracted via energy-driven sampling," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2012, pp. 13–26.

[14] A. P. Moore, S. J. D. Prince, J. Warrell, U. Mohammed, and G. Jones, "Superpixel lattices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.

[15] C. Xu and J. J. Corso, "Evaluation of super-voxel methods for early video processing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 1202–1209.

[16] O. Veksler, Y. Boykov, and P. Mehrani, "Superpixels and supervoxels in an energy optimization framework," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2010, pp. 211–224.

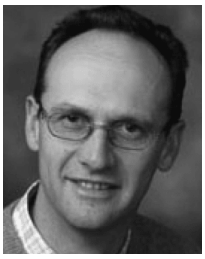
[17] D. Weikersdorfer, A. Schick, and D. Cremers, "Depth-adaptive supervoxels for RGB-D video segmentation," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2013, pp. 2708–2712.

[18] G. Gao, M. Lauri, J. Zhang, and S. Frintrop, "Saliency-guided adaptive seeding for supervoxel segmentation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 4938–4943.

[19] P. Xu, J. Li, J. Yue, and X. Yuan, "Scale adaptive supervoxel segmentation of RGB-D image," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2016, pp. 1303–1308.

[20] Y. Zhou, L. Ju, and S. Wang, "Multiscale superpixels and supervoxels based on hierarchical edge-weighted centroidal Voronoi tessellation," *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 3834–3845, Nov. 2015.
[21] J. Papon, A. Abramov, M. Schoeler, and F. Worgotter, "Voxel cloud connectivity segmentation—Supervoxels for point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2027–2034.
[22] Y. Lin, W. Cheng, D. Zhai, L. Wei, and J. Li, "Toward better boundary preserved supervoxel segmentation for 3D point clouds," *ISPRS J. Photogramm. Remote Sens.*, vol. 143, pp. 39–47, Sep. 2018.
[23] Z. Sha, Q. Zhu, Y. Chen, C. Wang, A. Nurunnabi, and J. Li, "A boundary-enhanced supervoxel method for 3D point clouds," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Sep. 2020, pp. 2771–2774.
[24] X. Dong, Y. Xiao, Z. Chen, J. Yao, and X. Guo, "GPU-based supervoxel segmentation for 3D point clouds," *Comput. Aided Geometric Des.*, vol. 93, Feb. 2022, Art. no. 102080.
[25] H. Ni and X. Niu, "SVLA: A compact supervoxel segmentation method based on local allocation," *ISPRS J. Photogramm. Remote Sens.*, vol. 163, pp. 300–311, May 2020.
[26] L. Landrieu and M. Boussaha, "Point cloud oversegmentation with graph-structured deep metric learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7440–7449.
[27] L. Hui, J. Yuan, M. Cheng, J. Xie, X. Zhang, and J. Yang, "Superpoint network for point cloud oversegmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 5510–5519.
[28] R. L. Larkins, M. J. Cree, and A. A. Dorrington, "Analysis of binning of normals for spherical harmonic cross-correlation," *Proc. SPIE*, vol. 8290, Jan. 2012, Art. no. 82900L.
[29] T. Riemersma. (2019). *Color Metric*. [Online]. Available: https://www.compuphase.com/cmetric.htm
[30] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 1–4.
[31] F. P. León, *Forum Bildverarbeitung 2012*. Karlsruhe, Germany: KIT Scientific Publishing, 2012.
[32] A. Schick, M. Fischer, and R. Stiefelhagen, "Measuring and evaluating the compactness of superpixels," in *Proc. 21st Int. Conf. Pattern Recognit. (ICPR)*, 2012, pp. 930–934.

V. ANIRUDH PULIGANDLA (Member, IEEE) was born in Hyderabad, Telangana, India, in 1992. He received the B.Tech. degree in electronics and communications engineering from Amity University, Jaipur, Rajasthan, India, in 2014, the B.Sc. degree in computer vision and robotics from the University of Burgundy, France, in 2016, and the M.Sc. degree in computer vision and robotics from the University of Burgundy, France; University of Girona, Spain; and Heriot Watt University, U.K., in 2018, as part of an Erasmus Mundus Joint Master's Degree Program. He is currently pursuing the Ph.D. degree with the University of Zagreb, Zagreb, Croatia, under a Marie-Curie Actions ITN Fellowship. His research interests include discrete and continuous optimization, signal and image processing, and 3D reconstruction from multiple camera systems using multi-view stereo.



SVEN LONČARIĆ (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Cincinnati, Cincinnati, OH, USA, in 1994. From 2001 to 2003, he was an Assistant Professor at the New Jersey Institute of Technology, USA. He was a Fulbright Scholar. He is currently a Professor of electrical engineering and computer science at the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia. He is also the Director of the Center for Computer Vision, University of Zagreb, and the Head of the Image Processing Laboratory. He is the Co-Director of the Center of Excellence in Data Science and Cooperative Systems. He was the principal investigator on a number of research and development projects. He coauthored more than 250 publications in scientific journals and conferences. His research interests include image processing and computer vision. He is a member of the Croatian Academy of Sciences and Arts. He received several awards for his scientific and professional work.

Publication 4

V.A. Puligandla, S. Lončarić, ”A Continuous Camera Placement Optimization Model For Surround View”, *IEEE Transactions on Intelligent Vehicles*, 2023, Jul 26, doi: 10.1109/TIV.2023.3299199

A Continuous Camera Placement Optimization Model For Surround View

V. Anirudh Puligandla, *Member, IEEE*, Sven Lončarić, *Senior Member, IEEE*

***Abstract*—Increasing use of Advanced Driver Assistance Systems (ADAS) in autonomous vehicles is rising the demand for advanced perception systems. With more sensors being placed on the vehicle than ever, a need arises to optimize the placement of the sensors on the vehicle's body to maximize coverage at minimal cost. Camera Placement Optimization (CPO) methods tailored for multi-camera networks to maximize a vehicle's surrounding view coverage are limited. While existing CPO methods tend to sample the simulation space to work in the discrete domain using integer programming-based problem formulation, this article proposes a novel approach to optimize for camera poses for vehicle surround-view in the continuous domain using gradient-free blackbox optimization techniques by defining a non-linear objective function. Experimental results on more than 100 instances of real world 3D models of vehicles of various shapes and sizes show that the proposed method is effective in maximizing coverage for vehicle surround-view by a multiple camera network in a reasonable amount of time. Comparisons against a discrete CPO formulation show that the proposed method significantly improves coverage accuracy by optimizing poses of multiple cameras for vehicle surround-view.**

***Index Terms*—Optimal camera placement, global optimization, black-box optimization, surround-view vision, autonomous driving**

I. Introduction

SENSOR placement optimization (SPO) finds applications in a variety of industries such as identifying defects in structures, [1], [2], drone navigation, [3], 3D reconstruction, [4], IoT and wireless networks, [5], and surveillance, [6]. Camera placement optimization (CPO) is a sub-category of this vast field that deals with placement optimization of visual sensors. Most CPO methods focus on surveillance applications, [7], [8]. Few works address the problem of optimizing sensor poses on vehicles for applications in autonomous driving, [9], [10]. Advanced Driver Assistance Systems (ADAS) to provide surrounding-view/top-view/rear-view visualization are finding uses in vehicles, [11]. Besides providing new perspectives of the surrounding environment to the vehicle operators, ADAS systems are capable of eliminating fatal accident-causing blind spots, particularly for larger and heavier vehicles. A typical surround-view system consists of at least four fish-eye lens cameras with additional lidar/ultrasonic sensors, [12]. As the cost increases with the increasing number of sensors for larger vehicles, sensor placement optimization is necessary to increase efficiency of the multi-camera network and to keep the costs low.

Our work to estimate camera poses in 3D may aid in bypassing the camera calibration step as camera pose estimation forms the basis of *camera calibration*, [13]. A common and widely used approach to model CPO problems is the use of Binary Integer Programming (BIP) formulation in a discretized space model. The simulation space for the problem is sampled to collect sets of points that approximate the underlying surface (floor plans for e.g.). These samples are modelled as the set of possible locations for placing cameras and the set of points that need to be covered by the optimally placed cameras. Continuous space models are avoided due to the complexity of computing camera coverage through geometrical intersections (surface-surface, surface-volume) in a reasonable amount of time. However, discrete space models have limitations in terms of approximation accuracy and scalability of the problem. To accurately approximate the underlying surface, sampling more points is required which increases the complexity of the optimization algorithm. Sampling accuracy is important for surround-view CPO problems where a small deviation in the camera's orientation may result in artefacts during 360^o video stitching and 3D reconstruction.

We propose a new method to optimize camera poses directly on the vehicle's surface in the continuous space. Optimization in the continuous space achieves sub-sample accuracy and ensures that crucial intermediate values between samples are not missed when searching for the poses of the optimal multi-camera network. Our proposed method approximates the non-convex surface of 3D vehicle models by fitting a hollow hemisphere around the 3D model and optimizing for camera poses on the surface of the hemisphere. By using spherical coordinates, a camera's position on the hemisphere can be represented using only two variables, i.e., the polar and azimuthal angles. The camera's orientation is represented using a unit *view*-direction (or *look-at* direction) vector which is estimated using only one variable in our work. By using our proposed method, a camera's pose can be optimized using only 3 variables instead of thousands as in the case of discrete CPO problem formulations. Camera coverage is modelled using discrete points sampled from the ground plane around the vehicle's 3D model and a user-defined number of cameras' poses are optimized to maximize overall coverage.

For calculating coverage, we propose a non-linear objective function that can be solved using *gradient-free* and *blackbox*

This work is supported by the ImmerSAFE (Project number 764951) project funded under the EU's H2020-MSCA-ITN-2017 call and is part of the Marie Sklodowska-Curie Actions - Innovative Training Networks (ITN) funding scheme.

V. Anirudh Puligandla and Sven Lončarić are with Image Processing Group, Department of Electronic Systems and Information Processing, Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia (e-mail: apuligandla@fer.hr; sven.loncaric@fer.hr)

Manuscript received April 19, 2021; revised August 16, 2021.

optimization methods, [14]. As it is desired to place cameras on the surface of the vehicle, we propose a ray-tracing-based approach to project a camera's location from the surface of hemisphere to the vehicle's 3D model. While existing methods use camera Field of View (FoV) models based on geometrical shapes such as, circular sectors, pyramids, etc., we propose a novel method to use the camera projection matrix to estimate coverage of the surrounding points. We test the proposed continuous CPO problem on 3D models of vehicles with varying shapes and sizes using two approximate optimization algorithms namely, particle swarm optimization (PSO) and Bayesian optimization (BO). Comparison of our proposed continuous space model against a discrete CPO problem formulation shows that our proposed method performs significantly better in terms of coverage without any significant overhead in computational complexity. The rest of the article is organized as follows: Section II introduces relevant CPO problems and optimization methods; Section III introduces the proposed continuous CPO problem formulation; and Section IV presents an analysis of the proposed method and a comparison against a similar discrete CPO model.

II. Background Work

Pioneering works in CPO can be attributed to [15], [16]. Numerous CPO methods have been proposed for various use-cases, in the past two decades, with most of them focusing on multiple-camera networks for surveillance, [17]. Despite the availability of a vast amount of literature on CPO problems, few address the problem of camera placement optimization for vehicle surround-view vision. In [9], Indu et al. proposed a multi-camera optimization model designed specifically for driver-less cars. Their model works in discrete space with a weighted coverage function to differentiate between critical and non-critical areas in the vehicle's surrounding region. They use the camera's intrinsic and extrinsic parameter matrices like our method but, they construct a view frustum from the parameters and use the geometrical frustum model to construct visibility and coverage matrices as part of a standard Mixed Integer Programming (MIP)-based discrete optimization model. They solved the optimization problem using heuristic algorithms such as PSO and grey wolf optimization.

The authors in [10], proposed an optimization framework to place a combination of distinct types of sensors (not only visual sensors) as part of an ADAS system. They also model the problem in discrete space by first identifying 10 zones around the model of a car. A particular type of sensor's pose is optimized in only its designated zones. This approach of dividing the search space into zones simplifies the SPO problem, but the use of discretized space model limits its accuracy. As the orientation changes in steps of 1^o, important poses might be missed between two steps. A similar problem of SPO for autonomous vehicles using only lidar sensors is proposed in [18]. They place a set of vertical *lidar occupancy boards* around the vehicle and measure coverage by placing a grid on the board and counting the number of cells covered by the lidar's FoV. Their optimization problem is formulated in discrete space and suffers from the same limitations as the other discrete optimization models.

As BIP-based CPO models are $\mathcal{NP}$-complete, many heuristic algorithms were proposed to solve large real-world CPO problems in reasonable time given a reasonable amount of resources. Most prominently used categories of approximate algorithms, [19], include, greedy heuristics, [20], differential evolution algorithms, [21], local search algorithms, [22], [23], genetic algorithms, [24], [25], and particle swarm optimization, [26], [27]. In [28], the authors propose a continuous objective function for a CPO problem by modelling a continuous *observation measure* function over camera poses which are modelled using discrete variables. In contrast our method proposes to use continuous variables for camera poses to improve coverage. In [29], Smith et al. proposed a continuous optimization approach for path planning of UAV positions to achieve multi-view 3D reconstruction. The optimal paths for the UAV are selected from a predefined set of navigable paths. Such a simple approximation is, however, not suitable for our CPO problem as it is not possible to approximate the non-convex structure of a vehicle's surface. Some CPO models use application-specific relaxations to allow some variables to take values in the continuous domain. In [30] the author proposed a MIP-based CPO problem that uses continuous variables only for the camera's orientation while the camera's position and surrounding points are discrete. [31] proposes an approximation to define sensor locations using continuous variables. Their approximation is not generalizable to other use cases and cannot be used for vehicle surround-view.

We model the CPO problem for vehicle surround-view using 3D polygon mesh models of vehicles, [32]. Although one way of discretizing the CPO problem is by voxelizing the vehicle's 3D model, [33], another approach may be to use the vertices present in the 3D mesh model as possible camera positions. This however, may degrade the coverage results compared to voxelization as the vertices are irregularly distributed in space and thus, do not represent the vehicle's surface uniformly. Moreover, some faces in the 3D mesh model may be large and, as a result, crucial camera poses between those vertices may be missed during optimization. To avoid under representation of the vehicle's surface in the optimization problem, we propose to model the camera poses using variables in the continuous domain. We test our optimization problem using a PSO algorithm and a Bayesian Optimization (BO) algorithm. According to our knowledge, BO has not yet been studied in the context of CPO problems. BO is a method that is increasingly finding interest in machine learning applications including in neural network training, [34], [35]. It appears suitable for our CPO model as it falls under the category of black-box or *surrogate methods*, where a surrogate function is developed to model the objective function. While PSO methods have been studied extensively in the context of CPO or combinatorial optimization, [36], [37], BO methods are not studied as much in this context.

III. Camera Placement Optimization for Vehicle Surround View

Any camera placement optimization (CPO) model requires the definition of 1) A space model, 2) Decision Variables, 3)

Camera Model, and 4) Objective function. In this article, we present a novel method where we model part of the variables in the continuous domain. Our space model consists of a 3D polygon mesh model of a vehicle placed at the center, surrounded by 3D points (commonly known as *control points* in OCP literature). The goal of the optimization problem is to identify locations and orientations of multiple cameras such that they are placed on the outer surface of the 3D mesh to maximize coverage of the *control points*. Camera poses are optimized using global, gradient-free, black-box optimization methods. We use a flexible camera projection matrix-based camera model that can be easily extended for distinct types of real-world cameras, camera lenses and visibility models. The following sub-sections describe these aspects of the camera placement optimization problem for vehicle surround-view in detail.

A. Continuous Space model

The simulation environment consists of a vehicle's 3D model placed at the center/origin, O, where the *xy-plane* describes the ground plane with *z-axis* pointing towards the height of the vehicle. The size of the vehicle's 3D model is given by the size of the smallest cubical bounding box that encompasses the polygon mesh model, as $B = (b_x, b_y, b_z)$ units. A set of J points, $C_j = (x_c, y_c, z_c) \forall j = 1 : J$, are sampled on the ground plane around the vehicle model at equal intervals in a square region, between a distance, c_b, of $\sqrt{2} \cdot \max\{b_x, b_y\} \leq c_b \leq 2 \cdot \sqrt{2} \cdot \max\{b_x, b_y\}$ units. The reasoning behind choosing these limits for c_b is explained in Section III-B. Depending on the size of the bounding box of a 3D model, the sampling interval varies between the range $[2, 20]$. Control points are decision variables for the optimization problem as the quality of the optimization model is evaluated based on the number of control points viewed by the optimized multi-camera network. For our optimization problem, control points are modelled as discrete binary decision variables. Each point C_j is represented using a binary variable and can be defined as,

$$c_j = \begin{cases} 1 & \text{if } \textit{control point } \text{j is covered} \\ & \text{by at least one camera} \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

A visualization of the simulated environment is shown in Figure 1. The visualization shows a model of a JCB machine placed at the center. For visualization purpose, the control points are shown as spheres on the ground plane around the 3D model.

B. Camera and Coverage Models

The position and orientation of a camera in 3D space together describe the *pose* of a camera, [38]. In the fields of computational geometry and computer vision, a camera's pose is described using a 3×4 transformation matrix, M, [39], which describes transformations of a rigid body in 3D space. The transformation matrix can be decomposed as, $M = K[R\ T]$, where, K is a 3×3 matrix known as the intrinsic parameter matrix which encompasses hardware-related properties of the camera such as, focal length of the lens, sensor dimensions, etc. The 3×4 matrix $[R\ T]$ describes the pose of the camera in three dimensional space, where R is a 3×3 *rotation matrix* describing a composition of the rotations of a camera's local coordinate axes about the three axes of a common referenced coordinate system, and T is a 3×1 vector describing a camera's *translation* w.r.t. a referenced origin. The actual position of the camera, $P = (x, y, z)$, in the referenced coordinate system is computed as $P = -R^T T$. The rotation angles (*Euler* angles) about the coordinate axes are computed from the rotation matrix, and vice-versa, using Euler's rotation theorem, [40].

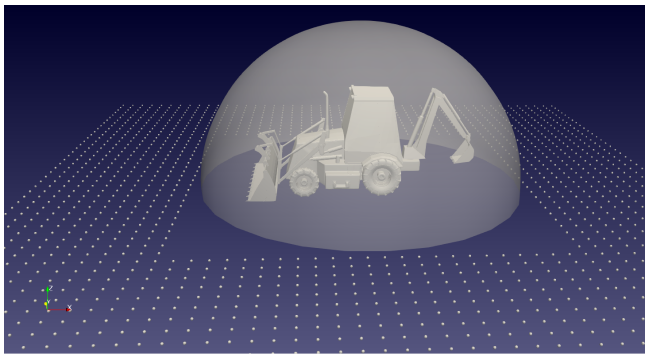


Fig. 1. Visualization of the simulation environment.

The matrix R can be further decomposed into three 3×1 vectors which form an orthogonal basis. In computer graphics, the three vectors are commonly referred to as the camera's *view*, *up*, and *right* direction vectors which represent the z, y, and x axes of the camera's local coordinate system, respectively. The camera's z-axis is same as the direction in which the camera looks at in 3D space. Our CPO problem estimates a camera pose in terms of the camera transformation matrix, M. Estimating M implies estimation of the translation vector, T, and the three rotation vectors for R. To simplify the estimation of R, two assumptions are commonly made in computer graphics: 1) The camera is placed upright, i.e., the camera's y-axis is always oriented towards the positive z-axis of the simulated environment; 2) The camera cannot rotate about itself. If the simulated environment's z-axis is given as, $z_w = [0.0, 0.0, 1.0]$ then, with the above two assumptions, a camera's *up* and *right* direction vectors can be calculated given its *view* vector, as,

$$\begin{aligned} right &= view \times z_w \\ up &= right \times view \end{aligned} \quad (2)$$

where '$\times$' represents vector cross product. Thus, we need to estimate only a camera's *view* vector (referred to as, $\hat{P}$) to construct it's R matrix. Therefore, only a camera's 3D position, P, and it's view direction vector are required to be estimated to construct it's camera matrix, M, which is used to compute coverage of control points, as will be described later.

A polygon mesh model is a composition of predefined vertices, edges and faces that describe the structure of an object. Popular file formats for polygon meshes support triangles,

quadrilaterals and general n-sided convex/concave polygons as faces. For this work, we assume that the vehicle's 3D model is a polygon mesh consisting of only triangles. Every triangle in the model is described by it's three vertices and three edges but, the positions of the vertices do not follow any specific structure, making it impossible to model the vehicle's surface using continuous functions. To simplify the problem, we model a hemispherical surface, S, around the vehicle's polygon mesh with a radius, $r = \sqrt{2}\max\{b_x, b_y\}$, with it's center placed at O. Figure 1 shows a translucent visualization of S encompassing the vehicle's polygon mesh model. Position of a camera is first estimated on S and then mapped to the vehicle's 3D model. In spherical coordinates, S can be described using it's radius, and polar and azimuthal angles, θ and ϕ respectively, as (r,θ, ϕ). To aid in camera pose estimation, we introduce a new variable, h, which represents the height of a point from the ground plane on the line passing through O along the z-axis of the simulated environment. This variable can take values only within a certain range, defined as, $0 \leq h \leq r$.

An illustration of the process of mapping a point from S to the surface of the vehicle's polygon mesh model is shown in Figure 2. As r is a constant for a given 3D model, we require three variables, θ, ϕ, h to estimate a camera's pose. A point $A(r, \theta, \phi)$ on S is converted into Cartesian coordinates as, $A'(x', y', z')$. $B(0, 0, h)$ describes a point on the line passing through the world z-axis and the origin O. The line $\overrightarrow{A'B}$ is checked for intersection with all the triangles of the vehicle's polygon mesh model using a simple ray tracing algorithm[1]. The closest point of intersection of $\overrightarrow{A'B}$ with any of the triangles of the 3D model gives the required camera's position, P, on the surface of the vehicle's model. The *view* direction vector of the camera placed at P is defined as $\hat{P} = -\overrightarrow{A'B} = A' - B$. $\hat{P}$ is normalized to make it a unit normal vector. Although this method simplifies camera pose estimation, it introduces challenges in covering the control points close to the vehicle.

This limitation is illustrated in Figure 2 in red. The red camera on the left presents a valid camera position, but due to limited vertical FoV it cannot cover the control points located close to the vehicle. The red camera on the right in Figure 2 can potentially cover the control points close to the vehicle, but it is an invalid camera pose as the value of h for this camera is beyond it's limits. To circumvent these limitations, we simply chose to not sample control points inside S. However, given the importance of covering the region close to the vehicle, innovative solutions such as, using camera models with larger FoV, increasing the limits of h and ϕ, etc., may be considered for future work.

Coverage of a control point is verified by projecting it into the camera's image plane. A control point is said to be covered by a given camera if it's projection lies within the camera's image dimensions, and the corresponding variable c_j is set to 1. Using homogeneous coordinates (see [39] for details on projection of a point using the camera matrix), a control point is projected into the camera's image plane as,

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = K[R\ T] \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}, \tag{3}$$

where, u, v are the pixel coordinates of the projected control point in the estimated camera's image plane and w is the distance of the point from the camera's origin along it's z-axis in the camera's local coordinate frame. In this work, for simplicity, we consider only the model of a pinhole camera for which the intrinsic parameter matrix K becomes a 3×3 identity matrix. A control point lies in the FoV of the camera if the pixel coordinates computed using equation (3) are within the size of the image. When $K = I$, the image plane becomes the same as the camera's projection plane and u, v are divided by w for the point's coordinates in the camera plane. For this special case, the point is said to lie within the camera's FoV if $-1 \leq u/w, v/w \leq 1$. Therefore, a variable c_j takes a value of 1 if $|\frac{u}{w}| \leq 1$ and $|\frac{v}{w}| \leq 1$ for at least one camera. The total coverage of the optimization model is given as, $\sum c_j$.

The presented camera coverage model using the camera projection matrix, to our knowledge, has not been used previously in any CPO problems. Discrete CPO models use geometric coverage models such as, disk sectors, pyramids, cones, etc., to model a camera's FoV, [41], [42]. Coverage is estimated using point in plane calculations. For e.g., in the case of a pyramidal FoV, each control point is checked against five planes of the pyramid and if it falls inside all five planes, the point is marked as covered by that particular camera. Coverage models based on CPM have certain advantages over geometrical shape based coverage models: 1) they do not need to define near and far planes explicitly as focal length parameter does it implicitly; 2) defining different types of lenses is easier as changing the geometrical shape is not required; and 3) visibility parameters such as, resolution, distortion, etc. can be defined easily using the camera intrinsic parameter matrix.

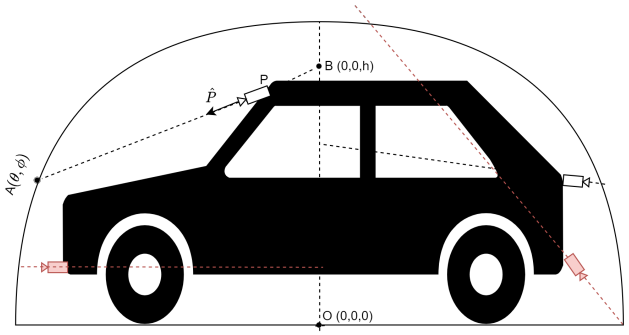


Fig. 2. Illustration of obtaining the camera pose on vehicle's surface.

C. Black-Box Optimization

The problem is formulated as a constrained optimization problem. It means that the decision variables are constrained to only take values within a specified range. The optimization

[1]https://www.scratchapixel.com/lessons/3d-basic-rendering/ray-tracing-rendering-a-triangle/ray-triangle-intersection-geometric-solution.html

problem handles both continuous and discrete variables. The control points are modelled as binary variables whereas, θ, ϕ and h are modelled as continuous variables. The problem of optimizing cameras poses of a multi-camera network consisting of N cameras to maximize surround-view coverage is formulated as,

$$\begin{aligned} max \ \sum_{j} c_j \qquad & \forall j \in J \qquad (4) \\ \text{subject to:} \quad & 0 \leq \theta_n < 2\pi, \\ & 0 \leq \phi_n < \frac{\pi}{2}, \\ & 0 \leq h_n \leq r, \end{aligned}$$

where, $n = 1 : N$. The objective function is a simple maximization of the sum of binary variables c_j but, estimating a camera's pose and computing coverage of the estimated camera involve non-linear calculations. This problem is optimized using black-box optimization techniques where, the optimization methods are designed to find a balance between exploration and exploitation of the search space while moving towards maxima/minima of any given unknown function. Black-box optimization methods define a search space for the provided unknown fitness functions based on the bounds and constraints placed on the values that can be taken by the variables. The optimization method works by sampling a set of values for the variables from the search space at each iteration. The fitness of the sampled values to the defined objective function is calculated. The fitness function is required to take the sample as an input at each iteration and output the sample's fitness as a real value which is used to adapt the exploration strategy for the next iteration.

Our proposed algorithm for fitness evaluations requires that the number of cameras, N, constituting the multi-camera network is defined by the user. The parameters, B and r for the given polygon mesh model are pre-calculated and control points, C_j are sampled as described in Section III-A. Intrinsic parameter matrix, K, is initialized as a 3×3 identity matrix, I. For representation in homogeneous coordinates, a value 1 is appended to the control points to represent them as 4×1 vectors. After the optimization starts, the variables are sampled from the search space within their bounds and passed to our defined fitness function. The algorithm starts by looping over the number of cameras N. The points $A(r_n, \theta_n, \phi_n)$ and $B(0, 0, h_n)$ are created from the N samples (θ_n, ϕ_n, h_n) for a particular iteration. Estimating camera poses and calculating total coverage follows the steps mentioned in III-B. Sequentially, for each camera, the point A is converted into Cartesian coordinates as A' and the line $\overrightarrow{A'B}$ is computed. A' is mapped to the surface of the vehicle using line-triangle intersections and the closest point of intersection on the vehicle's polygon mesh model is given as P. In the case, for any of the cameras, an intersection between $\overrightarrow{A'B}$ and any of the vehicle model's triangles is not found, the coverage/fitness value is returned as 0 and the algorithm skips to the next iteration with new samples.

Once P is found, the orientation for the camera n is set as $\hat{P} = -\overrightarrow{A'B}$ and the rotation vectors are found as given in 2 and the matrix R is constructed. The translation vector is calculated and the matrix M is constructed by concatenating R and T. Here we introduce a step to check if the camera n is occluded by the vehicle. Occlusion checking is done in the same way as coverage of control points is calculated. Consider the set of all vertices of the polygon mesh model, $V_i \ \forall i = 1 : I$ where, I is the number of vertices the polygon mesh model is comprised of. Similar to control point coverage calculations, every vertex V_i is projected into the camera's image plane using (3). If any V_i falls within the FoV, then the camera n is considered as occluded and the algorithm immediately proceeds to the next iteration by returning a fitness value of 0. While complex occlusion checking strategies present an accurate simulated representation of the problem, we consider only occlusion by the vehicle in individual cameras to keep the model simple and concise.

Our model sequentially computes occlusion of individual cameras by the vehicle, but it can be easily extended to compensate occluded views by comparing with adjacent cameras, or other complex occlusion checking methods by exploiting our camera projection matrix-based coverage model. If the camera n is not occluded, the algorithm loops over all J control points to calculate coverage. Each control point is projected into the camera's image plane and the corresponding variable c_j is marked as 1 if the control point falls within the camera's FoV. After computing coverage by N cameras, the fitness value or the overall coverage value by N cameras is returned as the sum of all variables c_j. The pseudo-code for the fitness function is presented in Algorithm 1.

Depending on the optimization algorithm and it's convergence criteria, the poses of N cameras corresponding to the best obtained fitness value are saved as the solution to the optimization problem. The complexity of the algorithm depends on the number of vertices and faces (or triangles) that a vehicle's polygon mesh model is composed of. In all cases, the number of triangles of a model is much higher than the number of control points. Hence, the line-triangle intersection checking and occlusion checking parts of the algorithm require the highest computational time as those functions are executed over all the vertices or faces of the polygon mesh model, for every iteration. The worst-case complexity of the algorithm can be given as $\mathcal{O}(nt)$ where, nt is the number of triangles (or faces) present in the vehicle's 3D model. The actual complexity, however, varies on a case-to-case basis due to the presence of break statements. In case an invalid camera pose is encountered, the algorithm may skip a majority of the calculations for that iteration, resulting in lower overall computational times.

IV. Results

We test the proposed continuous CPO model on over 100 simulation instances. These instances mainly consist of 3D polygonal models of vehicles, which are all downloaded from open-source websites that host free 3D models. The models represent various types of vehicles, including cars, buses and heavy vehicles used in the construction industry. The experiments are conducted in two parts. The experiments

Algorithm 1: Fitness function

Input: N, r, $\theta_n, \phi_n, h_n\ \forall n = 1:N$, $C_j\ \forall j = 1:J$, $V_i\ \forall i = 1:I$, $z_w = [0,0,1]^T$;
Result: $\sum_j c_j$
Initialize: $c_j = 0\ \forall j = 1:J$, $K = I_{3\times3}$;
for $n = 1$ **to** N **do**
  $(x, y, z) = (r, \theta_n, \phi_n)$ from Spherical to Cartesian coordinates;
  $A' = [x, y, z]^T$;
  $\overrightarrow{A'B} = [-x, -y, h_n - z]^T$;
  P = point of intersection between a line from A' in the direction $\overrightarrow{A'B}$ and the 3D polygon mesh;
  if *intersection not found* **then**
    $c_j = 0\ \forall j = 1:J$;
    break;
  end
  $\hat{P} = -\overrightarrow{A'B} = [x, y, z - h_n]^T$;
  $right = \hat{P} \times z_w$, $up = right \times z_w$;
  R = concatenate column vectors $\hat{P}$,*up*,*right* into 3×3 matrix;
  T = -(R×P), M = [R T];
  $\forall i$ check for occlusion by V_i using M;
  if *camera is occluded* **then**
    $c_j = 0\ \forall j = 1:J$;
    break;
  end
  for *j=1* **to** *J* **do**
    $[u\ v\ 1]^T$=$K \times M \times C_j^T$;
    if $|u/w| \leq 1$ **then**
      if $|v/w| \leq 1$ **then**
        $c_j = 1$;
      end
    end
  end
end

in the first part are conducted on 50 instances to validate the working of our proposed model. For the first part, the camera poses are optimized using two open-source black-box optimization methods. We use a particle swarm optimization (PSO) heuristic method called Fuzzy Self Tuning PSO (FST-PSO) that uses fuzzy logic to estimate appropriate values for the involved parameters without any user intervention, [43]. We used the openly available source code[2] of this PSO-variant with the default parameters. For the second method, we chose an open-source Bayesian Optimization library[3] built in python programming language. This library has a simple interface, specifically designed for black-box functions. In the second part, experiments are conducted on the remaining 54 instances using the FST-PSO algorithm and the percentage coverage for all instances is compared against the coverage from a discrete optimization model. For the discrete problem, we use a PSO variant, LH-RPSO [26], that is designed to handle discrete problem formulations.

All tests are run on an Intel i5 1235u processor with 16GB of RAM. The ray-tracing and occlusion functions are run on an Nvidia MX570 GPU in python using *numba* library. The geometry computations required to build a visibility matrix for the discrete optimization problem were run on the same GPU using the OpenCL library in C++ programming language. The interface for loading, processing, and visualizing the data is programmed in Qt in C++ and the optimization algorithms were run in python using their python interfaces. For the first part of the experiments, the number of cameras were set to $N = 4$, with each camera having a horizontal FoV of 90^o. For the second part, we set $N = 5$ with each camera having a horizontal FoV of 67.5^o. FST-PSO algorithm was run for 100 iterations with 30 particles for all cases. BO algorithm was run for 300 steps of Bayesian optimization and 50 steps of random exploration (total of 350 iterations). The FST-PSO method was run for 10 times and the best result of the 10 runs is presented. The BO method is a deterministic method, and hence, was run only one time. The BO method requires one parameter called *window size*, [44] to be set by the user. The authors claim that the BO algorithm is relatively unaffected by this parameter. Through our experiments, we observed that this parameter has high influence on the coverage result. We set it to a value in the range [0.1, 100.0], depending on the size of the bounding box of the model, B, through trial and error, changing it for every vehicle model. The total coverage and the optimization times obtained using both the optimization methods on 50 instances (first part of the experiments) are presented in Table I.

The proposed CPO model in continuous space is tested on 50 instances of different types of vehicle's 3D polygon mesh models, ranging from small cars to heavy vehicles such as, mining truck, JCB machine, etc. In the table, entries marked with a $*$ next to the instance number show results on mesh models of heavy vehicles. The small cars group includes models of small geometric shapes resembling a car, van, truck and bus, and 3D models of real-life cars, such as, camaro, dodge, compact and mini cars, BMW, tesla truck, and a truck in 2-4 sizes (varying number of triangles). The other group of heavy vehicles includes polygon mesh models of a JCB machine, digger, mining truck, dumper, lift truck, forklift, tank, tractor-scraper, bulldoser, school bus, and tour bus, in 2-4 sizes of each model. The instances are categorized into four groups for better readability, based on the number of triangles the polygon mesh models are composed of. In the table, the better coverage values and the shorter computation times between the two methods are marked in bold font. The number of cameras were set equal to four because, each camera has an FoV angle of 90^o. Hence, in the ideal case, four cameras should provide a 360^o coverage when the cameras are placed adjacent.

It can be seen from the table that coverage values for some instances using the PSO method have indeed reached the ideal expected case of 100% coverage. 100% coverage was achieved for instances 9, 14, 19, 35, 37, 41 and 50, which are mesh models of a mini car, a JCB machine, a tesla truck, a lift truck, a dumper, a digger machine, and a truck, respectively. These models are diverse with varying shapes and irregularity

[2]https://github.com/aresio/fst-pso
[3]https://github.com/fmfn/BayesianOptimization

TABLE I
TEST RESULTS OF THE CONTINUOUS CPO MODEL ON 50 INSTANCES OF VEHICLE MODELS USING PSO AND BO METHODS.

Instance #	# of △s	Coverage (%)		Time (s)	
		FST-PSO	BO	FST-PSO	BO
1	382	**98.1**	90.8	**18**	205
2	416	**96.6**	88.0	**12**	96
3	427	90.4	**95.2**	**19**	201
4	552	86.5	**87.5**	**22**	182
5	668	**93.9**	90.0	**29**	149
6	760	90.7	**91.7**	**24**	204
7	840	87.0	**88.2**	**35**	263
8	972	85.1	**86.3**	**26**	115
9	2,548	**100.0**	95.0	**58**	181
10	2,188	**91.9**	67.2	**54**	75
11	2,340	**93.3**	91.2	94	**75**
12	2,844	**92.7**	85.7	**71**	189
13	3,180	84.3	**85.0**	**78**	119
14*	3,343	**100.0**	92.3	**51**	232
15*	3,382	**95.4**	58.5	**30**	110
16*	3,759	**96.9**	81.5	**68**	167
17	4,134	**93.5**	79.6	**63**	111
18	4,326	**93.5**	91.7	**82**	175
19	4,926	**100.0**	87.1	303	**210**
20	5,256	89.8	**89.9**	**83**	244
21	5,268	**95.8**	91.7	**50**	144
22*	6,078	**97.9**	93.2	**74**	411
23	6,644	86.1	**87.0**	250	**197**
24*	7,434	**95.3**	90.0	**108**	268
25*	8,986	84.3	**85.2**	**138**	168
26*	12,839	**87.2**	72.0	206	**89**
27*	13,140	**95.1**	93.0	240	**133**
28*	15,286	**90.5**	66.7	332	**207**
29	15,568	**97.5**	88.1	**180**	409
30*	26,851	**96.2**	51.4	972	**180**
31*	35,120	79.6	**79.9**	650	**121**
32	36,952	**95.0**	80.8	**260**	320
33*	40,065	**97.5**	61.4	439	**220**
34*	44,753	**86.5**	45.0	438	**142**
35*	48,636	**100.0**	67.6	1082	**295**
36*	48,734	**99.2**	73.2	704	**144**
37*	53,380	**100.0**	88.2	1539	**309**
38	82,930	**98.0**	68.0	1046	**172**
39*	112,080	**87.4**	58.8	2308	**737**
40*	132,977	**81.3**	73.2	3968	**310**
41*	133,082	**100.0**	62.41	2008	**348**
42	159,060	**93.7**	86.9	2881	**783**
43*	162,016	**89.6**	62.3	2094	**255**
44*	165,088	**88.6**	78.4	3220	**499**
45*	171,004	**93.8**	63.8	2047	**232**
46*	274,694	**93.9**	64.4	4371	**713**
47*	282,636	**99.6**	88.4	3408	**669**
48	299,640	**96.9**	62.5	2504	**520**
49	353,172	**94.4**	57.1	1957	**392**
50*	513,612	**100.0**	69.6	3732	**623**

of the outer surface of the vehicle models which shows that the proposed continuous model based CPO problem can be used for the general case of camera poses optimization to achieve surround-view coverage for any types of vehicles. A coverage value of more than 80% was obtained for 49 instances using the FST-PSO method, and for 29 instances using the BO optimization method. The one instance ($\#31$) for which a coverage of less than 80% was obtianed using the PSO method is of a mining truck. The mining truck model is the largest of all vehicle models, in the measure of volume filled within S or in terms of surface area of the vehicle's model. As a consequence, the search space of this model is relatively large, posing a challenge for the optimization methods to find an acceptable solution within 100 iterations. Over all the instances, the PSO method has performed better than the BO method with an average coverage of 93.2%, against the BO method with an average coverage of 78.6%.

Across the four groups categorized according the number of triangles present in the model, the average coverage obtained by the PSO method is 92.2%, 93.3%, 94.0%, and 93.3%, whereas, the coverage obtained by the BO method is 88.1%, 85.6%, 71.9%, and 69.0%, for the first to the fourth groups, respectively. These average coverage values show that the PSO method has performed well on all instances irrespective of the vehicle's size in terms of the number of triangles it is composed of or the structure and shape of the vehicle. Whereas, the BO method has found acceptable camera poses for the first two groups of vehicle models but, the obtained coverage values for the last two groups highlight the poor performance of the BO method on larger 3D models of vehicles. The BO method's performance, however, did not show any dependability on the type or structure of the vehicle, as high and low coverage values were obtained by the BO method for models of both small vehicles and heavy vehicles. It is natural to expect a method's performance or accuracy to decrease as the size of the search space increases. While this behaviour is evident in BO method, the PSO method's consistency over vehicle models of all sizes shows the versatility of PSO-based methods.

The computational times results for the two methods show an increasing trend as the number of triangles in polygon mesh models increases. This increasing trend is more evident for the results from PSO method than the results from BO method. The BO method's complexity does not directly depend on the number of triangles in the model. it rather depends on the estimation of posterior probability from the prior which is constructed from the sampled points during the *random exploration* stage. For this reason, the computationally intensive steps of ray-triangle intersection calculation and occlusion checking are not executed for all the iterations, resulting in lower overall computational times. On the other hand, particles in the PSO method sample the search space and their fitness is calculated in every iteration. This increases the time for optimization as the geometric calculations need to be done for all the triangles in every iteration. The break statements in the algorithm however, may help in skipping the calculations for some iterations depending on whether the sampled point represents valid poses for all the cameras or not. In the N estimated camera poses, when any one of them has an invalid pose, then the remaining computations are skipped for that iteration, resulting in comparatively shorter computation times.

This behaviour is evident for the PSO method where, the computational times are relatively higher for similarly sized models (for e.g., instances 19 and 20, 33 and 34, 35 and 36, etc.). This behaviour by PSO method is entirely dependent on the structure of the vehicle as vehicle models with a greater surface area inside the hemisphere S have a higher chance that an estimated camera pose on S can intersect with any of the model's triangles. As the intersection results in a valid

position, all the computations need to be performed for the iteration, resulting in higher computational time. Overall, the PSO method was faster than the BO method for 22 out of the 25 instances in the first two groups. However, in the last two groups, the BO method is faster than the PSO method for 23 out of the 25 instances. On an average, the required computational time using the PSO method is 888.5s whereas, it is 266.3s using the BO method. The average computational times using the PSO method are 38s, 106s, 622s, and $2,875$s, for the first to the fourth groups, respectively. From these averages it can be seen that the computational complexity increases w.r.t. the number of triangles present in the vehicle's polygon mesh model. Whereas, the required computational times of 161s, 196s, 211, and 506s for the first to the fourth groups, respectively, using the BO method show that the complexity of the BO method is relatively constant to the number of triangles in the vehicle's mesh model.

A. Comparison against discrete model

To further evaluate the performance of the proposed method for optimization of camera poses in continuous space, we compare it against a similar CPO problem modelled in discrete space, [33]. The two methods are tested on 54 instances of different categories of 3D models of vehicles. To test the discrete method on the same simulated environments, the polygon mesh models were voxelized to convert them into a 3D volume. Voxelization discretises the vehicle model from irregular vertices and faces into a structured grid with equal spacing between consecutive voxel centers. Each grid point on the voxelized vehicle model's surface represents a camera position. Each point is associated with a surface normal which becomes the camera's orientation (*view* direction vector) for that camera position. Each orientation is then rotated about it's *up* and *right* directions at equal steps of $\sim 14^o$. Further details about the discrete space model can be seen in, [33]. The only difference between the simulated environments for both the methods is that the camera poses for the discrete space model are modelled using discrete binary variables. For the discrete method, coverage for all possible camera poses is pre-computed and stored in a lookup table which is used by the optimization algorithm. This pre-computation step filters out part of the camera poses on the basis of occlusion and minimum required coverage by an individual camera. The complexity of the algorithm for the discrete model depends on the number of entries present in the lookup table. For a fair comparison between the continuous and discrete models, we adjust the parameters during this filtering step so that the number of entries in the lookup table roughly matches with the number of triangles in the polygon mesh model. The discrete space model is tested using the LH-RPSO method with 30 particles and run for 100 iterations. All tests were run with $N = 5$ cameras with a horizontal FoV of 75^o and a vertical FoV of 56.25^o.

Table II presents a comparison between the coverage values and computational times for the continuous and discrete models on 54 instances of different types of vehicle models. The instances are arranged in the table in ascending order of the number of triangles, nt, present in the corresponding vehicle's polygon mesh models. Of these, 27 instances (11-15,18,24,27-28,30,33-41,45-48 and 51-54) are of models of heavy vehicles used in the construction industry such as, bulldozer, dumper, lift truck, tractor, etc. The remaining instances are models of smaller vehicles such as, different types of cars and vans. Although care was taken to ensure that the number of variables present in the discrete model matches the number of triangles

TABLE II
COMPARISON OF COVERAGE AND OPTIMIZATION TIMES OF CONTINUOUS MODEL AND THE DISCRETE MODEL.

Instance #	# of △s	Coverage (%)		Time (s)	
		C	D	C	D
1 1	468	**100.0**	82.7	15	**3**
2	472	**100.0**	83.6	16	**3**
3	560	**100.0**	84.6	23	**5**
4	592	**99.0**	85.1	17	**5**
5	686	**100.0**	80.0	17	**8**
6	812	**100.0**	83.6	26	**5**
7	904	**100.0**	80.2	28	**5**
8	1080	**99.5**	83.6	30	**6**
9	1104	**98.8**	77.5	31	**8**
10	1213	**100.0**	82.5	62	**10**
11	1367	**100.0**	84.4	24	**7**
12	1543	**100.0**	88.2	38	**11**
13	1805	**100.0**	80.0	35	**11**
14	2078	**93.8**	84.6	20	**10**
15	2122	**100.0**	81.5	95	**18**
16	2240	**99.7**	87.1	277	**38**
17	2612	**100.0**	79.7	163	**23**
18	2688	**100.0**	86.7	276	**43**
19	2776	**100.0**	83.9	94	**22**
20	3309	**100.0**	86.9	43	**27**
21	3520	**100.0**	78.7	177	**28**
22	3600	**99.2**	83.8	113	**30**
23	3753	**100.0**	87.0	51	**18**
24	4910	**97.2**	80.6	77	**21**
25	5468	**100.0**	83.3	149	**25**
26	5704	**100.0**	80.6	277	**32**
27	6160	**100.0**	82.3	94	**29**
28	6799	**100.0**	86.8	119	**47**
29	6800	**100.0**	86.1	136	**40**
30	6924	**98.7**	82.7	62	**42**
31	7282	**99.9**	80.8	628	**237**
32	7536	**100.0**	82.4	385	**53**
33	10164	**100.0**	89.0	182	**162**
34	11607	**100.0**	82.2	246	**70**
35	12496	**98.7**	82.5	260	**67**
36	13861	**100.0**	82.8	333	**66**
37	14191	**100.0**	81.9	226	**57**
38	16204	**100.0**	88.4	130	**112**
39	16565	**100.0**	83.1	511	**133**
40	17570	**92.8**	84.8	267	182
41	22127	**97.6**	87.5	403	**144**
42	22962	**99.3**	86.8	340	**115**
43	23201	**100.0**	86.4	200	**124**
44	26119	**100.0**	88.0	441	**140**
45	30840	**98.7**	85.0	563	**139**
46	31084	**99.9**	86.7	976	**403**
47	31328	**96.1**	81.6	**360**	410
48	31817	**97.1**	78.6	1118	**248**
49	34666	**100.0**	85.6	410	**195**
50	36694	**100.0**	86.2	510	**176**
51	38313	**100.0**	83.4	694	**281**
52	52110	**98.6**	86.5	724	**259**
53	64165	78.8	**83.9**	**421**	755
54	64834	**95.4**	81.3	**534**	1083

in the continuous model, it is impossible to match them exactly as we do not have direct control over how many variables get selected after the visibility checking step in the discrete model. During optimization, on an average across all the instances, the discrete model has 900 variables more than the number of triangles in the continuous model. From the coverage results it can be observed that the continuous model clearly performs better than the discrete model. The instances are divided into four groups based on a threshold on nt. On an average, the percentage of control points covered by the continuous model is better than the discrete model by 17.0, 15.8, 15.3, and 12.4 percentage points for the four groups of instances, respectively.

Looking at near 100% coverage for most cases, it may seem that N is redundant. Considering the horizontal FoV of the cameras, the ideal number of cameras for full coverage is $\frac{360}{75} = 4.8$. This justifies our choice of selecting $N = 5$ cameras as $N = 4$ cameras may not produce 100% coverage for any of the cases. As the discrete model collects samples at equal intervals, the values between two samples are missed during optimization. There is a chance that a camera placed at one of the intermediate values between two sampled positions may provide better coverage. Additionally, there is a gap of 14^o between two consecutive camera orientations. This gap is significant as small variation in the orientation angle may severely effect coverage by the camera. While accuracy may be improved by reducing the step size between two consecutive rotations of the camera view but doing so results in a higher number of possible camera poses in the lookup table. The continuous model does not have any such limitations as the camera poses can be precisely located on the vehicle's surface with floating point accuracy. The continuous model can also estimate camera orientations to an accuracy of less than 1^o. Due to these limitations in the placement of cameras in the discrete model, the placed cameras may have a relatively higher overlap between adjacent cameras. Whereas, cameras in the continuous model can be precisely placed to minimize overlap and increase overall coverage.

In only one case, instance 53, the discrete model performed better than the continuous model in terms of coverage of control points. This instance is a model of a tractor scraper. The vehicle has an irregular structure with many holes through the structure and perturbations covering over the holes only on some sides. Due to the presence of holes, the traced rays from the outer hemisphere on to the vehicle model's surface may end up in the holes. As a result, the camera pose gets obstructed by the other surface which may be covering the hole partly making the camera pose invalid. As fewer valid camera poses were found during the given number of iterations of the optimization algorithm, the coverage was relatively lower for this particular vehicle model. On the other hand, the possible camera poses in the discrete model are collected directly form the outermost surfaces. Due to this reason, camera poses present inside the holes on the vehicle surface's structure do not end up in the look-up table, leaving only valid camera poses for the optimization algorithm to choose from. As a result, the discrete model performed better for this instance.

In Table II, the shorter computation times between the continuous and discrete models are highlighted in bold. The discrete model has shorter computation times for all instances except for instances 47,53, and 54. Instances 47 and 54 are models of a mining truck in two different sizes and instance 53 is a polygon mesh model of a tractor scraper. Both these models have relatively large surface area, resulting in a greater number of discrete samples. This increases the computational time for the visibility checking part and the optimization part for the discrete model. Whereas, the continuous model must have encountered a greater number of invalid camera poses resulting in fewer calculations and lower computational times. Although the computational times show an increasing trend with increasing number of triangles in the polygon mesh model, it is not so evident for the continuous model as the time varies greatly for models with similar size, in some cases. The reasoning for this behaviour is same as stated previously that computational times are higher when a majority of the estimated camera poses turn out to be valid as all the computations are performed.

Whereas, most of the computations are skipped when a camera pose is found to be invalid in any of the iterations. Whereas, computational times for the discrete method show a more stable increase with increasing size of the models. There are stray cases of high computational times for some models compared to other models of similar size such as, instances 31,33,46, etc. 31, 33, and 46 are models of a humvee, a digger machine, and a tour bus, respectively. The outer surface of these vehicles have a box-like structure with large planar body on all sides, resulting in more camera poses that pass through the visibility checking steps when compared to other similarly sized vehicle models. As the number of variables increases, number of computations also increase resulting in higher computational times. On an average the discrete model is faster than the continuous model by 14s, 113s, 141s, and 233s in the four groups, respectively. Over all the 54 instances, the discrete optimization model is faster than the continuous optimization model by 128 seconds.

To evaluate the performance of the proposed continuous CPO model, we show visualizations of the simulated environment with the $N = 5$ cameras placed at the optimized poses. The visualizations for four instances are presented in Figure 3. The visualizations show results of both the continuous and discrete methods for each instance. From the visualizations we can see that optimized camera poses for the continuous model are placed such that the overlap between them is minimized. Whereas, camera poses for the discrete model have relatively more overlap between them, resulting in more gaps between camera FoVs and lower overall coverage. Another limitation of the discrete model is also evident from the visualizations. The FoV of a camera in the discrete is described by five planes one of which is the far plane. The distance of this plane form the camera position needs to be carefully specified as points lying beyond this plane may not be counted as covered even though an extended version of the same camera's FoV may cover them. These points do not get counted into the overall coverage resulting in lower coverage percentage for the discrete model.

Another downside of the discrete model can be seen in Figure 3 (d), where there is a large gap between camera FoVs at the rear of the vehicle. Additional sampling of camera

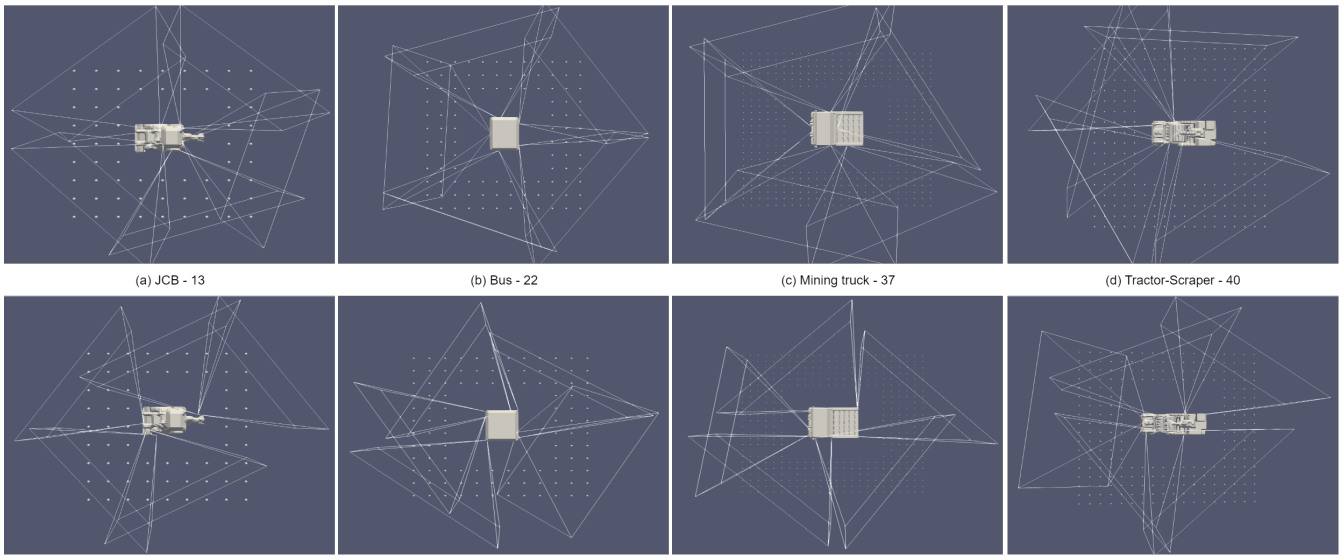


Fig. 3. Comparison of visualizations of the optimal solutions from the continuous CPO model and the corresponding discrete CPO model. The labels in the middle show the instance number and the name of the vehicle present in the model. The image above the label refers to the result from continuous CPO problem, and the one below the label refers to the result from discrete CPO problem, for that instance.

poses and decreasing the step size between orientations of cameras may have helped in reducing this gap. While the continuous method provides better coverage, the discrete model is faster by at least two times. The discrete model is faster primarily because the geometric calculations are done only once whereas, for the continuous model, coverage must be calculated in every iteration. However, the additional coverage obtained by the continuous method outweighs the advantage of faster computational speed of the discrete model.

V. Conclusion and Future Work

This article presented a new camera placement optimization formulation in continuous space domain for optimizing poses of a multi-camera network on the surface of 3D polygon mesh models of vehicles for capturing surrounding view. The camera poses are defined in the continuous domain and a non-linear objective function, suitable for gradient-free balckbox optimization methods is modelled as the sum of discrete binary variables representing coverage of the vehicle's surrounding area. The method was tested on 50 instances of 3D mesh models of vehicles using two approximate optimization algorithms in the first part. Experiments from the first part show that the proposed model can be used for 3D mesh models of vehicles of any shape and size. In the second part of experiments, it is compared against a discrete problem formulation on another 54 instances using a continuous and a discrete variant of particle swarm optimization. Results show that the problem formulation in the continuous domain is more effective than optimization in the discrete domain in terms achieving maximum coverage of the vehicle's surrounding area and in terms of computational time required. Visualizations of the results show that the proposed approach can effectively tackle the combinatorial aspect of the problem with only a few exceptions of vehicle models with irregular surfaces. For future work, more optimization methods need to be tested which can better exploit the combinatorial aspect of the problem. Different types of cameras can be tested by exploiting camera matrix-based coverage model to further test the versatility of the proposed method. Advanced ray tracing algorithms may be used to decrease the computational time for the PSO optimization method.

References

[1] W. Ostachowicz, R. Soman, and P. Malinowski, "Optimization of sensor placement for structural health monitoring: A review," *Structural Health Monitoring*, vol. 18, no. 3, pp. 963–988, 2019.

[2] H. An, B. D. Youn, and H. S. Kim, "A methodology for sensor number and placement optimization for vibration-based damage detection of composite structures under model uncertainty," *Composite Structures*, vol. 279, p. 114863, 2022.

[3] A. V. Savkin and H. Huang, "Navigation of a uav network for optimal surveillance of a group of ground targets moving along a road," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 9281–9285, 2021.

[4] T. Koch, M. Körner, and F. Fraundorfer, "Automatic and semantically-aware 3d uav flight planning for image-based 3d reconstruction," *Remote Sensing*, vol. 11, no. 13, p. 1550, 2019.

[5] X. Wang, H. Zhang, S. Fan, and H. Gu, "Coverage control of sensor networks in iot based on rpso," *IEEE internet of things journal*, vol. 5, no. 5, pp. 3521–3532, 2018.

[6] A. A. Altahir, V. S. Asirvadam, P. Sebastian, N. H. B. Hamid, and E. F. Ahmed, "Optimizing visual sensors placement with risk maps using dynamic programming," *IEEE Sensors Journal*, vol. 22, no. 1, pp. 393–404, 2021.

[7] M. S. Suresh, A. Narayanan, and V. Menon, "Maximizing camera coverage in multicamera surveillance networks," *IEEE Sensors Journal*, vol. 20, no. 17, pp. 10 170–10 178, 2020.

[8] J. Kim, Y. Ham, Y. Chung, and S. Chi, "Systematic camera placement framework for operation-level visual monitoring on construction jobsites," *Journal of Construction Engineering and Management*, vol. 145, no. 4, p. 04019019, 2019.

[9] S. Indu, S. Srivastava, and V. Sharma, "Optimal camera placement and orientation of a multi-camera system for self driving cars," in *Proceedings of the 2020 4th International Conference on Vision, Image and Signal Processing*, 2020, pp. 1–5.

[10] J. Dey, W. Taylor, and S. Pasricha, "Vespa: A framework for optimizing heterogeneous sensor placement and orientation for autonomous vehicles," *IEEE Consumer Electronics Magazine*, vol. 10, no. 2, pp. 16–26, 2020.

[11] D. Buljeta, M. Vranješ, Z. Marčeta, and J. Kovačević, "Surround view algorithm for parking assist system," in *2019 Zooming Innovation in Consumer Technologies Conference (ZINC)*. IEEE, 2019, pp. 21–26.
[12] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, "A systematic review of perception system and simulators for autonomous vehicles research," *Sensors*, vol. 19, no. 3, p. 648, 2019.
[13] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.
[14] C. Audet and W. Hare, *Derivative-free and blackbox optimization*. Springer, 2017.
[15] E. Hörster and R. Lienhart, "On the optimal placement of multiple visual sensors," in *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, 2006, pp. 111–120.
[16] U. M. Erdem and S. Sclaroff, "Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements," *Computer Vision and Image Understanding*, vol. 103, no. 3, pp. 156–169, 2006.
[17] J. Kritter, M. Brévilliers, J. Lepagnot, and L. Idoumghar, "On the optimal placement of cameras for surveillance and the underlying set cover problem," *Applied Soft Computing*, vol. 74, pp. 133–153, 2019.
[18] T.-H. Kim and T.-H. Park, "Placement optimization of multiple lidar sensors for autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 5, pp. 2139–2145, 2019.
[19] J. Zhao, R. Yoshida, S.-c. S. Cheung, and D. Haws, "Approximate techniques in solving optimal camera placement problems," *International Journal of Distributed Sensor Networks*, vol. 9, no. 11, p. 241913, 2013.
[20] K. Veenstra and K. Obraczka, "Grid partition: an efficient greedy approach for outdoor camera iot deployments in 2.5 d terrain," in *2020 29th International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2020, pp. 1–9.
[21] M. Brévilliers, J. Lepagnot, L. Idoumghar, M. Rebai, and J. Kritter, "Hybrid differential evolution algorithms for the optimal camera placement problem," *Journal of Systems and Information Technology*, vol. 20, no. 4, pp. 446–467, 2018.
[22] C. Gao, X. Yao, T. Weise, and J. Li, "An efficient local search heuristic with row weighting for the unicost set covering problem," *European Journal of Operational Research*, vol. 246, no. 3, pp. 750–761, 2015.
[23] W. Lin, F. Ma, Z. Su, Q. Zhang, C. Li, and Z. Lü, "Weighting-based parallel local search for optimal camera placement and unicost set covering," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, 2020, pp. 3–4.
[24] B. Cao, M. Li, X. Liu, J. Zhao, W. Cao, and Z. Lv, "Many-objective deployment optimization for a drone-assisted camera network," *IEEE transactions on network science and engineering*, vol. 8, no. 4, pp. 2756–2764, 2021.
[25] Y. Zhang, H. Luo, M. Skitmore, Q. Li, and B. Zhong, "Optimal camera placement for monitoring safety in metro station construction work," *Journal of construction engineering and management*, vol. 145, no. 1, p. 04018118, 2019.
[26] X. Wang, H. Zhang, and H. Gu, "Solving optimal camera placement problems in iot using lh-rpso," *IEEE Access*, vol. 8, pp. 40 881–40 891, 2019.
[27] Y. Morsly, N. Aouf, M. S. Djouadi, and M. Richardson, "Particle swarm optimization inspired probability algorithm for optimal camera network placement," *IEEE Sensors Journal*, vol. 12, no. 5, pp. 1402–1412, 2011.
[28] C.-H. Chen, Y. Yao, W.-W. Hsu, A. Koschan, and M. Abidi, "Continuous camera placement using multiple objective optimisation process," *IET Computer Vision*, vol. 9, no. 3, pp. 340–353, 2015.
[29] N. Smith, N. Moehrle, M. Goesele, and W. Heidrich, "Aerial path planning for urban scene reconstruction: A continuous optimization method and benchmark," *ACM Transactions on Graphics*, vol. 37, no. 6, pp. 1–15, 2018.
[30] N. Kirchhof, "Optimal placement of multiple sensors for localization applications," in *International Conference on Indoor Positioning and Indoor Navigation*. IEEE, 2013, pp. 1–10.
[31] Z. Ismail, S. Mustapha, M. A. Fakih, and H. Tarhini, "Sensor placement optimization on complex and large metallic and composite structures," *Structural Health Monitoring*, vol. 19, no. 1, pp. 262–280, 2020.
[32] R. F. Tobler and S. Maierhofer, "A mesh data structure for rendering and subdivision," *WSCG'2006*, 2006.
[33] V. A. Puligandla and S. Lončarić, "A multiresolution approach for large real-world camera placement optimization problems," *IEEE Access*, vol. 10, pp. 61 601–61 616, 2022.
[34] A. H. Victoria and G. Maragatham, "Automatic tuning of hyperparameters using bayesian optimization," *Evolving Systems*, vol. 12, pp. 217–223, 2021.
[35] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, "Hyperparameter optimization for machine learning models based on bayesian optimization," *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019.
[36] C. Hocine and A. Benaissa, "New binary particle swarm optimization algorithm for surveillance and camera situation assessments," *Journal of Electrical Engineering & Technology*, pp. 1–11, 2021.
[37] X. Wang, H. Zhang, S. Bai, and Y. Yue, "Design of agile satellite constellation based on hybrid-resampling particle swarm optimization method," *Acta Astronautica*, vol. 178, pp. 595–605, 2021.
[38] W. A. Hoff, K. Nguyen, and T. Lyon, "Computer-vision-based registration techniques for augmented reality," in *Intelligent Robots and Computer Vision XV: Algorithms, Techniques, Active Vision, and Materials Handling*, vol. 2904. SPIE, 1996, pp. 538–548.
[39] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.
[40] E. W. Weisstein, "Rotation matrix," *https://mathworld. wolfram. com/*, 2003.
[41] A. Mavrinac and X. Chen, "Modeling coverage in camera networks: A survey," *International journal of computer vision*, vol. 101, pp. 205–226, 2013.
[42] C. Qian and H. Qi, "Coverage estimation in the presence of occlusions for visual sensor networks," in *Distributed Computing in Sensor Systems: 4th IEEE International Conference, DCOSS 2008 Santorini Island, Greece, June 11-14, 2008 Proceedings 4*. Springer, 2008, pp. 346–356.
[43] M. S. Nobile, P. Cazzaniga, D. Besozzi, R. Colombo, G. Mauri, and G. Pasi, "Fuzzy self-tuning pso: A settings-free algorithm for global optimization," *Swarm and evolutionary computation*, vol. 39, pp. 70–85, 2018.
[44] N. Stander and K. Craig, "On the robustness of a simple domain reduction scheme for simulation-based optimization," *Engineering Computations*, vol. 19, no. 4, pp. 431–450, 2002.

V. Anirudh Puligandla was born in Hyderabad, Telangana, India in 1992. He received B.Tech degree in electronics and communications engineering from Amity University, Jaipur, Rajasthan, India in 2014 and B.Sc. degree in computer vision and robotics from University of Burgundy, France in 2016. He received M.Sc. degree in computer vision and robotics from University of Burgundy, France, University of Girona, Spain and Heriot Watt university, Scotland in 2018 as part of an Erasmus Mundus Joint Masters degree program. He is currently pursuing a Ph.D. degree program at University of Zagreb, Zagreb, Croatia under a Marie-Curie Actions ITN fellowship. His research interests include discrete and continuous optimization, signal and image processing, 3D reconstruction from multiple camera systems using multi-view stereo.

Dr. Sven Lončarić is a professor of electrical engineering and computer science at the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia. As a Fulbright scholar, he received a Ph.D. degree in electrical engineering from the University of Cincinnati, OH in 1994. From 2001-2003, he was an assistant professor at the New Jersey Institute of Technology, USA. His areas of research interest are image processing and computer vision. He was the principal investigator on a number of R&D projects. Prof. Lončarić co-authored more than 250 publications in scientific journals and conferences. He is the director of the Center for Computer Vision at the University of Zagreb and the head of the Image Processing Laboratory. He is a co-director of the Center of Excellence in Data Science and Cooperative Systems. He is a fellow of the Croatian Academy of Sciences and Arts and a senior member of IEEE. Prof. Lončarić received several awards for his scientific and professional work.

Biography

Venkata Anirudh Puligandla was born in Hyderabad, Telangana, India in 1992. He received B.Tech degree in electronics and communications engineering from Amity University, Jaipur, Rajasthan, India in 2014 and B.Sc. degree in computer vision and robotics from University of Burgundy, France in 2016. He received M.Sc. degree in computer vision and robotics from University of Burgundy, France, University of Girona, Spain and Heriot Watt university, Scotland in 2018 as part of an Erasmus Mundus Joint Master's degree program. He obtained his Ph.D. degree in computer science, in April 2024, from the Department of Electronic Systems and Information Processing, Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia under a Marie-Curie Actions ITN fellowship from 2019. From May 2023, he is working as an Application Engineer at dSPACE d.o.o.. His research interests include linear, non-linear and inverse programming, optimization, image reconstruction, compressed sensing, 3D reconstruction, and image segmentation.

List of Publications

Journal Publications

- Puligandla VA, Lončarić S. A Continuous Camera Placement Optimization Model For Surround View. IEEE Transactions on Intelligent Vehicles. 2023 Jul 26.
- Puligandla VA, Lončarić S. A Supervoxel Segmentation Method With Adaptive Centroid Initialization for Point Clouds. IEEE Access. 2022 Sep 12;10:98525-34.
- Puligandla VA, Lončarić S. A multiresolution approach for large real-world camera placement optimization problems. IEEE Access. 2022 May 23;10:61601-16.

Conference Publications

- Puligandla VA, Lončarić S. Optimal Camera Placement To Visualize Surrounding View From Heavy Machinery. InProceedings of the 2020 2nd Asia Pacific Information Technology Conference 2020 Jan 17 (pp. 52-59).

Životopis

Venkata Anirudh Puligandla rođen je 1992. godine u Hyderabadu; Telengana, Indija.. Stekao je diplomu B.Tech. iz elektronike i komunikacijskog inženjerstva na Sveučilištu Amity, Jaipur, Rajasthan, Indija 2014. godine te 2018. godine stječe B.Sc. diplomu iz područja računalnog vida i robotike na Sveučilištu Burgundija, Francuska; Sveučilištu Girona, Španjolska i Sveučilištu Heriot Watt, Škotska u sklopu zajedničkog magistarskog programa Erasmus Mundus. Od 2019. godine pohađa doktorski studij na Zavodu za elektroničke sustave i obradbu informacija u sklopu Marie-Curie Actions ITN stipendije na Fakultetu elektronike i računarstva, Sveučilište u Zagrebu. Od svibnja 2023. radi kao Application Engineer u dSpace d.o.o. Njegovi istraživački interesi uključuju linearno, nelinearno i inverzno programiranje, optimizaciju, rekonstrukciju slike, komprimirani senzor, 3D rekonstrukciju i segmentaciju slike.