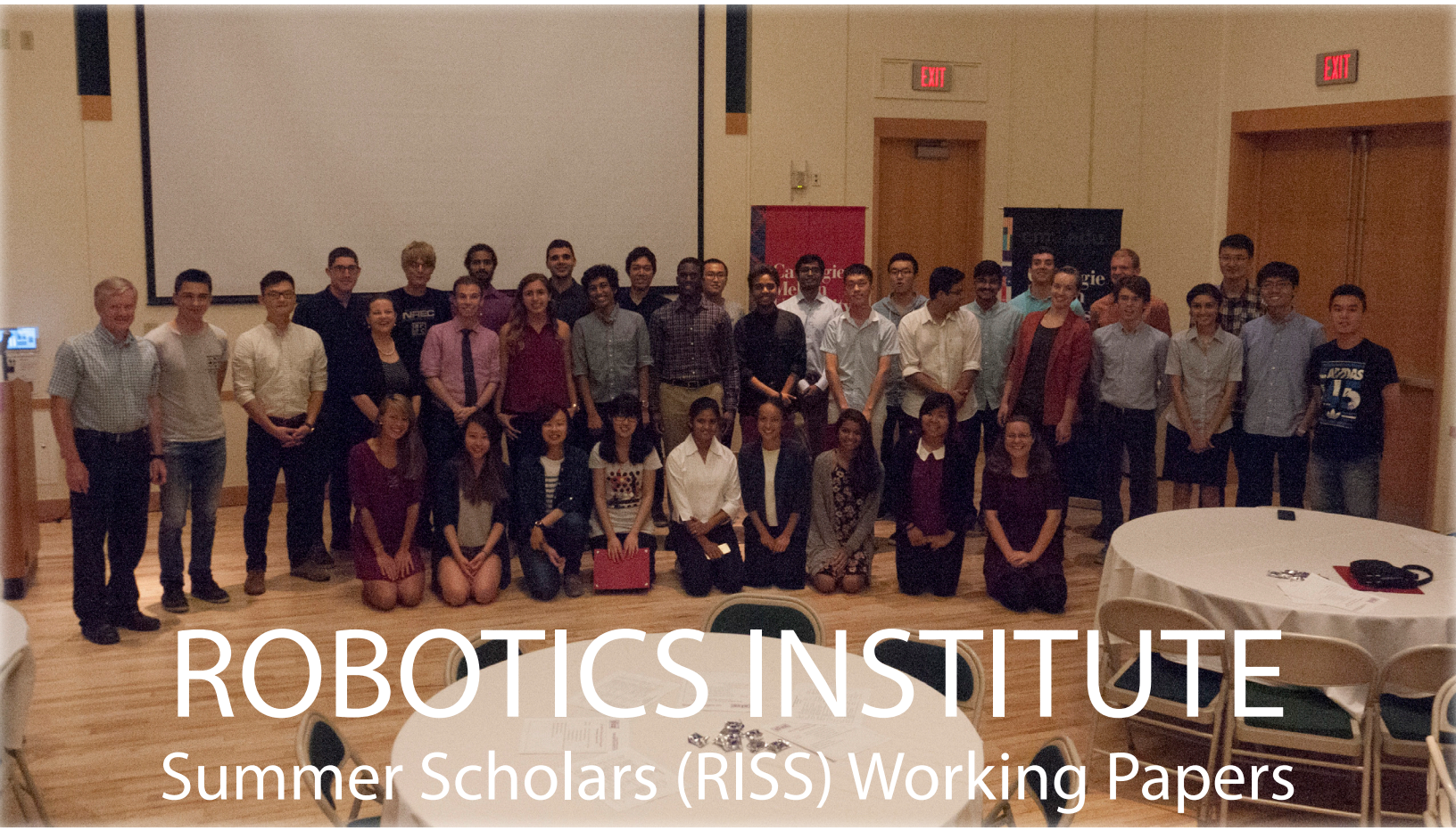
VOLUME 4 FALL 2016



ROBOTICS INSTITUTE
Summer Scholars (RISS) Working Papers

JOURNAL

ROBOTICS INSTITUTE SUMMER SCHOLARS
Working Papers Journal

The Robotics Institute Summer Scholars Working Papers Journal is an annual publication of the Robotics Institute's Summer Scholars Program at Carnegie Mellon University.

Volume 4 Fall 2016

TABLE OF CONTENTS

SUMMER SCHOLAR PROGRM

Carnegie Mellon
THE ROBOTICS INSTITUTE

RISS | ROBOTICS INSTITUTE SUMMER SCHOLARS

At the core of the program are incredibly talented and dedicated faculty, graduate students, staff, and RISS alumni.

We are incredibly thankful for their support, participation, leadership, and vision that make this one of the best research experiences in robotics and intelligent systems in the world.

THANK YOU TO THE RISS 2016
Participating Labs

AIR Lab

Auton Lab

Biorobotics Lab

Computer Vision Lab

Field Robotics Center (FRC)

Intelligent Coordination and Logistics Lab (ICLL)

Learning, Artificial Intelligence, and Robotics Lab

Microdynamic Systems Lab

Navlab

Platypus LLC

People Image Analysis Consortium (PIA) Lab

Personal Robotics Lab

RoboTutor

Robust Adaptive Systems Lab

Search-Based Planning Lab

Tekkotsu

THANK YOU TO THE RISS 2016 Advisors

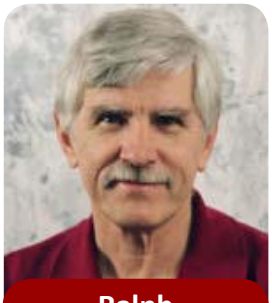

Howie Choset


Stelian Coros


John M. Dolan


Artur Dubrawski


Martial Hebert


Ralph Hollis


Isaac K. Isukapati


Michael Kaess


George Kantor


Maxim Likhachev


Simon Lucey


Christoph Mertz

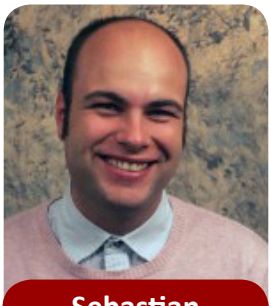

Nathan Michael


Jack Mostow


Stephen T. Nuske


Zachary Rubinstein

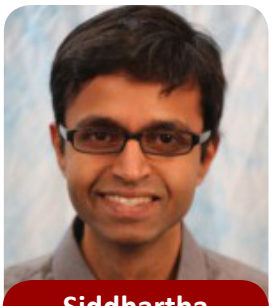

Paul Scerri


Sebastian Scherer

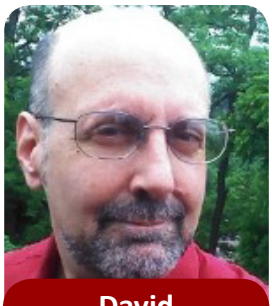

Yaser Sheikh


Stephen Smith

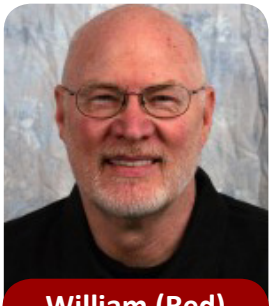
THANK YOU TO THE RISS 2016 Advisors



Siddhartha Srinivasa



David Touretzky



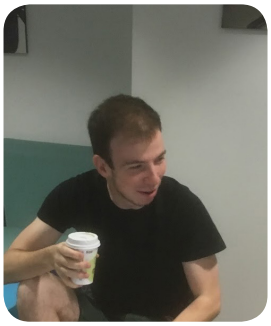
William (Red) Whittaker

CONGRATULATIONS
Cohort of 2016 !

Akanimoh Adeleye	University of Oklahoma, USA
Jahdiel Alvarez	University of Puerto Rico, Mayaguez, Puerto Rico
Simon Bloch	Swarthmore College, USA
Jon Cruz	Harvard University, USA
Mosam Dabhi	National Institute of Technology, Surat, India
Aditya Dhawale	Indian Institute of Technology, Guwahati, India
Deepthi Hegde	National Institute of Technology, Surathkal, India
Grace Kumagai	University of Toronto, USA
Yi Chun Kuo	The Chinese University of Hong Kong, China
Vince Kurtz	Goshen College, USA
Minae Kwon	Cornell University, USA
Michael Lee	Princeton University, USA
Anjana K N	National Institute of Technology , Surathkal, India
Chirag Nagpal	University of Pune, Maharastra, India
Cormac O'Meadhra	University College Dublin, Ireland
Maggie Oates	Indiana University, USA
Elizabeth Olson	University of Oregon, USA
Jay Patravali	VIT University, Chennai, India
Vinitha Ranganeni	Carnegie Mellon University, USA
Ressa Reneth Sarreal	University of Maryland, Baltimore County, USA
Saul Shanabrook	University of Massachusetts, USA
Joseph Shepley	Columbia University, USA
Neel Shihora	Sardar Vallabhbhai National Institute of Technology, India
Li Shixin	Sichuan University, China
Sudharshan Suresh	National Institute of Technology, Trichy, India
Griffin Tabor	Worcester Polytechnic Institute, USA
Shohei Wakayama	Kyushu University, Japan
Junpeng Wang	Nanjing University of Science and Technology, China
Jiatian Wu	Nanjing University of Science and Technology, China
Sha Yi	The Hong Kong Polytechnic University, China
Qi Zhu	University of Illinois, USA

THANK YOU !
Special Thanks to the RISS Working Papers Journal

Akanimoh Adeleye
University of Oklahoma (USA)

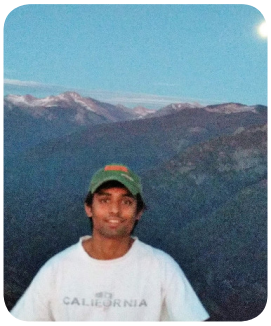


Simon Bloch
Swarthmore College (USA)

Vince Kurtz
Goshen College (USA)

Minae Kwon
Cornell University (USA)

Maggie Oates
Indiana University (USA)

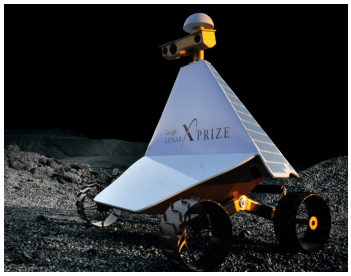


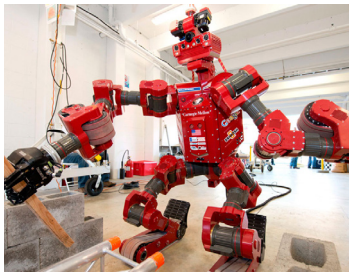
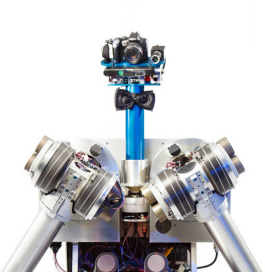
Jay Patravali
Vit University (India)

Sudharshan Suresh
NIT Trichy (India)

Why Join RISS ?

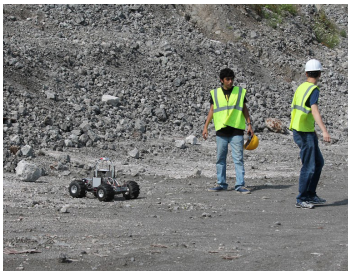
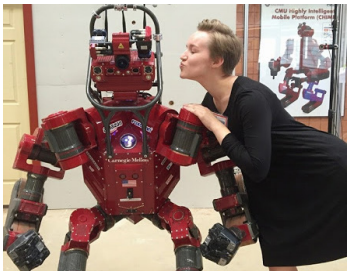
As an RI summer scholar you will be given the opportunity to work on cutting edge robotic research. Some of you will be familiar with your lab or the work you are doing, for others, this will be a new experience entirely.



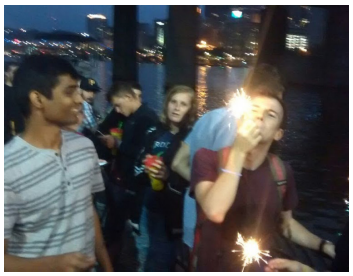
All scholars will work on an individual project and learn how to conduct research at the level of a graduate student. More excitingly, scholars will have a chance to work with and learn from some of the leading experts in robotics. The program ends with a poster session and chance to submit to the RISS Working Papers Journal, where you will also learn how to effectively communicate the work that you've done.

The program is unique in that as long as you have a passion for robotics, all applicants will be equally considered. The program will fund students in most cases, and it is also one of the few robotics programs that accepts international students. Students from smaller schools and schools that don't have robotics programs are considered just as much as students from larger research universities.

When you are applying, it's okay if you don't have extensive experience in robotics many of us did not. It's more important to show that you have a good idea of what kind of research you would be interested in and why you are passionate about it.

The Robotics Institute offers unparalleled resources that you are expected to take advantage of. For any possible topic in robotics that you can think of, it's very likely that you will be able to find someone who is working on it at the RI. The program also tries its best to supply any piece of equipment or technology you might need in completing your project.



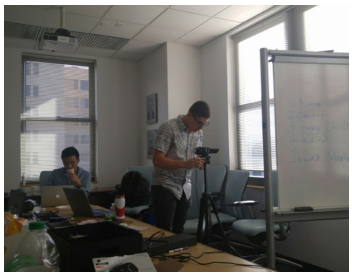


However, remember to immerse yourself not only in your research, but your cohort and the research community. Get to know the projects of your fellow scholars, visits labs, and talk to graduate students and professors about their work. Learn as much as you can about what it means to be a roboticist.

Furthermore, the small size of the RISS cohort will allow for you to get a lot of individual attention and support from the program organizers, research mentors, and fellow cohort members.

When conducting research, we encourage you to delve deep and make it your own. Take initiative and be sure to remember that research is a collaborative process and that you are not alone in your endeavors. Things often take three times longer than you expect, so it is important to be disciplined and schedule your time. How much you can accomplish and gain from RISS will be dependent on this.





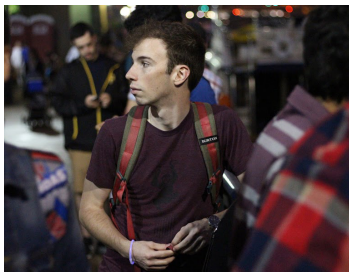

The RISS program is selective and if accepted, you will be a part of a cohort with multi-talented individuals. Take advantage of this and get to know the other amazing scholars. They will enrich you greatly as you do the same. We encourage you to have fun together! Explore Pittsburgh, attend social events, organize sports games. For us, one of the best parts of the RISS program was the community and we hope that it will be the same for you too.

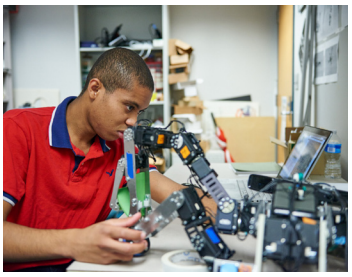
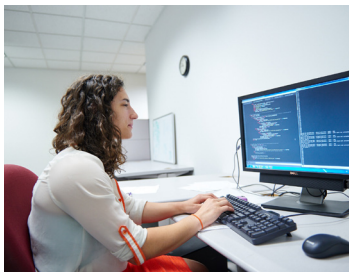
Sincerely

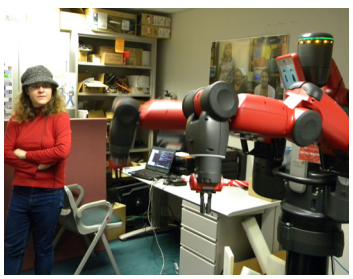
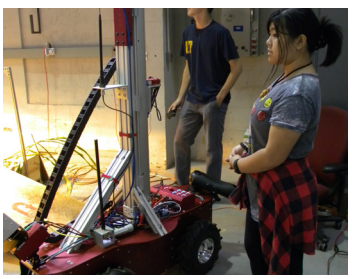
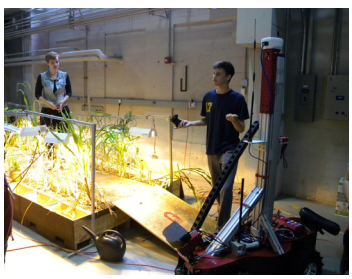
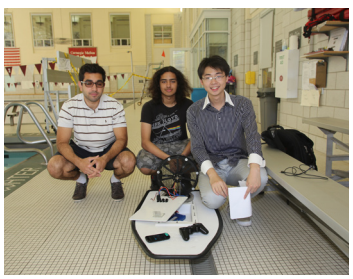
Akanimoh Adeleye Simon Bloch Minae Kwon

RISS Cohort 2016





About the Robotics Institute Summer Scholars Program

Carnegie Mellon's Robotics Institute Summer Scholars (RISS) program (http://riss.ri.cmu.edu/) is an eleven-week summer undergraduate research program that immerses a diverse cohort of scholars in cutting-edge robotics projects that drive innovative and have real-world impact. Launched in 2006, RISS is among the best and most comprehensive robotics research programs for undergraduates in the world.

The quality and breadth of research, high-level of institute and university engagement, extensive professional development curriculum, graduate school application counseling, and alumni network create transformative experiences and remarkable postprogram trajectories.

The RI Summer Scholars Program:

1. Immerses a highly motivated and diverse cohort of students (hereafter referred to as "scholars") in a guided research process;

2. Challenges scholars to build an understanding of research philosophy that serves as a basis for creative problem-solving that transcends the summer research experience;

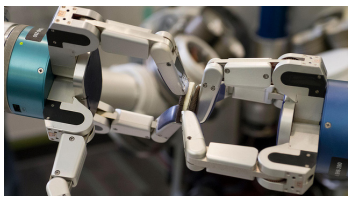
3. Introduces scholars to the highly interdisciplinary nature of robotics and the vast potential to impact and improve the world's quality of life;

4. Provides professional development components that prepare students for successful transitions to graduate school and research careers;

5. Engages scholars in reflective service learning experience that imparts the value of broadening participation and engagement in STEM fields;

6. Inspires scholars to pursue careers in robotics and related fields, such as graduate programs, and equips them with new skills and knowledge;

7. Helps scholars to build collaboration and lifelong connections with a dynamic global community of robotics researchers and entrepreneurs.



The Robotics Institute at Carnegie Mellon University is the largest university affiliated robotics research group in the world. It offers a remarkably diverse breadth of research with an extensive range of applications. With hundreds of active research projects, together with both graduate and undergraduate programs, the Institute is a global leader in robotics research, education, and innovation. The Institute has the nationally recognized research capacity, educational programming, and student development experience to provide, through the RISS program, high-quality research experiences and a developmentally appropriate professional development curriculum to a diverse cohort of undergraduate students.

RISS Core Research Areas:

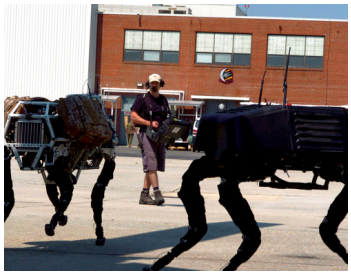
1.**Intelligence**: core AI technologies, motion planning, control theory, planning under uncertainty, POMDPS, game theory, data mining, and machine learning

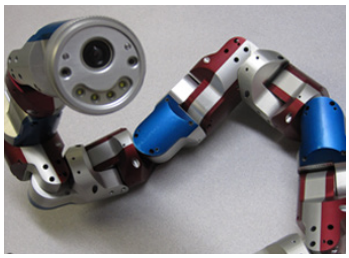
2.**Perception**: computer vision, stereo processing, understanding ladar and 3D sensing, state-estimation, and pattern recognition

3.**Action**: work mechanisms, actuators, their design and control

Recent sample scholar projects have included:

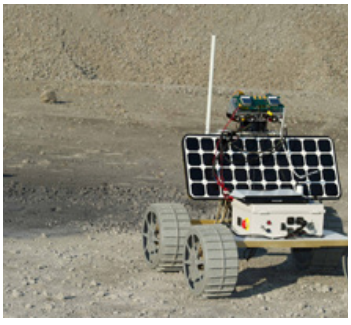
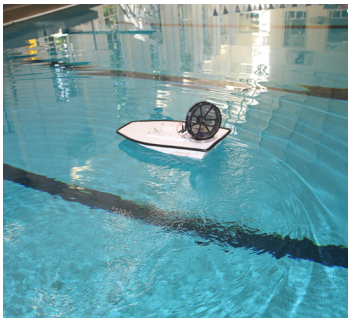
- 3D Manufacturing of a Liquid Metal Microchannel for Soft Sensing
- Autocalibration and Hybrid Position Control for the Cerberus Cardiac Robot
- Autonomous Object Recovery in Manipulation Experiments
- Design and Characterization of Map Based Lunar Rover Localization
- Generating Spatial Paths to Express Attentional Attitudes
- Highly Flexible and Stretchable Sensor Using Soft Optical Waveguides



- Improving Power and Vision Systems on Autonomous Quadrotors
- Monocular Visual Features for Fast Flight Through Forests
- New Visual Programmer Converter that Allows the Hummingbird
- Persistent Deployment of Micro-Aerial Vehicles
- Pothole Detection with Cell Phone Data
- Trajectory Following in GPS Denied Environments for UAVs
- Using Receding Horizon Control
- Visual Programmer and New Efficient File System

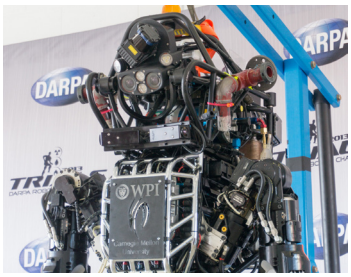
The RISS program also has a remarkable mentor and alumni community with wide participation and support from across the university. In 2016, over 31 researchers and professionals contributed to the RISS curriculum - presenting workshops, laboratory tours, and other programming elements. Over 50 members of the Robotics Institute (RI) community participated as research mentors in the same period.

Unique partnerships with robotics intelligent systems, and data science companies extend the scholars exposure from the robotics lab to product development, commercialization, and technology transfer.

Scholars benefit from RI's institutional experience from hosting undergraduate research programs for over nine years and an alumni network of over 200 previous undergraduate researchers now in graduate programs, academia, and industry both domestically and globally.



In the fall of 2016, 33 RISS Program alumni will be attending CMU graduate programs in both Masters and PhD programs (29 within the School of Computer Science) or will be working as technical staff at the Robotics Institute. Also in the fall of 2016, a core group of RISS alumni are launching Pittsburgh-based alumni programming. This extensive community helps scholars successfully prepare for graduate school and maximize the summer experience.

One of the program's strategic priorities is to extend access to robotics research opportunities to students from underrepresented groups and those from higher education institutions with fewer research opportunities. Human capital is one of the most important resources driving today's knowledge economy.

RISS connects a diverse group of talented undergraduate students from around the world to the Carnegie Mellon University community. The valuable contributions and connections that summer scholars make continue beyond this eleven week summer program.

A Note From The 2016 Cohort

The Robotics Institute Summer Scholars program over the summer 2016 has been an intense 11 weeks of robotics research, packed with a schedule comprising of various technical events and social activities. The large diversity of students groups from across the world, gives an opportunity for the scholars to interact with each other and share their culture and background.

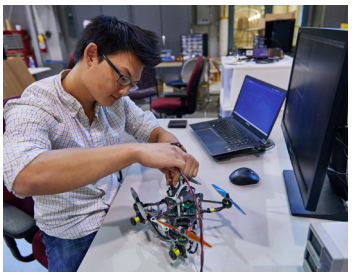
The RISS programming comprised of the weekly lunch seminars where professors and scientists working at RI were invited to present their research with the scholars. The seminar followed by engaging discussions were quite helpful and provided the exposure to the wide variety of robotics research being carried out at RI. The scholars got a chance to interact with the best minds in robotics and discover newer areas of research beyond their domain.


An interesting part about this year's cohort was the active involvement of all the scholars in organizing and planning program events. Some highlights of this include scholars presenting their research to local students from Pittsburgh, hosting a delegation of high school students from Mexico. We also witnessed few scholars taking initiative for improving the program structure, holding brainstorming sessions for a revised RISS journal format and the RISS website.

As part of broader impact and social outreach, we had members of the cohort planning a visit to meet Mayor of Pittsburgh, Mr Bill Peduto. Key social issues like legal reforms and impact of robotics on jobs were raised.

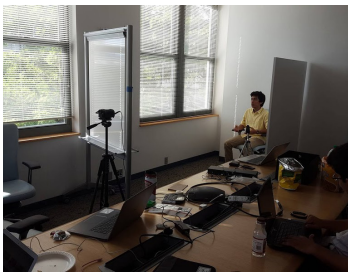
The students lead discussions on how public policy would be framed public policy with the advent of robotics technology, and how is Pittsburgh as a city is taking steps for the future and adapt to the changes in industry.

We also are proud that many of the scholars were able to push through with their research and submit their work at top tier conferences for publications. Summarizing this journey, we can conclude that the cohort of RISS 2016, collectively and individually were able to contribute towards the success of this program. As put in the words of RISS'14 alumnus Shushman Choudhury, "We have raised the bar higher for the future scholars".



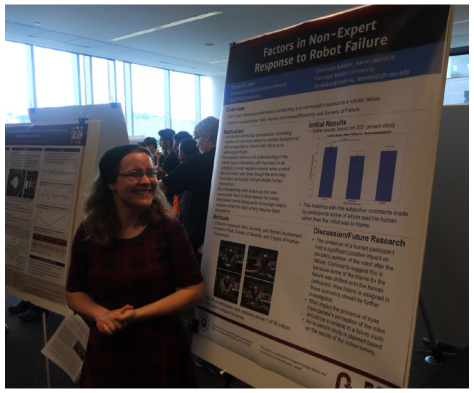


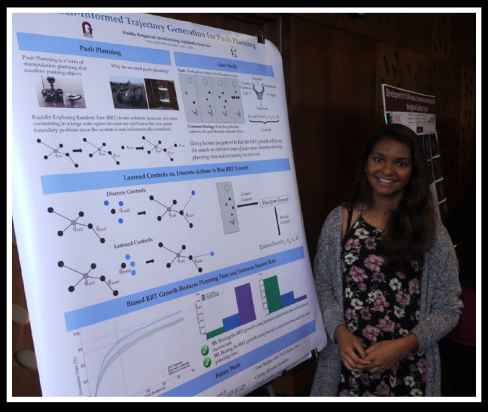


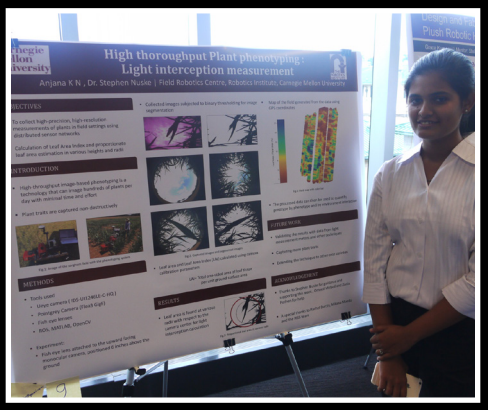



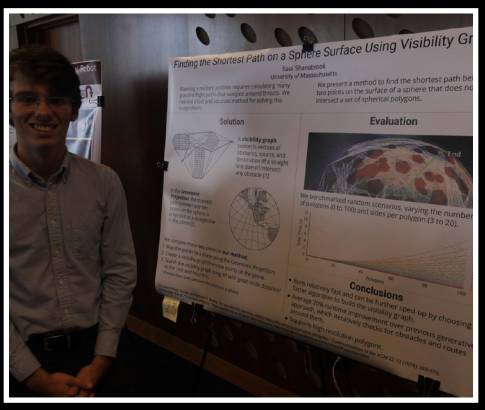


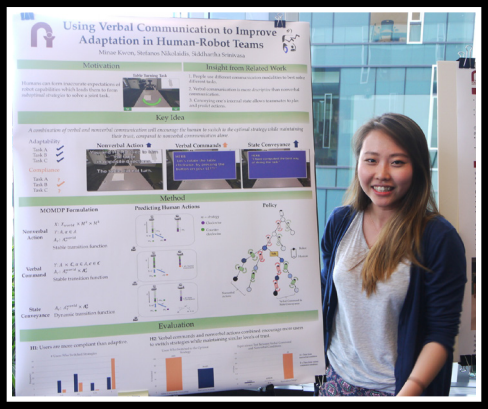
RISS Cohort Gallery 2016

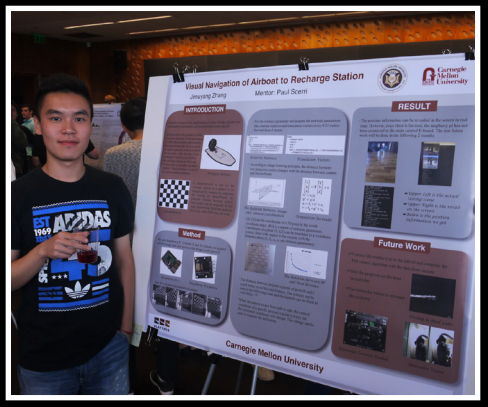


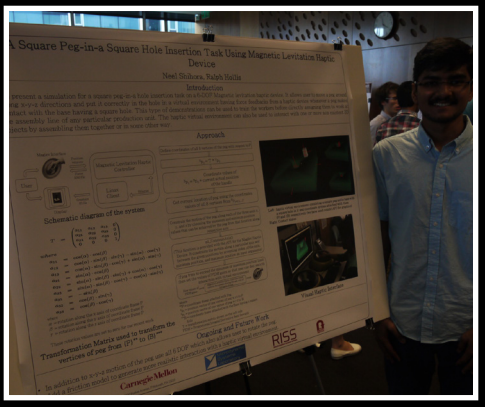

Push Planning
Learned Controls vs. Discrete Actions in Rss RRT Growth

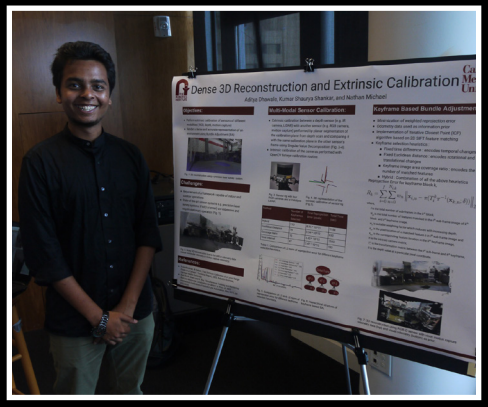

High thoroughput Plant phenotyping:
Light interception measurement
Anjana K N , Dr. Stephen Nuske | Field Robotics Centre, Robotics Institute, Carnegie Mellon University
OBJECTIVES
INTRODUCTION
METHODS
RESULT
FUTURE WORK
ACKNOWLEDGMENT

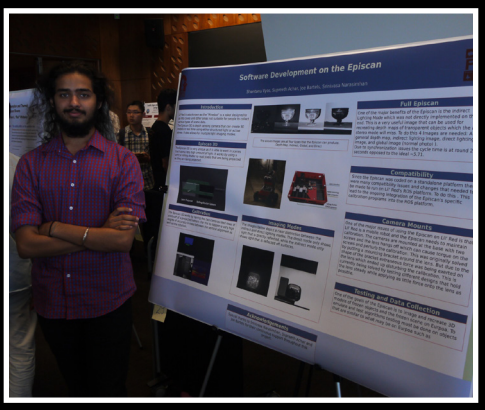

Finding the Shortest Path on a Sphere Surface Using Visibility Gr
Saul Shanabrook
University of Massachusetts
Solution
Evaluation
Conclusions

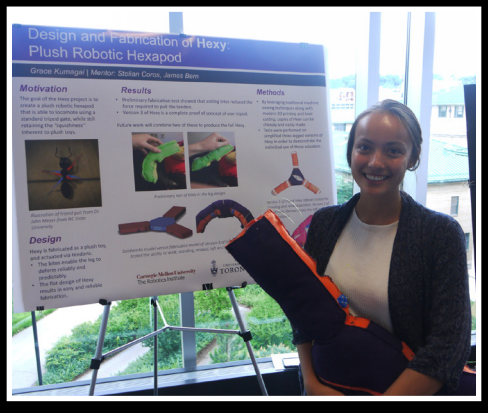

Using Verbal Communication to Improve Adaptation in Human-Robot Teams
Motivation
Insight from Related Work
Key Idea
Method
Evaluation

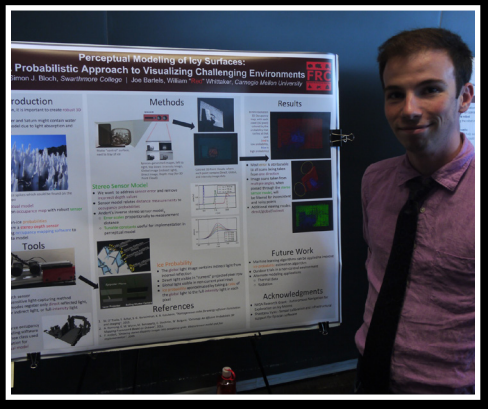

Visual Navigation of Airboat to Recharge Station
Jinyang Zhang
Mentor: Paul Scerri
Carnegie Mellon University
INTRODUCTION
RESULT
Method
Future Work

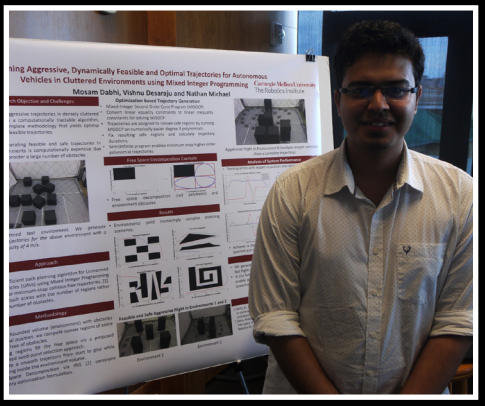

A Square Peg-in-a Square Hole Insertion Task Using Magnetic Levitation Haptic Device
Neel Shihora, Ralph Hollis
Introduction
Approach
Schematic diagram of the system
Conclusion and Future Work
RISS

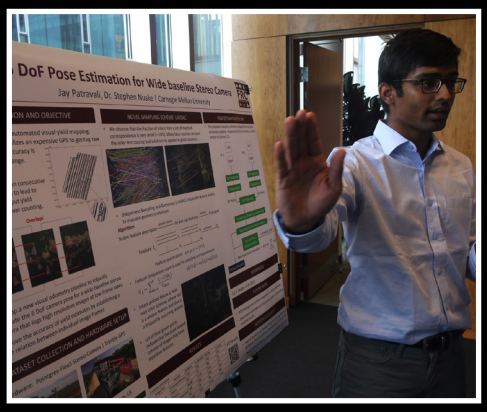

Dense 3D Reconstruction and Extrinsic Calibration
Multi-Modal Sensor Calibration:
Keyframe Based Bundle Adjustment

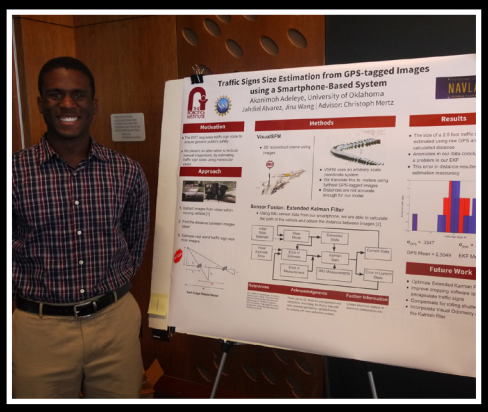

Software Development on the Episcan
Full Episcan
Compatibility
Camera Mounts
Motion Capture
Method and Data Collection

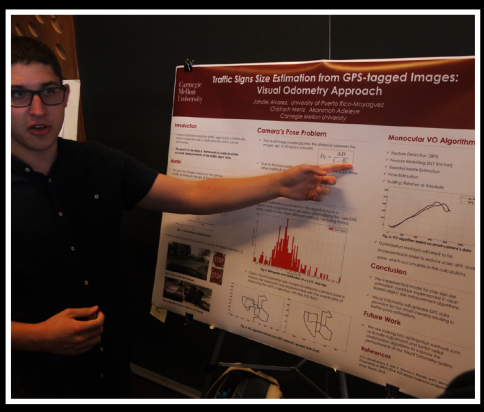

Design and Fabrication of Hexy
Plush Robotic Hexapod
Grace Kumaiyai | Mentor: Stollan Corrie, James Bern
Motivation
Results
Methods
Design

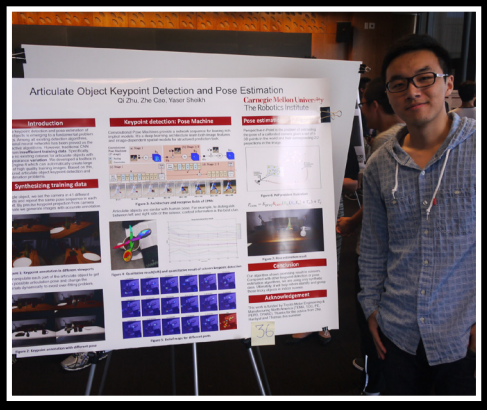

Perceptual Modeling of Icy Surfaces:
Probabilistic Approach to Visualizing Challenging Environments
Simon J. Bloch, Swarthmore College | Joe Bartels, William "Red" Whittaker, Carnegie Mellon University
Introduction
Methods
Results
Tools
Future Work
Acknowledgments
References

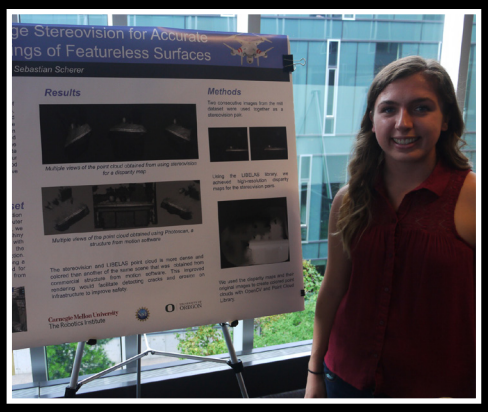

Aggressive, Dynamically Feasible and Optimal Trajectories for Autonomous Vehicles in Cluttered Environments using Mixed Integer Programming
Carnegie Mellon University
The Robotics Institute
Mosam Dabhi, Vishnu Desaraju and Nathan Michael
Optimization Based Trajectory Generation
Free Space Decomposition Concept
Results
Approach
Methodology

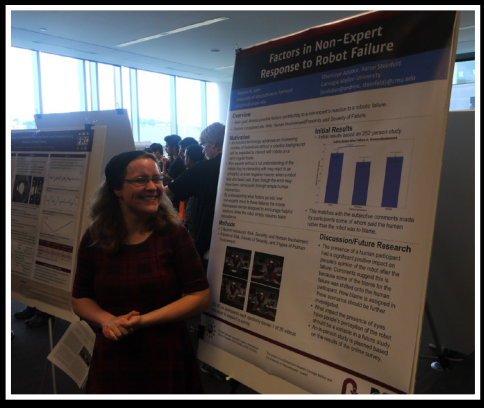

DoF Pose Estimation for Wide baseline Stereo Camera
Jay Patravali, Dr. Stephen Nuske | Carnegie Mellon University

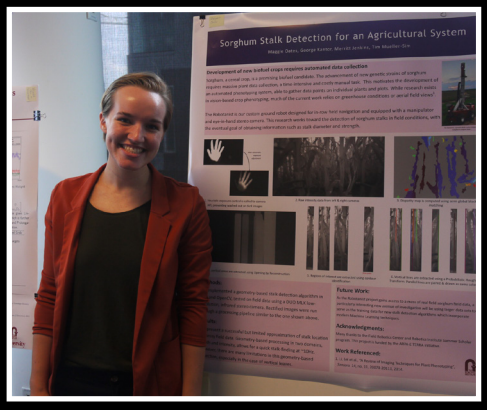

Traffic Signs Size Estimation from GPS-tagged Images using a Smartphone-Based System

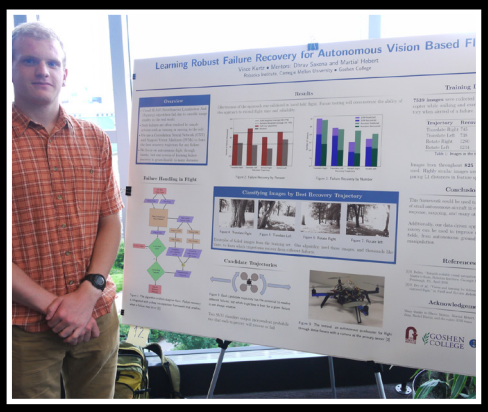

Traffic Signs Size Estimation from GPS-tagged Images: Visual Odometry Approach

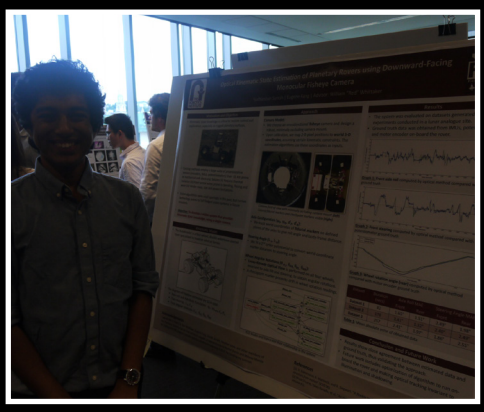

Articulate Object Keypoint Detection and Pose Estimation
Qi Zhu, Zhe Cao, Yaser Sheikh
Carnegie Mellon University
The Robotics Institute

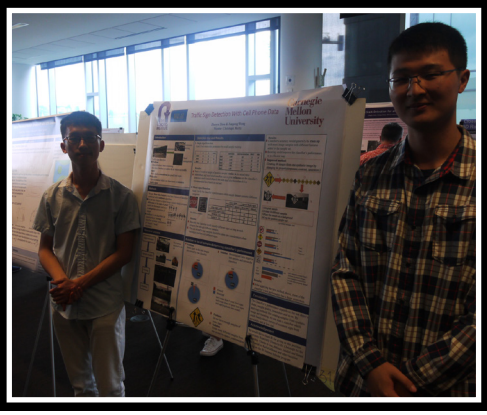

Results
Methods

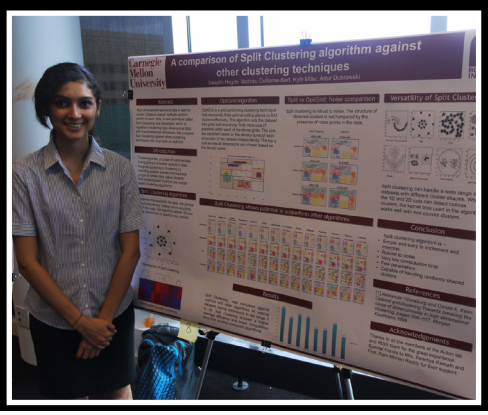

Factors in Non-Expert Response to Robot Failure

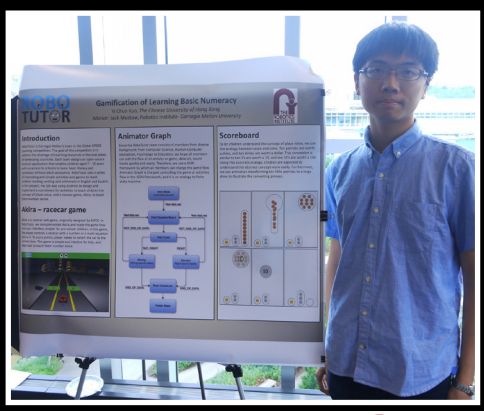

Sorghum Stalk Detection for an Agricultural System

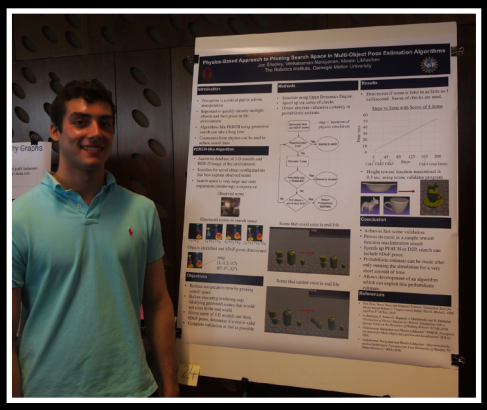

Learning Robust Failure Recovery for Autonomous Vision Based

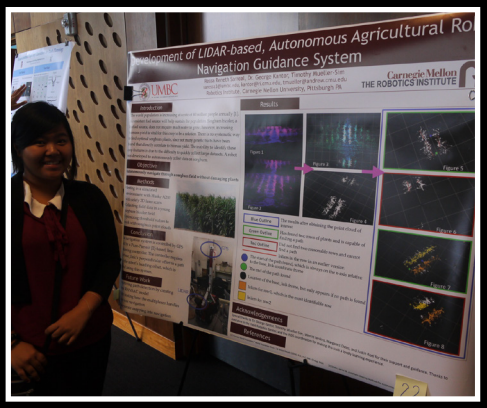


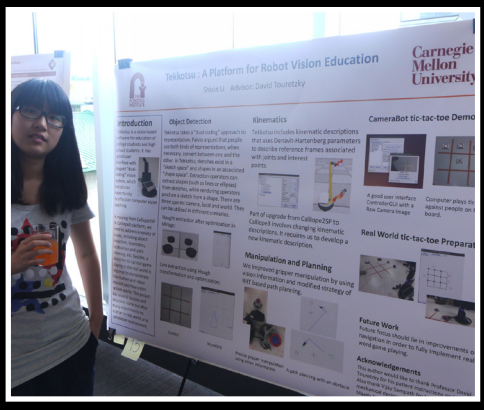

Carnegie Mellon University

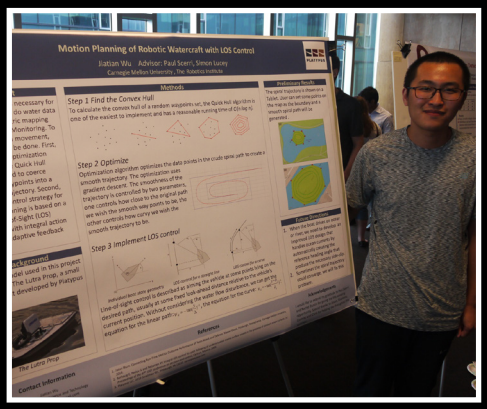

A comparison of Split Clustering algorithm against other clustering techniques

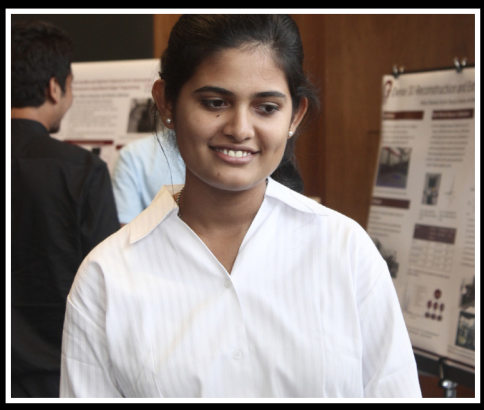

ROBO TUTOR
Gamification of Learning Basic Numeracy

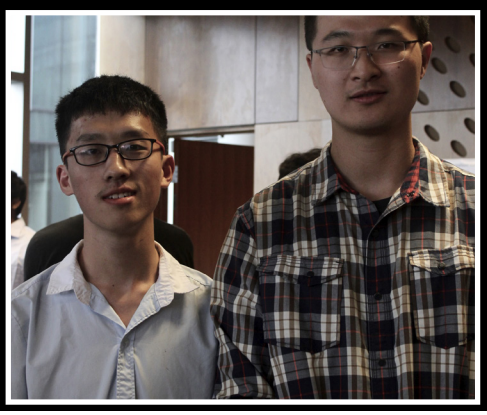

Physics-Based Approach to Picking Search Space in Multi-Object Pose Estimation Algorithms

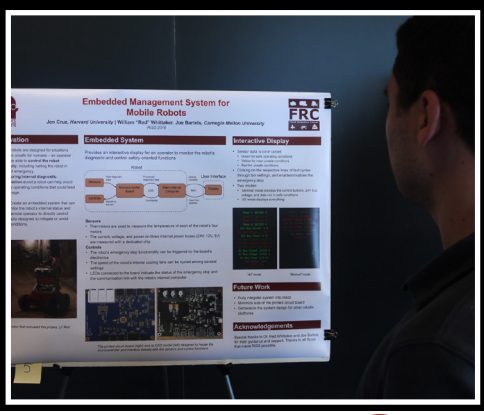

Navigation Guidance System
Carnegie Mellon
THE ROBOTICS INSTITUTE


Tekkotsu : A Platform for Robot Vision Education
Carnegie Mellon University


Motion Planning of Robotic Watercraft with LOS Control
Jiatian Wu Advisor: Paul Scerri, Simon Lucey
Carnegie Mellon University, The Robotics Institute
Step 1 Find the Convex Hull
Step 2 Optimize
Step 3 Implement LOS control






Embedded Management System for Mobile Robots
Embedded System
Interactive Display
Future Work
Acknowledgements





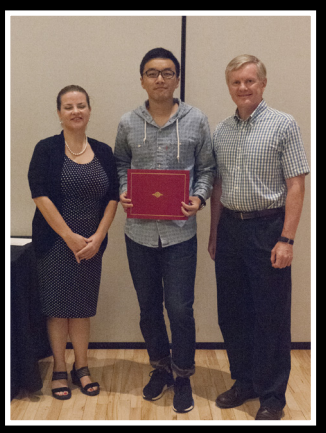


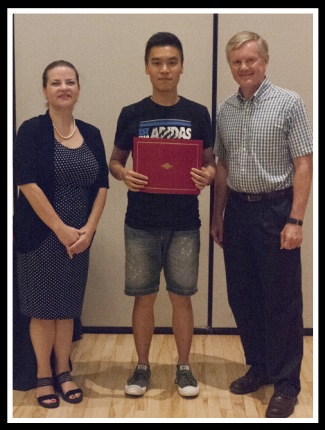




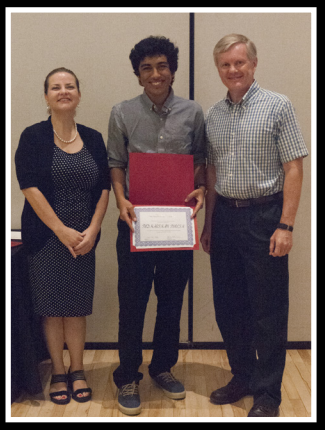




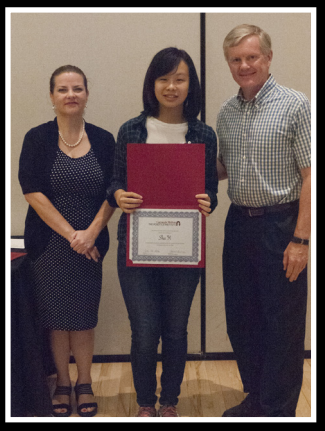




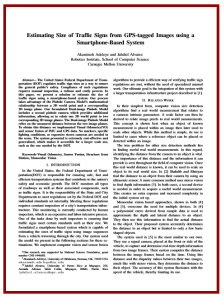








WORKING PAPERS
Authors


Estimating Size of Traffic Signs from GPS-tagged Images using a Smartphone-Based System

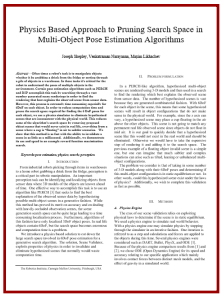
1. Akanimoh Adeleye and Jahdiel Alvarez

Estimating Size of Traffic Signs from GPS tagged Images Using a Smartphone Based System


Comparison of Split Clustering with other clustering techniques

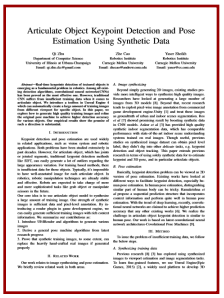
2. Deepthi Hegde, Mathieu Guillame-Bert, Kyle Miller and Artur Dubrawski

Comparison of Split Clustering with Other Clustering Techniques


Physics Based Approach to Pruning Search Space in Multi-Object Pose Estimation Algorithms

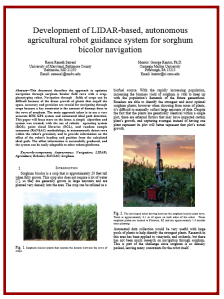
3. Joseph Shepley, Venkatraman Narayanan and Maxim Likhachev

Physics Based Approach to Pruning Search Space in Multi-Object Pose Estimation Algorithms


Articulate Object Keypoint Detection and Pose Estimation Using Synthetic Data

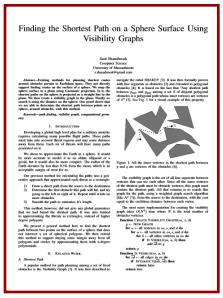
4. Qi Zhu, Zhe Cao and Yaser Shaikh

Articulate Object Keypoint Detection and Pose Estimation Using Synthetic Data


Development of LIDAR-based, autonomous agricultural robot guidance system for sorghum bicolor navigation

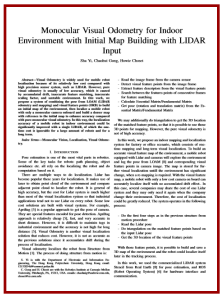
5. Ressa Reneth Sarreal and George Kantor

Development of LIDAR-based, Autonomous Agricultural Robot Guidance System for Sorghum Bicolor Navigation


Finding the Shortest Path on a Sphere Surface Using Visibility Graphs

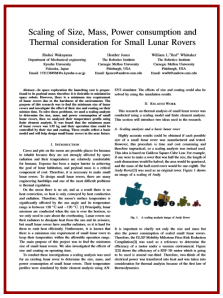
6. Saul Shanabrook

Finding the Shortest Path on a Sphere Surface Using Visibility Graphs


Monocular Visual Odometry for Indoor Environment with Initial Map Building with LIDAR Input

7. Sha Yi, Chaohui Gong and Howie Choset

Monocular Visual Odometry for Indoor Environment with Initial Map Building with LIDAR Input


Scaling of Size, Mass, Power consumption and Thermal consideration for Small Lunar Rovers

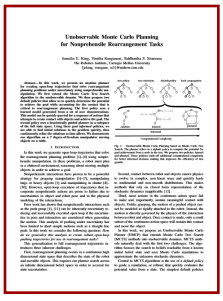
8. Shohei Wakayama, William L. “Red” Whittaker and Heather Jones

Scaling of Size, Mass, Power Consumption and Thermal Consideration for Small Lunar Rovers


Optical Kinematic State Estimation of Planetary Rovers using Downward-Facing Monocular Fisheye Camera

9. Sudharshan Suresh, William L. ”Red” Whittaker and Eugene Fang

Optical Kinematic State Estimation of Planetary Rovers Using Downward Facing Monocular Fisheye Camera


Unobservable Monte Carlo Planning for Nonprehensile Rearrangement Tasks

10. Jennifer E. King, Vinitha Ranganeni and Siddhartha S. Srinivasa

Unobservable Monte Carlo Planning for Nonprehensile Rearrangement Tasks


Traffic sign detection with cell phone data

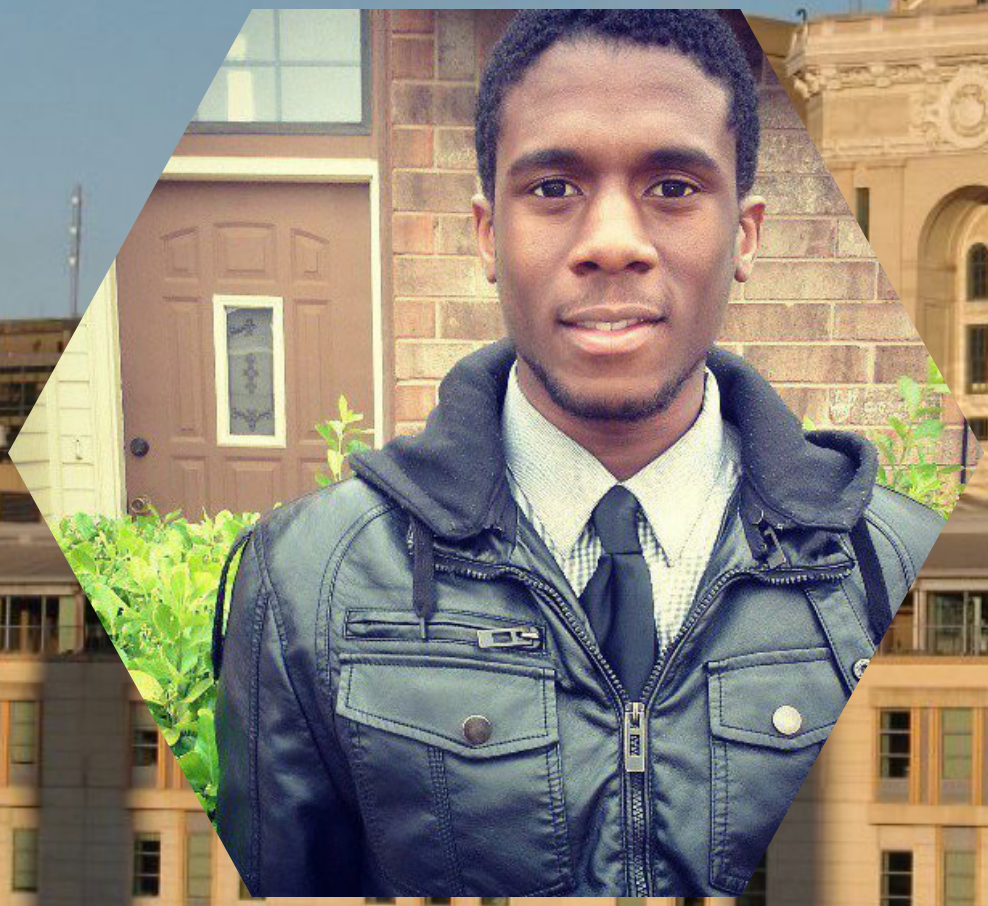
11. Zhenyu Zhou, Junpeng Wang and Christoph Mertz

Traffic sign detection with Cell Phone ata

Akanimoh Adeleye

University of Oklahoma

The Federal Department of Transportation regulates traffic sign sizes as a way to ensure the general public's safety.



https://www.linkedin.com/in/akanimoh-sanmi-adeleye-0a3821a3

Estimating Size of Traffic Signs from GPS-tagged Images using a Smartphone-Based System

Akanimoh Adeleye and Jahdiel Alvarez
Robotics Institute, School of Computer Science
Carnegie Mellon University

***Abstract*— The United States Federal Department of Transportation (DOT) regulates traffic sign sizes as a way to ensure the general public's safety. Compliance of such regulations requires manual inspection, a tedious and costly process. In this paper, we present a solution to estimate the size of traffic signs using a smartphone-based system. Our process takes advantage of the Pinhole Camera Model's mathematical relationship between a 3D world point and a corresponding 2D image plane. Our formulated Dual-image Pinhole Model includes a second pinhole camera which provides additional information, allowing us to relate one 3D world point to two corresponding 2D image planes. The Dual-image Pinhole Model relies on the measured distance between the two image planes. To obtain this distance we implemented Visual Odometry(VO) and sensor fusion of IMU and GPS data. No markers, specific lighting conditions, or expensive stereo cameras are needed in the scene. The system presented is extremely cost-effective and generalized, which makes it accessible for a larger scale use, such as the one needed by the DOT.**

Keywords-Visual Odometry, Sensor Fusion, Structure from Motion, Monocular Vision

I. INTRODUCTION

In the United States, the Federal Department of Transportation(DOT) is responsible for ensuring safe, fast and efficient transportation systems that contribute to the nation's safety and economic growth. The DOT monitors all types of roadways as well as their associated components, such as traffic signs. It is the responsibility of the State and City Departments to meet regulations set by the Federal DOT and individual standards set internally. Meeting these regulations requires constant inspection of a city's transportation infrastructure. This monitoring is currently conducted by human inspectors, which is an expensive and labor-intensive process. One of the tasks done by such inspectors is ensuring that traffic signs meet certain standards such as their size. We formulated a smartphone-based system which is capable of estimating the sizes of traffic signs using image sequences from a vehicle traversing through the city. Our system used the model we present in the following sections for these estimations. We implement computer vision and sensor fusion algorithms to provide a efficient way of verifying traffic sign regulations are met, without the need of specialized manual work. Our ultimate goal is the integration of this system with a larger transportation infrastructure project described in [1].

*This research was supported by National Science Foundation, grant number CNS-1446601.

[I] Akanimoh Adeleye, University of Oklahoma, Norman, OK, 73019 USA akanimoh.adeleye@ou.edu

[II] Jahdiel Alvarez, University of Puerto Rico at Mayagez, 259, Avenida Alfonso Valds Cobin, Mayagez, 00681, Puerto Rico jahdiel.alvarez@upr.edu

II. RELATED WORK

In their simplest form, computer vision size detection algorithms find a real world measurement that relates to a cameras intrinsic parameters. A scale factor can then be derived to relate image pixels to real world measurements. This concept is shown best when an object of known measurement is placed within an image then later used to scale other objects. While this method is simple, its use is limited to cases where a reference object can be placed or detected within an image.

The true problem for other size detection methods lies in finding useful real world measurements. In this regard, identifying the distance from the camera to an object is vital. The importance of this distance and the information it can provide is seen throughout the field of computer vision. Once this real world distance is known, it can be used to scale an object to its real world size. In [2] Shahdib and Bhuiyan find the distance to an object from their camera by using an ultrasonic sensor. A more common method uses stereo vision to find depth information [3]. In both cases, a second device is needed in order to acquire a useful world measurement. This creates an extra expense and increased complexity in the initial system set up.

Monocular vision based approaches, shown in both [4] and [5], overcome the need for multiple devices. In [4] a polynomial curve derived from sample data is used to approximate the depth and lateral distance to an object. They then use this information to find the actual distance to the object. Their proposed system accurately determines the distance to an object but is limited to only a few basic shaped objects.

The system used in [5] is the most similar to our own. They use a signal camera, placed at the front or side of the vehicle, to capture and determine real-time depth information from two image frames. Their system computes the distance between the image frames based on the time. Using this distance and the disparity values between their two images, much like with stereo vision, they determine the distance to their object. The accuracy of this system fluctuates with the speed of the vehicle, thereby limiting its use.

The novelty of our system is first shown in the mounting of our smartphone camera within our vehicle. This reduces cost and complexity by eliminating the need to weatherproof our camera. We use an approach independent of time to calculate the distance between the two sequential images we capture. Our formulated Dual-image model then relates this distance to our camera's intrinsic parameters, finding the distance to our object for the necessary scaling.

III. Data Collection

Fig. 1. Image capturing system.

The Samsung Galaxy Camera was our main device in data collection [1]. Our system is mounted on the windshield (Fig. 1) of the vehicle. A mobile app is used for the collection of the information. Eventually, this setup can be placed on city vehicles, which traverse through the city, providing up-to-date information about the status of the infrastructure.

IV. Technical Approach

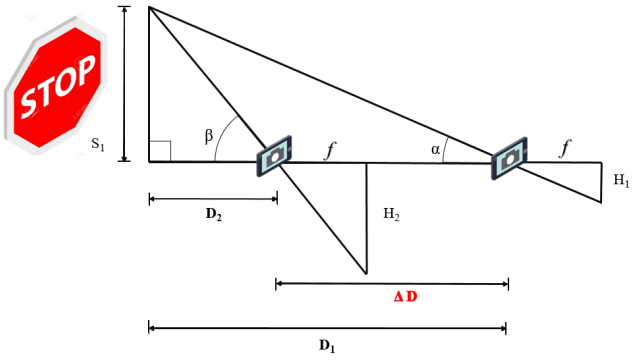


Fig. 2. Dual-image model.

Our Dual-Image Pinhole model is shown in Figure 2. It is the geometric model used to detect the size of an object from two images once ΔD, the distance between successive images, is known.

A. Dual-Image Pinhole Model

The model assumes that the focal length (f) is known and is constant throughout the processed images. The images are also assumed to be taken sequential with a change in the position of where the images were taken. Based on the Pinhole-camera model [1] (Fig. 3) the size of an object in an image can be determined by the use of similar triangles, $S = \frac{hd}{f}$. The two parameters, h and f are known in our model meanwhile, d is unknown; h, f, d being the size of the cropped traffic sign, the focal length of the images, and the distance from the camera to the traffic sign, respectively.

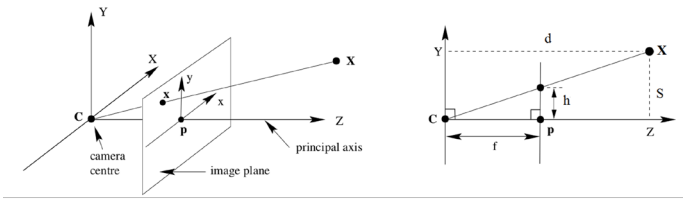


Fig. 3. Pinhole-Camera Model.

Given that the distance from the camera to the sign is an unknown we resorted to utilize two consecutive images, resulting in a new geometrical model shown in Fig 2. Solving for D_1 is possible using only the aforementioned parameters and ΔD, the physical distance between where the images were taken. Following the mathematical steps taken from Figure 2 results in:

$$S_1 = \frac{H_1 D_1}{f} = \frac{H_2 D_2}{f} \tag{1}$$

$$D_2 = D_1 \frac{H_1}{H_2}$$

Because we were able to write D_2 in terms of D_1 we could solve for D_1 :

$$\Delta D = D_1 - D_2$$

$$\Delta D = D_1 - D_1 \frac{H_1}{H_2}$$

$$D_1 = \frac{\Delta D}{1 - \frac{H_1}{H_2}} \tag{2}$$

Once we solve for D_1 we can solve for S_1 using similar triangles, which would result in the estimated size of the object, in this case the traffic signs. The accurateness of our model relied, mainly, on the accurateness of the measurement of ΔD,therefore the main effort was focused on obtaining an accurate estimation of the distance between the positions of the images.

To gauge the accuracy of our dual-image model, we first conducted a simple ground truth test. In both the controlled environment of our lab and the unstructured environment of the outdoors, we took two sets of pictures. Each picture captured a stop-sign of known size, which was used as ground truth and later compared to our estimates. We got ΔD by using a measuring tape to measure the distance between where we took the images. We then manually cropped these images to fit the size of the stop-signs, since it is the object we want to scale. We chose to do this manually in order to make our initial experimental test as controlled as possible.

We determined the height H1 and H2 by the pixel height of our cropped images. Providing these images and the known parameters into our algorithm, we received estimated values within 0.027 meters of the true size of our stop-signs. A percentage error of 3.54%.

Thus, the heart of our work became accurately determining the distance between any two images we wished to use for our model.

B. Global Positioning System (GPS)

To determine the distance between our two images, we first tried to simply use the smart-phone's GPS coordinates and the Haversine formula to calculate distance. We soon found this method unreliable due to the GPS system within our smart-phone. Standard receivers in smart-phones have an accuracy within 7.8 meters of a true location [6]. While this is useful for most general applications, it is not nearly precise enough for our system.

C. VisualSFM

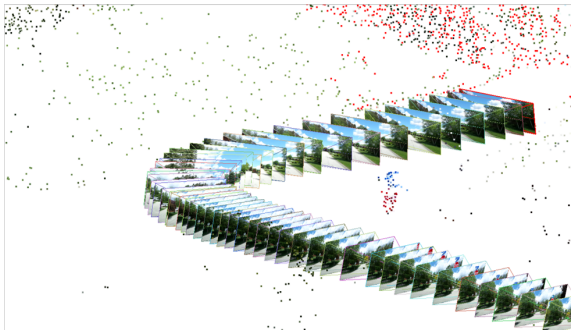


Fig. 4. VSFM sparse reconstruction.

The methods described in this sections and ones below were tested and implemented as a way to fix this inaccuracy within our GPS. VisualSFM stands for Visual Structure from Motion [7]. It is a free software for 3D reconstruct through Structure from Motion. As a part of the reconstruction procedure, VSFM determines the location and orientation of all photos used relative to one another (see figure 4). The goal of our first approach was to extract and use the calculated distance between our images that VSFM provides. To do this, we first had to scale VSFM's arbitrary coordinate system to a real world measurable distance.

To understand of the capabilities of VSFM we first took pictures of stop signs outside of our vehicle setup, simply to see how well they would reconstruct. Our pictures were taken on a linear track, as if our camera was inside our vehicle driving towards a stop sign. We took care to ensure the orientation of our camera was about the same as within our vehicle. As expected, our pictures produced minimal reconstructions. Feature matching is intrinsic to SFM and pictures taken linearly only have one projection view. A good reconstruction requires two or more projections of the same 3D point [9] [8].This however did not impact our objectives. As long as we could accurately reconstruct a part of our scene using images containing the stop sign, then no matter the quality of the reconstruction, we would gain calculated distances between our images.

These calculated distances were the direct product of SFM and as a result, were based on an arbitrary scale. To scale these distances to real world measurements, we created a scale factor using the distance between the furthest apart VSFM reconstructed images and their corresponding GPS distance. To reduce complexity, we transformed our GPS coordinate system to Universal Transverse Mercator (UTM) coordinates and calculated the euclidean distance between our two points. By using the furthest apart possible GPS tagged images for our reconstruction, we had the smallest relative error in our GPS distance. We applied this scale factor to our VSFM coordinate system then chose a different set of images and their newly scaled distance, for our model. Figures 5 through 8 show this process.

We found that VSFM was able to accurately determine the distance between our images, within a meter and a half of the true distance. Although this was still not the accuracy we wanted, it was enough for us to effectively cluster the size of an estimated stop sign to the true sign size. When testing this method using images collected from within our vehicle, we found that our results were not as accurate as our test set.

Due to variables inherent to our vehicle setup, such as cases where excessive vegetation was visible or a rolling shutter effect was seen, images extracted from within our vehicle failed to reconstruct more often than our test sets. These variables also corrupted the feature matching capabilities of VSFM, causing successful reconstructions to inaccurately determine the location and orientation of our images, thereby increasing the distance error rather than minimizing it. Large moving objects such as other vehicles on the road also contributed to corrupting the feature matching of VSFM.

There were a few methods we thought of to eliminate these limitations, such as minimizing the rolling shutter effect through stabilization techniques or filtering vegetation and larger moving objects. Due to the time constants of our work, we ultimately decided not to purse either of these methods and instead shift our effort to both the Visual Odometry and sensor fusion methods described below.

D. Monocular Visual Odometry (VO)

The designed monocular visual odometry system implemented the basic VO 2-D-to-2-D correspondences algorithm presented in [10]. The system processed image sequences captured from a moving ground vehicle, in our experiments from a sedan car. The implemented algorithm is composed of seven steps.

1) Capturing Frames: A Samsung Galaxy Camera was calibrated for the purposes of this project. The frames in use are the ones we obtain from the data collection, the resolution of each frame was of 1920x1080 px. The algorithm iterated through the dataset and processed each frame individually. The camera intrinsic matrix was obtained for the camera calibration process and therefore we used it for the camera's pose estimation.

Fig. 5. Sample input image taken from vehicle set up. For this run we used 12 gps-tagged images for reconstruction

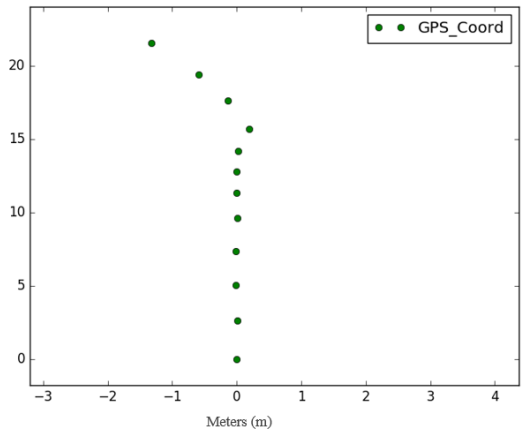


Fig. 6. Plot of images'location by GPS distance.

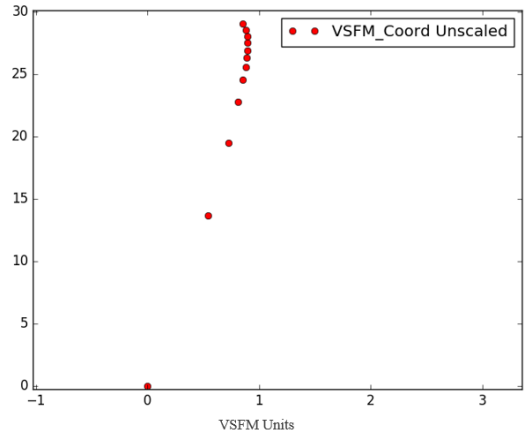


Fig. 7. Plot of images'location by VSFM distance.

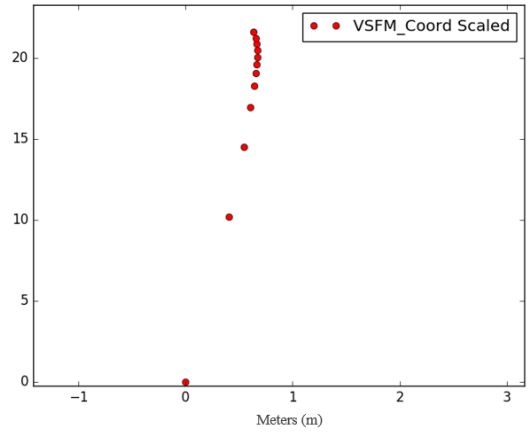


Fig. 8. Plot of images'location by scaled VSFM distance.

2) Feature Detection and Feature Matching: Scale Invariant Feature Transform (SIFT) [11] was the adopted feature detector because of its robustness and precision to outperform the other detectors in a visual odometry application, as shown in [9]. The Kanade-Lucas-Tomasi (KLT) tracker [12] was used for feature matching, given its great performance in optical flow and visual odometry scenarios, due to its efficiency with consecutive image samples. The features captured by SIFT are tracked along a window of images until the number of tracked features is less than a certain threshold, where new SIFT features are computed. The feature correspondence provided by the KLT tracker is taken for the estimation of the essential matrix.

3) Essential Matrix Estimation: Every image has a projective matrix which maps a point in the 3-D world space to a 2-D image plane, $x = PX'$. This matrix is described as $P = K[R|t]$, where K is the Camera Intrinsic Matrix, obtained by calibrating the camera, and $[R|t]$ the essential matrix, which is composed of a rotation matrix and translation vector. Through feature correspondence and the following constraint, $(y')^T Ey = 0$, E can be calculated, y' and y being image coordinates which correspond to the same 3-D point in the world scene. We calculate the essential matrix for each subsequent image pair, $I_{k-1}$and I_k.

4) Rotation and Translation from Essential Matrix: The essential matrix is decomposed into the rotation matrix, R_k and the translation vector, t_k. The translation vector t_k, gives the relative movement of the camera from one image to another, producing the trajectory seen in our visual odometry implementation.

5) Relative and Absolute Scale: Because the implemented system is monocular, the translations are up to scale, therefore it is only possible to calculate a relative scale from one image to another. We used GPS data in order to obtain an absolute scalar for which we could scale the trajectories provided by the concatenation of the translation vectors.

6) Absolute Pose and Repetition: By concatenating the scaled transformation matrices we are able to calculate the whole trajectory of the vehicle. The absolute scalar was used to approximate the distance between images. The algorithm is repeated from 1, as new images are processed.

The OpenCV library was utilized for the implementation of the algorithm using Python. The resulting system produces clearer trajectories when the lens distortion in the images is removed. KITTI Vision Benchmark Suites odometry dataset was used to test our monocular VO implementation along with data obtained by driving through Pittsburgh. Figure 9 shows the performance of the py-MVO, name of our implementation, in comparison to the GPS data, taken from

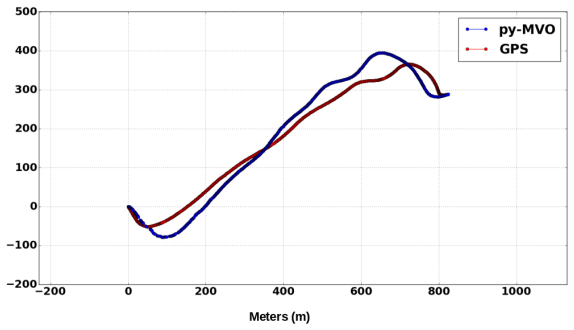


Fig. 9. Monocular VO compared to GPS trajectories

a data set of the city Pittsburgh, PA.

E. Sensor Fusion: Extended Kalman Filter

Using an Extended Kalman Filter (EKF) [13] [14] we fused the gyroscope and accelerator Inertial Measurement Unit (IMU) sensor data received from our smart-phone, with our GPS data; reducing the error in localization. This allowed us to plot the path of our vehicle and match our GPS-tagged images to specific approximated points along our path. We could then calculate the distance between these new approximated points and use them within our model.

We decided to use the Kalman Filter based on the type of sensor data we were receiving [15]. We implemented and modified the EKF from a previous project in NavLab, denoted in [16]. Figure 10 shows a basic diagram representation of the filter.

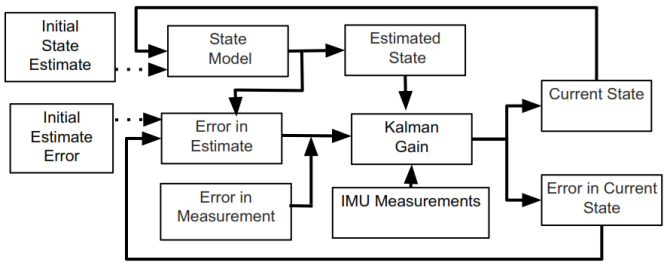


Fig. 10. Extended Kalman Filter Model.

V. Results

The standard sizes of stop signs in the Pittsburgh area are 0.610 meters (in width and height), 0.762 meters, and 0.914 meters. 0.914 meters are the new regulatory standard size that the city is required to meet. We tested our EKF method on 15 sets of image data, all containing a stop-sign of .762 meters. We tested this size specifically because it was in between the other two standard sizes. The case shown in Figure 11 is an example of the EKF Estimated GPS and Raw GPS paths of our vehicle plotted simultaneously. We compared the accuracy of our EKF estimated distances to that of our GPS coordinates. Figure 12 shows a histogram of our results.

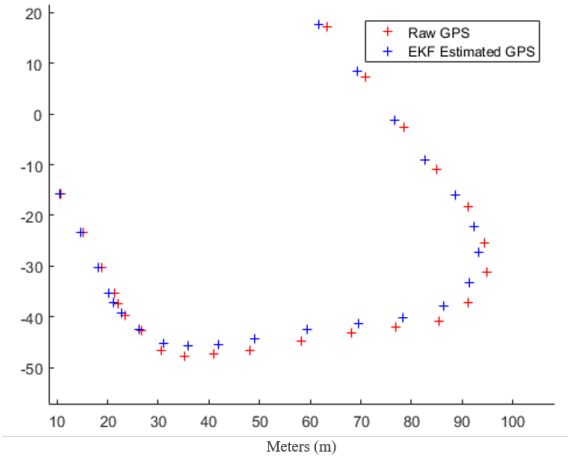


Fig. 11. Sample of a plotted path of our vehicle. Each "+" represents a point where a picture was extracted

Using our EKF values, the average size estimated was .768 with a standard deviation of .108. Using our GPS coordinates, we received an average size estimation of .763 and a standard deviation of .102. We noticed that two data points within our EKF estimated sign sizes and one data point within our GPS values fell outside of two standard deviations. Due to the error in GPS, this one data point was unsurprising. The two in our EKF were unexpected.

We traced these two data points back to anomalies within our Kalman Filter. Following the plot in figure 11 from the y-axis of the graph, we see the EKF Estimated GPS plots shift from slightly under the Raw GPS plots to slightly above. We recognize this as an anomaly because at the point of the shift, our EKF Estimated GPS behaves almost independent on of our Raw GPS. Given the error of the Raw GPS plots, our EKF Estimated GPS plot should not have shifted up unless our IMU data changed noticeably. When reviewing the outputs in our filter surrounding this point, we did not find anything to indicate such a change.

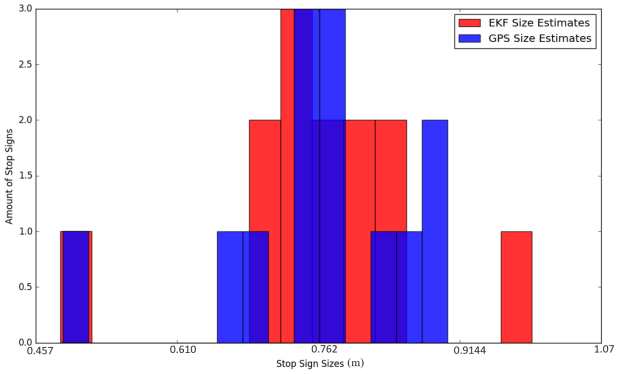


Fig. 12. Plot of images location by GPS distance.

VI. Future Work

Based on our results, it is clear that our EKF needs to be optimized. We want to be able to detect when an anomaly occurs and better account for it. After this we plan

to infuse the data from our VO into our EKF, making our size estimations more accurate. Lastly, we plan to continue exploring our efforts using VSFM. We believe minimizing the rolling shutter effect seen will increase its ability to correctly determine distance between our images.

VII. Conclusion

Our Dual-image model demonstrated that given the distance between two sequential images, we could correctly estimate the distance from the camera to the object and then determine the size of the object. Our efforts to correctly determine the distance between our images using sensor fusion of GPS and IMU sensor data, as well as monocular VO proved the concept viable. However, our findings yielded a substantially large standard deviation.

ACKNOWLEDGMENT

We would like to sincerely thank Dr. Christoph Mertz for his mentorship and support through the project. We would also like to thank Rachel Burcin and everyone who made the Robotics Institute Summer Scholar program possible.

References

[1] S. Varadharajan, S. Jose, K. Sharma, L. Wander and C. Mertz, "Vision for Road Inspection", IEEE Winter Conference on Applications of Computer Vision, 2014.

[2] F. Shahdib, W. Ullah, K. Hasan and H. Mahmud, "Obstacle Detection and Object Size Measurement for Autonomous Mobile Robot using Sensor", International Journal of computer Applications, vol. 66, no. 9, 2013.

[3] Y. Mustafah, R. Noor, H. Hasbi and A. Azma, "Stereo Vision Images Processing for Real-time Object Distance and Size Measurements", International Conference on Computer and Communication Engineering, 2012.

[4] A. Rahman, A. Salam, M. Islam and P. Sarker, "An Image Based Approach to Compute Object Distance", International Journal of Computational Intelligence Systems, vol. 1, no. 4, 2008.

[5] Y. Mustafah, R. Noor, H. Hasbi and A. Azma, "Stereo Vision Images Processing for Real-time Object Distance and Size Measurements", International Conference on Computer and Communication Engineering, 2012.

[6] "Global Positioning System Standard Positioning Service Performance Standard", U.S. government, 2008.

[7] C. Wu, "VisualSFM : A Visual Structure from Motion System", Ccwu.me, 2016. [Online]. Available: http://ccwu.me/vsfm/. [Accessed: 11- Jun- 2016].

[8] A. Avinash, S. Kumar and P. Singh, "CS676: 3D Reconstruction from Several Images(Structure from Motion)", Kanpur, 2014.

[9] N. Govender, Evaluation of feature detection algorithms for structure from motion, Council for Scientific and Industrial Research, Pretoria, Technical Report, 2009.

[10] D. Scaramuzza and F. Fraundorfer, Visual Odometry Part 1: The First 30 Years and Fundamentals, in IEE" Robotics & Automation Magazine, 2011.

[11] D. Lowe, "Distinctive image features from scale invariant keypoints", in International Journal of Computer Vision, ser. 91-110, vol. 60, no. 2, 2004.

[12] C. Tomasi and T. Kanade. "Detection and Tracking of Point Features", Carnegie Mellon University Technical Report CMU-CS-91-132, April 1991.

[13] R. Faragher, "Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation", 2012.

[14] S. Levy, The Extended Kalman Filter: An Interactive Tutorial for Non-Experts, 2016.

[15] W. Higgins, "A Comparison of Complementary and Kalman Filtering", IEEE Transactions on Aerospace and Electronic Systems, vol. -11, no. 3, 1975.

[16] C. Thorpe, R. Aufrere, J. Carlson, D. Duggins, T. Fong, J. Gowdy, J. Kozar, R. MacLachlan, C. McCabe, C. Mertz, A. Suppe, C. Wang and T. Yata, "Safe Robot Driving", International Conference on Machine Automation, 2002.

Deepthi Hegde

NIT, Surathkal

The goal of this research was to implement and compare different clustering algorithms with a new and simple marginal clustering algorithm developed at the Auton Lab

www.linkedin.com/in/deepthi-hegde-684312100

Comparison of Split Clustering with other clustering techniques

Deepthi Hegde, Mathieu Guillame-Bert, Kyle Miller, Artur Dubrawski

Abstract— Clustering is a method of finding groups of similar objects in data. Real world data, however, is inherently complex. Although many clustering techniques work well on low dimensional data, they fail in higher dimensions. Clustering in higher dimensions is not only computationally intensive, but also takes a toll on the effectiveness of the clusters. The presence of noise makes the task all the more tedious. Hence, we developed a new clustering technique called Split Clustering to cluster high dimensional data with low dimensional structures. In this paper, we focus on comparing the complexity and effectiveness of Split Clustering algorithm against existing clustering techniques. We take a close look at OptiGrid, a grid partitioning technique which resembles the Split Clustering algorithm. This paper does not cover the details and discussion of the Split Clustering algorithm.

Keywords- clustering; high-dimensionality; grid partitioning;

I. INTRODUCTION

Cluster analysis divides data into meaningful groups (clusters) [2]. Clustering finds its application in a wide range of fields spanning from medicine to psychology. It serves as a means to an end as in the case of data compression and efficient finding of nearest neighbors of points or an end in itself as in the case of document similarity, genes similarity and drug prediction for patients with similar symptoms. With the rapid increase in the amount of data being made available, the advancement of data manipulation is of prime importance. Image and video datasets, along with traditional datasets, are of growing complexity and dimension due to the improved technology with which the data is being captured and the precision with which we want to make predictions [1]. As a result, data often contains hundreds to thousands of features.

While clustering has a long history and a large number of clustering techniques have been developed in statistics, pattern recognition, data mining and other fields, significant changes still remain [2]. A number of present clustering techniques are either applicable only to low dimensional data or take too much time in high dimensions. We developed a simple yet non-trivial technique that scales well for high dimensional data having low dimensional structures. We report through our experimental study that our algorithm is able to perform better than the other algorithms we compared it with.

The rest of the paper is structured as follows: Section II introduces the various notions behind the goodness of a cluster. Section III describes the OptiGrid algorithm and Section IV introduces the Split algorithm. Section V presents the results of comparison of time and efficiency of various clustering techniques.

II. GOODNDESS OF A CLUSTER

Since clustering is usually unsupervised, there is no established notion of what constitutes a good cluster [2]. Different algorithms use different definitions to form clusters. The metric chosen determines the shape and boundary of the clusters. K-means and PAM (Partitioning Around Medoids) consider distance of points from a prototype to decide what data points should belong to a cluster. These distances based methods perform poorly on high dimensional data as distance does not always make sense in higher dimensions due to the sparsity of points in that space. Connectivity based clustering techniques, like hierarchical clustering, use the concept that closely lying data points are more related to each other than their farther counterparts, which relates back to the distance method. Density based clustering techniques, like DBSCAN and OPTICS, define cluster regions to have higher density than the surrounding regions. Low density regions are said to contain noise or outliers. A region of low density separates any two clusters. Grid partitioning techniques like OptiGrid use projections of the data to determine optimal cuts that separate the data. The limitation of this algorithm is that the data needs to be linearly separable. Split clustering is similar to OptiGrid in the sense that it uses margins to separate clusters. It can however handle data which is not linearly separable by means of feature transformations.

III. OPTIGRID

OptiGrid algorithm is a grid partitioning clustering algorithm which recursively finds optimal cutting planes that separate the dataset into meaningful clusters. The algorithm cuts the dataset into grids and recursively finds more cuts (if possible) within each of the dense grids. The cuts are identified based on the density curve of each dimension of the dataset independently. The top q cuts along all dimensions are chosen based on the density score.

Hinneburg et al. describe the OptiGrid algorithm as follows: [1]

OptiGrid (dataset D, q, max_cut_score, noise_level, cells, k)

1. Determine the projections of points from each of the axes.
2. For I = 0 to n do:
   - Cut <- Determine best_local_cuts
   - CutScore <- Score best_local_cuts
   - Insert all cuts <= max_cut_score into BEST_CUT
3. If BEST_CUT = None and length(D) > cells then return D as a cluster
4. Determine the top q cutting planes with the least score from BEST_CUT and delete the rest
5. Construct a multidimensional grid G defined by the cutting planes in BEST_CUT and insert all data points within the boundaries into G
6. Distinguish grids with population > cells as highly populated and add them to the recursion list
7. For each grid in recursion list, do:

OptiGrid (grid dataset, q, max_cut_score, noise_level, cells, k)

q – Maximum number of cutting planes
k – Number of clusters
n – Number of dimensions

In step 2 of the algorithm, the local best cutting planes are determined for each dimension independently using the density plots of the projections of the dataset. We first determine the leftmost and rightmost maxima (peaks of the curve) that are above a certain threshold value called the noise_level. This is to avoid detecting cutting planes on the edges of the dataset that have negligible density. We then identify q-1 maxima in between the leftmost and rightmost maxima's and their respective points with minimal density (valleys of the curve). The position of the minima gives the cutting plane and the density at that point serves as the score of the cutting plane. The valleys chosen to determine BEST_CUT should also satisfy the condition that their score should be less than a defined maximum (max_cut_score). The various parameters of the algorithm are highly sensitive and need to be tuned based on the dataset at hand through experimentation and analysis of the density plots. After the cuts are obtained, data points are placed in the grids formed by the cuts. The population of each of the grids is computed and only those with a minimum number of points defined by the parameter 'cells' are explored further recursively for distinct clusters. The recursion stops when 'k' number of clusters have been found or no good cuts can be found or when the grid becomes sparsely populated, indicating that the grid contains only outliers.

To exactly find 'k' number of clusters, we define two lists- waiting and selected. Waiting contains grids that are waiting to be cut further and selected contains the grids that have been explored. At every recursion, the grid with the maximum number of points from the waiting list is given as input to the OptiGrid function. Another list called result contains the clusters returned in step 3. At any point of the algorithm, if length(waiting + result) > k, the recursion is stopped. The grids in waiting and result at the end of the recursion contain the final clusters.

"Fig.1" shows the visualization of optiGrid on a synthetic dataset with 6 clusters and 1046 data points. The lines in the figure indicate the cuts and the shaded regions indicate the clusters. The blue lines indicate primary cuts, the red lines indicate secondary cuts and the black lines indicate the tertiary cuts.

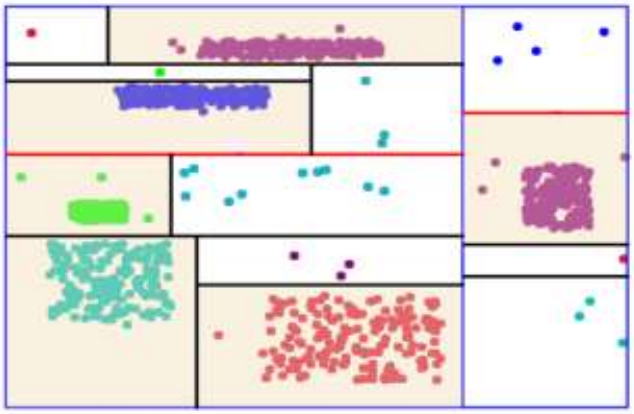


Figure 1. OptiGrid on synthetic dataset with q = 1

IV. SPLIT CLUSTERING

Split Clustering algorithm is similar to OptiGrid in concept. Like OptiGrid, it tries to find an optimal curve that separates clusters by using dimensional projections of the data. The cut is chosen in such a way that it does not pierce through a large cluster. Split clustering algorithm however, generalizes better to clusters with different density and differently shaped clusters. It can also detect non-convex and linearly inseparable clusters. The algorithm achieves this by means of transforming the features into a pseudo kernel space in which the data is possibly linearly separable and by finding an optimal cut in the transformed space. We achieve non axis aligned cuts by using Hough transformation of the points. This is yet another major distinction from the OptiGrid algorithm. We have however tested for only axis aligned cuts and two dimensional rotation cuts (planes that are non axis aligned in two dimensions). Fig. 2 shows two dimensional rotation hyperplanes. We use a greedy recursive approach to make binary partitions of the data at each stage. Further details of the algorithm and the pseudocode will be discussed in an upcoming publication.

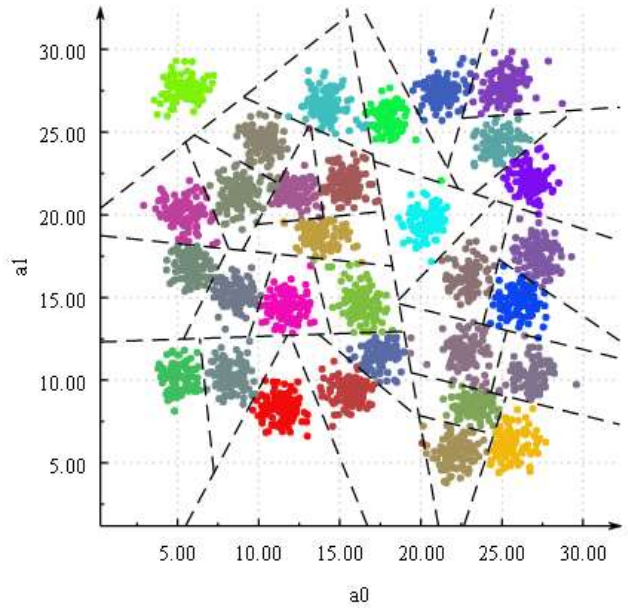


Figure 2. Split Clustering on a dataset with 31 clusters

V. RESULTS

The first three rows of Table I and Table II indicate times and efficiencies of three versions our algorithm. Split 1D makes one dimensional cuts, that is, the plane separating the clusters is axis aligned. Split 2D makes two dimensional cuts that are axis aligned to n – 2 dimensions (as shown in Fig. 2). The Split GRB version uses a transformation into a different space to make non-linear cuts. The numbers in bold indicate the best values in each column.

A. Comparison of Time

The time taken by the algorithms to finish clustering was recorded in seconds. The reading and writing times of the data were ignored to keep it fair for all algorithms. One-dimensional version of Split Clustering takes 0.04 seconds on averaging over the datasets which is the least amount of time among the algorithms compared. The two-dimensional version takes 0.14 seconds, which is longer than one-dimensional version due to the Hough transformation evaluations. The kernel version takes longer than one-dimensional version due to increased number of features in the transformed space.

B. Comparison of Efficiency

We ran the experiments on classification datasets, each of which had truth values. We provided the number of clusters as an input parameter for each of the algorithms. We measured efficiency as the ratio of the number of correctly clustered points to the total number of points in the dataset. Split Clustering shows the best average efficiency among the algorithms compared.

VI. CONCLUSION

In this paper, we have compared Split clustering, a new technique, with eleven other existing algorithms. We described the metrics used by different clustering techniques. We then described OptiGrid in detail and briefly described Split Clustering and brought out the differences between the two. We then presented the comparison of time and efficiency of various algorithms and found that Split Clustering performs better both in terms of time taken for clustering and the efficiency of the clusters. Our aim for the future is to refine the algorithm and explore its application in real world scenarios.

ACKNOWLEDGMENT

We thank all the members of the Auton lab for their continuous support through the course of the project. Deepthi Hegde thanks the Department of Information Technology, National Institute of Technology, Karnataka, for giving the opportunity to work at the Auton Lab. Special thanks to Mrs. Sowmya Kamath, National Institute of Technology, Karnataka, for her encouragement.

REFERENCES

[1] A. Hinneburg and D.A. Keim, *Optimal Grid-clustering: Towards Breaking the curse of Dimensionality in High-dimensional Clustering, pp. 506-517, 1999*

[2] Anil J. Jain, M. N. Murty, P. J. Flynn, *Data Clustering: A Review*, ACM Computing Surveys, 31(3): 264-323 (1999)

TABLE I. Table of time taken by different algorithms

Algo\Data	synthetic	synthetic	synthetic	compound	flame	pathbased	spiral	aggr	iris	wine	breast	Average
split 1D	**0.015131**	0.0401	0.015677	**0.015515**	**0.004916**	**0.006307**	**0.01674**	**0.017677**	0.005388	0.014052	0.003836	**0.040061**
split 2D	0.029833	0.123612	0.100896	0.02847	0.013148	0.014568	0.011089	0.043898	0.031945	0.290508	0.699712	0.140296
split GRB	0.147466	0.477341	0.092316	0.109553	0.026122	0.038423	0.04809	0.139388	0.006096	0.023613	0.037789	0.140441
optiGrid	1.329	3.687	3.792	4.194	0.484	0.49	0.516	3.645	3.404	5.499	6.121	4.869417
optics	2.58	16.5	0.38	0.22	0.42	0.68	1.35	7.74	0.05	0.03	0.16	4.135833
pdfc	11.78	696.46	4.23	13.09	1.11	2.41	2.88	14.7	1.68	7.16	2710.21	325.745
spectral	52.41	27797.05	42	3.99	1.51	2.59	3.06	20.07	1.43	0.36	15.49	2399.971
k-means	1.48	**0.03**	**0**	0.17	0.34	0.5	0.67	7.66	**0**	**0**	**0**	2.229167
pam	5.28	492.32	0.45	0.51	0.39	0.36	1.15	0.33	0.08	**0**	0.05	57.6525
hclust	1.71	10.01	0.04	0.01	0.59	0.92	1.17	0.1	**0**	0.02	0.04	2.49
agnes	4.77	2950.53	1.5	0.53	0.63	0.89	1.25	0.54	0.05	**0**	0.21	254.21
diana	3.8	5312.5	2.98	0.31	0.72	0.97	0.75	8.69	0.03	0.01	0.27	456.7675
clara	1.28	0.18	0.09	0.49	0.7	0.53	0.74	11.73	0.02	**0**	0.01	2.745833
fanny	4.17	1063.07	2.77	1.39	0.77	1.33	1.73	25.29	0.04	0.03	0.16	209.5275

TABLE II. Table of efficiency of different algorithms

Algo\Data	synthetic	synthetic	synthetic	compoun	flame	pathbased	spiral	aggr	iris	wine	breast	Average
split 1D	**1**	**1**	0.833333	0.736842	0.5375	0.6966667	0.426282	0.996193	**0.96**	0.606742	0.866432	0.787272
split 2D	**1**	**1**	**1**	**0.867168**	0.829167	0.7066667	0.403846	0.998731	**0.96**	0.44382	0.653779	0.805743
split GRB	**1**	**1**	**1**	**0.867168**	0.975	0.7466667	0.474359	0.996193	0.826667	0.539326	0.838313	**0.842154**
optiGrid	**1**	0.801921	**1**	0.804511	0.479167	0.6966667	0.442308	0.923858	0.846667	0.38764	0.637961	0.729154
optics	**1**	0.333333	**1**	0.681704	0.975	0.65	**1**	0.859137	0.793333	0.578652	0.720562	0.781066
pdfc	0.982932	0.994298	**1**	0.651629	**0.991667**	0.79	0.365385	0.991117	0.893333	0.651685	0.727592	0.821785
spectral	0.833333	0.708583	0.833333	0.696742	0.645833	**0.87**	**1**	0.993655	0.9	0.629213	0.862917	0.815783
k-means	0.710843	0.87375	0.833333	0.666667	0.8375	0.7433333	0.346154	0.784264	0.893333	0.702247	0.85413	0.749596
pam	0.964859	0.982293	**1**	0.646617	0.85	0.74	0.349359	0.777919	0.893333	**0.707865**	0.86819	0.798221
hclust	0.936747	**1**	**1**	0.691729	0.720833	0.76	0.371795	0.837563	0.893333	0.696629	0.778559	0.789744
agnes	0.666667	0.75	0.333333	0.862155	0.833333	0.73	0.358974	**1**	0.906667	0.61236	0.662566	0.70146
diana	0.666667	0.685974	0.5	0.689223	0.854167	0.7133333	0.358974	0.798223	0.88	0.52809	0.850615	0.684115
clara	0.967871	0.962685	**1**	0.656642	0.85	0.7433333	0.358974	0.784264	0.893333	**0.707865**	0.86819	0.799378
fanny	0.97992	0.990896	**1**	0.513784	0.841667	0.6666667	0.352564	0.681472	0.913333	**0.707865**	**0.887522**	0.775972
Clusters	6	12	6	6	2	3	3	7	3	3	2	NA
Rows	1046	10496	1046	399	240	300	312	788	150	178	569	NA
Dimension	2	2	4	2	2	2	2	2	4	13	30	NA

Joseph Shepley

Columbia University

Often times a robot's task is to manipulate objects whether it be grabbing a drink from the fridge or picking items off shelves.

https://github.com/jshepley14

Physics Based Approach to Pruning Search Space in Multi-Object Pose Estimation Algorithms

Joseph Shepley, Venkatraman Narayanan, Maxim Likhachev

Abstract—**Often times a robot's task is to manipulate objects whether it be grabbing a drink from the fridge or sorting through a pile of objects in a warehouse. In these tasks it's critical for the robot to understand the poses of multiple objects in the environment. Certain pose estimation algorithms such as PERCH and D2P accomplish this task by searching through a vast number generated scene renderings in order to find the rendering that best explains the observed scene from sensor data. However, this process is extremely time consuming especially for 6DoF on each object. In order to reduce computation time and prune the search space required for finding the 6 DoF poses for each object, we use a physics simulator to eliminate hypothesized scenes that are inconsistent with the physical world. This reduces some of the algorithm's search space by removing proposed object scenes that would never exist in real life, everything from a scene where a cup is "floating" in air to subtler scenarios. We show that this method is as fast with the ability to invalidate a scene in as little as a millisecond. Additionally, we demonstrate its use and speed in an example reward function maximization search.**

Keywords-pose estimation; physics; search; perception

I. INTRODUCTION

From industrial robots picking through items in warehouses to a home robot grabbing a drink from the fridge, perception is a critical part in robotic manipulation. An important perception task can be identifying and localizing objects from sensor data where 3D models of the objects are known ahead of time. One effective way to accomplish this task is to use an algorithm like PERCH [3] that seeks to find the best explanation of the observed sensor data by hypothesizing possible multi-object scenes in a generative fashion. While this method has proved its merit on accuracy and on dealing with heavily occluded observation scenes, the scene generation search space can be quite large leading to a time consuming localization process. Furthermore, algorithms of this fashion have only localized 3DoF object poses. In real life objects contain 6DoF, but the search space becomes enormous and computation time is a problem.

We introduce a physics based solution to cut down the large search space involved in 6DoF pose estimation for a generative search algorithm. The solution, Scene Validator, exploits properties of physics in order to invalidate and eliminate hypothesized scenes that normally would waste computation time.

The Robotics Institute, Carnegie Mellon University, Pittsburgh, USA

II. PROBLEM FORMULATION

In a PERCH-like algorithm, hypothesized multi-object scenes are rendered using 3-D models and then used in a search to find the rendering which best explains the observed scene from sensor data. The number of hypothesized scenes is vast because they are generated combinatorial fashion. With 6DoF for each object in the scene, this means that some hypothesized scenes will result in object configurations that do not make sense in the physical world. For example, since the z axis can vary, a hypothesized scene may place a cup floating in the air above the other objects. This scene is not going to match any permanent real life observed scene since objects do not float in mid air. It is our goal to quickly decide that a hypothesized scene like this would not exist in the real world and should be eliminated. Otherwise we would have to take the expensive step of rendering it and adding it to the search space. The previous example of a floating object invalid scene is a simple one, but one can imagine that much subtler and complex situations can arise such as tilted, leaning or unbalanced multi-object configurations.

The problem we consider is that of taking in some number of 3-D models along with their 6DoF poses and determining if this multi-object configuration is in static equilibrium or not. In other words, could this hypothesized scene exist under the laws of physics? Additionally, we wish to complete this validation as fast as possible.

III. METHODS

A. Physics Engine

The crux of our scene validation relies on exploiting physical laws to determine if the scene is in static equilibrium. We used a physics engine to simulate real world behavior. With a physics engine one may simulate physics by stepping through the simulator in an iterative fashion. One iteration is referred to as a step and calculations and forces are applied to the objects during this time. Several physics engines were considered such as DART, Bullet, PhysX, and ODE [1]. Because of the physics engine comparison results from [1] and [2], we chose ODE (Open Dynamics Engine) for its speed and accuracy relating to our specific application which mainly involves contact forces between distinct mesh models, and the effect of gravity in a simulated world.

B. Scene Validation

The process of validating a scene first involves loading the mesh models (.obj files) and their 6DoF poses into the physics simulation world. Then, some number of simulation steps are run. The initial *x,y,z* positions of the objects are compared to their final positions and if the difference between the positions is greater than some threshold, the scene is considered invalid. If the scene is still valid at this point, more simulation steps are run, and delta positions are checked again. This process is repeated a number of times (in our implementation four times) until a simulation step limit is reached, and at this point if the delta positions are still below the threshold, the scene is considered valid.

Additionally, rather than merely returning true or false as to a scene's validity, one can stop the cycle early and return a probabilistic estimate of whether the scene is valid or not. This probabilistic estimate is returned when a lazy step limit, or early termination is reached. For example, after a short amount of simulation steps, the objects may have moved, but are still below the threshold. In this case, comparing how close the delta positions are to the threshold number can provide information about the probability of the scene's validity if more simulation steps were to be taken.

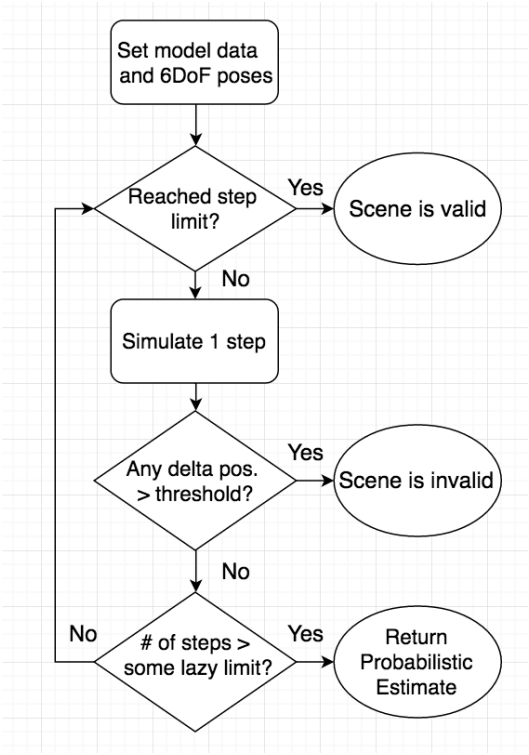


Figure 1. Scene validation flow chart

IV. Results

A. Simulator Speed

To evaluate the performance of the scene validator, measurements were taken to determine the speed. Fig. 1 shows that a scene can be determined as false in as little as 1 millisecond. A series of checks indicates areas where early termination could take place. The step limit is at check 4, and it takes 50 ms to reach that limit at which point the the scene can be labeled valid. Additionally, fig. 2 shows that as more and more 3-D models are placed in the simulator, computation time increases proportionally.

B. Height Maximization

The PERCH-like search algorithm that would benefit from the scene validator will be maximizing some reward function as it searches for the best hypothesized scene. To demonstrate the scene validator's use in a reward function maximization search, we used it in a simple search to find the highest stable pile of objects. Three models were given along with an order, and the models' *z* positions were varied until height and stability were maximized. Fig. 3 shows the initial and final result. This search process involved many possible object configurations that could be quickly discarded by completing stability validation until the best one was found. The process took 0.3 seconds. This demonstration serves as a proof of concept of scene validation used in a simple reward function maximization search. In a pose estimation algorithm, all 6DoF variables will be altered instead of just the *z* position.

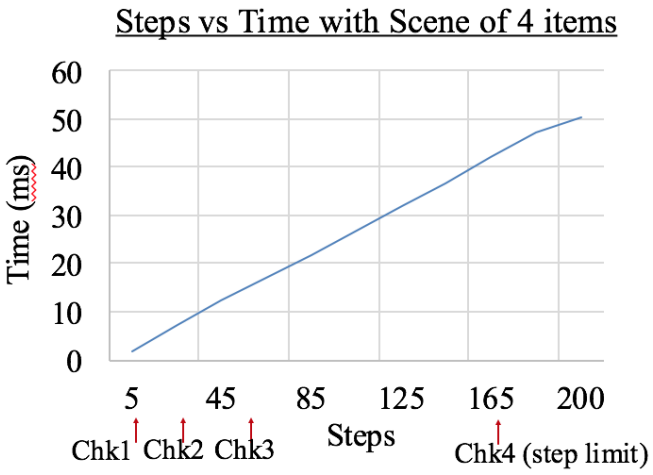


Figure 2. Number of simulation steps and how long the program takes to execute them. Tests are run on a scene of four 3-D models. Shows series of checks which allow early termination.

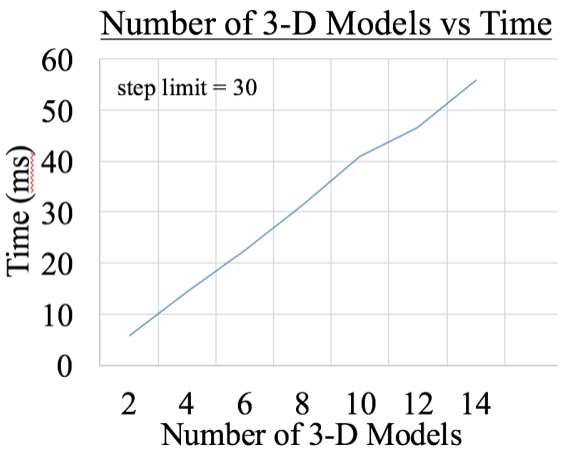


Figure 3. Number of 3-D models and how long the program takes to execute scene validation with a step limit of 30.

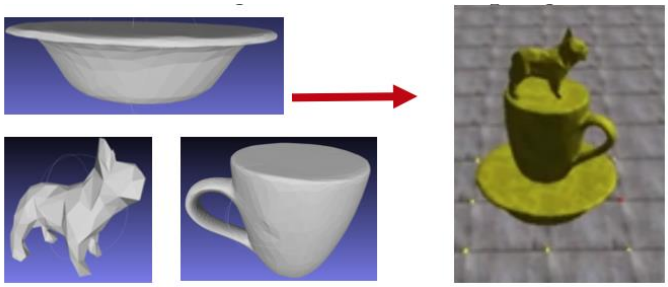


Figure 4. Height maximization search

V. Conclusions

We present a tool that can cut down the search space involved in generative pose estimation algorithms. The program achieves fast scene validation, in as little as 1 millisecond. We demonstrated the program's merit in a simple reward function maximization search. This program can be used to speed up pose estimation algorithms like PERCH [3] or D2P [4] which search for 6DoF poses. Additionally, rather than merely returning true or false as to a scene's validity, a probabilistic estimate can be made after only running the simulation for a short amount of time. This allows for the development of algorithms which can exploit a probabilistic estimate thus speeding up the perception process even more.

Acknowledgment

This work was funded by the NSF for the Robotics Institute Summer Scholars program.

References

[1] Tom Erez, Yuval Tassa and Emanuel Todorov "Simulation Tools for Model-Based Robotics: Comparison of Bullet, Havok, MuJoCo, ODE and PhysX" (ICRA) 2015.

[2] A. Roennau, F. Sutter, G. Heppner, J. Oberlaender and R. Dillmann "Evaluation of Physics Engines for Robotic Simulations with a Special Focus on the Dynamics of Walking Robots" (ICAR) 2016.

[3] Venkatraman Narayanan and Maxim Likhachev "PERCH: Perception via Search for Multi-Object Recognition and Localization" (ICRA) 2016.

[4] Venkatraman Narayanan and Maxim Likhachev "Discriminatively-guided Deliberative Perception for Pose Estimation of Multiple 3D Object Instances" (RSS) 2016.

Qi Zhu

University of Illinois

This project tends to use synthetic data to train an accurate model for articulate objection pose estimation.

https://www.linkedin.com/in/qi-zhu-22633598

Articulate Object Keypoint Detection and Pose Estimation Using Synthetic Data

Qi Zhu
Department of Computer Science
University of Illinois at Urbana-Champaign
Email: qi.zhu.ckc@gmail.com

Zhe Cao
Robotics Institute
Carnegie Mellon University
Email: zhecao@andrew.cmu.edu

Yaser Sheikh
Robotics Institute
Carnegie Mellon University
Email: yaser@cs.cmu.edu

***Abstract*—Real-time keypoints detection of textured objects is emerging as a fundamental problem in robotics. Among all existing detection algorithms, convolutional neural networks(CNNs) has been proved as the most effective one. However, traditional CNN suffers from insufficient training data when it comes to articulate object. We introduce a toolbox in Unreal Engine 4 which can automatically create a large amount of training images from different viewport for specific object. In this paper, we explore how to generate high quality training images and refine the original pose machine to achieve higher detection accuracy for various objects. Our empirical results show the promise of such a direction is substantial.**

I. Introduction

Keypoint detection and pose estimation are used widely in related applications, such as vision system and robotic applications. Both problems have been studied extensively in past decades. However, for articulate object, which has joints or jointed segments, traditional keypoint detection methods like SIFT, can easily generate a lot of outliers regarding the large appearance variation. For training based methods, there is insufficient data for those objects. Typically, it's impossible to have well-annotated image for each articulate object. In robotics, robotic manipulation techniques are already stable and effective. Robots are expected to take charge of more and more sophisticated tasks like grab object or manipulate scissors in the future.

Our core idea is to use articulate object model to synthesize a large amount of training image. One strength of synthetic images is sufficient data and pixel-level annotation. By introducing a render plugin in game development engine, we can easily generate sufficient training images with rich context information. We summarize our contributions as:

1. Introduce UE4Render and algorithms to generate various images
2. Derive a general pose machine algorithms from latest research progress
3. Prove that synthetic training images, to some extent, can replace the heavily hand-crafted real images if generated properly

II. Related Work

Our work relates to image synthesizing and pose estimation. We briefly review related work in both areas.

A. Image synthesizing

Beyond simply generating 2D images, existing studies provide more intelligent ways to synthesize high quality images. Researchers have looked at generating a large number of images from 3D models [8]. Beyond that, recent research tends to exploit pixel-wise image annotation from commercial game development engine-Unity [1] and treat these images as groundtruth of urban and indoor scenes segmentation. Ros *et al* [7] showed promising result by boosting synthetic data in CNN models. Ankur *et al* [3] has provided high quality synthetic indoor segmentation data, which has comparable performance with state-of-the-art indoor scene understanding systems trained on real images. Though useful, previous studies on synthesized image dataset can obtain pixel level label, they didn't dig into other *delicate* tasks, e.g. keypoint detection and object tracking. This paper extends previous research in terms of using solely synthetic data for to estimate keypoint and 3D pose, and in particular articulate objects.

B. Pose estimation

Basically, keypoint detection problem can be viewed as 2D version of pose estimation. Existing works have looked at different ways to facilitate keypoint detection, especially human pose estimation. In human pose estimation, distinguishing similar part of human body can be tricky. Ramakrishna *et al* propose a sequential prediction structure that incorporates context information and perform quite well in human pose estimation. With the trend of deep learning, recently, convolutional neural networks are claimed to achieve higher prediction accuracy that any other existing works [6]. We realize the challenge in articulate object keypoint detection is similar to human pose. Our work is based on latest convolutional neural network architectures-Covolutional Pose Machines [9].

III. Method

To issue the problem of insufficient training data, we follow the below steps.

A. Synthesizing training data

Previous research [8] [3] has explored using synthesized images to viewport estimation and image segmentation tasks. To learn fine-grained features, we use Unreal Engine(Epic Games, 2015) [2], a widely used platform to develop 3D

game. It has excellent lighting and physics simulations, which highly resemble real images. UETorch [5] is an open-source combination of the Unreal game engine and the Torch deep learning environment. UETorch scripts can be embedded into any blueprint file and dynamically change the attribute of the objects with a few lines of code.

1) UE4Renderer: Platform and philosophy: UE4Renderer is a package that embeds image rendering and groundtruth output into Unreal Engine 4 game loop based on UE4Torch. UE4Render is written by Lua and Python, Lua is the original script language for UE4. UETorch provides interfaces for capture screenshots, segmentation, and set up game loop parameters. On top of that, UE4Renderer has several basic functions: articulate object animation, game loop simulation and data cube output. It can both do physics simulations and per-frame pre-defined simulation. To address different type of articulation relations, UE4Renderer doesn't embed articulation in the game loop, e.g. when joint angle of scissors changes. Instead keypoint location in object coordinate is designated by model configuration script. In UE4Renderer, we try to reduce the calculation amount in simulation stage as mush as possible.

2) Image rendering and data processing: To rendering satisfied images, basically, there are several steps:

1. Set up UE4 scene and standard blueprint
2. Create configuration files for scene, articulate object model and object animation
3. Evaluate scene rendering intervals and set up screenshots/segmentation intervals
4. Run UE4 and wait for high resolution synthesized data

The most important component in UE4Renderer is the simulation module. Given all configuration files, simulation module fetch keypoint data in object coordinate, object rotation and translation and camera viewport location and then manipulate articulate object with precise 2D and 3D keypoint output in camera coordinate. Specifically, as mentioned above, every mode of articulate object is treated as different models. First, UE4Renderer transform object keypoint X_o into world coordination with translation and rotation $\{T_w, R_w\}$. Specifically, R_o encodes the mode of articulate object:

$$X_w = R_w(R_o X_o) + T_w \tag{1}$$

According to different scene center and camera viewport in world coordinate, UE4Renderer transform all the 3D keypoint into camera coordinate. The rotation matrix R_c can be derived from camera location in configuration file. Specifically, each camera must orient at scene center and camera's y axis is parallel with *xy* plane (in UE4, *xzy* Cartesian coordinate is used and camera's optical axis is X axis). With these two constraints, $\{R_c, T_c\}$ is unique.

$$X_c = R_c(R_w(R_o X_o) + T_w) + T_c \tag{2}$$

To get 2D keypoint groundtruth data, all the 3D keypoints are projected onto camera plane. Unfortunately, we don't know the intrinsic parameters of UE4 cameras. We employed *camera calibration* in UE4, that is, using segmentation functions to get pixel distance of known standard objects. With intrinsic parameter matrix K_{proj}, 2D keypoint can be represented as:

$$X_{2d} = K_{proj}(R_c(R_w(R_o X_o) + T_w) + T_c) \tag{3}$$

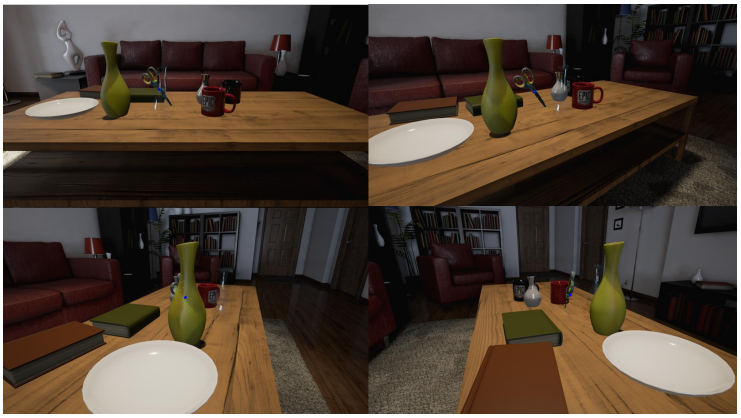


Fig. 1. A toy example that illustrates how our UE4Renderer can generate sufficient data from different viewport.

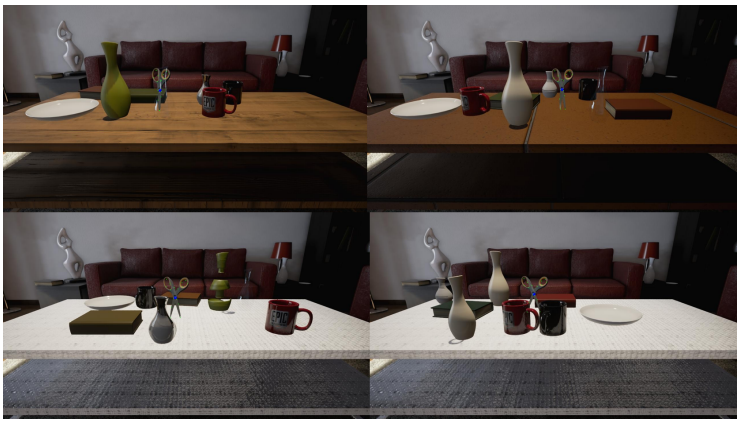


Fig. 2. Using dynamic backgound, our training data shows more diversity

3) Data sufficiency: One promising feature of UE4Renderer is, theoretically, people can generate sufficient data for any object. In our experiment settings, we uniformly pick up 41 viewports in a semi-sphere that center at scene center. In each viewport, we play the same animation sequence. In other words, in world coordinate, the same animation sequence is executed 41 times to promise data sufficiency. Fig. 1 shows synthetic examples with keypoint annotation from four different viewport.

4) Data variance: Different with previous tasks, synthesized images easily suffers from the problem of overfitting in training stage. To alleviate this issue, we first randomly place the background objects in animation sequence. However, our learning algorithms - pose machines [9] [6] only take a small image patch around object as training data. Random placement results an sparse distribution in space, which lead to similar image patch. Thus, we create various "micro" backgrounds", **i.e.** context objects around object of study, and dynamically change the materials and light conditions, i.e. place other objects around object of study.Fig. 2 shows synthetic examples adding dynamic micro background

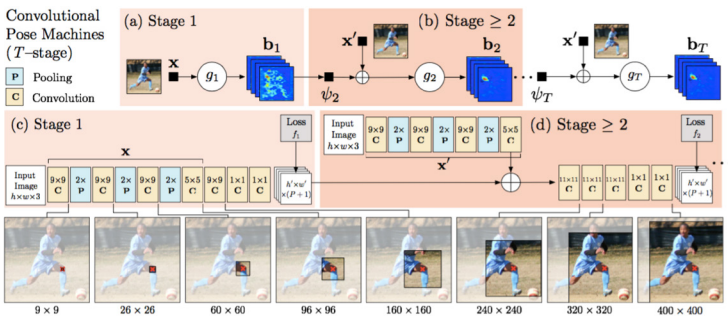


Fig. 3. Architecture and receptive fields of CPMs

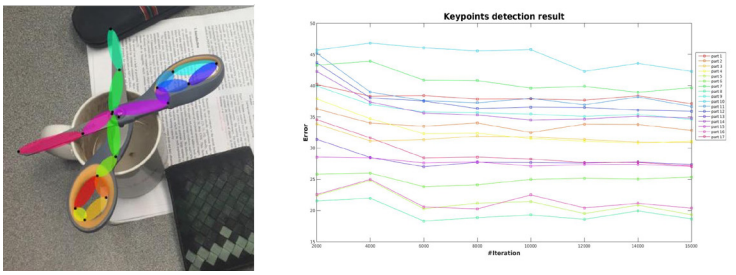


Fig. 4. Qualitative and quantitive of scissors keypoint detection

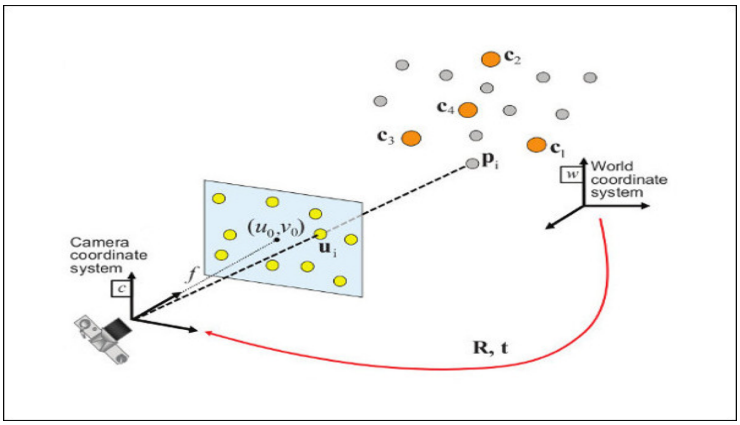


Fig. 5. Perspective n point problem illustration

B. Keypoint detection: Convolutional Pose Machine

1) CPM characteristics: Convolutional Pose Machine [9] is one derivation of traditional pose machines, which consists of a sequence of multi-class predictors. In each stage $t \in \{1..T\}$, the classifiers g_t predict belief maps for every part based on feature extracted from specific location and context information in previous stage. When $t = 1$, classifier g_1 produces below belief values:

$$g_t(x_z) \rightarrow \{b_1^p(Y_p = z)\}_{p\in\{0..P\}}, \tag{4}$$

where $b_1^p(Y_p = z)$ represents belief scores for part p at image location z. For stages $t > 1$, the classifier predicts belief values by both features and contextual information:

$$g_t(X_z^{'}, \Phi_t(z, b_{t-1})) \rightarrow \{b_t^p(Y_p = z)\}_{p\in\{0...P+1\}}, \tag{5}$$

Here P parts plus one for backgound and $\Phi_{t>1}$ map from belief maps b_{t-1} to feature space. In each stage, the computed beliefs refine location of each part. CPM has shown these consecutive pattern can by enhanced by deep convolutional architecture. Basically, we follow their deep architecture for keypoint detection. Articulate object shares a lot of similarities with human pose, for example, scissors also have large appearance variation and symmetric structure. Thus, context information is crucial in articulate object keypoint detection.

2) CPM architecture: Like pose machines, CPM use only local image evidence. Fig. 3 shows per-layer structure of CPM. In the first stage, only small image patches are considered. In pratice, we crop an 368×368 pixels window as input, and the receptive field of the first stage network is 160×160 pixels.

For the following stages, convolutional pose machines incorporate previous belief maps and new local features from a larger receptive field. From Stage 2, the receptive field is 400×400 pixels in original image, which approximately covers any bi-part relations. In this paper, we use scissors in Fig. 3 as one typical articulate object, we found that 3-stage network can capture every part precisely.

3) Keypoint detection: Using UE4Renderer, we focus on scissors. We generate 20000 images with various background and different pose. Besides, we give out scissors mask in each image and scissors center, which helps crop the image into small batch. We feed these synthetic data into convolutional pose machine and store the model. To evaluate the training result and look into overfitting issues, we generate 3000 images containing scissors with a totally different environment as synthetic testing data and capture several videos of real scissors. In Fig. 4 , we explore both qualitative and quantitive performance with our novel keypoint detector. We can see that keypoint detection accuracy rise with number of training iterations, which indicates our synthetic data don't suffer from overfitting too much.

C. Pose estimation: Perspective-n-point problem

In Fig. 5, traditional PnP algorithms take feature point as input to estimate the camera viewport, and they introduced RANSAC algorithms to exclude outliers. EPnP [4] algorithm is a non-iterative solution with $O(n)$. It expresses the n 3D points as a weighted sum of four control points by solving a small scale eigenvector problems. EPnP has been widely used in rigid object pose estimation. However, when it comes to articulate object, e.g. scissors, feature points are not representative for all different joint angles. Pre-defined feature points can not indicate structure deformation clearly. So we pick keypoint described above as input of PnP algorithm. Given synthetic testing data, we are able to evaluate the performance of PnP algodrithms. The other question is, how we can get joint angle between different part, here we simply treat every part as an single object and estimate angle via rotation matrix difference. In Fig. 6, we present PnP result in synthetic image.

IV. Conclusion

We provide a thorough pipeline to create synthetic dataset and train both a keypoint detector and a pose estimator. With UE4Renderer, we suppose to overcome technical bottleneck, i.e. insufficient training for those tasks. Experimental results are promising. Our algorithm shows promising

Fig. 6. 3D pose estimation result on synthetic image

result in scissors. Compared with other keypoint detection or pose estimation algorithms, we are using only synthetic data. Interestingly, renderer can dynamically relate with CNN architecture. Exploring how to build an open-loop system is left to future work.

ACKNOWLEDGMENTS

This work is funded by Toyota Motor Engineering & Manufacturing North America (TEMA, TOC, PE, PEPD, TPARC). Thanks for the advice from Zhe, Hanbyul and Thomas this summer.

REFERENCES

[1] Unity Game Engine. Unity game engine-official site. *Online][Cited: October 9, 2008.] http://unity3d. com*.

[2] Epic Games. Unreal engine 4. *Online] https://www.unrealengine.com*.

[3] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. Scenenet: Understanding real world indoor scenes with synthetic data. *arXiv preprint arXiv:1511.07041*, 2015.

[4] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155–166, 2009.

[5] Adam Lerer, Sam Gross, and Rob Fergus. Learning physical intuition of block towers by example. *arXiv preprint arXiv:1603.01312*, 2016.

[6] Varun Ramakrishna, Daniel Munoz, Martial Hebert, James Andrew Bagnell, and Yaser Sheikh. Pose machines: Articulated pose estimation via inference machines. In *European Conference on Computer Vision*, pages 33–47. Springer, 2014.

[7] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3234–3243, 2016.

[8] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694, 2015.

[9] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. *arXiv preprint arXiv:1602.00134*, 2016.

Ressa Reneth Sarreal

University of Maryland

Autonomous navigation through fields of crops is difficult for robots due to the dense growth of plants that engulf the space, which is especially true when navigating through sorghum bicolor.

https://www.linkedin.com/in/ressa-reneth-sarreal-519064123

Development of LIDAR-based, autonomous agricultural robot guidance system for sorghum bicolor navigation

Ressa Reneth Sarreal
University of Maryland, Baltimore County
Baltimore, MD 21250
Email: saressa1@umbc.edu

Mentor: George Kantor, Ph.D.
Carnegie Mellon University
Pittsburgh, PA 15213
Email: kantor@ri.cmu.edu

***Abstract*—This document describes the approach to optimize navigation through sorghum bicolor field rows with a crop-phenotyping robot. Navigation through fields of crops can be difficult because of the dense growth of plants that engulf the space. Accuracy and precision are crucial for navigating through crops because a key constraint is the amount of damage done to the rows of sorghum. The main approach taken is to use a very accurate RTK GPS system and automated ideal path detection. This paper will focus more on the latter. A simple algorithm and system was created, with the use of robotic operating system (ROS), point cloud libraries (PCL), and random sample consensus (RANSAC) methodology, to autonomously detect rows within the robot's proximity and to provide information on the offset of the robot's heading and position from the calculated ideal path. The offset information is successfully produced, and the system can be easily adaptable to other robots/platforms.**

Keywords-component; Autonomous; Navigation; LIDAR; Agriculture; Robotics; RANSAC; Sorghum

I. INTRODUCTION

Sorghum bicolor is a crop that is approximately 20 feet tall when fully grown. This crop also does not require a lot of water [7], so they are generally grown in large amounts and are planted very densely into the area. The crop can be utilized as a biofuel source. With the rapidly increasing population, increasing the biomass yield of sorghum is vital to keep up with the population's demands of the future generations. Breeders are able to identify the strongest and most optimal sorghum plants; however when choosing from acres of plants, it's difficult to manually collect large amounts of data. Despite the fact that the plants are genetically identical within a single plot, there are external factors that may have impacted certain plant's growth, and capturing averages instead of having one plant represent its plot will better represent that plot's actual growth.

Fig. 1. Sorghum bicolor plants that express the density between the rows of crops.

Fig. 2. The developed robot driving between the sorghum bicolor plant rows. There is approximately 0.1 m of space on both sides of the robot. These sorghum plants are located in Florence, SC and are approximately 1.5 months since planting.

Automated data collection would be very useful with large pools of plants to help identify the strongest plants. Research in this area has been applied to vineyards, and orchards, but there has not been much research on navigation through sorghum. This is part of the challenge since sorghum is so densely packed, leaving many constraints for the robot itself.

II. Related Works

The application of robots in agriculture have been done to certain scales. There have been works on citrus groves, maize, orchards, roses and generic fields. LIDAR was used in all of these approaches.

A full-sized tractor was used when navigating through citrus groves. It depends highly on real time kinematics global positioning system, RTK GPS, and differential global positioning system, DGPS; however the trees overhead the system interrupt the signals. Laser radar, LADAR/LIDAR, was used for obstacle avoidance. Additional visual data is captured by a color camera to discern the ground from the plants surrounding it. Information is then passed to a proportional integral derivative controller, PID, for steering control. The actual paths followed by the tractor allows room for approximately a meter of error on both sides of the vehicle, but when testing a narrower path was simulated using hay bales [6]. This research project and the following projects emphasize and reiterate the value in using RTK GPS.

Another full-sized tractor was the medium for navigating through an orchard. A LIDAR is attached to the front of the tractor and used single, planar laser scans to capture the orchard rows. RTK GPS is also used here for general navigation. A controller is used to adjust the tractor's path based off of the heading offset of the tractor. An ideal speed is found for the tractor, then the heading is adjusted as it drives [1]. The controller used is similar to the controller that will be implemented with the algorithm developed. The speed of the robot will not be as fast as the tractor due to the limited space allotted by the sorghum rows. It's heading will be adjusted carefully as data is read from the fields. The following project describes how a ground robot will interact with maize, a crop similar to sorghum.

In the maize row detection project, RANSAC and LIDAR is used to navigate a land robot through maize fields. The purpose of this was to see how row detection handled the crops when they have grown at different heights. An ideal height found was 0.268m for best heading detection [5]. Though this information is not directly related to the work in this document, their implementation of RANSAC can also be used for this project; the scale of this project is very similar, which provides information that will be beneficial to consider when designing algorithms for this project's ground robot, like how to handle attachments used for gathering input from the environment.

A four-wheeled, ground robot, developed by another lab at the Field Robotics Center, is applied to navigate through an orchard. This lab uses methodology that will be very close to the methodology applied to the robot described in this document. This lab implements certain features that may not be implemented in this iteration of the robot's development; however, these features may be useful in the future to capture different phenotype data, for example: motorized, rotating LIDAR for capturing multiple plane layers [8]. The point clouds and photographs of the canopies shown look similar to the data sets that are processed in this project so many similarities are clear in the two research projects.

All the projects described have successfully used a combination of GPS and LIDAR to have a robot navigate through some agricultural field. Due to the accessibility to the point cloud library, RANSAC became a possible approach for processing the laser scans. At this point in development, the RTK GPS has already been implemented and tested in a field. An issue that arose was that due to the swaying of the mast the GPS is attached to, the detected location of the robot

III. Methods

The robot already is based off of ROS , and has a LIDAR attached. The use of LIDAR was approached in two ways: using single-planar laser scans that are parallel to the ground plane and aggregating many angled laser scans into one point cloud.

Fig. 3. The phenotyping robot's navigation relies solely on the RTK GPS and the LIDAR, which are circled and labeled in the figure.

A. Single Plane Laser Scan

The LIDAR is aimed so that the laser scan captured is parallel to the ground plane, which is shown in Figure 4. This method has been tested with a simulated environment where there were only two rows of crops that the robot had to drive between. This approach was briefly tested with data taken from a sorghum field in Clemson, South Carolina. The scans taken from the simulated environment looked very similar to the scans taken in the field. The most significant and only relevant difference is that the rows are much narrower than the ones in the simulated environment. Despite that fact, this means that the robot should be able to detect paths in this environment. This method requires that the rows not adjacent to the robot must be removed from the scan. Unfortunately, the method is not very versatile and can only be used to detect nearby rows.

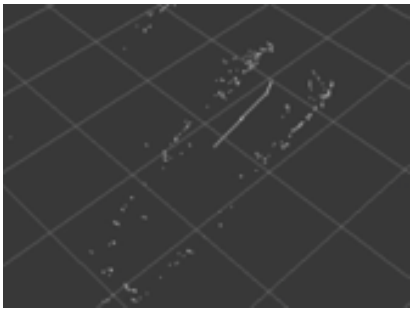


Fig. 4. The point cloud and path detected when using data from the simulated environment. The solid lines in between the points is the path found.

B. Aggregated Laser Scans

The method selected is to take a LIDAR that is angled 30° below the xy-plane, defined by the base_link coordinate frame, and aggregate multiple laser scans into a point cloud and repeat this aggregation over time, replacing old scans with more recent scans.

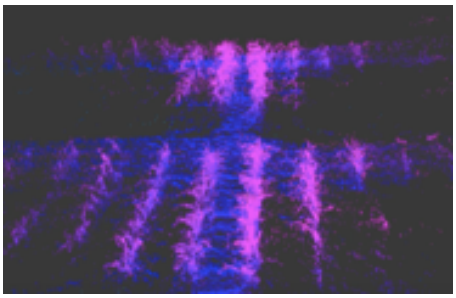


Fig. 5. The aggregated laser scans captured during a specified time frame. For example, if Tc is the current time, n is the amount of time that multiple scans are taken, and ds/dt is the rate at which scans are being taken, the total point cloud should have scans captured between the time frame (Tc - n) until Tc, and the total number of laser scans captured during that time can be found by multiplying ds/dt and (Tc - n).

Unfortunately, six rows not adjacent to the robot are captured by the LIDAR, shown in Figure 5. Also, since the ground plane is captured, due to the downward-directed angle, many more points must be done to isolate the adjacent rows into a point cloud of interest, shown in Figure 6. To avoid using leaves that hung over the pathway and to focus more on the points encompassing the stems of the sorghum, the points above a certain height from the ground plane were also removed. This point cloud's points are then all translated into the xy-plane so that a point cloud similar to the first method was attained, then similar RANSAC algorithms were used for identifying rows. When points are translated onto the xy-plane, there is some error that must be accounted for or can be declared negligible, and declaring negligibility is explained in Figure 7. The error will not be significant enough for the robot to collide with the crops since it is assumed that the sorghum fields will not be angled more or less than 10°. Data used to test these algorithms was taken from the sorghum bicolor fields in Clemson, South Carolina.

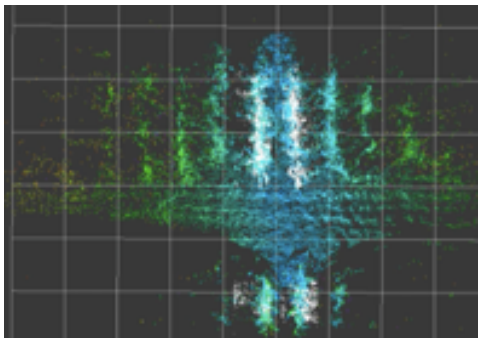


Fig. 6. The point cloud of interest is the white points among the colorful points. The colorful points represent the total aggregated point cloud explained in Figure 5. The point cloud of interest only represents the sorghum bicolor crops within the rows adjacent to the robot's current location.

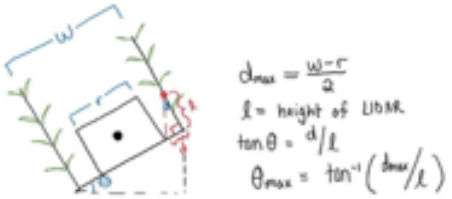


Fig. 7. When the points captured by the LIDAR are projected onto the z-plane of the base_link coordinate frame, data points may not be perfectly translated downward due to slope of the field. It was found that the field must not be sloped more than approximately 11°. The method of just projecting points onto the xy-plane is valid because the fields will not be sloped so much and since the reference coordinate frame will be the base_link coordinate frame, which follows the robot, as long as the robot is on a slope identical to the plants' ground plane slope, error in point translation is negligible.

Distance thresholds are determined by knowing the diameter of a sorghum plant. This method was selected because it is more versatile with the potential to be used for 3d imaging and mapping.

IV. RESULTS

The product of this research is a ROS executable that only requires a robot with defined transforms, and a laser range finder, LIDAR. The robot is able to aggregate many laser scans and, from that, be able to identify potential rows that are around the robot. The system only works if the robot is within the rows already. Once rows are identified, if the rows are relatively parallel, then the robot will be able to identify a line that represents the ideal path that the robot should be on, shown in Figure 8. The perpendicular distance of the robot's base_link coordinate frame to the path and the heading offset, the angle between the path found and the base_link coordinate frame's y-axis, is communicated to a line-following controller that will correct the robot's trajectory. If the angle between the two potential rows found is not parallel enough, the offset data will not be calculated, because there is not enough certainty that the path found would be the correct path to follow, which is shown in Figure 9.

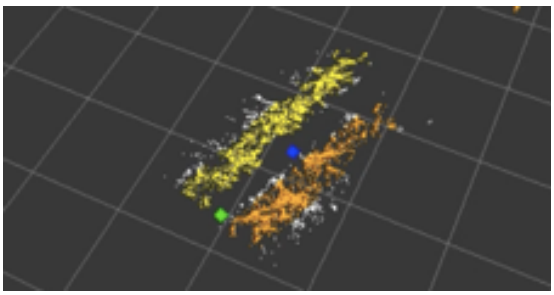


Fig. 8. The RANSAC algorithm has found two of the most prominent rows on both sides of the robot. The blue and green dots indicate the path. The path is expressed as the vector on the xy-plane, drawn from the blue dot to the green dot. The blue dot is always along the x-axis of the base_link coordinate frame.

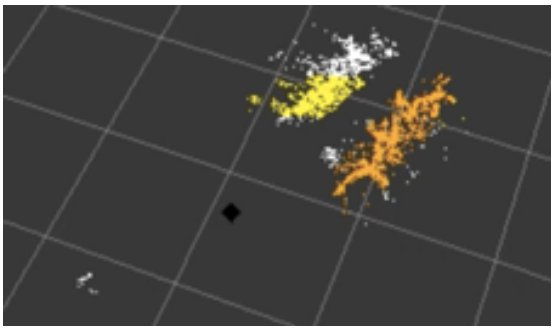


Fig 9. No path is able to be detected with certainty. The black dot represents the origin of the base_link coordinate frame. The yellow and orange colors are the most prominent rows, which do not line up well with the actual rows in the point cloud.

The general algorithm is as follows:

1. Have laser scans be collected for a certain amount of time, then aggregate laser scans into one point cloud

2. Remove ground plane

3. Remove points past 30 inches to left and right because the robot is assumed to always be inside the row and a row is no wider than 30 inches.

4. Fit the remaining points to a RANSAC line model

5. If that line was found to the left of the robot, fit another RANSAC line model to the points to the right of the robot, or vise versa.

6. If these lines form an angle no greater than $\pi/6$ radians, then it is safe to assume that a decent path can be found.

7. Using the equations of the lines that represent the rows, find an equation of a line of the path whose angle is the average angle of the two rows and a point is found by finding the middle point between the two rows at the x-axis.

8. Relative to the path line found, publish the perpendicular distance of the robot's base_link frame and the heading offset of the robot.

9. Update point cloud very quickly so that points captured closer to present time are represented in the cloud and perform steps 1-8 on every point cloud generated.

V. Conclusion & Future Work

A four-wheeled ground robot has been developed to autonomously phenotype sorghum bicolor plants. Its navigation is based solely on RTK GPS and algorithms relying on LIDAR laser scans. The base is ROS and any robot using ROS can apply this system easily. Further testing of the RTK GPS and LIDAR algorithms navigation combination must be done to identify what the robot is capable of in a real field setting. Additional future work includes creating a RANSAC model that is used for the sole purpose of finding two sorghum bicolor crop rows, and also implementing mapping for more robust navigation. Overall development of the robot includes testing of phenotyping features which includes a robotic arm attachment that uses a stereo camera to identify sorghum stalks and servos to the identified stalk and a vineyard camera developed by Dr. Stephen Nuske [4] calibrated to work with sorghum stalks. To sensors will be placed along the robot's mast to be sense temperature, relative humidity, ambient light and CO_2 levels.

Acknowledgments

Special thanks to my mentor, Dr. George Kantor, Timothy Mueller-Sim, Merritt Jenkins, Margaret Oates, and Justin Abel for their guidance and support. Also, thanks to the Field Robotics Center and the Robotics Institute Summer Scholars coordinates for making this such a lovely learning experience.

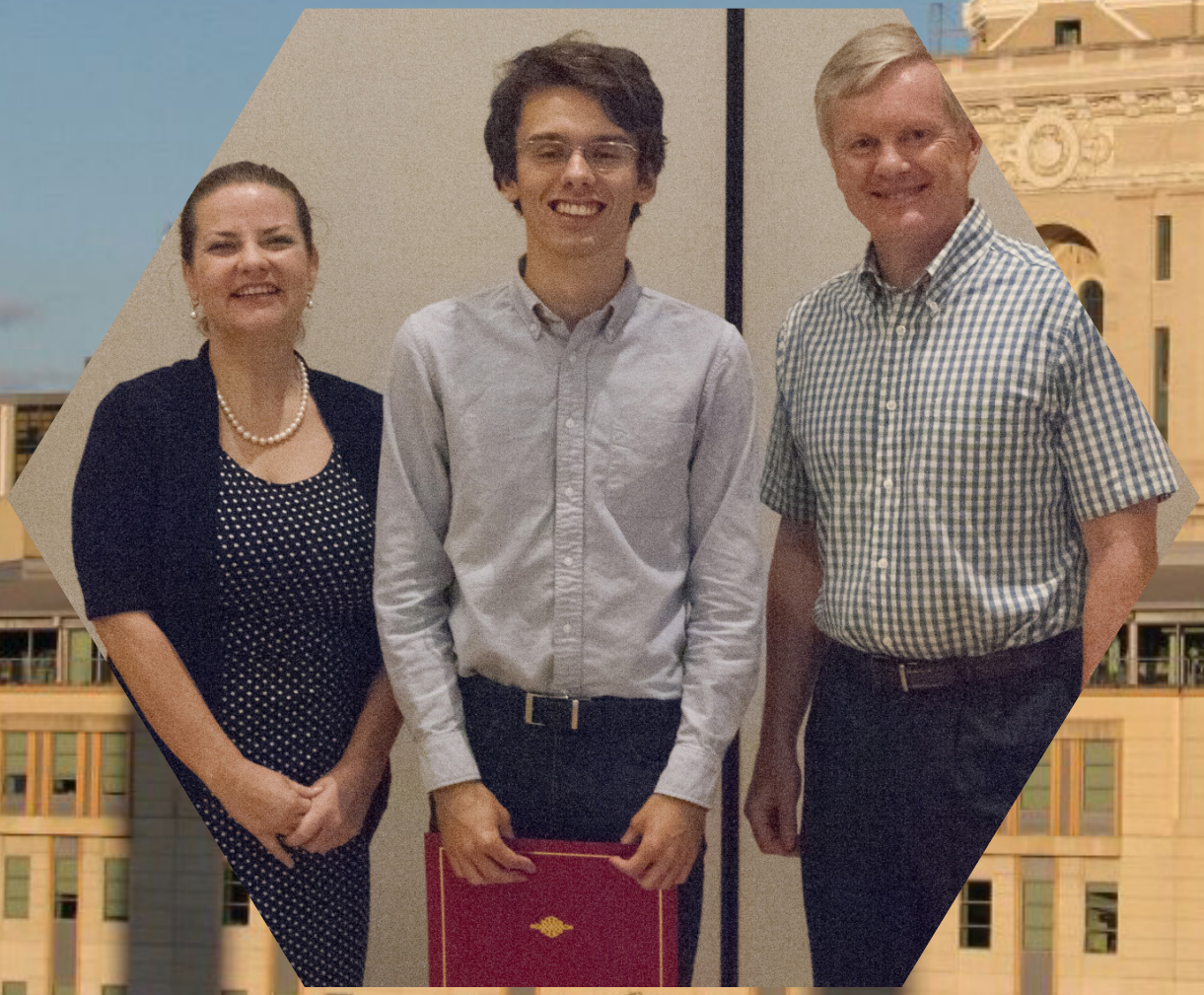
References

1. O. C. Barawid Jr., A. Mizushima, K. Ishii, and N. Noguchi, "Development of an Autonomous Navigation System using a Two-dimensional Laser Scanner in an Orchard Application," Japan, Biosystems Engineering, vol. 96, pp. 139–149, February 2007.
2. R. C. Coulter, "Implementation of the Pure Pursuit Tracking Algorithm", Carnegie Mellon University, Pittsburgh, PA, 1992.
3. Natural Sorghum Producers "Sorghum 101," http://sorghumgrowers.com/sorghum-101/, Lubbock, Texas, 2016.
4. S. Nuske, K. Wilshusen, S. Achar, L. Yoder, S. Narasimhan, and S. Singh, "Automated Visual Yield Estimation in Vineyards", Carnegie Mellon University, Pittsburgh, PA, September 2013
5. D. Reiser, G. Miguel, M. V. Arellano, H. W. Griepentrog, D. S. Paraforos, "Crop Row Detection in Maize for Developing Navigation Algorithms Under Changing Plant Growth Stages," Robot 2015: Sec. Iber. Robo. Conf., vol. 417, pp. 371–382, December 2015.
6. V. Subramanian, T. F. Burks, and A. A. Arroyo, "Development of machine vision and laser radar based autonomous vehicle guidance systems for citrus grove navigation," Florida, Computers and Electronics in Ag., vol. 53, pp. 130–143, September 2006.
7. United States Department of Agriculture, "SORGHUM: Sorghum bicolor (L.) Moench Plant Symbol=SOB12," Natural Resources Conservation Service, Plant Guide. Tuscan Plant Materials Center, AZ, February 2013.
8. J. Zhang, A. Chambers, S. Maeta, M. Bergerman, and S. Singh, "3D Perception for Accurate Row Following: Methodology and Results," IEEE Inter. Conf. on Intelligent Robots and Sys., Tokyo, Japan, November 2013.

Saul Shanabrook

University of Massachusetts

This research is aimed to develop a novel to compute shortest paths on spheres, by projecting the points into euclidean space using the Gnomonic projection and then using a visibility graph approach.



https://github.com/saulshanabrook/

Finding the Shortest Path on a Sphere Surface Using Visibility Graphs

Saul Shanabrook
Computer Science
University of Massachusets
s.shanabrook@gmail.com

***Abstract*—Existing methods for planning shortest routes around obstacles pertain to Euclidean space. They not directly support finding routes on the surface of a sphere. We map the sphere surface to a plane using Gnomonic projection. In it, the shortest paths on the sphere is projected as a straight line in the plane. We then create a visibility graph in the plane. Finally we search it using the distance on the sphere. Our proof shows that we are able to determine the shortest path between points on a sphere, around obstacles, with this method.**

***Keywords*—path finding, visibility graph, computational geometry.**

I. Introduction

Developing a global high level plan for a military airstrike requires calculating many possible flight paths. These paths must take into account threat regions and stay some distance away from them. Each set of threats will have many paths generated on it.

We chose to approximate the Earth as a sphere. It would be more accurate to model it as an oblate ellipsoid or a geoid, but it would also be more complex. The radius of the Earth deviates by less than 0.5% from the mean, which is an acceptable margin of error for us.

Our previous method for calculating the paths was a generative approach that approximated each threat as a rectangle:

1) Draw a direct path from the source to the destination
2) Determine the first obstacle this path will hit, and try going to the left or right of it. Repeat until it hits no more obstacles.
3) Smooth the path to minimize it's length.

This method, however, did not give any global guarantees that we had found the shortest path. It was also limited to approximating the threats as rectangles, instead of higher degree polygons.

We present a general method here to find the shortest path between two points on the surface of a sphere that does not intersect a set of spherical polygons. We then extend this method to support staying some margin away from all polygons and circles by approximating them with n-degree polynomials.

II. Related Work

A. Shortest Path

A popular method for path planning among a set of fixed obstacles is the Visibility Graph [1]. It was first described to navigate the robot SHAKEY [2]. It was then formally proven with line segments as obstacles [3] and extended to polygonal obstacles [4]. It is based on the fact that "Any shortest path between p_{start} and p_{goal} among a set S of disjoint polygonal obstacles is a polygonal path whose inner vertexes are vertexes of S" [5]. See Fig. 1 for a visual example of this property.

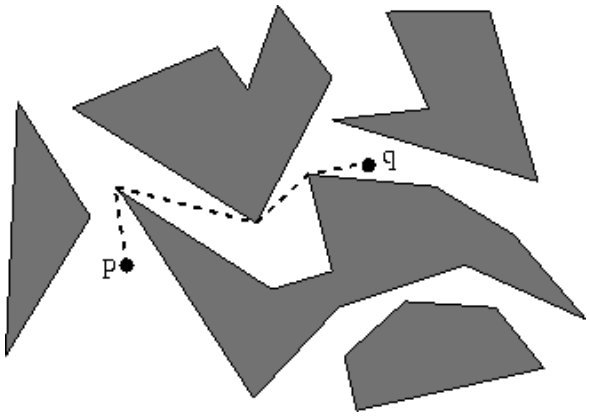


Figure 1: All the inner vertexes in the shortest path between p and q are vertexes of the obstacles [6].

The visibility graph is the set of all line segments between vertexes that can see each other. Since all the inner vertexes of the shortest path must be obstacle vertexes, this graph must contain the shortest path. All that remains is to search this graph for the path, using a weighted graph search algorithm (like A* [7]), from the source to the destination, with the cost equal to the euclidean distance between each vertex.

The most naive implementation for creating the visibility graph takes $O(N^3)$ time where N is the total number of obstacles vertexes:

```
function CREATE VISIBILITY GRAPH(os, s, d)
    g ← NEW GRAPH
    for o ← all vertexes in os, s, and d do
        for a ← all vertexes in os, s, and d do
            for b ← all other vertexes in os do
                if IS VISIBLE(os, a, b) then
                    add ab to g
    return g
function IS VISIBLE(os, a, b)
    for cd ← all edges in os do
        if INTERSECTS(ab, cd) then
            return false
    return true
```

There have been a variety of other algorithms proposed that achieve faster algorithmic complexity, while trading simplicity or by constraints [5]. Many of these require non overlapping obstacles.

The ”Virtual Stretched String” method is a more recent attempt to plan a path around obstacles, that only requires looking at the obstacles near the direct path, instead of all obstacles in the space [8]. It is also possible to combine the Voronoi diagram with the visibility graph to plan a natural looking path that stays some margin away from all surfaces [9].

B. Sphere Surfaces

In our method, we treat the obstacles a spherical polygons. These are defined as a list of points on a sphere, connected by shortest paths [10] similar to Euclidean polygons. The difference is that the shortest path between two points on a sphere is a great circle arc. A great circle is a ”section of a sphere that contains a diameter of the sphere” [11]. Great circle arcs are one type of geodesic, which is a shorest path between two points on a surface [12]. Any two non antipodal points on a sphere have a unique great circle arc between them. If we restrict ourselves to a portion of the sphere without antipodal points (i.e. less than one hemisphere), then we can say that two points are mutually visible if the great circle arc between them is unobstructed [13]. Any set of points on a sphere that are contained in a hemisphere are in Euclidean position. Points on any surface are in Euclidean position, if they can be translated to a surface and retain some key geometric behaviors [14]. This concept can be understood intuitively [15] [sic]:

> Most people along the human history have believed in a flat earth. Even nowadays there exit persons that still hold the flatness of the world. This is because most of our daily experience takes place in a restricted region of a sphere-like surface, so that there are no significant errors if it is considered as a plane. This idea can be easily extended to the Computational Geometry context, where in multiple applications it is assumed that if a given data set is constrained to a small portion of a surface, it presents a planar behavior.

C. Geometric Projection

The Gnomonic Projection is a method for translating points on Euclidean position on a sphere to a plane that maintains some geometric behaviors. It is unique amoung all projects in that “all great circle paths — the shortest routes between points on a sphere — are shown as straight lines” [16]. Since it can only transform a set of points in Euclidean position, it requires choosing a center point parameter. All other points must be within $\frac{\pi}{2}$ away from this center point, so that they lie in a hemisphere. Fig. 2 shows as example of this projection.

To transform a point with latitude ϕ and longitude λ, given a center point of latitude ϕ_1 and longitude λ_0 [17]:

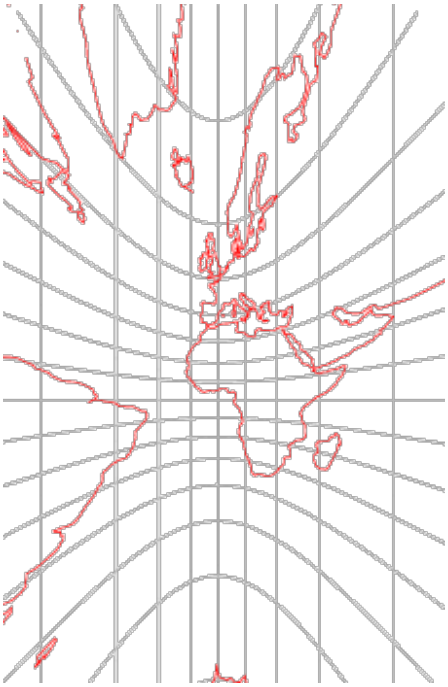


Figure 2: The Gnomonic projection centered on 0 latitude and 0 longitude [17].

$$x = \frac{\cos\phi \sin(\lambda - \lambda_0)}{\cos c}$$
$$y = \frac{\cos\phi_1 \sin\phi - \sin\phi_1 \cos\phi \cos(\lambda - \lambda_0}{\cos c}$$
$$\cos c = \sin\phi_1 \sin\phi + \cos\phi_1 \cos\phi \cos(\lambda - \lambda_0)$$

We can also do the inverse transform, from a euclidean point and a center on the sphere to a spherical point.

III. Methods

First we will look at the method we developed, in the abstract and a proof for it’s validity. Then we will consider our usage of this method in our larger pipeline.

A. Abstract Solution

We present a method to find the globally shortest path between two points that is not obstructed by a set of spherical polygons. All of the vertexes of the polygons and the points need to be on one hemisphere.

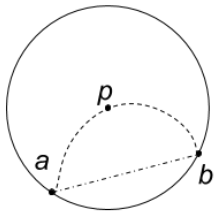
1) Choose a center point, where the great circle distance between it and all of the points lie on the hemisphere centered here.
2) Project all of the points onto a plane, using the gnomonic projection with the chosen center point.
3) Create a visibility graph with the projected points, using any euclidean visibility graph algorithm.
4) Search the graph using A* from the source to the destination, using the great circle distance between the spherical versions of each vertex, as the path cost and heuristic.

B. Proof

We are interested in a method to search for the shortest from points S to G on the surface of a sphere. Let O be a set of spherical polygons that the path should not intersect. The edges of a spherical polygon are great circle arcs between their vertexes. Let C be S, G, and all vertexes of O. We restrict ourselves to the case where there exists some center point R where the great circle distance between R and all members of C is $< \frac{\pi}{2}$ (i.e. all points lie on one hemisphere). Let $P(p) =$ be the gnomonic projection of p, with center R, given that $p \in C$. P is a bijective function. All great circle arcs between members of C are projected as lines, because that is a property of the gnomonic projection.

Lemma 1. *The shortest path between S and G is a polygonal path whose inner vertexes are the vertexes of O.*

Proof: Lemme 15.1 of [5] states "Any shortest path between p_{start} and p_{goal} among a set S of disjoint polygonal obstacles is a polygonal path whose inner vertexes are vertexes of S." Their proof will hold if we let $p_{\text{start}} = P(S)$, $p_{\text{goal}} = P(G)$, $S = P(O)$, and substitute spherical distance between the inverse projection of points for euclidean distance. They use the fact that the straight line from two points a and b on the edge of a circle with center p is shorter than or equal to the length of a line from a to b that intersects p. This is also true for the spherical distance between the inverse projection of a and b, because a straight line is inversely projected as a great circle arc, which is the shortest path between two points on a sphere, so it has to be the shortest path on the sphere between $P^{-1}(a)$ and $P^{-1}(b)$. ■



Lemma 2. *The shortest path between S and G will consist entirely of edges from a visibility graph VG where the vertexes are C and an edge exists between two vertexes iff they have a shortest path between them that is not obstructed by any polygons from O.*

Proof: We prove this by contradiction. Let an edge E be in the shortest path but not in the visibility graph. The endpoints of E must either be vertexes of O, if they are inner vertexes of the shortest path because of the previous lemma, or be S or G because the outer vertexes of the path must be the start and end points. So the endpoints of E must be in C. Since the shortest path cannot intersect any polygons from O, E also cannot intersect any polygons in O. Since E is does not intersect any polygons from O and its endpoints are in C, it must be an edge in the visibility graph. ■

Lemma 3. *Creating a visibility graph VG' from $P(S)$, $P(G)$, and $P(O)$ and taking the inverse projection of its nodes, will give us a visibility graph VG that is equal to the definition in the previous lemma.*

Proof: We know that the vertexes of VG will be C, because the vertexes of VG' are $P(S)$, $P(G)$, and $P(O)$ and we apply P^{-1} to these to get VG and $P^{-1}(P(a)) = a$, because P is bijective. Now we need to prove an edge will exist in VG iff it exists in VG'. In VG, an edge between two points a and b exists iff the great circle arc ab does not intersect any edges in O.

We first consider the case where ab does intersect an edge de and so should not appear in VG. It will cross at some point f, which is on the great circle arcs ab and de. $P(a)P(b)$ and $P(d)P(e)$ will be projected as straight line that must intersect because they both contain point $P(f)$. So $P(a)P(b)$ must not exist in VG', so ab will not exist in VG.

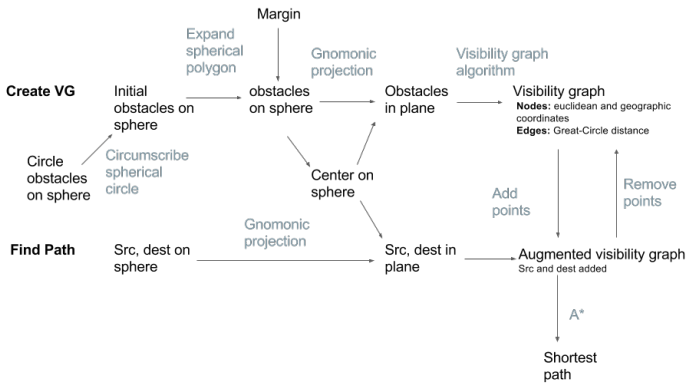
Now let us consider the other case where ab does not intersect an edge and so should appear in VG. Let us prove this by contradiction, by starting with the premise ab does not intersect an edge and does not appear in VG. Therefore $P(a)P(b)$ must not exist in VG'. So then there must exist some edge de and some point f which intersects with $P(a)P(b)$ on the plane. If we inversely project this edge, onto the sphere, then it must contain the point $P^{-1}(f)$ and the great circle ab must also contain this point, since straight lines are inversely projected as great circles. So ab does intersect an edge. ■

Lemma 4. *We can use the A* algorithm to find the shortest path between S and G on VG, using great circle distance as the edge cost and heuristic.*

Proof: Great circle distance is a valid heuristic because there is no shorter distance than the great circle minor arc between two points on a sphere, so it is a valid lower bound on the cost. ■

C. Concrete Implementation

We use the above method as part of a pipeline to compute possible flights paths in our airstrike planner.



We separate this into two different stages. First we create the visibility graph (VG) for a set of unique obstacles. Then for every path we want to generate with those obstacles, we use that augment the visibility graph and search it. This allows us to reuse it for many paths. This saves some time, because the runtime cost of adding and removing the endpoints is much less than that of recreating the whole visibility graph. For one scenario, we also compute many different visibility graphs, with different unique sets of threats.

We found that creating and updating the visibility graph accounted for 90% of the runtime of the whole pipeline. In order to make this approach viable, we used a simple visibility

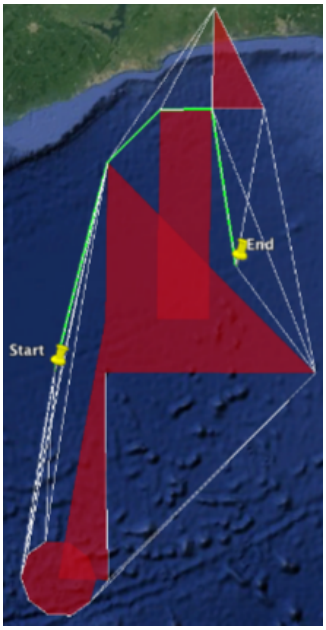


Figure 3: Overlapping obstacles

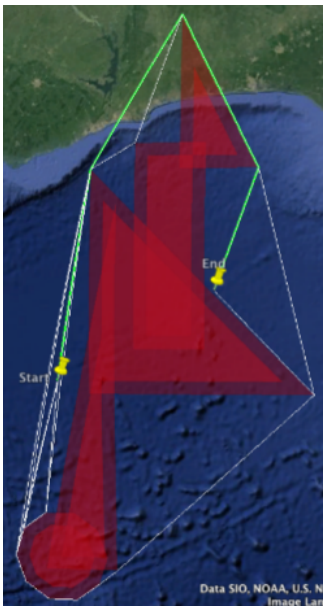


Figure 4: Expanded obstacles

graph algorithm and optimized it heavily. By making sure to check each edge of the graph once, instead of twice, we can halve the visibility graph creation time [18]. Since we have overlapping polygons, we also precomputed whether every vertex is inside a different polygon. If it was, we didn't both to check if any other points could see it. When iterating through the list of obstacles, we also added the edge from each vertex to it's neighbors directly. Then we could skip checking a vertex of an obstacle against all other vertexes of that obstacle.

IV. Results

We first tested this method on manually created edge case problems. We verified that it worked correctly with overlapping polygons (Figure 3) and that it correctly expanded polygons by some margin (Figure 4).

We then benchmarked the performance of finding paths as a factor of the number of polygons. We created a set of normal polygons, within a subsection of the sphere surface (Figure 5). We randomly varied their size and position. We varied the number of polygons (n = 0 to 100) and the number of sides per polygon (n = 3 to 20). We found that the total time to create the visibility graph and find the path increased with the number of sides and number of polygons (Figure 6).

We then integrated this new process into our codebase and compared it against the previous iterative approach. We choose one representative scenario (Figure 7) and ran both versions on it multiple times (n = 100). In this scenario, 8 different sets of threats were used and approximately 2k paths were found. We compare the time spent only in the path finding step.

The mean total time decreased by 20% (Figure 8). Also, when we did not have to generate a new visibility graph, the runtime increased as the path length increased, for both the old and the new method, but was significantly lower in the new ¡insert graph¿.

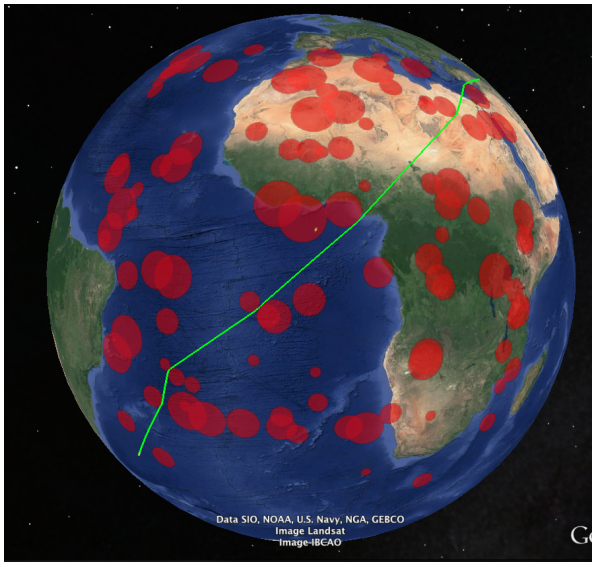


Figure 5: Visualization of randomly created benchmark, with 100 polygons, each with 20 sides

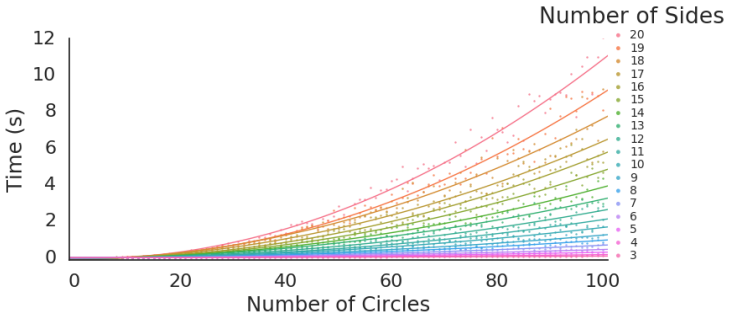


Figure 6: The total time (to create the visibility graph and find the shortest path between two points) increased with the number of circles (obstacles, approximated by polygons) and the number of sides per circle, i.e. the total number of obstacle vertexes.

V. Conclusions

We succeeded in creating a method to find paths across a sphere, around obstacles, that could deal with high resolution polygons and was fast enough. It is possible to use the method with any algorithm to create a visibility graph. It is also a conceptually simple method that can be implemented in multiple discrete modules.

However, it is limited to points on one hemisphere of the sphere. The runtime is also highly dependent on the visibility graph implementation and the number of obstacle vertexes.

VI. Future Work

This work could be expanded to any points in Euclidean position on any surface. The key is to find a way to map those points to a euclidean plane that maintains visibility.

It would also be possible to compute the visibility graph directly on the sphere, without projecting the points. This would allow us to find paths between points across hemispheres,

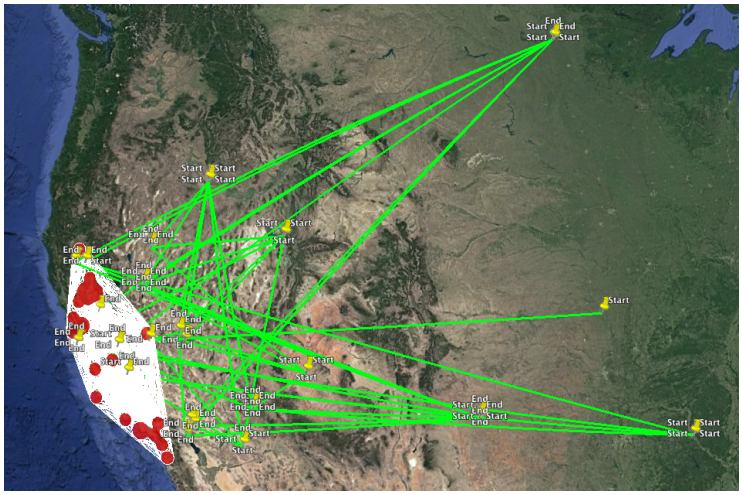


Figure 7: Overlay of some of the paths and visibility graphs created for our scenario we used for benchmarking.

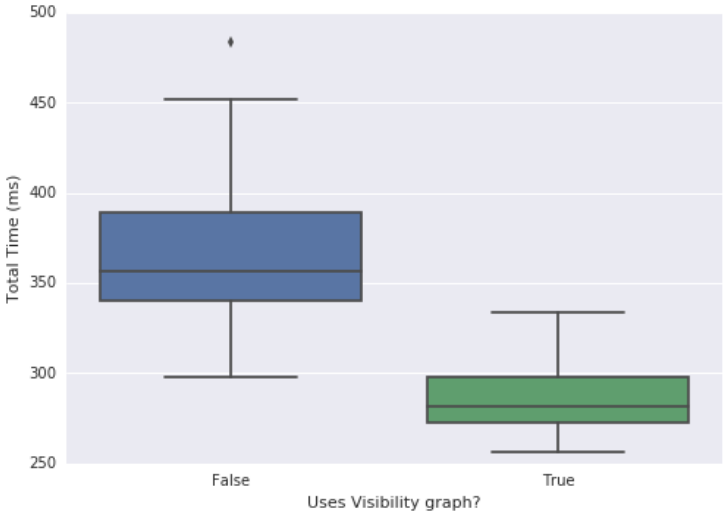


Figure 8: Total runtime spent finding paths (including creation of visibility graph) decreased when we switched to the method described in this paper.

by looking at both the great circle minor and major arcs to check visibility. However, this would require implementing more spherical trigonometry and entangles the visibility graph creation code with the spherical geometry.

Acknowledgment

I would like to thank Zachary Rubinstein for proposing this problem and bouncing ideas around with me, as well as Laura Barbulescu for working through the concepts in the proof.

Also to the fellow students and staff at RISS for editing my paper and providing the structure that made this work possible.

References

[1] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, 1979. [Online]. Available: http://dl.acm.org/citation.cfm?id=359164

[2] N. J. Nilsson, "A mobius automation: An application of artificial intelligence techniques," in *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, ser. IJCAI'69. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1969, pp. 509–520. [Online]. Available: http://dl.acm.org/citation.cfm?id=1624562.1624607

[3] D.-T. Lee, "Proximity and reachability in the plane." Ph.D. dissertation, Champaign, IL, USA, 1978, aAI7913526.

[4] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Commun. ACM*, vol. 22, no. 10, pp. 560–570, Oct. 1979. [Online]. Available: http://doi.acm.org/10.1145/359156.359164

[5] M. De Berg, M. Van Kreveld, M. Overmars, and O. C. Schwarzkopf, "Visibility Graphs," in *Computational Geometry*. Springer Berlin Heidelberg, 2008, pp. 323–333. [Online]. Available: http://link.springer.com.silk.library.umass.edu/chapter/10.1007/978-3-540-77974-2_15

[6] D. Z. Chen, "Developing algorithms and software for geometric path planning problems," *ACM Computing Surveys (CSUR)*, vol. 28, no. 4es, pp. 18–es, Dec. 1996. [Online]. Available: http://dl.acm.org/ft_gateway.cfm?id=242246&typehtml

[7] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, July 1968.

[8] A. Sengupta, R. Ray, and S. N. Shome, "Virtual stretched string: An optimal path planning technique over polygonal obstacles," in *Proceedings of Conference on Advances In Robotics*, ser. AIR '13. New York, NY, USA: ACM, 2013, pp. 95:1–95:5. [Online]. Available: http://doi.acm.org/10.1145/2506095.2506128

[9] R. Wein, J. P. van den Berg, and D. Halperin, "The visibility–voronoi complex and its applications," in *Proceedings of the Twenty-first Annual Symposium on Computational Geometry*, ser. SCG '05. New York, NY, USA: ACM, 2005, pp. 63–72. [Online]. Available: http://doi.acm.org/10.1145/1064092.1064104

[10] E. W. Weisstein, "Spherical polygon. From MathWorld—A Wolfram Web Resource," last visited on 31/7/2016. [Online]. Available: http://mathworld.wolfram.com/SphericalPolygon.html

[11] ——, "Great circle. From MathWorld—A Wolfram Web Resource," last visited on 31/7/2016. [Online]. Available: http://mathworld.wolfram.com/GreatCircle.html

[12] ——, "Goedesic. From MathWorld—A Wolfram Web Resource," last visited on 31/7/2016. [Online]. Available: http://mathworld.wolfram.com/Geodesic.html

[13] A. Rosenfeld and A. Y. Wu, "Geodesic convexity in discrete spaces," *Information Sciences*, vol. 80, no. 1, pp. 127 – 132, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0020025594900604

[14] C. Grima and A. Márquez, *Computational Geometry on Surfaces: Performing Computational Geometry on the Cylinder, the Sphere, the Torus, and the Cone*. Springer Science & Business Media, 2013.

[15] M. Abellanas, C. Cortées, and G. Hernéandez, "Maximum subsets in euclidean position."

[16] J. P. Snyder, "Map projections: A working manual," Washington, D.C. C6 - ET -, Tech. Rep. [Online]. Available: http://pubs.er.usgs.gov/publication/pp1395

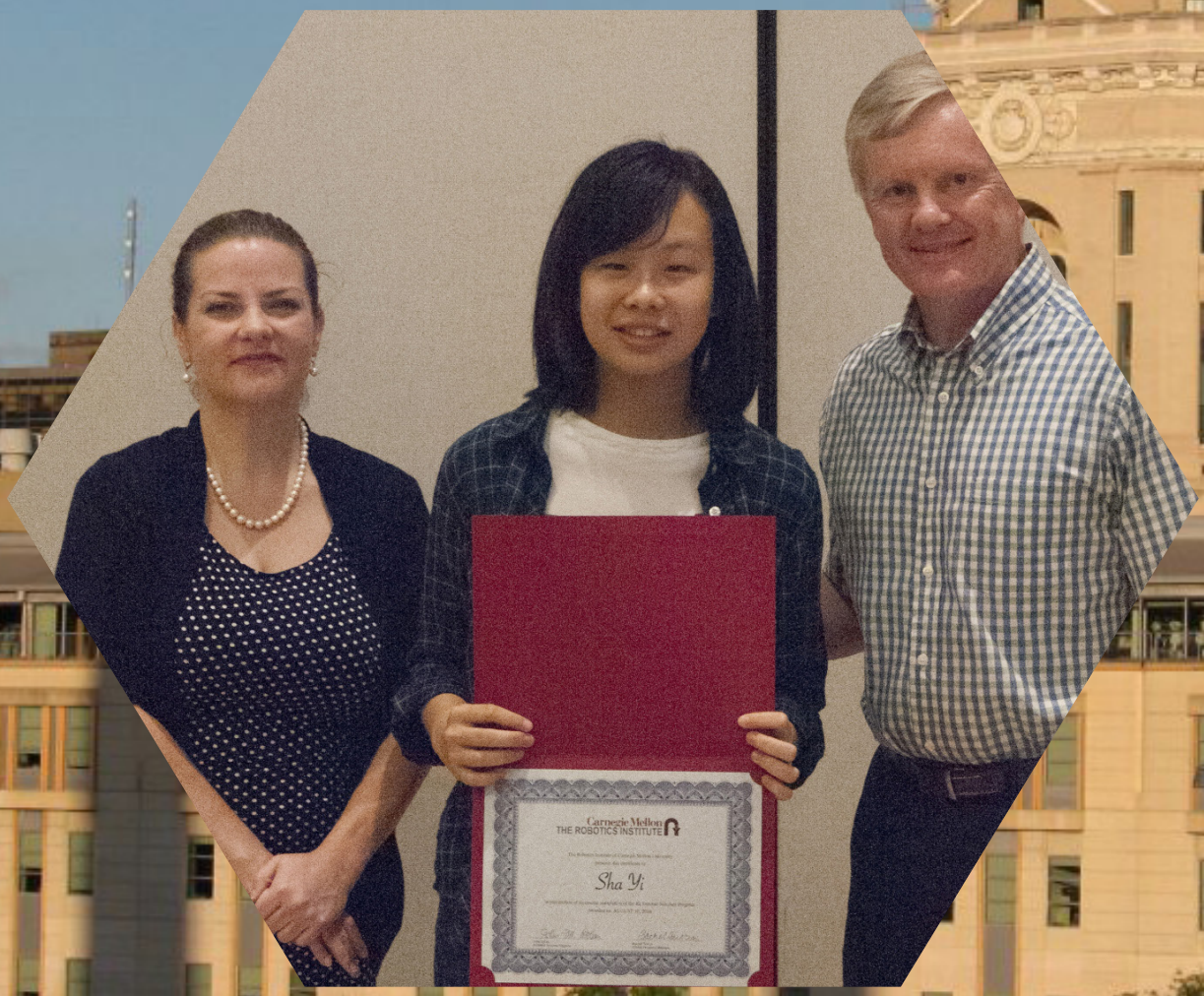
[17] E. W. Weisstein, "Gnomonic projection. From MathWorld—A Wolfram Web Resource," last visited on 31/7/2016. [Online]. Available: http://mathworld.wolfram.com/GnomonicProjection.html

[18] J. Kitzinger, "The visibility graph among polygonal obstacles: a comparison of algorithms."

Sha Yi

The Hong Kong Polytechnic University

Multi-camera Visual Odometry



https://hk.linkedin.com/in/sha-yi-128031a8

https://github.com/yswhynot

Monocular Visual Odometry for Indoor Environment with Initial Map Building with LIDAR Input

Sha Yi, Chaohui Gong, Howie Choset

***Abstract*—Visual Odometry is widely used for mobile robot localization because of its relatively low cost compared with high precision sensor system, such as LIDAR. However, pure visual odometry is usually of low accuracy, which is caused by accumulated drift, inaccurate feature matching, inaccurate scaling factor, and unstable environment. In this work, we propose a system of combining the pose from LOAM (LIDAR odometry and mapping) and visual feature points (ORB) to build an initial map of the environment, then localize a mobile robot with only a monocular camera onboard and build a denser map with reference to the initial map to enhance accuracy compared with pure monocular visual odometry. In this way, the localization accuracy of a mobile robot in indoor environment could be significantly improved with a single LIDAR, of which the one-time cost is ignorable for a large amount of robots and for a long term.**

***Index Terms*—Monocular Vision, Localization, Visual Odometry.**

I. Introduction

Pose estimation is one of the most vital parts in robotics. Some of the key tasks for robots: path planning, object avoidance etc. all rely on the localizing the robot and do computation based on it.

There are multiple ways to do localization. Lidar has become popular these years for localization. It makes use of laser to obtain point cloud of the environment and compare adjacent point cloud to localize the robot. It is general of high accuracy, but the cost for Lidar system is much higher than most of the visual localization system so that industrial applications tend not to use Lidar on every robot. Some low cost solutions are built with visual systems. For example, Apriltag [5] is a popular approach to get the pose of camera. They are special features encoded for pose detection. Apriltag approach is relatively cheap [5], fast, and very accurate in short distance. However, it is difficult to maintain tags in industrial environment and the accuracy is not high for long distance [5]. Visual Odometry is another visual localization solution that reduces cost. However, it is not as accurate as the previous solutions since it accumulates drift during the process of localization.

Visual odometry localizes the robot from Structure from Motion [1]. The process of doing structure from motion is:

S. Yi is with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic Unverisity, Hong Kong email: 12132054d@connect.polyu.hk

C. Gong and H. Choset are with the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA, 15213, USA. emails: chaohuig@andrew.cmu.edu, choset@cs.cmu.edu

- Read the image frame from the camera sensor
- Detect visual feature points from the image frame
- Extract feature descriptors from the visual feature points
- Search between the features points of consecutive frames for feature matching
- Calculate Essential Matrix/Fundamental Matrix
- Get pose (rotation and translation matrix) from the Essential Matrix/Fundamental Matrix

We may additionally do triangulation to get the 3D location of the matched feature points, so that it is possible to use those 3D points for mapping. However, the pure visual odometry is not of high accuracy.

In this work, we propose an indoor mapping and localization system for factory or office scenario, which consists of one-time mapping and long-term visual localization. To build an accurate visual feature map of the environment, a mobile robot equipped with Lidar and cameras will explore the environment and log the pose from LOAM [8] and corresponding visual feature points in camera image. The map is stored for further visual localization untill the environment has significant change, when a re-mapping is required. With the visual feature map, a mobile robot with only a low cost camera on board can accurately localize itself with no accumulated drift effect.. In this case, several companies may share the cost of one Lidar system and they may only need it again when the company change their environment. Therefore, the cost of localization could be greatly reduced. The system operates in the following process:

- Do the first four steps as in the previous structure from motion procedure
- Read the Lidar pose
- Do triangulation on the matched feature points based on the input Lidar pose
- Get the 3D location of the visual feature points

With those feature points, it is possible to build and save a 3D map of the environment and the robot could localize itself later in the tracking process.

In this work, we used the commercialized LIDAR system Stencil from Real Earth [8] for pose estimation, and ROS (Robot Operating System) [6] for hardware interface and communication.

II. Related Work

A. Project Tango

Google developed the Project Tango [2] system to do visual odometry for localization. The Tango system fuses the visual estimation with the pose information of inertial measurement unit (IMU). The system operates by creating an initial map of the environment. However, the Tango system is not open source, thus there is little flexibility to build and modify with industrial specifications. Our goal is to develop a system to replace the dependency on Project Tango on the current robot.

B. ORB-SLAM

Mur-Artal et al. [3] proposed the ORB-SLAM system that can be used for monocular vision. It is a pure vision approach so that it is impossible to get the scaling factor of the odometry. The system structure is shown in Fig. 1:

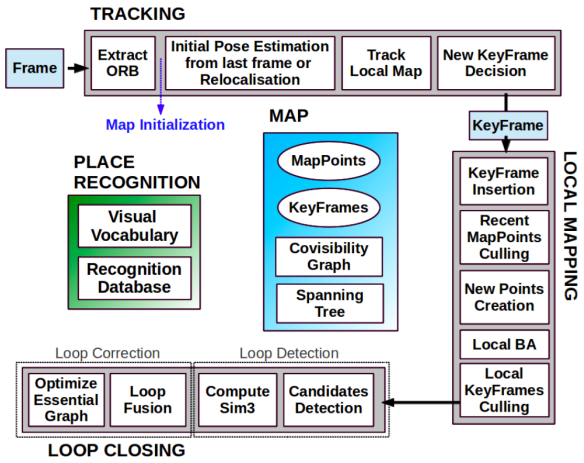


Fig. 1: ORB-SLAM System [3]

The ORB-SLAM system combines most of the popular visual odometry approaches including bundle adjustment [7], loop closing, key frame culling, etc. The monocular system runs with an initialization process with structure from motion first, then assumes a constant velocity motion model and only do least square optimization on this model. Therefore, it is relatively robust and fast enough. Thus, we build our system on the ORB-SLAM and since it is open source, we do our modification on the code it provides.

III. Methodology

A. Triangulation

The 3D location of feature points are computed using triangulation [1]. We could obtain the previous pose $g_{l0} = \{ R_{l0} \mid t_{l0} \}$ and the current pose $g_{l1} = \{ R_{l1} \mid t_{l1} \}$ from LIDAR odometry. We can then get the camera pose:

$$g_c = g_l^c g_l \tag{1}$$

where g_l^c is the transform from LIDAR to camera. Therefore the Projection matrix could be obtained:

$$\boldsymbol{P_c} = \boldsymbol{K}\,[\boldsymbol{R_c}|\boldsymbol{t_c}] \tag{2}$$

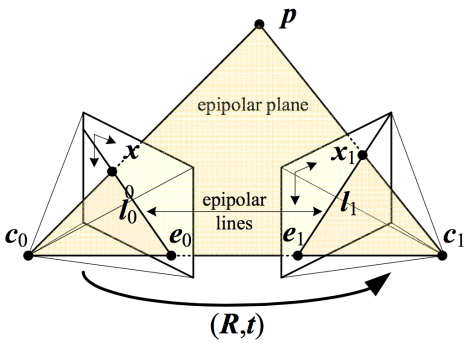


Fig. 2: Epipolar Geometry [1]

where $\boldsymbol{K}$ is the 3-by-3 intrinsic matrix of the camera.

As shown in the Fig. 2, $\boldsymbol{c_0}$ and $\boldsymbol{c_1}$ are camera centers which could be obtained from

$$\boldsymbol{t_j} = -\boldsymbol{R_j c_j} \tag{3}$$

Then the direction of rays $\hat{\boldsymbol{v}}_{\boldsymbol{j}}$ as shown in the Fig. 2 could be obtained [1]:

$$\hat{\boldsymbol{v}}_{\boldsymbol{j}} = \boldsymbol{R_j^{-1} K_j^{-1} x_j} \tag{4}$$

where $\boldsymbol{x_j}$ is the homogenous coordinate of each 2D feature point. Then the optimal point $\boldsymbol{p}$, as proven in [1], could be computed with:

$$\boldsymbol{p} = \left[\sum_j (\boldsymbol{I} - \hat{\boldsymbol{v}}_j \hat{\boldsymbol{v}}_j^T)\right]^{-1} \left[\sum_j (\boldsymbol{I} - \hat{\boldsymbol{v}}_j \hat{\boldsymbol{v}}_j^T)\boldsymbol{c_j}\right] \tag{5}$$

Then all the triangulated feature points will be saved as the map of the environment.

B. Map Building

With the pose input from LIDAR [8], it is not necessary to do the initialization process for the monocular tracking. In this case, we replace the initialization process with map building. The detailed process is shown in Fig. 3.

The reprojection process is done by triangulation mentioned in the previous section.

Then Least-square method Levenberg-Marquardt algorithm [7] is used to optimize the reconstructed map points, with pose fixed. The reason for this is that the pose output from Stencil LIDAR system is accurate enough, within 2 cm, to act as ground truth for the measurement.

The loop closing thread will keep detecting loop throughout the map building process. When a loop is detected, global bundle adjustment will start to avoid the drift generated from the visual odometry and optimize the 3D map points once again.

Thoughout the entire map building process, all the original key frame culling algorithm and reference frame tracking is kept. Therefore, it is more convenient for later tracking process since it preserves optimal information of the initial environment and odometry information.

When all the above process is done correctly, the map, map points, essential graph, key frames, key frame database are saved to the file system for future use.

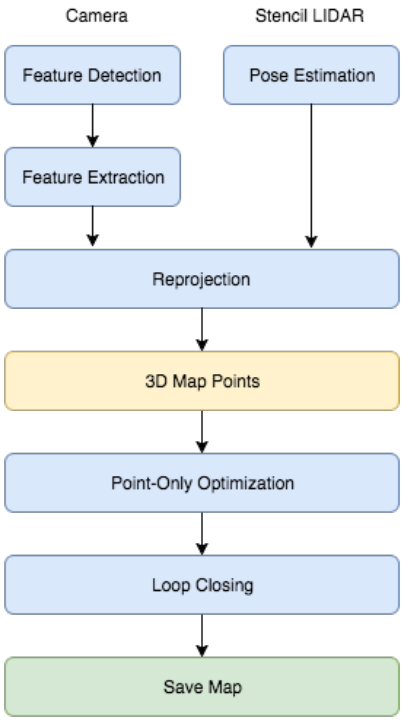


Fig. 3: Map Building Process

C. Tracking

During the tracking process, most of the original ORB-SLAM is kept for the system. The major difference is that the new system has a comprehensive ROS wrapper that broadcast topics on its current pose, initialization stage, number of features points so that it is easier to record data.

There are still three threads, tracking, loop closing and local mapping. The local mapping thread will be called once 3 seconds to relocate the robot with reference to the initial map built in the previous process.

D. Bag of Words

Nister and Stewenius [4] suggest the use of indexing feature descriptors to build a vocabulary tree for scalable recognition. In our case, vocabulary tree is built before hand and we make use of this for search on feature matching. As the ORB-SLAM [3] system suggested, the vocabulary tree is built on the detection of comprehensive dataset of indoor office environment, thus it fits our need in this case.

The bag of words matching will assign different clusters to feature point descriptor and only those that lie in the same cluster will be matched. This improves the searching speed to a great extent compared with brute force searching and matching of all feature points.

E. Constant Velocity Model

To speed up tracking process, the system is assuming a constant velocity model [4]. It makes assumption that when the camera frame rate is fast enough, the speed is more or less constant between adjacent frames.

Then when a new frame comes into the system, it first predicts its camera pose by multiplying constant velocity to the previous frame pose. The previous velocity v between pose g_0 and g_1 could be represented as:

$$v = g_0^{-1} g_1 \tag{6}$$

Therefore, by assuming constant velocity, the next pose g_2 could be predicted:

$$g_2 = g_1 v \tag{7}$$

Then least-square method is made on the current frame pose g_2. It takes the visible 3D map points and projects onto the current frame and minimize the projection error on the frame.

IV. Tests and Results

All the tests are done with the default ORB-SLAM parameters [3]. LIDAR is running with the Localization mode [8]. The robot we are using is shown below in Fig. 4:

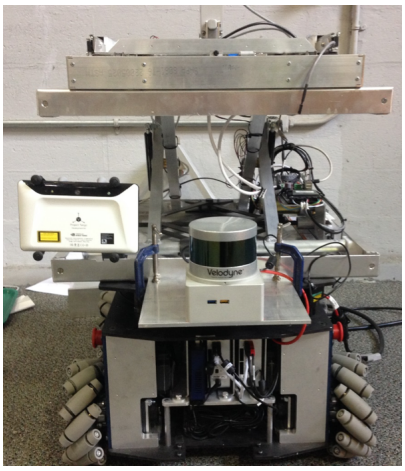


Fig. 4: Intelligent Mobile Robot (IMR)

A. Constant Velocity Model

The result is shown in Fig. 5 with a 10 Hz monocular camera, taking LIDAR [8] as ground truth.

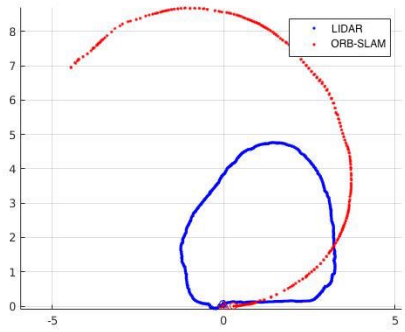


Fig. 5: Constant Velocity Model Test Result

However, as seen in the graph, speed is not always constant and it performs extremely bad when doing sharp turns. Though it might be better with a high frame rate camera, it is still necessary to have a better motion model to speed up the matching and optimizing process.

B. Joy Stick Twist Input

Since the constant velocity model is not accurate, we tried to use joy stick twist command information as motion model for prediction and do optimization on the predicted pose, i.e. replace the velocity v from the previous pose with the velocity input from the joy stick twist command. The result is in Fig. 6.

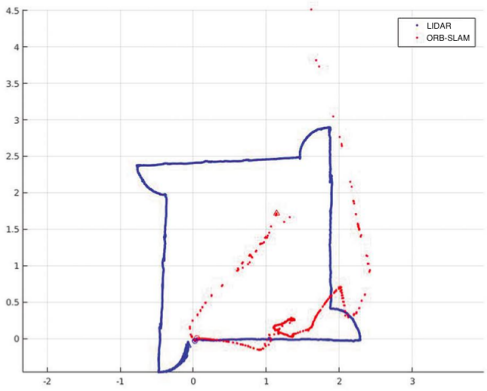


Fig. 6: Joy Stick Input Test Result

However, the joy stick twist information is far more inaccurate than the desired result.

V. Discussion

A. Image Processing

During the tracking process, it is very easy to lose track, especially when encountering challenging environment. These challenging environments include feature-less environment like walls and floor, and scenes that only have high contract and simple pattern.

To enhance the robustness, image filter is added to the camera frame before inputting to the visual odometry system. The current filter is a sharpen filter that deducts the original image with a Gaussian blurred image and this reduces the number of track lost on the snake robot when watching the floor.

B. Data Security

Another concern of using initial map instead of mapping every time when localizing is for the security reason.

LIDAR operates by collecting and comparing dense point cloud for mapping and localization, which means the detailed shape of surrounding object could be reconstructed from the LIDAR point cloud. It is possible that some of the factories or industries are high concerned about privacy so that they do not want their entire environment to be reconstructed by LIDAR point cloud so that people might hack into their system and get all the details of their environment.

The map-based visual odometry only store map point and is mapped offline every time after building the map. This is a much more secure and stable way if the environment does not change regularly.

VI. Future Work

A. Multiple Camera Pose Fusion

During our tests, we found that cameras on different position have various accuracy for the movement. For instance, when doing a left turn, the camera on the right side gives more accurate result than the left one, since it has longer trajectory and the features points move more obviously than those on the left. In another case when moving straight forward, the front camera might not provide as good pose result as the side cameras, since pin hole camera is better for rotation than translation or scaling.

If we could fuse multiple odometries together dynamically by taking the body twist as reference, we may obtain much better result than the result from only one camera.

B. Shaky Vision

In some cases, when the motion is very noisy, like on snake robot, the system is more likely to lose track because of blurred image or the drawback of constant velocity model.

One way to enhance the robustness during shaky motions is to change the camera with a better camera that has higher frame rate. Another way is to replace the constant velocity model with other models. For instance, if the snake robot is moving in a circular motion, we may input the circular motion as a reference so that it might make optimization easier.

C. Information Gain Evaluation

Since the visual odometry is usually connected with path planning or related decisions, it is possible to evaluate the environment and do better planning on this. For example, when the robot is about to turn to a clean wall or feature-less ground, by calculating the information gain we might be able to tell the upcoming scene is good or bad for the visual system. If the result is bad, we might need to avoid those scenes so that we get a more robust automation system.

VII. Acknowledgement

Acknowledgement to Chaohui Gong for guiding me all the way. Thanks to Elena Morara, Alexander Ansari, Lu Li, Zhongqiang Ren, Jin Dai, Puru Rastogi for giving suggestions. Thanks to Prof. Howie Choset for having me in the Biorobotics lab.

References

[1] David A Forsyth and Jean Ponce. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.

[2] R. Dugan J.C. Lee. Google project tango.

[3] Raul Mur-Artal, JMM Montiel, and Juan D Tardós. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

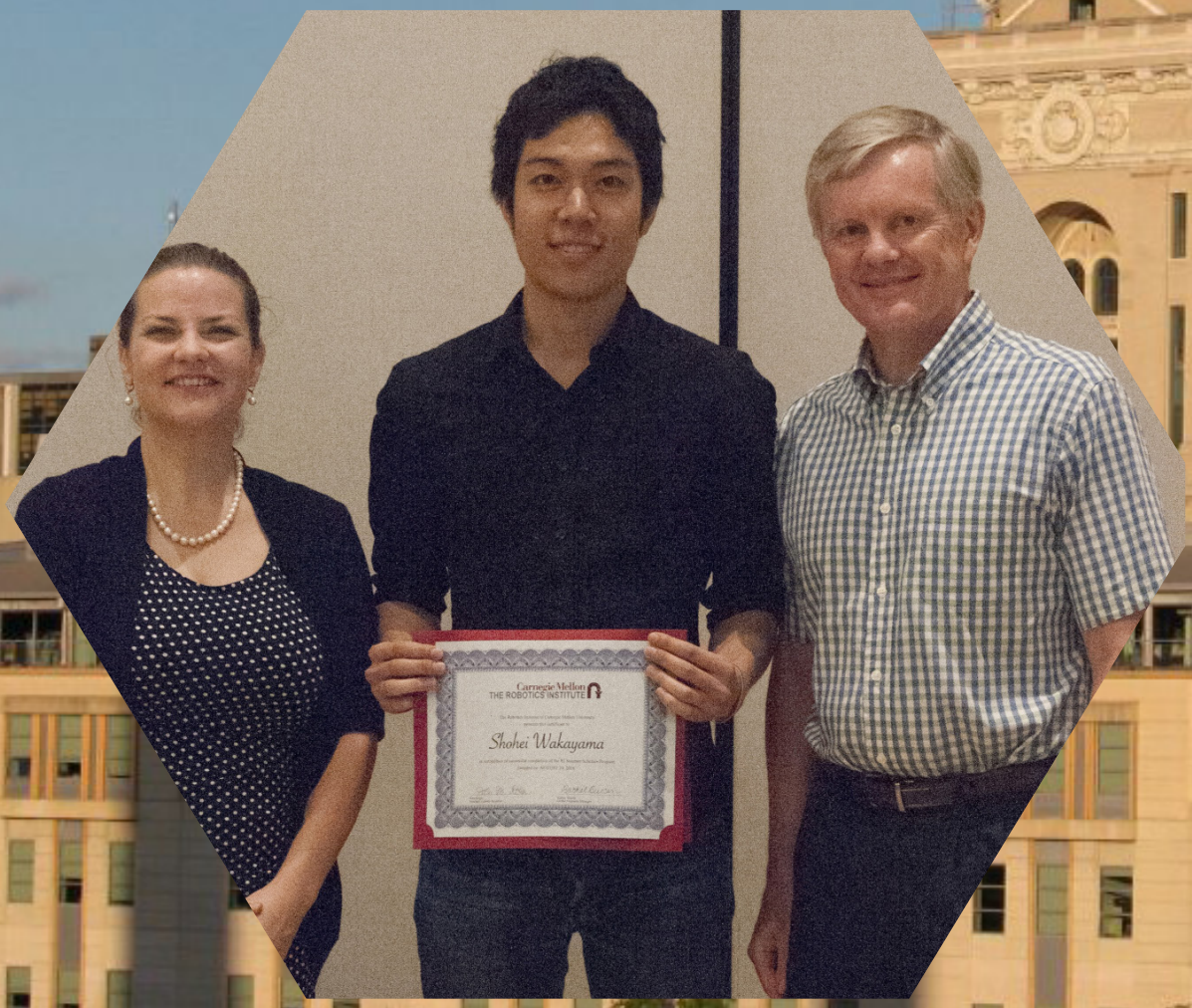
[4] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2161–2168. IEEE, 2006.

[5] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3400–3407. IEEE, 2011.

Shohei Wakayama

Kyushu University

The objective of this research is to determine the minimum size of small lunar rovers which can operate on the moon and investigate the duration of their missions by using a scaling and thermal analysis.



Scaling of Size, Mass, Power consumption and Thermal consideration for Small Lunar Rovers

Shohei Wakayama
Department of Mechanical engineering
Kyushu University
Fukuoka, Japan
Email: 1TE13849M@s.kyushu-u.ac.jp

Heather Jones
The Robotics Institute
Carnegie Mellon University
Pittsburgh, USA
Email: hjones@andrew.cmu.edu

William L."Red" Whittaker
The Robotics Institute
Carnegie Mellon University
Pittsburgh, USA
Email: ww0t@andrew.cmu.edu

***Abstract*—In space exploration the launching cost is proportional to its payload mass; therefore it is desirable to miniaturize space robots. However, there is a minimum size requirement of lunar rovers due to the harshness of the environment. The purpose of this research was to find the minimum size of lunar rovers and investigate the effect of their size and coating on their mission time. To solve these problems, we used a scaling analysis to determine the size, mass, and power consumption of small lunar rovers, then we analyzed their temperature profile using finite element analysis. It was found that the minimum mass of lunar rovers was 1.95 kg and their operation time could be controlled by their size and coating. These results reflect a basic model and will help design small lunar rovers in the near future.**

I. Introduction

Caves and pits on the moon are possible places for humans to inhabit because they are not strongly affected by space radiation and their temperatures are relatively comfortable for humans. Expense has been a major barrier in achieving the goal of lunar habitation, and payload mass is a critical component of cost. Therefore, it is necessary to make small lunar rovers. To design small lunar rovers, there are many engineering hardships and one of the most difficult problems is thermal regulation.

On the moon there is no air, and as a result there is no heat convection, so heat is only conveyed by heat conduction and radiation. Therefore, the moon's surface temperature is significantly affected by the sun angle and its temperature range is between 120 °C and −170 °C. [1] Principally, lunar missions are conducted when the sun is over the horizon, so we only need to care about the overheating. Lunar rovers use their radiators to dissipate heat from the sun and its avionics, but small lunar rovers have smaller radiators, so it is hard for them to emit heat efficiently. Furthermore, it is known that there is a minimum size requirement of small lunar rover to keep their temperature within the allowable operation range. The main purpose of this project was to find the minimum size of small lunar rovers. We also investigated the effects of size and coating on operating time.

To conduct these investigations a scaling analysis was used for an existing lunar rover to determine the size, mass, and power consumption of small lunar rovers. Their temperature profiles were simulated by finite element analysis using ANSYS simulator. The effects of size and coating could also be solved by using the simulation results.

II. Related Work

This research on thermal analysis of small lunar rovers was conducted using a scaling model and finite element analysis. This section will introduce two ideas used in the research.

A. Scaling analysis and a basic lunar rover

Highly accurate results could be obtained if each possible size of a small lunar rover was manufactured and tested. However, this procedure is time and cost consuming and therefore impractical, so a scaling analysis was instead used. This idea is based on Galileos Square-Cube Law. For example, if one were to make a rover that was half the size, the length of each dimensions would be halved, the area would be quartered, and the mass of the half-sized rover would be one-eighth. The Andy Rover[2] was used as an original rover. Figure 1 shows an image of a scaling of Andy.

Fig. 1. A scaling analysis image of Andy Rover

It is important to clarify not only the size and mass but also the power consumption of scaled small lunar rovers. Therefore, the GLXP Mobility Milestone Prize Risk Reduction Compilation[3] was used as a reference to determine the efficiency of a motor under a vacuum environment. Figure 2[3] shows the efficiency of a RSF-5B motor which is going to be used is around one-third. Therefore, two-thirds of the electrical power was transferred into heat and was taken into consideration for thermal analysis because of the first law of thermodynamics.

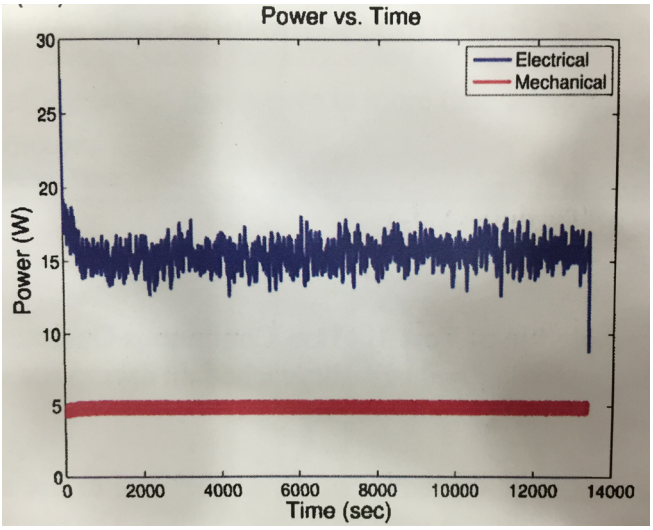


Fig. 2. Mechanical power efficiency of RSF-5B motor in vacuum

B. Thermal margin of the rover

In thermal design, it is necessary to include a thermal margin in the results because thermal design always includes several unknown factors, so the values fluctuate. To know how much of a thermal margin should be included in our models, the Margin Determination in the Design and Development of a Thermal Control System[4] was used from the Jet Propulsion Laboratory. This document introduced thermal margin in several ways, such as thermal uncertainty margin, qualification thermal margin, and protoqualification thermal margin. In our case, it was hard to calculate the specific view factors, surface environment, and joint and interface conduction, so a thermal uncertainty margin was chosen. This document mentioned that when we had a calibrated data set, we could use an 11 °C thermal margin. In a study of twenty Earth-orbiting satellite programs in the early 1970s, this margin was applied because their thermal systems was correlated to thermal-balance test data. When we do not have these correlated data set, we use a 17 °C thermal margin. As previously mentioned, manufacturing all the possible size of lunar rovers is impractical, so a 17 °C thermal margin was used.

III. Methods

A scaling analysis was used to see the size effect of a target. It would be cost and time consuming if each sized rover would be manufactured from scratch. Therefore, a scaling analysis was beneficial to determine a good estimation. A solar panel and a radiator are the target of the scaling analysis in this research. Heat from the body to the radiator can be ignored when the cool coating was used on the surface of the body and insulators were located on the junction between the body and the radiator. Therefore, only heat from the sun, the solar panel, and avionics were needed to be considered in this analysis.

A. A scaling analysis of a basic lunar rover

In this research, Andy Rover was used as a model of the scaling analysis. The dimension of its radiator and solar panel were $60cm$*$25cm$*$0.8cm$* and $100cm$*$50cm$*$1.27cm$, respectively. Therefore, theoretically the dimension of the half size radiator and solar panel were $30cm$*$12.5cm$*$0.4cm$ and $50cm$*$25cm$*$0.635cm$, respectively. The dimensions of other sized rovers could be similarly determined.

B. Solar radiation

There is no air on the moon, so the moon's surface temperature is significantly affected by the solar radiation which is based on the sun angle from the horizon. Furthermore, because there is no air, heat can only be conveyed by heat conduction and radiation. Mainly lunar rovers emit heat from their radiators to the black space, however it is difficult for small lunar rovers to do that because of smaller radiators. Hence, to emit heat effectively, white coating which has low absorptivity and high emissivity was applied on the surface of the radiator. In this research, AZW/LA-II Inorganic Low Alpha White, nonspecular thermal control coating[5] which has 0.09 absorptivity and 0.91 emissivity was used. It is known that heat flux from the sun to the moon is around 1369 W/m^2 , so the heat which is absorbed by the unit area with the coolest coating could be expressed

$$Thesolarradiation[W/m^2] = 1369 * 0.09 * sin\theta \quad (1)$$

In this equation, θ is the sun angle from the horizon.

C. Heat effect from the solar panel

In this rover, because of the mounting angle between the solar panel and radiator, the solar panel also provides heat to the radiator by heat radiation. So the temperature distribution of the solar panel and its effect for the radiator were needed to be considered.

1) Temperature distribution of a solar panel: The sun is the only heat source for lunar rovers, so they cannot operate when the sun is under the horizon unless lunar rovers contain some radioactive heat source inside them. So, a solar panel should receive the solar radiation. In order to collect as much as it can, the angle of a solar panel against the sun was adjusted for each lunar mission. In this mission, small lunar rovers were going to be launched on the Lacus Mortis[6] which is at 45 latitude on the moon and the sun rises until it is at 45 degrees from the horizon. Hence, the angle between the vertical line and the solar panel was determined as 22.5 degrees. The electrical transfer efficiency of this solar panel is around 25 %[7] and this means that 75 % of the solar radiation is wasted heat. Because of this high quality, if the solar cells are spread all over the solar panel, the wasted heat cannot emit effectively and the temperature of the panel would be very high. This high temperature might cause the solar panel to breakdown or affect the temperature of its radiator. Therefore, in order to prevent overheating of the solar panel, room was made on the solar panel and this was coated with the cool coating. Also, a multi-layer insulator which has 0.025 emissivity[8]

was attached to the back of the solar panel and this prevented the heat transferring from the hot ground. Regarding the body of the panel, although the aluminum honeycomb which has a low thermal conductivity was used, this effect could be ignored because the solar panel was thin. Considering these ideas, the solar panel's temperature distribution at each sun angle was analyzed to give a temperature profile shown in figure 3.

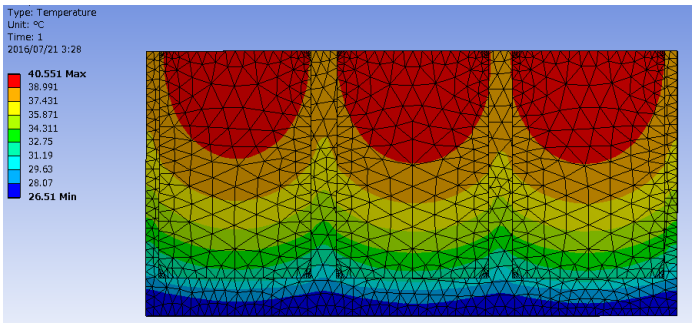


Fig. 3. Temperature diffusion of the solar panel:The sun angle is at 15°

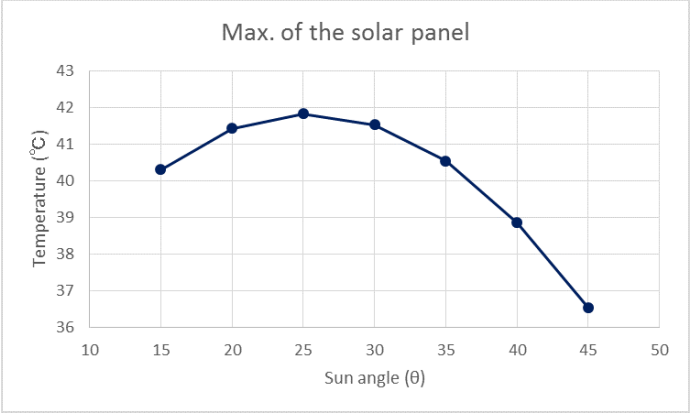


Fig. 4. Temperature profile of the solar panel

2) Heat from the solar panel to the radiator: Heat radiation between the surfaces is dependent on the configuration of the surfaces relative to each other and is independent of the surface properties and temperature. The effects of the orientation on heat radiation between the surfaces is called the view factor. As mentioned above, the angle between the solar panel and the radiator was 112.5 degrees and this view factor was estimated to be around 0.15 by referring a catalog of radiation heat transfer configuration factors. [9] Given these, the heat from the solar panel to the radiator was calculated to be

$$Sph[W/m^2] = Vf * \sigma * \{(Spt)^4 - (Rt)^4\} \quad (2)$$

In the above expression, Sph, Vf, Spt, and Rt stand for "Solar Panel Heat", "View Factor", "Solar Panel Temperature", and "Radiation Temperature", respectively. Also, σ is the Stephan-Boltzmann constant.

D. Avionics heat generation

The solar radiation has the most significant effect for the temperature of the radiator, so this heat is needed to be decreased first. In addition, the internal heat also needs to be considered when we make small lunar rovers because they have smaller radiators. This means the effect of avionics heat for the radiator is higher than the one when the larger rover has.

1) A scaling analysis of power consumption: The diagram below shows the electrical power of major electrical components of a 4.0 kg rover.

TABLE I
ELECTRICAL POWER DISTRIBUTION OF 4 KG LUNAR ROVER

Components	Power(W)
CPU	2
FPGA	1.5
Flash memory	0.25
Dynamic ram	0.5
Camera	0.5
IMU	0.1
Batteries	0.86
Battery management	0.3
Power management	0.5
Motors	9.6
Radio	1
Total	17.11

In this diagram, electrical power consumption of motors is the only variable for different size of lunar rovers. So, it was necessary to find out how to calculate the motor's power. Motors provide a mechanical power for movement and the relationship between the a mechanical power and rover's mass was acquired by Bekker's Derived Terramechanic Model [10].

$$F[N] = W * C_r \quad (3)$$

In this equation, F is the force needed to move a lunar rover, W stands for the weight and C_r is the resistant efficiency of the ground. The mechanical power is expressed shown below.

$$P[W] = E/t = (F * d)/t = F * v \quad (4)$$

In this expression, E is the work, d is the distance, t is the time, and v is the velocity. By combining the equation 3 and 4, the mechanical power can be expressed shown below.

$$P[W] = W * C_r * v = m * g * C_r * v \quad (5)$$

In this expression, m is the mass of a lunar rover and g is the gravitational acceleration. We had already known concrete except (We need to modify this sentence) the resistant efficiency of the ground, so by substituting the known values into the above equation, the resistance efficiency of the lunar ground was determined as *1.521*. Once we found the value, we can solve the mechanical power of other size rovers by substituting the mass into the equation 5. Although so far we assumed that the rovers move constantly, there is a time gap because of the communication from earth and a time lag because of large obstacles, so lunar rovers do not always move.

In this research, the duty cycle was set as 60 %. After we found the mechanical power of motors, if we used the efficiency of motors, we can find the heat from the motors like below because heat from the motor is 1.5 times mechanical power.

$$Heat[W] = 0.6 * P * (3/2) = 0.9 * P \tag{6}$$

E. ANSYS simulation

From the procedures described above, the external and internal heat related to the radiator were calculated. In this research, ANSYS simulator was used for determining the minimum size of lunar rovers and finding the effects of size and coating for operating time.

1) The minimum size of lunar rover: In this mission, small lunar rovers were going to be launched at 45 latitude, so when the sun is at 45 degrees, the temperature of the radiator is the highest. Therefore, the minimum sized rover was defined as the rover which radiators temperature is not over critical temperature, and the critical temperature is 45 °C from charging batteries.

2) The effects of size and coating for operating time: To determine the minimum size of lunar rovers, we just focused on the highest temperature of the radiator. However, it was also interesting to know how the size and coating could affect the operation time. Therefore, we simulated the temperature profiles for each rover between the sun angle at 15 degrees to 45 degrees and from 45 degrees to the sunset and saw when the radiator's temperature was lower than the critical minimum temperature depending on the size. In addition, regarding 3.0, 3.5 and 4.0 kg rovers, we also changed the absorptivity of the coating on the radiator and investigate how the coating change affected the operation time.

IV. Results

A. The result of the minimum size of lunar rover

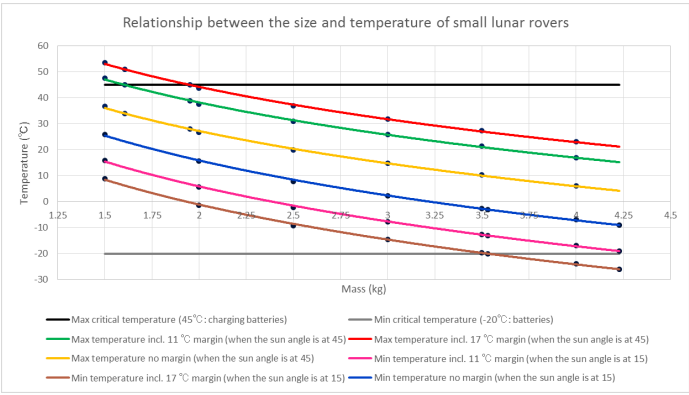


Fig. 5. The minimum size of small lunar rovers

Figure 5 shows the relationship between the maximum temperature and the mass of lunar rovers. The horizontal and vertical axes express the mass and the maximum temperature, respectively. The green line shows the maximum temperature including an 11 °C margin and the red line shows the maximum temperature including a 17 °C margin. Finally, the black and gray lines show the maximum and minimum critical temperatures of avionics from charging batteries. Therefore, where the black line and red is crossed shows the minimum size of lunar rovers when a 17 °C thermal margin was applied and the value is 1.95 kg, similarly where the black and green lines are crossed shows the minimum size of lunar rovers when an 11°C thermal margins was applied and the value is 1.6 kg.

B. The size and coating effect for the operation time

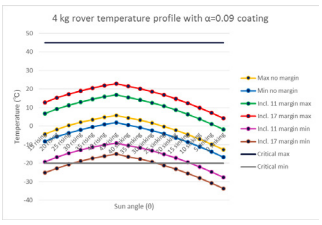


Fig. 6. 4 kg profile

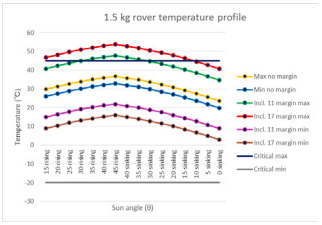


Fig. 7. 1.5 kg profile

Figures 6 and 7 show the size effect for the temperature profile. As it can be seen, smaller rovers have a higher temperature profile than the larger one. Regarding the maximum temperature, the maximum temperature of 4.0 kg is higher than the 1.5 kg one by around 30 centigrade.

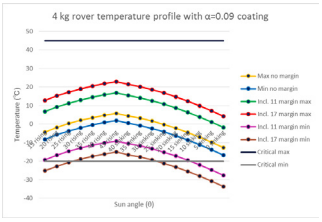


Fig. 8. α=0.09 profile

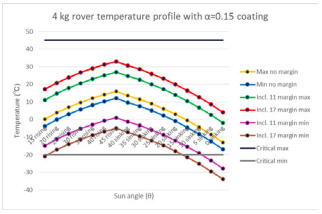


Fig. 9. α=0.15 profile

Next, these temperature profile shows that by changing the surface coating on the radiator, the operation time could be extended. When 0.09 absorptivity coating was used on the surface, a 4.0 kg rover can work until the sun angle is at 28 degrees, however when 0.15 absorptivity coating was used on the surface, the rover could work until the sun angle is at 17.5 degrees.

V. Conclusion

From this research, the minimum size of lunar rovers was determined to be 1.95 kg from only thermal perspectives. Regarding the effect of the size, we found that the temperature profile of smaller rovers was higher than larger one. We can conclude that this is because smaller rovers have smaller thermal inertia, so smaller rovers are affected by heat easily. Finally, as for the coating effect for operating time, the operation time could be extended by using a warmer coating on the radiator. However, we also need to be careful when we decide the coating because the coating affects not only the operation time but also the maximum temperature.

VI. Future work

In this research, heat radiation from the lunar surface to the body and heat from the body to the radiator were ignored. So, we will need to incorporate these heat in order to get a better estimate of thermal constraints on rover size. It is also necessary to manufacture different sizes of small lunar rovers, set up the Moons environment and conduct thermal experiments to calibrate this analysis. Finally, we need to integrate thermal results with mechanical requirement to make the flight prototype.

Acknowledgements

Special thanks to the Planetary Robotics Laboratory and Dr. Whittaker for admitting me this lab, and Heather Jones, Daniel Arnett, and Oleg Sapunkov for giving me educational ideas, and the RISS program for this precious opportunity.

References

[1] Hurley D, et al. An analytic function of lunar surface temperature for exospheric modeling, *2014*.

[2] Daniel Arnett. Any's physical information, *2014*, URL = http://lunar.cs.cmu.edu/

[3] William L."Red" Whitakker. GLXP Mobility Milestone Prize Risk Reduction Compilation, *2014*.

[4] Daniel P. Thunnissen, Glenn T. Tsuyuki. Margin Determination in the Design and Development of a Thermal Control System, *2004*.

[5] AZ Technology. Spacecraft Thermal Control and Conductive Paints/Coatings and Services Catalog, *2008*.

[6] Harold Hill. A Portfolio of Lunar Drawings, *2003*.

[7] Alta Devices. Single Solar Cell, *2014*.

[8] M.M. Finckenor, D. Dooling Multilayer Insulation Material Guidelines, *1999*.

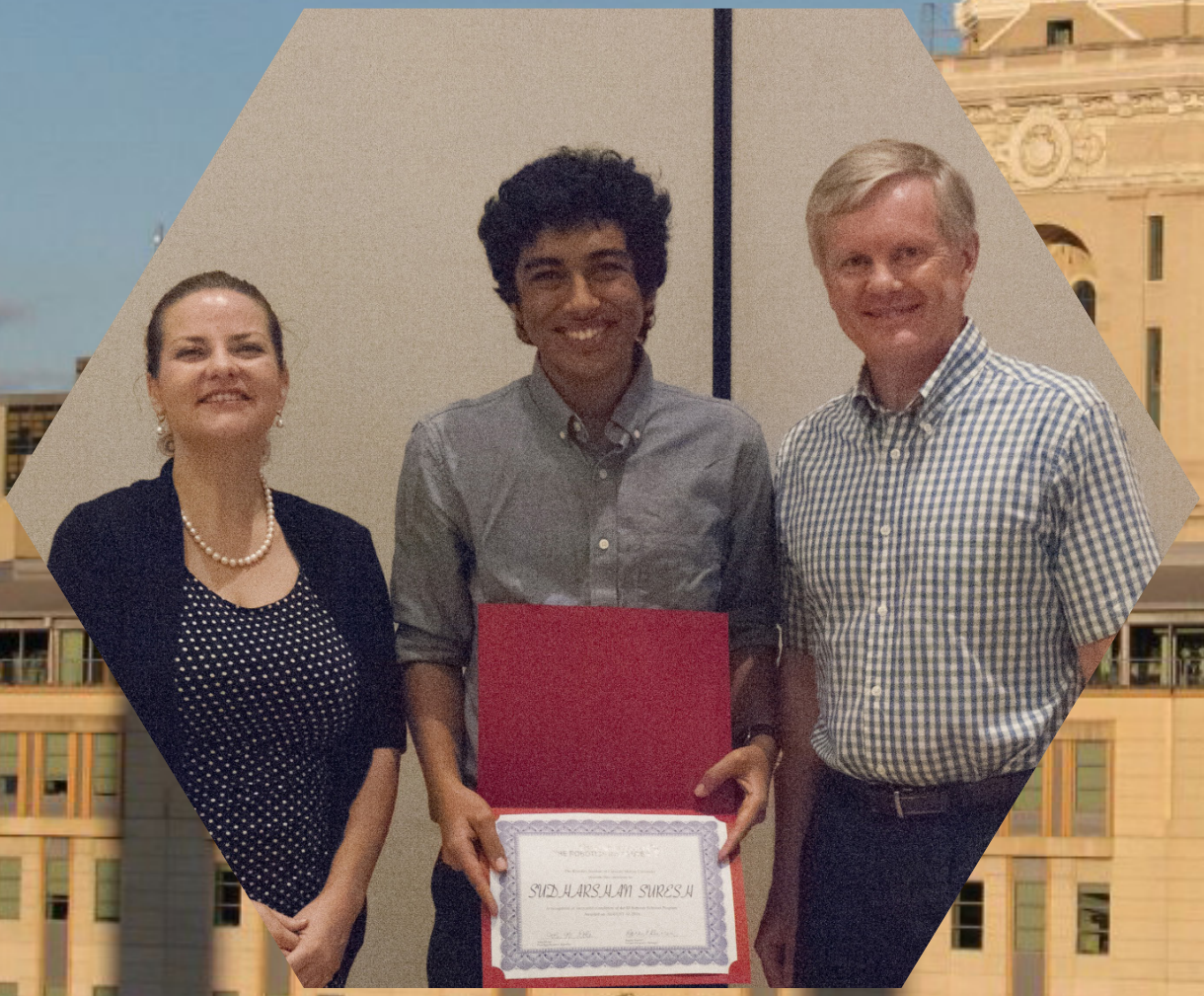
[9] John R. Howell. A Catalog of Radiation Heat Transfer Configuration Factors, *1982*.

[10] M. G. Bekker. Theory of land locomotion: the mechanics of vehicle mobility, *1956*.

Sudharshan Suresh

NIT, Trichy

Knowledge of the full kinematic state of a rover is critical to motion control and exploration, especially on the rugged terrain of a planetary surface.



https://in.linkedin.com/in/sudharshansuresh

https://github.com/suddhu

Optical Kinematic State Estimation of Planetary Rovers using Downward-Facing Monocular Fisheye Camera

Sudharshan Suresh
National Institute of Technology
Tiruchirappalli, India
Email: sudharshan.nitt@gmail.com

Eugene Fang
Robotics Institute
Carnegie Mellon University
Pittsburgh, USA
Email: fangeugene@cmu.edu

William L. "Red" Whittaker
Robotics Institute
Carnegie Mellon University
Pittsburgh, USA
Email: red@cmu.edu

***Abstract*—Knowledge of the kinematic state of rovers is critical to navigation, path reconstruction and exploration, especially on rugged terrain like planetary surfaces. Existing methods employ many encoders, potentiometers and hall sensors. These add components and wiring to moving parts. The components are susceptible to mechanical and electronic failures, add mass, and some require thermal regulation. In addition, the sensor wires are susceptible to bending, flexing and wear. Where miniaturization counts, the limitations on mass, size and power encourage elimination of sensors wherever possible. This paper presents a method to estimate the kinematic state of rovers using only a downward-facing fisheye camera. This novel approach implements a vision algorithm to obtain kinematic state information in planetary rovers. The two additional benefits of the technique are (1) redundancy to proprioceptive measurements, (2) means for perceptive visual odometry. The method uses a single camera to estimate 10 degrees-of-freedom (associated with steering, driving and suspension) on the *AutoKrawler*, a rover test platform for planetary exploration. Motions are estimated by self-perception - combining fiducial marker tracking, optical flow techniques and the kinematic constraints of rover mechanics. Experimental results, obtained from the rover operating in an environment analogous to the lunar surface, are presented. The results obtained are compared with ground truth data to validate the approach.**

I. Introduction

Planetary rovers have locomotive capabilities designed for traversing highly uneven terrain. A rover's axles or suspension act as passive hinges, allowing the wheels to make contact with uneven surfaces. Steering is critical to negotiating terrain. Understanding the real-time kinematic state of a rover provides timely information about abnormal motion and supports operational decisions. Such knowledge proves useful for rover guidance, navigation and safeguard. This information is also fundamental to modeling rover odometry. Proprioceptive sensing has for long been the modus operandi to obtain the aforementioned data, providing accurate estimates of position, orientation and motion of links and joints.

Space technology typically employs a large number of sensors to obtain kinematic state information. For example, the states of the Mars Rover prototypes are determined by six wheel encoders, three accelerometers, joint potentiometers for bogey angles and a sun sensor [1]. The problems with such a system are manifold, and of significant interest in the domain of space robotics. Optical encoders have rotating mechanical disks that reduce system reliability. Atmospheric dust, similar to that of Mars, may hinder optical wheel encoders [2]. A pressing concern with encoders is the numerous electrical conductors required. The Sojourner rover comprised of ten motors, each with an optical encoder. The motors required two conductors each to drive, but the encoders required six wires each to moving parts- translating to 80 conductors fed from the body of the rover [3]. In an articulated rover, special care must be taken in routing cables to prevent wear of wire harnesses. In addition to flexing and bending with suspension, when exposed to the frigid environment, the wires can stiffen and break [3]. Sensing elements with electronic components may require thermal protection and this additional insulation can compromise total system mass. 50% of a typical rovers mass distribution is generally dedicated to its subsystems [4], and the myriad sensors reduce the viable scientific payload limit.

Figure 1: The AutoKrawler traversing uneven terrain in a lunar analogue site.

In the past, vision algorithms were used sparingly on planetary missions, primarily due to the on-board computational constraints. A major leap in this direction was the NASA/JPL Mars Exploration Rover (MER) mission, where algorithms performed tasks such as visual odometry, stereo vision and feature tracking [5]. All processing was performed off-board, either on Earth or the lander. Subsequent FPGA implementation of the same algorithms have shown performance improvements of three orders of magnitude and are now utilized on-board large rovers [6]. These encouraging results point to full-fledged vision systems in future missions with minimum latency, to perform rover odometry and safeguarding, even on small, simple rovers.

This research conceives and demonstrates a novel method of optical kinematic state estimation of planetary rovers using a downward-facing monocular fisheye camera. In this paper, 10 degrees-of-freedom of a planetary rover are estimated. Fiducial marker tracking is used to obtain pixels of interest from the camera image. Using a spherical camera model, 2-D image coordinates are projected to 3-D points on a unit sphere around the camera's focal point. These are subsequently mapped to real-world coordinates using planes defined by the rover's kinematic constraints. 8 defined positions are tracked and a geometric approach is developed to determine the axle roll and steering angles. Odometry is performed for all four wheels by using an optical flow algorithm directly on the treads, as opposed to traditional egomotion estimation. The methodology is exhibited and evaluated by application to the AutoKrawler (see Figure 1), a highly-versatile, double-Ackermann rover test platform for planetary exploration.

The aforementioned system was evaluated on datasets generated from field experiments conducted in a lunar analogue site. The rover was tele-operated in conditions that mimic risk-prone traversals, where knowledge of kinematic state can aid control decisions. Results show close agreement between data from our technique and ground truth data from proprioceptive sensors onboard.

II. Related Work

The kinematic study of complex rocker-bogey mobility systems have been a prominent research problem. [7] and [8] describe a body of work surrounding the Rocky-7 Mars Rover that estimates position, velocities, contact angles and orientation of the robot. Systems inputs are generally accelerometer, gyroscope and encoder data, without involving visual observation. Lamon and Seigwart demonstrate the benefit of kinematic knowledge by performing 3-D odometry, showing a significant improvement in results on rough terrain. A controller that minimizes slip based on state information is also developed [9]. Visual self-perception (or the strategy of observing oneself), has been used to generate kinematic models of robotic manipulators [10]. Its joint configuration however, is determined solely by its own actuation and not by the environment.

Optical flow methods using pyramidal Lucas-Kanade [11] have been previously explored using a downward-facing fisheye by Seegmiller [12] to perform robust visual odometry. The algorithms employ RANSAC outlier rejection, which has been incorporated in this body of work. Fang's work also highlights the importance of a minimally obstructing, robust camera support structure for the selected configuration [13]. Fiducial markers have been used, albeit sparingly, in planetary robotics for accurate instrument positioning and as calibration targets [14]. Scaramuzza obtained a generalized omnidirectional camera model for fisheye and catadioptric systems, that has been used here [15].

III. Kinematic Model

A. Rover Overview

The AutoKrawler is a four-wheel, double-Ackermann steered rover specialized to traverse adverse terrain. A passive body-axle suspension system allows the rover to maintain compliance with uneven, rugged terrain. For an illustration of the Cartesian coordinate conventions and degrees-of-freedom, refer to Figure 2. Each articulated axle of the dual-axle configuration can perform roll about the y-axis and translation about the z-axis. The steering swivel joints can rotate in the range of [-30°, 30°] about the axle frame's z-axis.

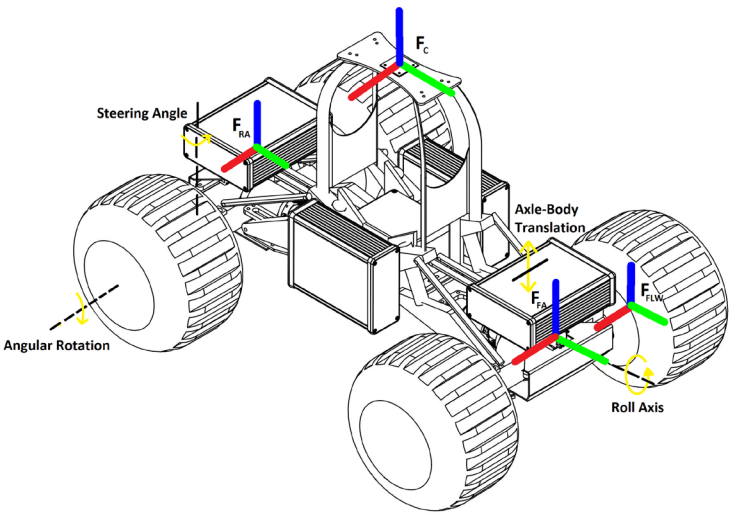


Figure 2: Depiction of (i) The coordinate frames of the camera, axle and wheels, illustrated with XYZ (**RGB**) axes. (ii) The permitted rotational and translational motions of the robot about their respective axes.

In conceptualizing the state estimation, several reasonable assumptions on rover kinematics are made:

- Angular movement of wheels on a common axle are assumed to be identical - giving each axle a single steering angle.
- The mechanical connection between the axle and body of the AutoKrawler comprises of a plurality of links and joints. However, for all practical purposes, the axles are

assumed to only exhibit one rotational degree-of-freedom (about y-axis).

- It is assumed that there is no relative translation in the x-direction between the axle and the body frame. This allows us to approximate the motion of each axle to be on a vertical plane perpendicular to the y-axis.
- The wheels of the rover are modelled as cylinders (of radius $r = 11.5cm$) that rotate with the wheel frame, centered at its origin.

B. Coordinate Frames and Variables

The translational and rotational displacements of rover joints and linkages are expressed via homogeneous transformation matrices. Examples of coordinate frame locations are provided in Figure 2. With transformation matrices, a kinematic chain of coordinate frames can be represented. Below, an equation is used to transform the center of the wheel (P) from the swivel joint frame to the camera frame. To do so, the 4×1 vector $[x \quad y \quad z \quad 1]^\intercal$ is successively pre-multiplied with ${}^{A}_{S}T$ (transform from swivel joint to axle frame) and ${}^{C}_{A}T$ (transform from axle to camera frame):

$$ {}^{C}P = {}^{C}_{A}T \, {}^{A}_{S}T P \tag{1}$$

In (1) ${}^{C}P$ represents the real-world coordinates of a wheel center, with the reference frame being the camera focal point. The degrees-of-freedom estimated are 10 in total:

(i) Two rotational (Axle roll angles) - ψ_F, ψ_R
(ii) Two translational (Axle-Body vertical distances) - d_F, d_R
(iii) Two steering - λ_F, λ_R
(iv) Four rotational - $\theta_{FL}, \theta_{FR}, \theta_{RL}, \theta_{RR}$

IV. System Description

A. Camera Model

The research problem necessitates that all motions (axle roll, steering and wheel rotation) of the rover be detected with a single camera. The best way to achieve this is with a downward-facing omnidirectional camera. The choice of a fisheye lens over a catadioptric lens is inconsequential, as a unified camera model [15] is applicable to both. The body frame is fit with a mount to house the downward-facing fisheye camera. The mount minimally occludes field-of-view, with four orthogonal plates along planes that intersect the camera center (as seen in Figure 3). Color images are acquired with a resolution of 640×480 at 30 frames per second. The chosen resolution provides adequate pixels to work with, while at the same time remaining computationally feasible.

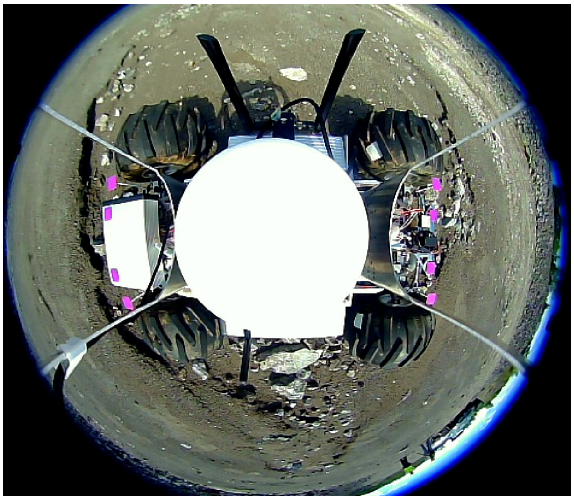


Figure 3: Fisheye lens's field-of-view with minimally occluding camera mount. The four thin plates in the image constitute the mount structure.

The MATLAB toolbox developed by Scaramuzza et al. [16] is used to calibrate the camera with a planar checkerboard pattern. The model uses affine transforms to handle misalignment between the camera's optic center and the focus point of the lens. A 4^{th} order polynomial is utilized to account for the camera's radial distortion. Any real-world point can be accurately projected to a point on a unit sphere, with the camera's focal point as its center.

B. Fiducial Markers

Passive, mono-colored square markers are used, and the paper does not attempt to validate tracking robustness. In our application, magenta is chosen due to its hue value being distinct to that of the rover's surrounding. Eight markers are placed on the rover - two collinear pairs on the front and rear axle plates, and two pairs on the Ackermann tie rods.

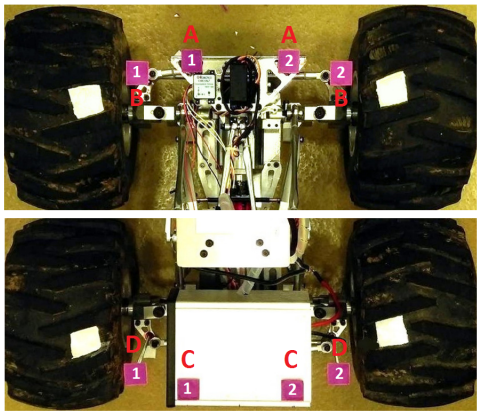


Figure 4: Mono-color markers visible on the front and rear axles. Pairs A and C track front and rear axle roll respectively, while pairs B and D track front and rear steering angles. Also seen are the white correction markers present on the wheels.

The process pipeline first performs color-based image segmentation in the HSV (Hue, Saturation, Value) color space. After noise-removal by erosion and dilation, pixel positions of the contour centers are obtained. These pixel positions are transformed to world coordinates based on the geometric constraints, as described in subsections V-A and V-B. Each wheel tread consists of a white marker, referred to henceforth as a correction marker. It is tracked via thresholding to correct wheel angular rotation, as described in subsection V-C.

C. System Design

A compositional view of the required variables is taken and the resulting system is simple and modular (see Figure 5). Axle roll, axle-body vertical distance and steering angles (ψ_F, ψ_R, $d_F, d_R, \lambda_F, \lambda_R$) are computed at once, requiring fisheye image data. Wheel angular rotations ($\theta_{FL}, \theta_{FR}, \theta_{RL}, \theta_{RR}$) require all the aforementioned variables *and* fisheye image data. This is because optical flow is performed only on dynamic observation windows, as described in V-C. If these variables are not present, due to momentary loss of marker tracking, flow operations are performed using the prior state of the rover. In the absence of computational overhead, the system operates at the frame rate of the camera. All vision tasks are optimized to defined regions-of-interest based on kinematic constraints, instead of being performed on the entire image.

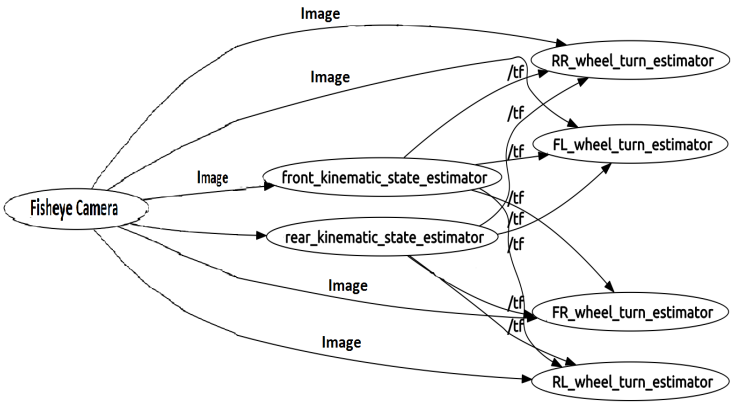


Figure 5: ROS Nodes and Topics and their relationship in the system. *tf* indicates transform matrices of the rover.

V. Estimation Methods

The following subsections describe how the degrees-of-freedom are obtained. All the estimation methods use **world coordinates** as inputs, and not pixel positions.

A. Estimation of Axle Configuration

The kinematic state estimation begins by calculating axle roll (ψ_F, ψ_R). By making the assumption that the axle has only one rotational and one translational degree-of-freedom (refer to Section III-A), markers are constrained to a plane. This is defined by:

$$y = \begin{cases} +c, & \text{front axle} \\ -c, & \text{rear axle} \end{cases} \tag{2}$$

where $c \in \mathbb{R}_{>0}$.

As explained in section IV-A, the omnidirectional camera projects a pixel to a point on a unit sphere - (a, b, c). The world coordinates of the marker centers are computed by finding the intersection point between the ray through the points $\{(0,0,0), (a,b,c)\}$ with the plane defined in equation 2. The two marker coordinates are subtracted to obtain a vector that represents axle orientation, $\vec{v} = x\hat{i} + y\hat{j} + z\hat{k}$. Finally, ψ is computed as the directed angle between $\vec{v}$ and vector $\vec{u} = \hat{i}$ (unit vector on x-axis). The midpoint of the two marker coordinates is the rotational center of the axle, and its vertical distance from the camera frame is the axle-body distance (d).

B. Estimation of Steering Angle

In steering angle estimation, a plane that rotates about the y-axis (similar to axle rotation) is considered. In the absence of axle excursion, the normal to the plane is $[0 \quad 0 \quad 1]^\intercal$. However, the axle roll must be accounted for by computing a new normal:

$$\vec{n'} = R_y(\theta) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \sin\theta \\ 0 \\ \cos\theta \end{bmatrix} \tag{3}$$

where θ is the angle of roll and $R_y(\theta)$ is an elemental rotation about the y-axis. The position vector of the axle's rotational center is represented by $\vec{r_o}$.

Thus, the plane considered is defined as:

$$\vec{n'} \cdot (\vec{r} - \vec{r_o}) = 0 \tag{4}$$

Similar to section V-A, the ray-plane intersection is computed to get centers of the steering markers in world coordinates. Euclidean distance between marker pairs is calculated and the given ratio is defined :

$$x = \begin{cases} \frac{d(\mathbf{B[1]},\mathbf{A[1]})}{d(\mathbf{B[2]},\mathbf{A[2]})}, & \text{front axle} \\ \\ \frac{d(\mathbf{D[1]},\mathbf{C[1]})}{d(\mathbf{D[2]},\mathbf{C[2]})}, & \text{rear axle} \end{cases} \tag{5}$$

where marker labels $(A, B, C, D; 1, 2)$ are assigned as per Figure 4. The geometric arrangement of the tie rod ensures that the ratio is ≈ 1 when the wheels aren't steered. The ratio is *greater* than 1 if steered in one direction, and *less* than 1 is steered in the other. To obtain a relation between the ratio x and wheel alignment, manual calibration is performed. x vs. steering angles is tabulated, where steering angle is set to values in the range of [-30°, 30°]. The calibration equation is a second-order polynomial $f(x)$ fit to the data, such that $\lambda = f(x)$ (see Figure 6).

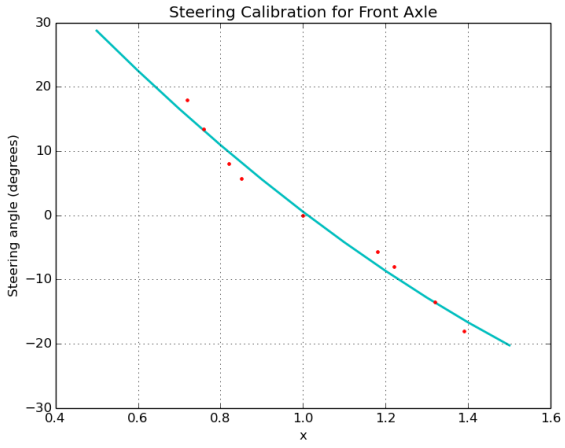


Figure 6: 2nd degree polynomial regression for steering angle vs. distance ratio. The red points indicate tabulated readings and the cyan curve is the polynomial fit.

C. Estimation of Wheel Angular Rotation

Optical flow provides vital knowledge of the arrangement of features in an image, and change in this arrangement over a sequence of frames. Egomotion estimates can be obtained from analysis of image sequences, as performed in [2], [12] and [13]. It has computational advantages over structure from motion algorithms, making it ideal to run online.

Lucas-Kanade optical flow is chosen, an established algorithm for correlated feature tracking [11]. Sparse feature sets are identified by the Shi-Tomasi corner detection algorithm [17]. Pyramidal implementation of Lucas-Kanade [18] relaxes the small displacement constraints of the algorithm and works with coarse variations. In our image sequences, flow is computed in the forward and backward direction with respect to time. All feature vectors that are inconsistent between frames are discarded.

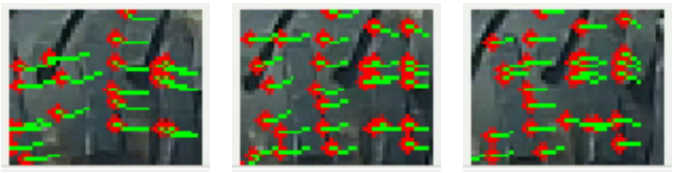


Figure 7: Lucas-Kanade feature tracking over three consecutive frames. The red dots indicate feature positions and the green trails show the direction of motion. There is a shift in features towards the left, and features may disappear and appear between frames.

The technique is applied on an unrectified fisheye image sequence to give 2-D displacement vectors. As highlighted in III-A, the rover's wheels are modelled as finite cylinders. It is defined by:

(i) A fixed radius of $r = 11.5cm$
(ii) Cylinder axis vector $\vec{v} = \vec{a} - \vec{b}$, where $\vec{a}$ and $\vec{b}$ are the base center position vectors in world coordinates. These points are obtained with knowledge of ψ and λ, along with the position of the swivel joints in the axle frame and the wheel centers in the swivel joint frame.

Similar to ray-plane intersection computed in V-A, the ray-cylinder intersection point is computed. The intersection point represents the world coordinate position of a point on the wheel's surface.

Feature search is performed in a dynamic observation window of fixed area, so as to only focus on the wheel tread (as seen in Figure 7). The position $w_{60} = [0 \quad \pm r\cos 60 \quad r\sin 60]$ with respect to the wheel coordinate frame is mapped back to a 2-D pixel coordinate and treated as the window center. This ensures that the flow is invariant to axle roll/translation and steering angle. Between frames, the arc-length travelled in world measurements is computed. To get angular displacement $\alpha(i)$ of a feature i, given its position on the circle's edge in consecutive frames (p_1 and p_2):

$$\alpha(i) = \cos^{-1}\left(\frac{2r^2 - |p_1 - p_2|^2}{2r^2}\right) \tag{6}$$

α is a $1 \times n$ size matrix, where n is the number of features tracked. Rather than computing the mean angular displacement directly, Random sample consensus (RANSAC) is used to reject outliers. These outliers may appear as a result of self-shadowing or occlusion. The RANSAC algorithm begins by selecting random angular displacements from the matrix α. The randomly selected value is compared with rest of the set, and all those that are approximately equal (within a tolerance range t) are treated as inliers for the selected model. After a defined number of iterations, the model with the maximum number of inliers is considered and its outliers are discarded. The mean of this set is taken to be the final angular displacement between the frames. The angle of rotation is incremented every frame, and the derivative of angular displacement with respect to time gives wheel angular velocity.

The demerit of a sparse feature set is the loss of tracking due to a lack of *interesting* features, or if there is image blur. When incremental angular rotation is observed, these inaccuracies accumulate. This can lead to θ drifting far from its ground truth over long traverses. Correction markers on the wheels (visible in Figure 4) clearly indicate each complete rotation. They are used to prevent this drift in angular rotation. The marker is segmented out by thresholding, and once every rotation, $theta$ is corrected:

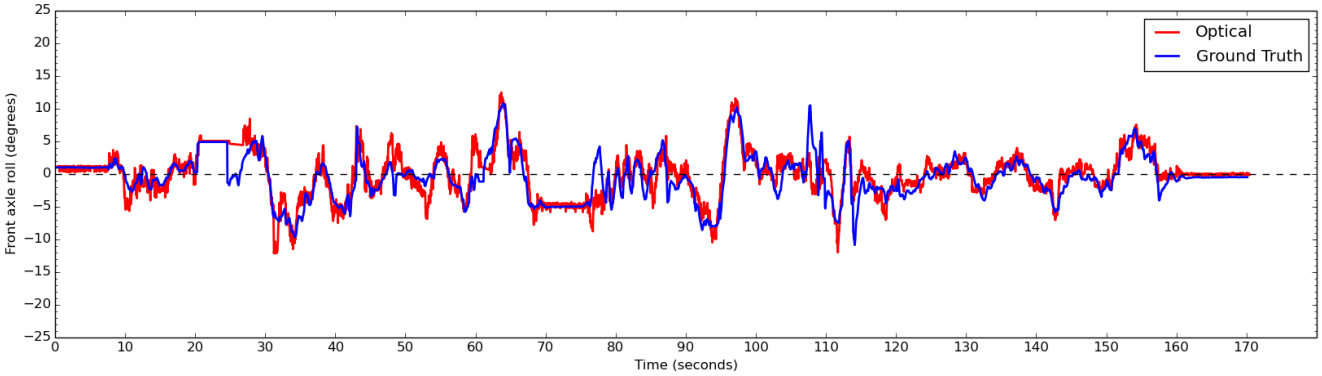
$$\begin{aligned} m &= (\theta - c_o) \ div \ 2\pi \\ \theta_{corrected} &= 2m\pi + c_o \end{aligned} \tag{7}$$

where $m \in \mathbb{Z}$ and c_o is the value of θ when the first correction marker was encountered.

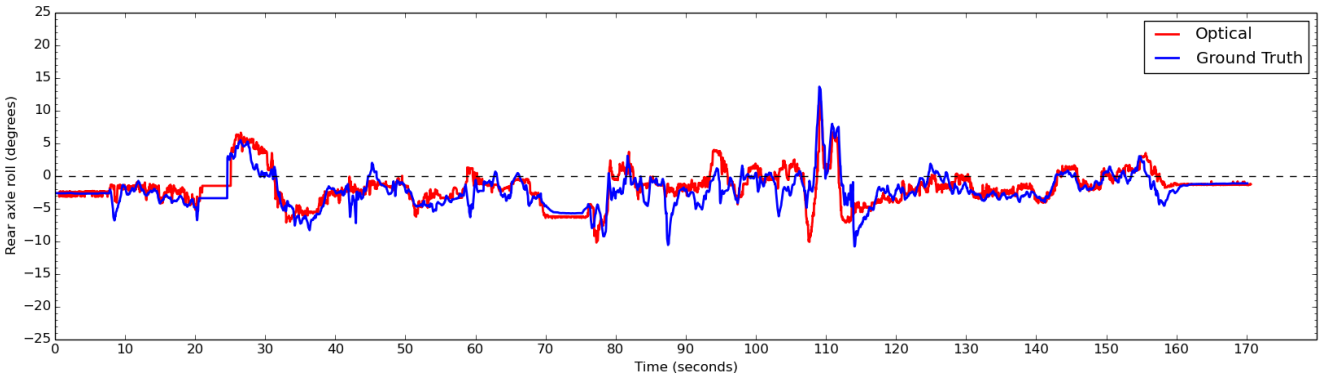
VI. Results

The performance of the method was evaluated on datasets acquired in an outdoor field experiment (see Figure 8). The rover was tele-operated at a location where the terrain characteristics and features served as an excellent analogue to lunar conditions.

Figure 8: Optical kinematic state estimation field test.

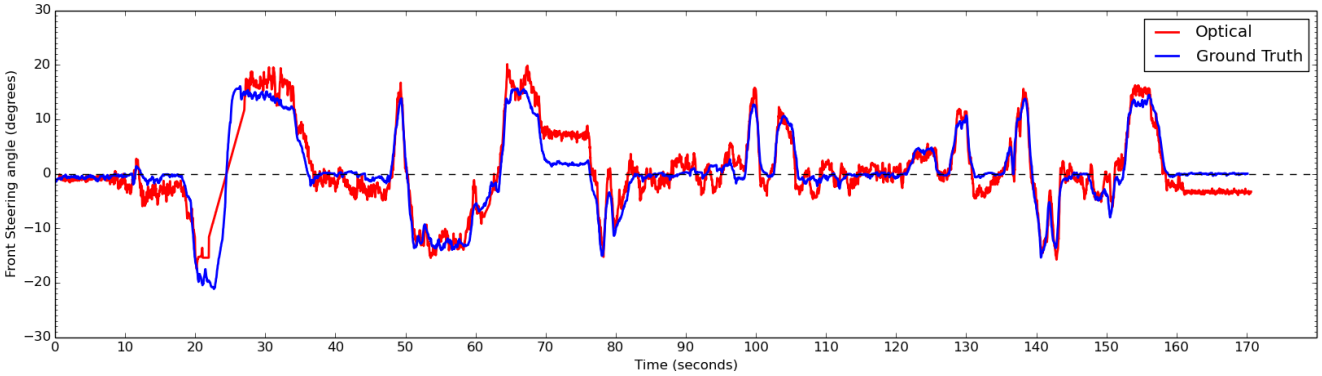


(a) Front Axle

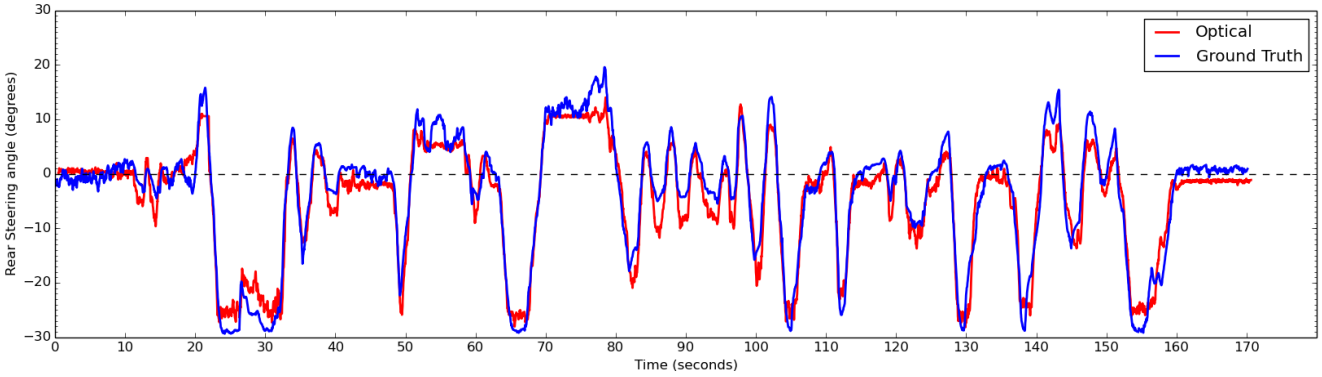


(b) Rear Axle

Figure 9: Estimated axle roll vs. Ground truth IMU data

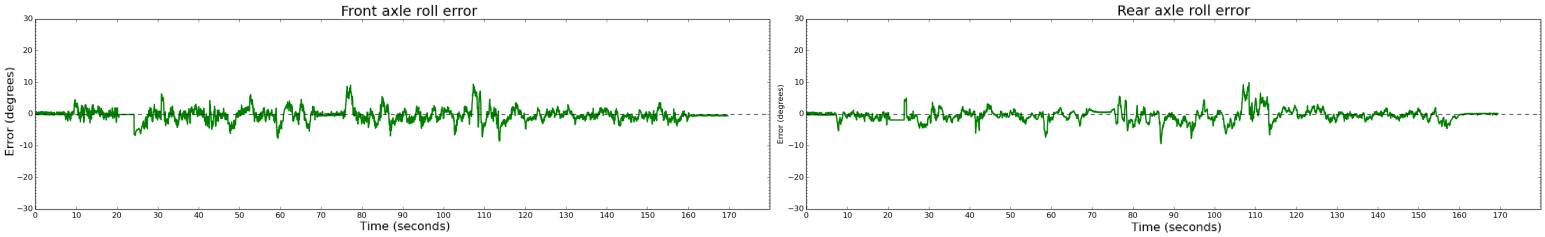


(a) Front Axle



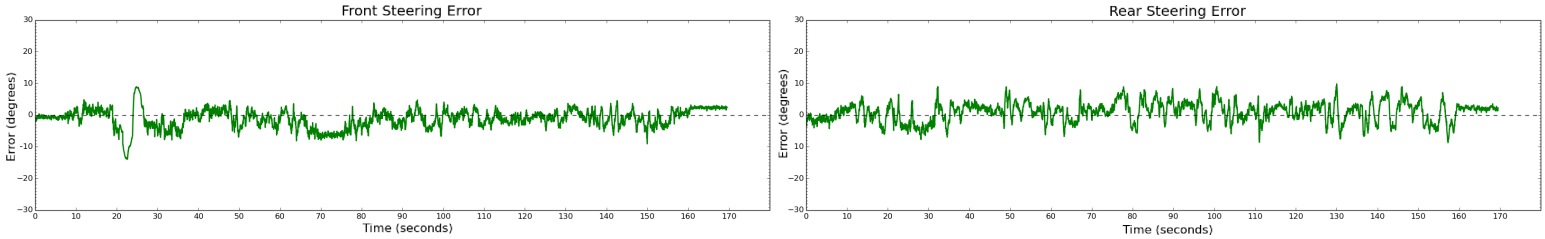
(b) Rear Axle

Figure 10: Estimated steering angle vs. Ground truth potentiometer data



(a) Front Axle **(b)** Rear Axle

Figure 11: Estimated axle roll error with respect to Ground truth



(a) Front Axle **(b)** Rear Axle

Figure 12: Estimated axle roll error with respect to Ground truth

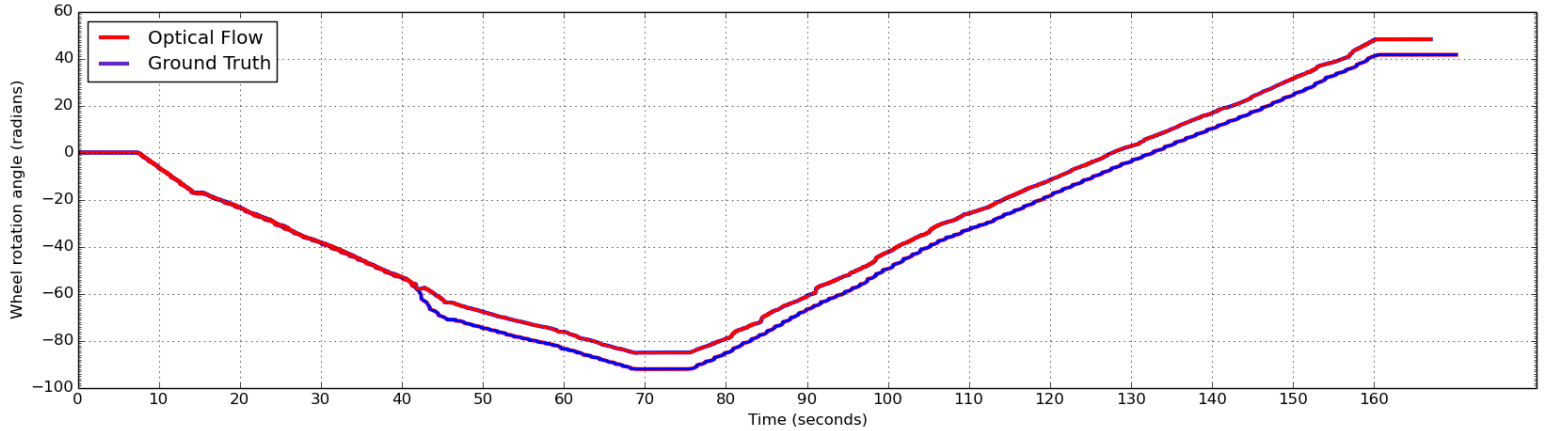


Figure 13: Estimated wheel rotation angle vs. Ground truth motor encoder data

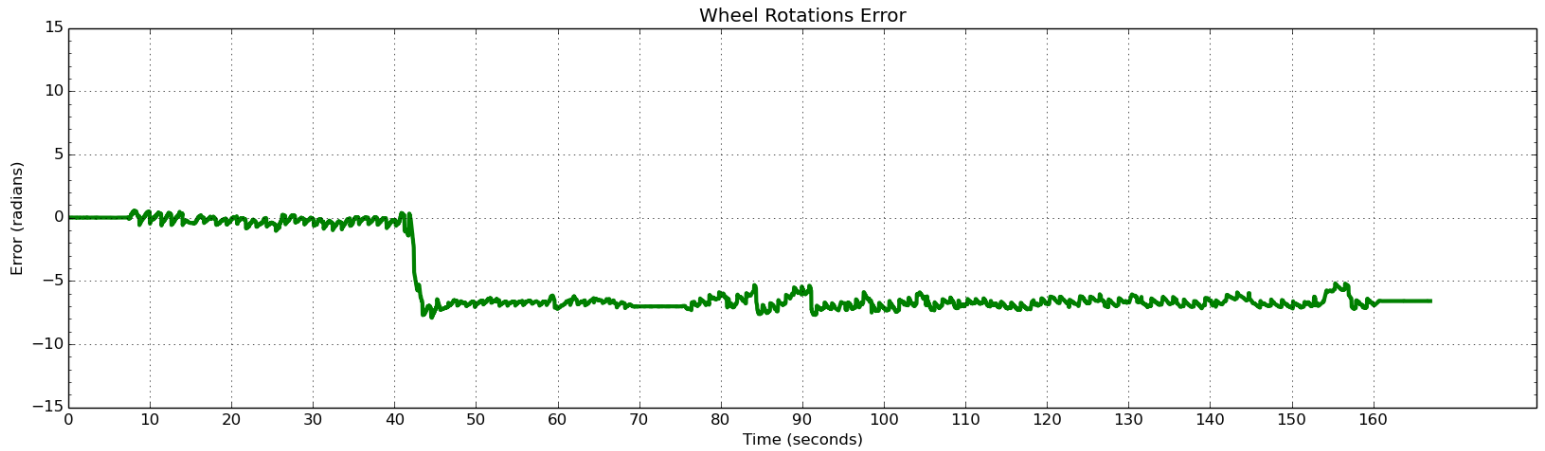


Figure 14: Drift in Estimated wheel rotation with respect to Ground truth data

Dataset	Duration (secs)	Axle Roll MAE		Steering Angle MAE		Maximum wheel rotation drift (rad)
		Front	Rear	Front	Rear	
Dataset 1	82	1.65°	1.51°	2.49°	1.98°	0.86
Dataset 2	170	1.41°	1.32°	2.40°	2.49°	3.70
Dataset 3	257	2.41°	1.97°	1.86°	2.55°	1.78

TABLE I: Mean Absolute Error (MAE) and Wheel rotation drift computed for three independent datasets

Ground truth data was recorded via three inertial measurement units for axle roll, two steering potentiometers for steering angle and a motor encoder for wheel angular rotation. The image sequences of three independent datasets, each traversing the span of the test-site, were used as inputs to estimate the 10 degrees-of-freedom.

Figures 9 to 14 are all plotted using **dataset 2**. Figure 9 and 10 illustrate the variations in ψ_F, ψ_R and λ_F, λ_R over time. The difference between the ground truth data and the estimated data is represented by Figure 11 and 12. The 2 methods correspond significantly well over all datasets, as seen in Table I, which shows the method's mean absolute error. Figure 13 plots the average of θ_{RL} and θ_{RR} compared against the rotations output from the rear motor encoder. Figure 14 shows the estimation error of the aforementioned graph.

The results are found to be highly satisfactory, and convey overall agreement between optical and conventional methods. Nevertheless, certain sources of error have been identified as below-

(i) Drop in frame rate results in the loss of continuous data. Frame rate is critical in the case of optical flow, where features are lost.
(ii) Excessive camera mount vibration results in noisy output.
(iii) Loss of marker positions due to self-shadowing causes gaps in kinematic state data.
(iv) Self-shadowing on wheels alter direction of optical flow vectors, giving erroneous angular rotation.

VII. Conclusion and Future Work

This paper introduces a novel approach to kinematic state estimation, and the results demonstrate high confidence in a single camera system. The method successfully precludes nine proprioceptive sensors, to achieve comparable kinematic state estimation with a single camera. As rovers minimalize and as reliability call for redundancy, state estimation via vision will be incorporated into planetary rovers. This is well-supported by the predicted increase in computational power in space robotics.

The method sets a precedent and provides a foundation for further work in vision-based kinematic state estimation. Future work would include developing an optimized version of the algorithm, capable of running on-board a planetary rover. Self-shadowing and abrupt changes in lighting affect the system adversely, and robustness to self-shadowing [12] and illumination-invariant tracking are rewarding avenues for further research. In addition, kinematic state estimation has no theoretical dependence on fiducial markers, and future work could focus on tracking inherent features of the robot. A downward-facing fisheye camera can also perform visual odometry, that tracks terrain features to give motion estimate. Future work will have visual odometry and optical kinematic estimation in conjunction to achieve unprecedented odometry with utmost simplicity. With two independent representations of motion, rover slip can also be detected.

Acknowledgments

The authors would like to thank the Robotics Institute Summer Scholars Program, Field Robotics Center and Carnegie Mellon for enabling and facilitating the above research.

References

[1] Richard Volpe. Navigation results from desert field tests of the rocky 7 mars rover prototype. *The International Journal of Robotics Research*, 18(7):669–683, 1999. doi: 10.1177/02783649922066493. URL http://ijr.sagepub.com/content/18/7/669.abstract.

[2] Michael Dille, Benjamin P Grocholsky, and Sanjiv Singh. Outdoor downward-facing optical flow odometry with commodity sensors. In *Proceedings Field & Service Robotics (FSR '09)*, July 2009.

[3] M.J. Roman. *Design and Analysis of a Four Wheeled Planetary Rover*. University of Oklahoma, 2005. URL https://books.google.com/books?id=d-hSNwAACAAJ.

[4] Ellery A. *Planetary Rovers: Robotic Exploration of the Solar System*. Springer Berlin Heidelberg, 2016. URL http://link.springer.com/book/10.1007%2F978-3-642-03259-2.

[5] Larry Matthies, Mark Maimone, Andrew Johnson, Yang Cheng, Reg Willson, Carlos Villalpando, Steve Goldberg, Andres Huertas, Andrew Stein, and Anelia Angelova. Computer vision on mars. *International Journal of Computer Vision*, 2007.

[6] T. M. Howard, A. Morfopoulos, J. Morrison, Y. Kuwata, C. Villalpando, L. Matthies, and M. McHenry. Enabling continuous planetary rover navigation through fpga stereo and visual odometry. In *Aerospace Conference, 2012 IEEE*, pages 1–9, March 2012. doi: 10.1109/AERO.2012.6187041.

[7] J. Balaram. Kinematic observers for articulated rovers. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 3, pages 2597–2604 vol.3, 2000. doi: 10.1109/ROBOT.2000.846419.

[8] M. Tarokh, G. McDermott, S. Hayati, and J. Hung. Kinematic modeling of a high mobility mars rover. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 992–998 vol.2, 1999. doi: 10.1109/ROBOT.1999.772441.

[9] Pierre Lamon and Roland Siegwart. 3d position tracking in challenging terrain. *The International Journal of*

Robotics Research, 26(2):167–186, 2007. doi: 10.1177/0278364906075170. URL http://ijr.sagepub.com/content/26/2/167.abstract.

[10] J. Sturm, C. Plagemann, and W. Burgard. Unsupervised body scheme learning through self-perception. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 3328–3333, May 2008. doi: 10.1109/ROBOT.2008.4543718.

[11] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

[12] N. Seegmiller and D. Wettergreen. Optical flow odometry with robustness to self-shadowing. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 613–618, Sept 2011. doi: 10.1109/IROS.2011.6094670.

[13] Eugene Fang . High-fidelity planetary route determination using computationally efficient monocular fisheye odometry and sun compass. Technical Report CMU-RI-TR-16-14, Robotics Institute, Pittsburgh, PA, May 2016.

[14] P. Backes, A. Diaz-Calderon, M. Robinson, M. Bajracharya, and D. Helmick. Automated rover positioning and instrument placement. In *2005 IEEE Aerospace Conference*, pages 60–71, March 2005. doi: 10.1109/AERO.2005.1559299.

[15] D. Scaramuzza, A. Martinelli, and R. Siegwart. A flexible technique for accurate omnidirectional camera calibration and structure from motion. In *Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*, pages 45–45, Jan 2006. doi: 10.1109/ICVS.2006.3.

[16] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. A toolbox for easily calibrating omnidirectional cameras. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5695–5701. IEEE, 2006.

[17] Jianbo Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, Jun 1994. doi: 10.1109/CVPR.1994.323794.

[18] Jean-Yves Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5(1-10):4, 2001.

Vinitha Ranganeni

Carnegie Mellon University

Push planning is a form of manipulation planning that involves pushing objects through a cluttered environment into a goal configuration or region.

https://github.com/vinitha910

Unobservable Monte Carlo Planning for Nonprehensile Rearrangement Tasks

Jennifer E. King, Vinitha Ranganeni, Siddhartha S. Srinivasa
The Robotics Institute, Carnegie Mellon University
{jeking, vrangane, ss5}@andrew.cmu.edu

***Abstract*— In this work, we present an anytime planner for creating open-loop trajectories that solve rearrangement planning problems under uncertainty using nonprehensile manipulation. We first extend the Monte Carlo Tree Search algorithm to the unobservable domain. We then propose two default policies that allow us to quickly determine the potential to achieve the goal while accounting for the contact that is critical to rearrangement planning. The first policy uses a learned model generated from a set of user demonstrations. This model can be quickly queried for a sequence of actions that attempts to create contact with objects and achieve the goal. The second policy uses a heuristically guided planner in a subspace of the full state space. Using these *goal informed* policies, we are able to find initial solutions to the problem quickly, then continuously refine the solutions as time allows. We demonstrate our algorithm on a 7 degree-of-freedom manipulator moving objects on a table.**

I. Introduction

In this work we generate open-loop trajectories that solve the rearrangement planning problem [1]–[4] using nonprehensile manipulation. In these problems, a robot must plan in a cluttered environment, reasoning about moving multiple objects in order to achieve a goal.

Nonprehensile interactions have proven to be a powerful strategy for pregrasp manipulation [5]–[7], manipulating large or heavy objects [1] and manipulating in clutter [8]–[10]. However, open-loop execution of trajectories that incorporate nonprehensile actions are prone to failure due to uncertainties in object and robot pose and in the physical modeling of the interactions.

Prior work has shown that nonprehensile interactions such as the push-grasp [1], [11] can be inherently uncertainty reducing and successfully executed open-loop if the uncertainties in pose and interaction are considered when generating the motion. This analysis of pushing under uncertainty has been limited to short simple motions such as a straight line push. In this work we consider the following question: *How do we generalize this analysis to create robust open-loop pushing trajectories for use in rearrangement tasks?*

This generalization to full rearrangement trajectories introduces three inherent challenges.

First, rearrangement planning occurs in a continuous high dimensional state space that describes the state of the robot and movable objects. This requires our planner search across an infinite dimensional belief space in order to account for state uncertainties.

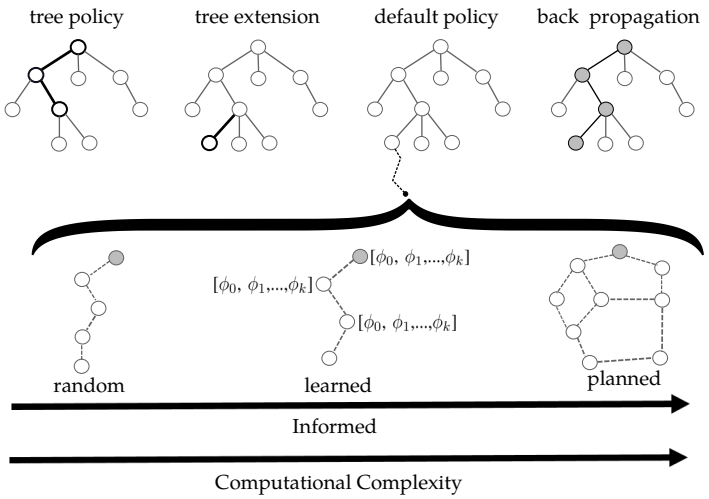


Fig. 1. Unobservable Monte Carlo Planning based on Monte Carlo Tree Search. The planner relies on a *default policy* to compute the potential for goal achievement from a node in the tree. We propose two policies: learned and planned. These policies trade-off additional computational complexity for better informed decision making that improves the efficiency of tree growth.

Second, contact between robot and objects causes physics to evolve in complex, non-linear ways and quickly leads to multimodal and non-smooth distributions. This makes methods that rely on closed form representation of the stochastic dynamics inapplicable [12].

Third, most actions in the continuous action space fail to make and, importantly, sustain meaningful contact with objects. Unlike grasping, the motion of a pushed object cannot be modeled as rigidly attached to the robot. Instead, the motion is directly governed by the physics of the interaction between robot and object. Once contact is made, only a small subset of the continuous action space will sustain this contact and move the object.

In this work, we propose an Unobservable Monte Carlo Planner (UMCP) that extends Monte Carlo Tree Search (MCTS) methods into unobservable domains. MCTS methods naturally deal with the first two challenges. The algorithm focuses the search to beliefs reachable from a known initial belief state and uses Monte Carlo simulations to approximate the unknown stochastic dynamics.

Central to MCTS algorithms is the use of a *default policy* to guide a simulation, or rollout, that quickly evaluates the potential value from a state. The simplest default policies

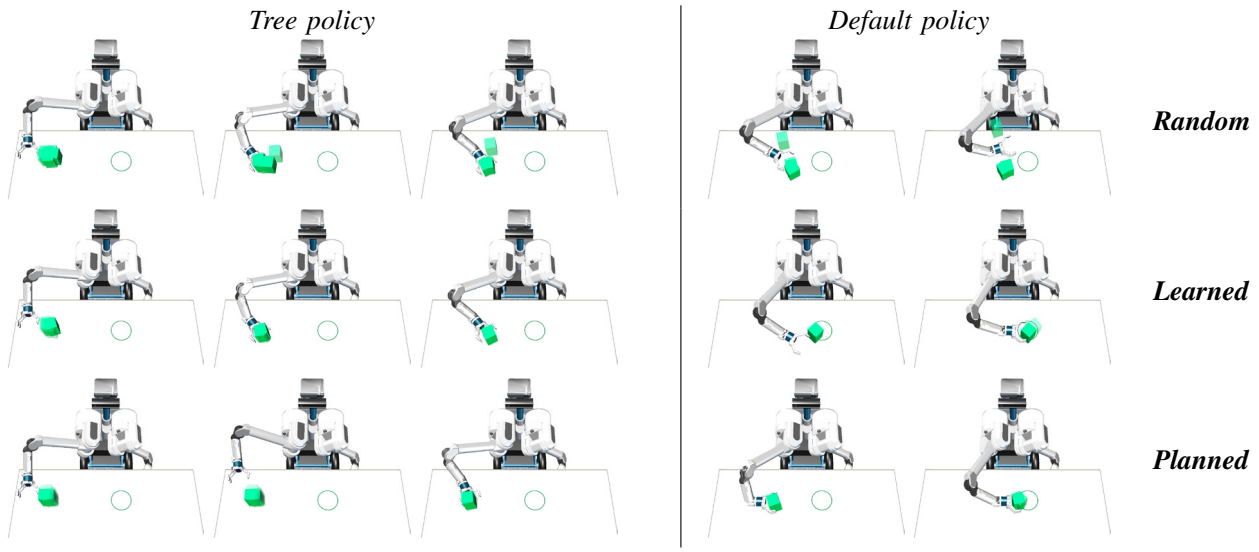


Fig. 2. An example path computed with the *random* default policy, *learned* default policy and *planned* default policy after $t = 45\,\mathrm{s}$ of planning time. The portion of the paths extracted using the tree policy look similar. However, the *random* policy quickly breaks contact with objects and is unable to move any portion of the belief to the goal. The *learned* and *planned* policies are able to use knowledge of the problem to maintain contact and achieve the goal.

perform random rollouts, drawing action sequences from a distribution over the control space defined for the robot. In rearrangement planning, random rollouts will rarely be informative, as most action sequences fail to make meaningful contact with objects. Relying on this default policy restricts the amount of information the planner can use to guide tree growth.

Our insight is that by carefully selecting a *goal informed default policy* that generates actions with the goals of rearrangement planning and nonprehensile interaction in mind, we can extract useful trajectories from the planner much earlier to better guide the search. We propose two such policies. The first uses user demonstrated trajectories to learn a mapping from state to control space. This mapping can then be used in place of random selection to generate rollouts. The second uses a planner that can solve rearrangement planning in the lower dimensional subspace containing only objects critical for goal achievement. The sequence of actions generated by the planner is then used to perform a rollout in the full state space.

These policies trade-off addition computational complexity for more informed decision making to guide tree growth (Fig.1). We test both policies on a manipulation task requiring a robot to push objects on a table. Our results show the *learned default policy* is useful when working in low clutter (Fig.2) while the *planned default policy* better guides the search through high clutter.

The remainder of this paper is structured in the following way. In Sec.II we formalize the rearrangement planning problem. In Sec.III we outline the UMCP planner and our proposed default policies. We demonstrate the effectiveness of the algorithm in Sec.V. Finally, we discuss limitations and areas for future work in Sec.VI.

II. The Rearrangement Planning Problem

A. Terminology

Our environment contains a robot $\mathcal{R}$, a set, $\mathcal{M}$, of objects that the robot is allowed to manipulate and a set, $\mathcal{O}$, of obstacles which the robot is forbidden to contact. We define the state space of the planner X as the Cartesian product of the state spaces of the robot and all objects in $\mathcal{M}$: $X = X^{\mathcal{R}} \times X^1 \times \cdots \times X^m$. We define the free state space $X_{free} \subseteq X$ as the set of states where the robot and objects are not contacting the obstacles and are not penetrating themselves or each other. Note that this definition specifically allows contact between robot and movable objects, which is critical for manipulation.

We consider pushing interactions. Thus, the motion of the movable objects is governed by the physics of the environment and the contact between the objects and the robot. As a result, the state x evolves non-linearly based on the physics of the manipulation. We describe this evolution as a non-holonomic constraint:

$$\dot{x} = f(x, u) \tag{1}$$

where $u \in \mathcal{U}$ is an instantaneous control input to the robot.

The task of rearrangement planning is to find a *feasible* trajectory $\xi : \mathbb{R}^{\geq 0} \to X_{free}$ from an initial state $x_0 \in X_{free}$ to any state in a goal region $X_{\mathrm{G}} \subseteq X_{free}$. A path is feasible if there exists a mapping $\pi : \mathbb{R}^{\geq 0} \to \mathcal{U}$ such that Eq.(1) holds throughout the duration T of the trajectory: $\dot{\xi}(t) = f(\xi(t), \pi(t))$ for all $t = [0, \ldots, T]$.

B. Uncertainty

We wish to generate open-loop plans robust to the uncertainties prevalent when executing trajectories in the real world. In general, open-loop plans are susceptible to failure due to uncertainty in initial state and poor modeling of both the motion of the manipulator and the physics of the interaction (Fig.3).

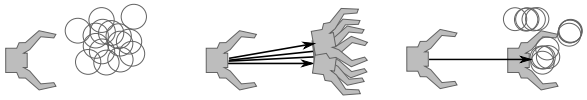


Fig. 3. Three sources of uncertainty in rearrangement planning with nonprehensile interaction: (*left*) Initial state. (*center*) Manipulator motion. (*right*) Physical interaction.

We represent initial state uncertainty as a belief $b_0 = p(x_0)$ that describes a probability distribution over possible initial states $x_0 \in X_{free}$. We represent modeling errors by assuming our state evolves as a stochastic non-holonomic system. This induces a distribution $p(\xi|\pi, x)$ over the trajectories that result from executing a control sequence π from a state x under the stochastic transition dynamics.

We represent the rearrangement problem as an instance of conformant probabilistic planning [13] where the goal is to maximize the probability that executing a sequence of actions π results in goal achievement. We can express this probability as an expectation:

$$p_\pi(x) = \int_\Xi \mathbf{1}_G(\xi) p(\xi|\pi, x) d\xi \tag{2}$$

where Ξ is the set of all trajectories from a state x and $\mathbf{1}_G : \xi \to \{0, 1\}$ is the indicator function that returns 1 if the endpoint of ξ is in X_G. Our goal is to generate a control sequence $\pi^* \in \Pi$ that maximizes the probability of success given all uncertainties:

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}}\, p_\pi \tag{3}$$

$$= \underset{\pi \in \Pi}{\operatorname{argmax}} \int_{x \in X} b_0(x) p_\pi(x) dx \tag{4}$$

In this work, we consider mappings π instantiated as a sequence of discrete actions $\pi = \{a_1, \ldots, a_j\}$ where each action $a_i = (u_i, \Delta t)$ represents a control input to the robot and a duration to apply the control.

III. Monte Carlo Tree Search

In our domain, the evolution of the uncertainty when using nonprehensile interactions is non-smooth and non-Gaussian. These characteristics make closed form representation of the system dynamics difficult. As a result, exact computation of Eq.(2) is not possible.

Monte Carlo methods have been used widely when the exact dynamics are unknown or difficult to model [14]–[18]. These methods use a generative model, G, or black-box simulator, to sample successor states given a current state and an action: $x' \sim G(x, a)$.

Monte Carlo Tree Search [17], [19] (MCTS) is one such algorithm that uses this paradigm. The MCTS algorithm iteratively builds a tree using Monte Carlo simulations. The tree estimates the value of action sequences by tracking the mean reward obtained from simulations of the sequences.

MCTS is a good fit for our problem. We can use a physics model to perform the black-box simulations. These physics simulations have some computational expense. The MCTS framework efficiently focuses computational resources to relevant regions of state space. In addition, the algorithm is anytime and highly parallelizable.

Algorithm 1 Unobservable Monte Carlo Planning

```
1:  s_0 ← GenerateInitialSamples()
2:  while not timeout do
3:      x ← SampleState(s_0)
4:      Simulate(x, {}, 0)
5:  function Simulate(x, h, d)
6:      if γ^d < ε then return 0
7:      if NotVisited(h) then
8:          InitializeHistory(h)
9:          return DefaultPolicy(x)
10:     a ← TreePolicy(h)
11:     x' ← G(x, a)
12:     r ← R(x, a)+
13:         γ· Simulate(x', h ∪ {a}, d + 1)
14:     B̂(h) ← B̂(h) ∪ {x}
15:     N(h) ← N(h) + 1
16:     Q̂(h, a) ← Q̂(h, a) + r
17:     return r
```

IV. Unobservable Monte Carlo Planning (UMCP)

The POMCP [17] algorithm applies the MCTS framework to partially observable environments. We use a similar approach to plan in our unobservable environment. We build a tree such that each node represents a unique history, $h = \{a_1, \ldots a_t\}$. Three values are stored for each node: $N(h)$ - the number of times the history, or action sequence, has been explored, $\hat{Q}(h, a)$ - an estimate of the value of taking action a after applying history h, and $\hat{\mathcal{B}}(h)$ - an estimate of the true belief achieved when applying the actions in h from known initial belief b_0.

Alg.1 shows the MCTS algorithm applied to our UMDP. The tree is rooted with an initial belief state s_0 that contains a set of states drawn from an initial distribution defined on the state space. Then, during the search an initial state $x \sim s_0$ is drawn from the belief state (Line 3). This state is propagated through the tree by using the *tree policy* to select actions (Line 10) and using a noisy physics model to forward propagate the state under the selected actions (Line 11). After applying the physics model, the new state is added to the belief state of the history (Line 14). The search recurses through the tree, propagating a single state through the noisy transition dynamics. Over time, the belief states represented at the nodes of the tree grow to represent the true belief distribution.

Once the search reaches a previously unvisited history, a *default policy* is used to rollout the remainder of a simulation and accumulate reward (Line 9). This reward is propagated back through the tree to update the value function estimate stored for each history.

A. Reward model

As stated in Sec.II, our goal is to generate paths that maximize the probability of successful execution. We encode this goal in our reward model:

$$R(x, a) = \mathbf{1}_G(x) \tag{5}$$

Here the indicator function returns 1 if $x \in X_{\mathrm{G}}$.

B. Action set

MCTS-based planners search across a discrete action set. The naive method for generating a discrete action set from our continuous space $\mathcal{U}$ is to divide the space into partitions, or bins, and select a single representative control from each bin and create an action that applies this control for a fixed duration: $a = (u, \Delta t)$. These *basic* actions are context agnostic: they ignore the goal of the planning instance.

We know *contact* is critical for goal achievement in rearrangement problems. We augment the action set to account for this by generating specific *contact* actions aimed at maintaining contact with objects important to goal achievement.

These *contact* actions are state dependent and must be dynamically generated for each node, or history, in the tree. We instantiate *contact* actions using the first state in the estimated belief $x \in \hat{B}(h)$ for each history. A *contact* action is generated by solving the two-point BVP in the robot's state space that moves the robot to a pose in contact with an object based on the object's pose in x. We create one *contact* action for each object in x defined in the goal.

The result is a discrete action set $\mathcal{A}_h = \mathcal{A}_{basic} \cup \mathcal{A}_{cont}$ for each history h composed of a set $\mathcal{A}_{basic}$ of *basic* actions that move the robot without the explicit intent of creating contact with objects and a set $\mathcal{A}_{cont}$ of *contact* actions that explicitly contact important objects in the scene.

C. Tree Policy

The tree policy is used to select actions, or edges, in the UMCP tree to traverse. On the first visit to a given node in the tree corresponding to history h, the method from the previous section is used to generate a discrete set of actions $\mathcal{A}_h$. On subsequent visits, we follow the UCT algorithm [15] and use UCB1 [20] to select a single action from this set to traverse as follows:

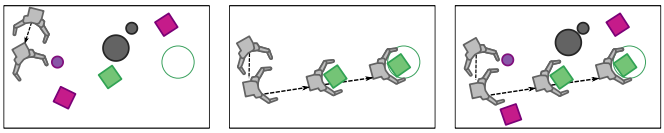
$$a_t = \underset{a \in \mathcal{A}_h}{\operatorname{argmax}} \frac{\hat{Q}(h, a)}{N(h \cup \{a\})} + c\sqrt{\frac{\log N(h)}{N(h \cup \{a\})}} \tag{6}$$

where $c > 0$ is an exploration constant. Note that this selection method requires all actions are tried at least once.

The use of such a method is ideal because it provides a formal method for trading between exploration and exploitation.

D. Default policy

Each time the search reaches a leaf in the tree, the default policy is used to estimate the reward that will be obtained if we follow a path that leads through this leaf. The most common default policy is to randomly select a sequence of actions to apply. For our rearrangement planning problem, this policy will rarely be informative: most action sequences fail to create and maintain the contact with objects that is critical to goal achievement.



(a) (b) (c)

Fig. 4. An example of MCTS applied to rearrangement planning using the planned default policy. (a) A *tree policy* is used to select initial actions. (b) The *default policy* plans in the lower dimensional space containing only objects important to goal achievement. (c) The resulting path is propagated through the full space to generate a reward.

Instead, we define two informed default policies that are capable of evaluating the potential to achieve the goal. The first uses a model learned from user demonstrated trajectories to generate a rollout. The second uses a heuristically guided planner in a subspace of the full state space. In the following sections we outline these policies.

Learned default policy

Our intuition is that we can use human judgment to learn a good default policy. Our goal is to learn a mapping: $\rho : X \to \mathcal{U}$ from a state $x \in X$ to a control $u \in \mathcal{U}$ from a set of human demonstrations. We can use any method for collecting these demonstrations and learning the mapping. We discuss a specific choice in the experiments.

Once we have learned a mapping ρ, we can use it as a default policy by creating a fixed length rollout. To do this we repeatedly use ρ to generate a control and G to forward propagate the control for a fixed duration Δt:

$$x_{t+1} = G(x_t, (\rho(x_t), \Delta t))$$

We compute the reward achieved by the rollout using the final state of the rollout.

Planned default policy

The learned default policy attempts to generalize demonstrations to solve an instance of rearrangement planning. In this section we explore an alternative approach: we apply a state space planner to quickly search for a solution to the specific rearrangement problem.

We perform the search in the lower dimensional subspace containing only the elements of the full state space that are defined in the goal.

After generating a sequence of actions that solve the problem in the lower dimensional subspace, the actions are then forward simulated through the full state space using G and reward is calculated from the final reached state. Fig.4 illustrates this method.

By reducing the dimensionality of the state space in the default policy search, we allow for the possibility of using fast planners or exact solvers that provide much more information than random action sequences. We discuss a specific example in the experiments.

E. Path extraction

We use our tree to create an anytime algorithm for extracting paths. Upon request, a path π is extracted from the tree as follows. First, we extract π_{tree} by repeatedly picking the action a_t such that:

$$a_t = \underset{a \in \mathcal{A}_h}{\operatorname{argmax}} \hat{Q}(h, a)$$

Once a leaf is encountered, we query the belief represented by the history $\hat{\mathcal{B}}(h)$ to obtain an estimated probability of success $\hat{p}_{\pi_{tree}} = \sum_{x \in \hat{\mathcal{B}}(h)} \mathbf{1}_G(x)$. If this probability is lower than the estimated probability of success of the best path found so far, $\hat{p}_{\pi^*}$, we randomly select a state from the belief $x \in \hat{\mathcal{B}}(h)$ and use the *default policy* to generate a path π_{def} from x to the goal. If successful, all remaining samples in $\hat{\mathcal{B}}(h)$ are forward propagated through this path to get an updated probability of success $\hat{p}_{\pi}$ of the combined path π formed from appending π_{def} to π_{tree}. If $\hat{p}_{\pi} > \hat{p}_{\pi^*}$ the path is returned. Otherwise, the previous best path π^* is returned.

The use of a *goal informed default policy* means we can often find path segments π_{def} that achieve the goal with non-zero probability. Our insight is that these path segments can be particularly useful when there is not enough planning time to deeply grow the tree, i.e. $\hat{p}_{\pi_{tree}} = 0$.

V. Experiments and Results

To test the capabilities of the UMCP algorithm, we task our robot HERB [21] with pushing a box on a table to a goal region of radius 0.1 m using the 7-DOF right arm. We test three hypothesis:

H.1 Using explicit **contact** actions allows the UMCP planner to generate paths with higher probability than a planner that uses a **basic** action set formed by discretization of each dimension of the control space.

H.2 The UMCP planner using the *goal informed* **learned** or **planned** default policy generates paths with higher probability of success than the UMCP planner with a **random** default policy.

H.3 The UMCP planner that uses **contact** actions and the *goal informed* default policy is able to produce paths that exhibit higher probability of success compared to baseline state space planners.

H.1 tests the need for actions that explicitly try to create contact with goal critical objects. **H.2** verifies our intuition that using a *goal informed* default policy will guide tree growth better than a random default policy. Finally, **H.3** verifies the need to track the evolution of uncertainty through sequences of actions during planning.

In the following sections we detail our planning setup and provide results for each hypothesis.

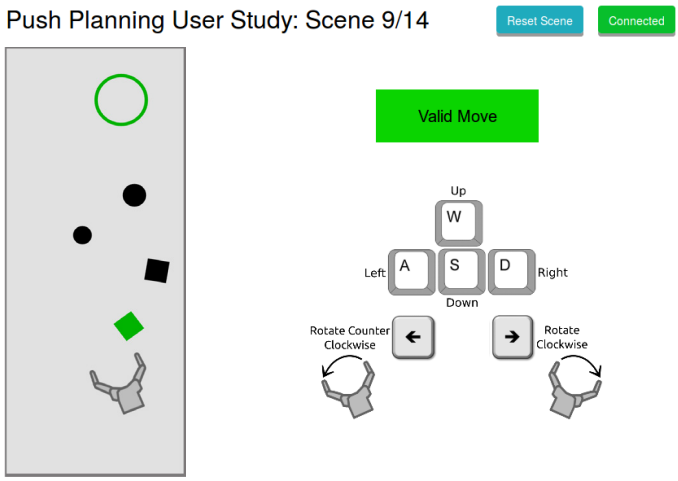


Fig. 5. An example interface used to collect user demonstrated rearrangement trajectories

A. Test setup

We run each version of the UMCP planner 50 times on a scene with only one movable object (denoted low clutter in all results) and a scene with six movable objects (denoted high clutter in all results). The scenes are depicted in Fig.6-*top*. In each scene, we generate the initial belief s_0 by sampling noise into the initial pose of each object from a Gaussian with distribution $\mu = \mathbf{0}, \Sigma^{1/2} = \texttt{diag}\{\texttt{2cm}, \texttt{2cm}, \texttt{0.1rad}\}$. We allow the UMCP planner to run for 300 s. We request and record a path every 15 s.

Following [10] we constrain the end-effector to move in the xy-plane parallel to the table in order to create motions likely to pushing objects. This allows us to define our control space $\mathcal{U}$ as the set of feasible velocities for the end-effector. We convert these end-effector velocities to full arm velocities using the Jacobian psuedo-inverse:

$$\dot{q} = J^{\dagger}(q)u + h(q)$$

where q is the current arm configuration, u is the end-effector velocity and $h : \mathbb{R}^7 \rightarrow \mathbb{R}^7$ is a function that samples the nullspace.

B. Learned default policy

We collect workspace trajectories from 97 users using Amazon Mechananical Turk. To collect these trajectories, we generate a set of seven scenes that require a robot to push an object on a table to a goal region. Each user is asked to solve each scene twice, for a total of 14 demonstrations from each user. For each demonstration, we record and save the sequence of state/action pairs, (x_t, a_t) used to solve the scene.

We provide users with an interface that allows them to apply discrete actions using keyboard input. Fig.5 shows an example interface. To simplify the user task, we render only the end-effector of the robot. In the example shown, the user must guide the robot hand to push the green box into the green circle. The action corresponding to the keyboard input

from the user is pushed through a physics model to generate an updated pose of the robot and objects.

We identify and extract a set of relevant features Φ of a state $x \in X$. Example features are the number of objects blocking a direct path to the goal or the distance to the goal. Then, for each of the 112,928 state/action pairs (x_i, a_i) collected from users, we create feature vectors ϕ_i from state x_i labeled by the action a_i to use as training data for a multi-class classifier. We train a random forest classifier using the scikit-learn python package [22]. This classifier serves as the mapping ρ needed to perform rollouts.

C. Planned default policy

We use a weighted A* search [23] with $w = 5.0$ for our planned default policy. The search uses the same discrete action set used to build the UMCP tree, including the *contact* actions critical to achieving success.

The search runs in the lower dimensional subspace containing only the robot and the box the robot must push to the goal. To guide the search, we compute the cost of an action a as the distance in workspace the robot moves during the action. Then we define an admissible heuristic, $h(x) = d_{cont}(x) + d_{move}(x)$, where d_{cont} is the minimum workspace distance the robot must move to make contact with the box and d_{move} is the minimum workspace distance the box must move to achieve the goal.

The planner is allowed 1 s to search. The result of the search is a sequence of actions $\{a_1, \ldots, a_k\}$ that describe robot motions to achieve the goal in the lower dimensional subspace (or an empty sequence if no solution is found). The action sequence is then applied in the full state space to check for validity and compute reward.

D. Baseline planners

We compare the estimated success rate of the paths generated by the UMCP algorithm to an anytime version of the B-RRT from [24]. Briefly, the B-RRT planner generates rearrangement plans by using a Rapidly-exploring Random Tree (RRT) [25] to search through state space. During tree growth, each action is evaluated for its robustness to uncertainties and this evaluation is used to bias tree growth– robust actions are more likely to be selected for extensions. Importantly, the planner does not track uncertainty through sequences of actions. Instead it makes local decisions about the robustness of each action. We test against two versions of the algorithm. First, we set $b = 0.0$ to eliminate all bias. This reduces the planner to a search over state space that does not consider uncertainty. Second, we set $b = 2.0$. This biases the search to prefer uncertainty reducing actions.

To create an anytime version of the planner, we make as many repeated calls to the planner as possible within 300 s. When a call completes we perform a set of 100 noisy rollouts on the resulting control sequence π to generate an estimate $\hat{p}_\pi$ of the probability of success. We generate these noisy rollouts using the same noise parameters used to create the initial belief s_0 in the UMCP planner. We keep π only if it has higher estimated success probability than all previous

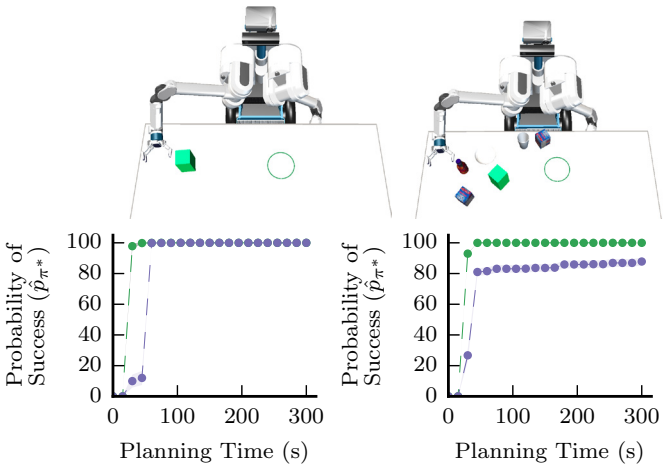


Fig. 6. Using **contact** actions (—) allows for finding better paths sooner than using only **basic** action (—) on both a low clutter scene (*left*) and a high clutter scene (*right*).

paths generated by the planner. This is a similar algorithm to the AMD-RRT described in [24] though we use probability of success rather than performance of the path under a divergence metric.

E. Effect of contact actions

We first examine the effect of the **contact** actions in the action set. We compare the UMCP planner with **contact** actions to the UMCP planner with only the **basic** actions formed from discretization of the control space. Both version of the planner use the planned default policy.

Fig.6 compares the probability of success $\hat{p}_{\pi*}$ of the path returned by the planner at each time step. In low clutter scenes, the usefulness of the **contact** actions is limited (Fig.6-*left*). The UMCP planner with the **contact** actions finds paths only slightly faster. This is due to the use of the **planned** default policy to complete paths extracted from the tree. Examination of the generated paths shows that the default policy is heavily relied upon to generate the contact needed for success when using only the **basic** action set.

The benefit of these actions is much more prevalent in high clutter scenes (Fig.6-*right*). Here, the **planned** default policy fails to be applied without first moving either the bowl or bottle. The **basic** action set is not rich enough to create useful contact. In contrast, the **contact** action easily moves both objects out of the way in order to make contact with the box. Then the **planned** default policy can be applied to achieve the goal. Fig.9 depicts an example path found by the UMCP planner with **contact** actions.

These results support **H.1**: Using explicit contact actions allows the planner to generate paths with higher probability than a planner that uses a basic action set formed by discretization of each dimension of the control space.

F. Effect of default policy

1) Low clutter scene: Next we examine the effect of our choice of default policy. Fig.7-*left* shows the estimated probability of success of the path output by the UMCP planner, $\hat{p}_{\pi^*}$, as a function of planning time for a low clutter

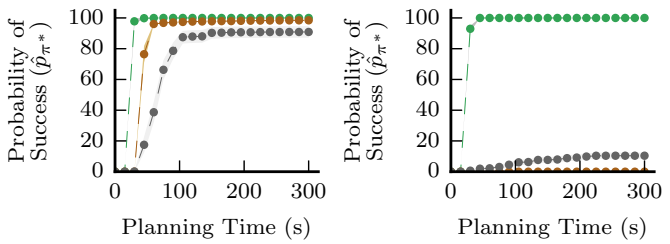


Fig. 7. Use of *goal informed* default policies such as the planned (—) and learned (—) result in overall better paths when compared to using a policy that randomly selects actions (—) for both low clutter (*left*) scenes and high clutter scenes (*right*).

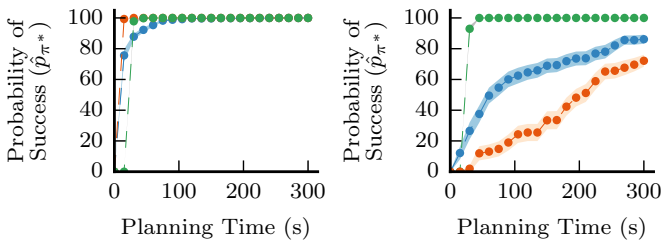


Fig. 8. On simple scenes (*top*), the B-RRT (b=2.0) from (—) is able to find better paths quickly. The UMCP algorithm (—) is able to find better paths more quickly than the B-RRT (b=0.0) (—). On difficult scenes (*bottom*), the UMCP outperforms the B-RRT.

scene. As can be seen, the use of the **planned** default policy allows us to generate path with high probability of success faster than the planner with the random default policy. This is despite the planned default policy taking almost 10x longer to compute than the random default policy (mean time $0.5\,\mathrm{s}$ and $0.06\,\mathrm{s}$ respectively). The **learned** default policy also outperforms the random default policy on this scene, finding solutions nearly as good as the **planned** policy.

Perhaps more interesting are the qualitative aspects of the results. Fig.2 shows an example path at $t = 45\,\mathrm{s}$ for the UMCP planner with each version of the default policy. The left column shows the portion of the path π_{tree} extracted from the tree. The right column shows the portion of the path π_{def} extracted using the default policy.

The tree paths for the **planned** and **random** default policies are similar, though the UMCP planner that uses the **planned** default policy finds a more robust sequence (Fig.2-*top*). This is because the default policy is more informative and allows better estimates of $\hat{Q}(h, a)$ early in the tree.

The main difference in the two results comes from the portion of the path extracted using the default policy. The **planned** default policy maintains contact with the goal object and eventually moves the full belief either into or near the goal region. The **random** default policy loses contact with the object quickly and fails to move any of the belief to the goal. Interestingly, the **learned** default policy is fast and informative enough that the tree is able to grow almost all the way to the goal within the $45\,\mathrm{s}$.

2) High clutter scene: Fig.7-*right* shows results for a high clutter scene. Again, the **planned** default policy performs well, returning paths with higher probability of success than the planner using the **random** policy. Interestingly, the **learned** default policy struggles to find solutions to this scene and performs worse than the **random** policy. This scene is difficult. The bottle blocks direct access of the box. It must be moved or avoided. The random policy fails to find any valid sequences of actions that achieve the goal for most of the search. As a result, no reward is propagated through the tree, and the tree fails to grow deep. Similarly, the **learned** policy fails to find good solutions, resulting again in a tree with almost no reward. Conversely, the **planned** policy is able to explicitly search for and find valid solutions for *this* problem. As a result, the tree contains sufficient reward to properly guide the search and find robust paths. Fig.9 depicts a solution found by the planner.common failure by

Our results partially support **H.2**: The *goal informed* **planned** default policy is useful in low and high-clutter scenes.

G. Comparison to baseline planners

Finally, we compare the UMCP planner using **contact** actions in the action set and the **planned** rollout policy to the baseline planners described in Sec.V-D. Fig.8 shows the estimated probability of success $\hat{p}_{\pi^*}$ of each planner as a function of planning time.

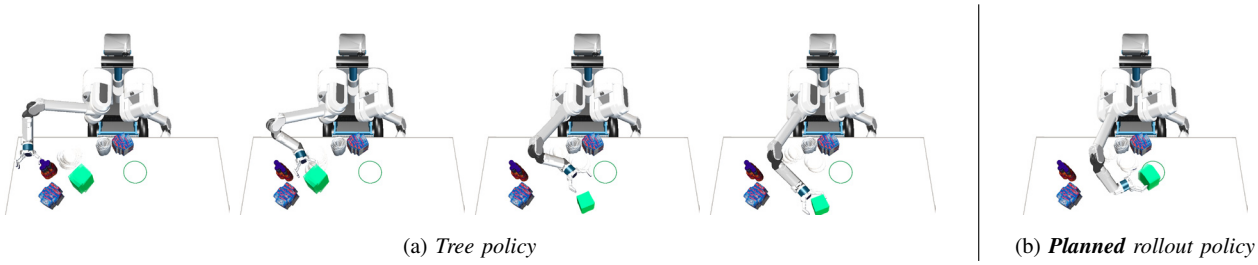
For the low clutter scene both versions of the B-RRT are easily able to find solutions (Fig.8-*left*). The B-RRT with $b = 2.0$ performs exceptionally well here because there exists a solution comprised almost entirely of uncertainty reducing, or low divergence, actions. These solutions are also found by the UMCP planner (Fig.2-*top*).

The advantage of the UMCP algorithm can be seen for the high clutter scene (Fig.8-*right*). Here, the B-RRT performs poorly because the actions that reduce uncertainty in the box increase uncertainty in the pose of other objects, such as the bottle or bowl (Fig.9). Such actions perform poorly under the divergence metrics used to bias the B-RRT. As a result, the B-RRT is slow to explore and find solutions. The UMCP allows for increasing uncertainty along dimensions that are not important for goal achievement. This allows the planner to find solutions more easily.

Overall, for planning time budgets greater than $30\,\mathrm{s}$ the UMCP algorithm finds solutions as good as the solutions found by the B-RRT algorithms in the low clutter scene. The UMCP algorithm outperforms both baseline planners in the high clutter scene. This supports **H.3**: Our UMCP planner that uses contact actions and the planned default policy is able to produce paths that exhibit higher probability of success compared to anytime versions of baseline planners.

VI. Discussion

In this work we propose an Unobservable Monte Carlo Planner. This algorithm extends MCTS to the unobservable domain. We show that by carefully selecting an informative default policy and an action set capable of generating contact with important objects in the scene, we are able to plan solutions that are robust to uncertainty. The result is an anytime algorithm that quickly returns good solutions in the example scenarios.



(a) *Tree policy* (b) ***Planned*** *rollout policy*

Fig. 9. In high clutter scenes, the UMCP algorithm with **contact** actions performs well. The algorithm allows for increasing uncertainty in the pose of the bowl, as long as it does not inhibit goal achievement.

Our experiments show that the *planned* default policy performed exceptionally well on the rearrangement tasks we consider. We concede that this task is particularly well suited for the planned default policy: only a single movable object is described in the goal, making the reduced state space containing only this object easy to search using a heuristic planner. More complicated rearrangement tasks that contain several objects defined in the goal, such as those in [26], may benefit less from this approach as the lower dimensional planning problem will be quite difficult to solve. We believe the *learned* default policy, trained with a sufficiently large and representative data set, may be more effective in these tasks.

This algorithm represents a step toward planning rearrangement tasks using nonprehensile manipulation in belief space. We believe there are two promising directions that may improve the quality of the planner. First, we can expand the set of actions considered by the planner by using gradient free methods to make local adjustments to the action set [27]. Second, we believe this algorithm could be extended to closed-loop planning by incorporating feedback as observations in a full POMDP formulation. Careful thought must be applied to allow us to maintain tractability under the exponential increase in histories due to the introduction of observations. However, recent work in using contact sensing [28] during push-grasping shows this may be possible.

References

[1] M. Dogar and S. Srinivasa, "A framework for push-grasping in clutter," in *Robotics: Science and Systems*, 2011.

[2] D. Nieuwenhuisen, A. Stappen., and M. Overmars, "An effective framework for path planning amidst movable obstacles," in *Workshop on the Algorithmic Foundations of Robotics*, 2008.

[3] M. Stilman. and J. Kuffner, "Navigation among movable obstacles: Real-time reasoning in complex environments," in *IEEE-RAS International Conference on Humanoid Robots*, 2004.

[4] J. van den Berg, M. Stilman, J. Kuffner, M. Lin, and D. Manocha, "Path planning among movable obstacles: a probabilistically complete approach," in *Workshop on the Algorithmic Foundations of Robotics*, 2008.

[5] J. Barry, K. Hsiao, L. P. Kaelbling, and T. Lozano-Pérez, "Manipulation with multiple action types," in *International Symposium on Experimental Robotics*, 2012.

[6] L. Chang, S. Srinivasa, and N. Pollard, "Planning pre-grasp manipulation for transport tasks," in *IEEE International Conference on Robotics and Automation*, 2010.

[7] J. King, M. Klingensmith, C. Dellin, M. Dogar, P. Velagapudi, N. Pollard, and S. Srinivasa, "Pregrasp manipulation as trajectory optimization," in *Robotics: Science and Systems*, 2013.

[8] M. Dogar, K. Hsiao, M. Ciocarlie, and S. Srinivasa, "Physics-based grasp planning through clutter," in *Robotics: Science and Systems*, 2012.

[9] M. Gupta and G. Sukhatme, "Using manipulation primitives for brick sorting in clutter," in *IEEE International Conference on Robotics and Automation*, 2012.

[10] J. King, J. Haustein, S. Srinivasa, and T. Asfour, "Nonprehensile whole arm rearrangement planning on physics manifolds," in *IEEE International Conference on Robotics and Automation*, 2015.

[11] M. Dogar and S. Srinivasa, "Push-grasping with dexterous hands: Mechanics and a method," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[12] J. van den Berg, P. Abbeel, and K. Goldberg, "LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information," in *Robotics: Science and Systems*, 2010.

[13] N. Hyafil and F. Bacchus, "Conformant probabilistic planning via CSPs," in *International Conference on Automated Planning and Scheduling*, 2003.

[14] B. Bonet and H. Geffner, "Labeled RTDP: Improving the convergence of real-time dynamic programming," in *International Conference on Automated Planning and Scheduling*, 2003.

[15] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo planning," in *European Conference on Machine Learning*, 2006.

[16] J. Tsitsiklis, "On the convergence of optimistic policy iteration," *Journal of Machine Learning Research*, vol. 3, pp. 59–72, 2002.

[17] D. Silver and J. Veness, "Monte-carlo planning in large POMDPs," in *Conference on Neural Information Processing Systems*, 2010.

[18] S. Thrun, "Monte Carlo POMDPs," in *Conference on Neural Information Processing Systems*, 2000.

[19] G.-B. Chaslot, S. Bakkes, I. Szita, and P. Spronck, "Monte-Carlo Tree Search: A new framework for game AI," in *Artificial Intelligence Interactive Digital Entertainment Conference*, 2008.

[20] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[21] S. Srinivasa, D. Ferguson, C. Helfrich, D. Berenson, A. Collet, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and M. Weghe, "HERB: A Home Exploring Robotic Butler," *Autonomous Robots*, vol. 28, no. 1, pp. 5–20, 2010.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[23] I. Pohl, "Heuristic search viewed as path finding in a graph," *Artificial Intelligence*, 1970.

[24] A. Johnson, J. King, and S. Srinivasa, "Convergent planning," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1044–1051, 2016.

[25] J. Kuffner and S. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation*, 2000.

[26] A. Krontiris and K. Bekris, "Dealing with difficult instances of object rearrangement," in *Robotics: Science and Systems*, 2015.

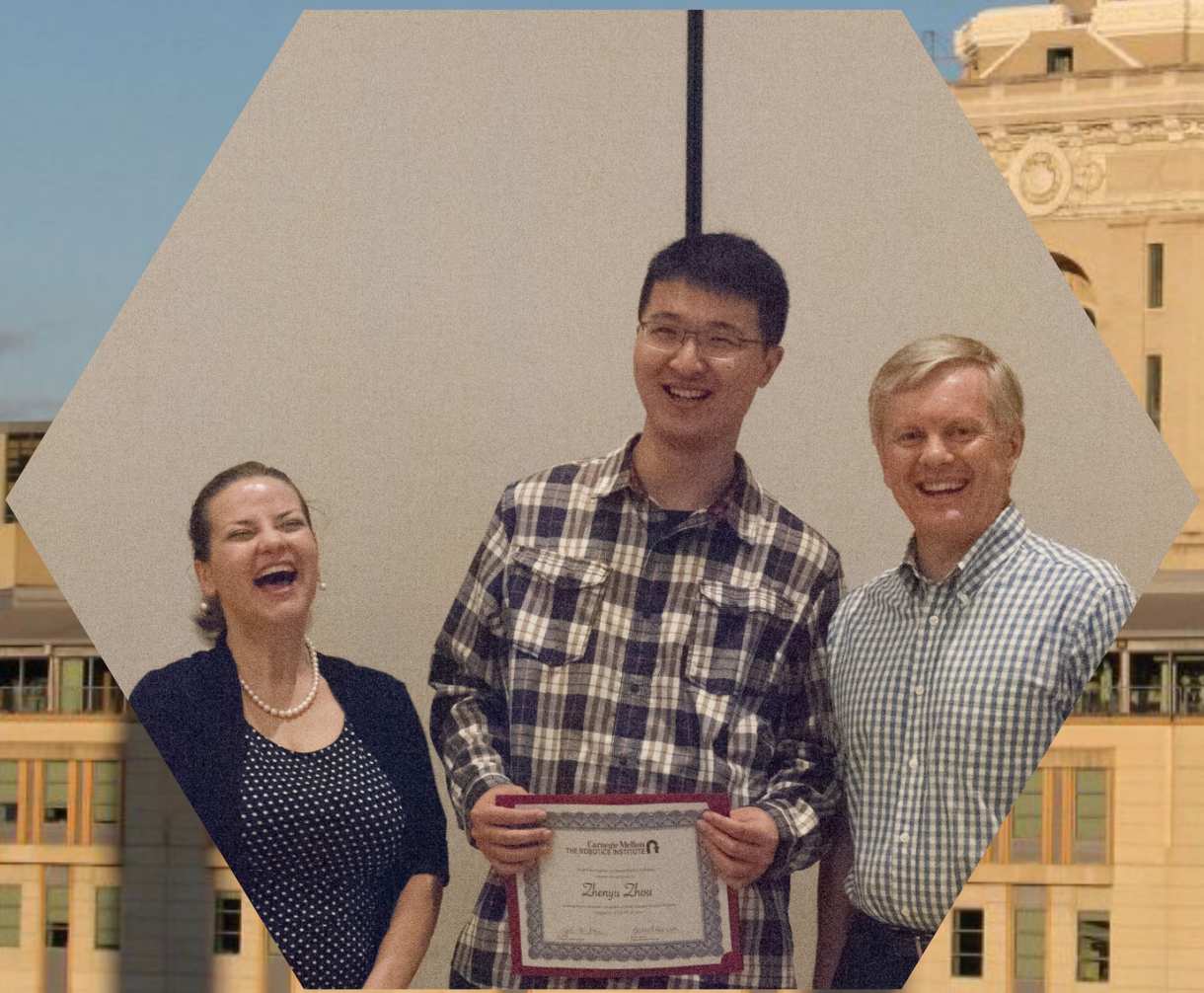
[27] K. Seiler, H. Kurniawati, and S. Singh, "An online and approximate solver for POMDPs with continuous action space," in *IEEE International Conference on Robotics and Automation*, 2015.

[28] M. Koval, N. Pollard, and S. Srinivasa, "Pre- and post-contact policy decomposition for planar contact manipulation under uncertainty," in *Robotics: Science and Systems*, 2014.

Zhenyu Zhou & Junpeng Wang

Nanjing university of Science and Technology

Traffic sign detection based on openCV.



Traffic sign detection with cell phone data

Zhenyu Zhou Junpeng Wang
Mentor Christoph Mertz

***Abstract*—Traffic detection and inventory has recently raised public interest because a highly accurate traffic detection system is essential to road inventory. The Navlab group has conducted research in stop sign, pedestrian and vehicle detection project in the past. However, in order to form a whole traffic sign detection system, it is necessary to build a detector that could detect varieties of traffic signs efficiently, which is most of our work. First, we compare different kinds of traffic signs in the US and put them into several categories, forming a sign database. For this process, a program labels and crops the interest areas of the picture. Second, aiming to obtain a best parameter, we use some of these samples to train several classifiers by applying an OpenCV algorithm based on Haar-like feature. Afterwards, we utilize some synthetic images to create training samples by altering their color property and conduct large-sample-based trials. A bootstrapping method is introduced to improve the classifiers performance in the training process. Finally, we evaluate the method and the final results indicate that a classifier based on Haar-like feature and AdaBoost algorithm could work well for common traffic signs.**

Index Terms—Traffic signs detection, Haar-like feature, AdaBoost, classifier training,

I. Introduction

Over recent years, much attention has shifted to recognizing object classes in the field of machine vision. Among them, Traffic Sign Recognition(TSR) is one of the areas where many challenges exist. Much progress has been made both in theoretical and commercial aspect in the past few years. But the research is still far from enough. Important problems have been solved with German Traffic Sign Recognition Benchmark(GTSRB) published in 2011[1] and the paper states that their system could work well in recognizing European traffic signs. Similar methods could be applied to detect USs traffic signs, however, there is far fewer publicly available data for US signs.

In order to obtain a proper traffic sign classifier and evaluate its performance in recognizing signs, the first thing we need to do is to obtain a large dataset of traffic sign. The appearances of the US traffic signs are significantly different from the other parts of the world. Hence, We need to build the dataset of US traffic signs. In theory, one could take thousands of photos of one standard traffic sign from different angles and use these images to train a classifier. But the results are usually disappointing due to the fact that neither the images are different from images in real environment nor are there many useful and distinguished features could be extracted from these images. Like mainstream field of large-scale data collection progress,our images are captured from a camera mounted in vehicles that traverse the road network on a regular basis[2]. Fortunately, we also found a North America dataset called LISA dataset on the internet, referred to its category organization and added some of images into our dataset. Owing to the fact that the camera recorded the images on a continuous basis, we have multiple similar images of the same traffic sign. Hence,we could just pick out only part of them into dataset for the purpose of decreasing sample training time.

More and more different object detection algorithms are emerging. One method was built upon the image thresholding followed by SVM classification[3]. But problems arised when setting the proper threshold for each image. Ruta introduced image color thresholding combined with shape detection[4]. Integral Channel Features offered good results on European signs[1]. We employed haar-like feature-based adaboost algorithma popular open source algorithm in the OpenCV. To guarantee the good performance of a classifier, false negative rate has to be slow and false positive rate should be reasonable. For the requirement of detection, taking into consideration that most of our work is related to road inventory, it is comparably not necessary for our TSR to work in a real-time condition than any other one which works in a autonomous driving system. On the other hand, to train a classifier in our system, we could discard some images which are not qualitied and utilize those with good quality. Comparably, an haar-like feature based algorithm would meet most of our requirement perfectly in a short time and high accuracy.

The primary goal of this article is to attempt to build a classifier which could detect common traffic signs. On the other hand, many traffic signs would be detected through the detection process, which could be added into dataset and form a larger sets. In the promising future, we would use a “HOG+SVM” method to conduct rigorous trials and compare the results with Haar+Adaboost method.

II. Detection Method

Detection Method In 2001, Paul Viola and Michael Jones published their new image representation called “Integal Image” to compute the features and brought about Adaboost, a learning algorithm used in the training process in their article[5]. A term called “cascade” was also discussed to find promising object efficiently. Later on, It has been proved that this algorithm could be utilized to detect objects in a much faster speed and higher classification precision. Since then, this method has been applied frequently in object recognition applications. Nowadays ,based on OpenCV, people can use an application called “opencv_traincascade” to train a classifier[6]. We employed this efficient utility in most of our work.

A. Haar-like feature and its computation

Haar-like features (Figure 1) are image features constructed on a framework for robust and extremely rapid object detection[5]. Typically it can be defined as the difference of the sum of pixels of areas inside the rectangles with any size. In the Figure 1, A, B, C, D are four different kinds of rectangle selection methods. In order to improve computing speed, Viola and Jones introduced integral images. In this algorithm,the

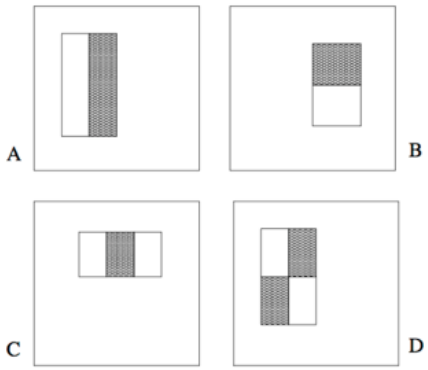


Fig. 1: Haar like features

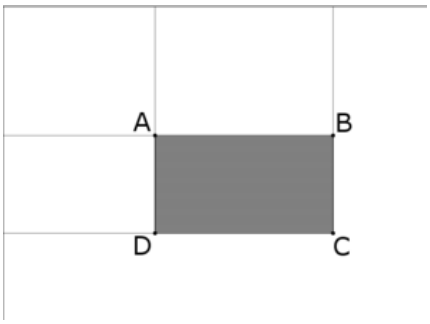


Fig. 2: sum of rectangular area

value at any point(x,y) can be defined as as sum of all the pixels above and to the left of (x,y).

Hence,the sum of particular region could be calculated by the equation below:
$sum = I(C) + I(A) - I(B) - I(D)$

B. AdaBoost

AdaBoost learning algorithm was proposed by Yoav and Robert as a general method for generating a strong classifier out of a set of weak classifier[7].The output of each weak classifier is assigned a weight and combined into a sum. It is adaptive in the sense that misclassified samples from previous classifiers would be focused more in the next classifier.

III. Data collection

In order to train a classifier properly, it is indispensable to have a large number of different traffic sign samples. However, there are limited traffic sign dataset online and most of them are European traffic sign dataset. American traffic signs are very different from those in Europe. Take prohibitory signs as an example (Figure 2). In contrast to the circular European prohibitory signs, American signs have a rectangular shape and express numerals and lettering on a white ground with a black border. Hence, we have to create a traffic sign dataset by ourselves. A vehicle mounting a camera drove around Pittsburgh to capture different kinds of traffic signs. Within 2 years, 7 million pictures are collected. We use some of these pictures to create our own dataset.

	Prohibitory signs			Warning signs		
	background	Speed limit	Weight limit	background	Signal ahead	Two way traffic
American		SPEED LIMIT 20	WEIGHT LIMIT 10 TONS			
European		30	6t			

Fig. 3: Differences of traffic signs between EU and US

Fig. 4: image collection process

IV. Preparation for the training

After dataset was built, another thing we should do before stepping into training process is creating positive samples. Based on OpenCV, an application called “opencv_createsamples” could realize the function. This application can be used to create many positive samples (traffic sign samples) from one image or several images as long as the original image and the annotation which tells the size and location of the traffic signs in the image are provided[8].Also, to obtain the samples, some pths and parameters like -img, -bg and -info keys should be specified.

OpenCV also provides an application called “opencv_traincascade.exe”, which can generate a classifier from Haar-like feature extraction and AdaBoost learning algorithm. We just need to call the function by Command Prompt. When using this application, some parameters should be customized according to different goals. Below are some important ones:

V. Trials and analysis

To find out the best parameters that could feed the traffic sign training, we use small number (47) of speed limit signs to have a test. In contrast,there is only one variable between two tests. The results are shown below.

W H	width and height of training samples(in pixels)
minHitRate	Minimal desired hit rate for each stage of the classifier.
maxFalseAlarmRate	Maximal desired false alarm rate for each stage of the classifier

Fig. 5: improtant training parameters

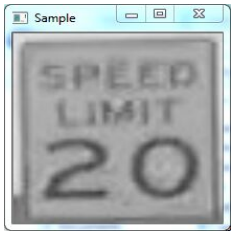


Fig. 6: 16*16 speed limit sign

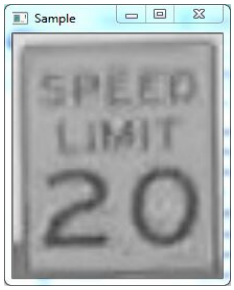


Fig. 7: 16*20 speed limit sign

	w	h	detect	missed	amount
data1	**16**	**16**	**37**	**10**	**47**
data2	**16**	**20**	**38**	**9**	**47**
data3	**20**	**20**	**22**	**25**	**47**

Fig. 8: speed limit sign trial 1

A.

Changing the size of positive samples by altering the training parameters w and h in OpenCV_traincascade while keeping minHitRate and maxFalseAlarmRate constant at the same time (usually we alter the w, which means the width of positive samples, and h, which means the height of positive samples)

When setting of positive samples 16*20 , the most signs could be detected. In fact, the ratio of 16 to 20 is similar to the actual ratio of a speed limit sign. Consequently, when we resize the positive samples, the actual size of the samples should be considered.

B.

altering minHitRate or maxFalseAlarmRate while keeping the ratio of w to h constant

The minHitRate and maxFalseAlarm Rate have great influence on the test results. And we find when we set minHitRate with maxFalseAlarmRate to 0.998 with 0.6, the most samples can be detected.

When keeping minHitRate the same and changing maxFalseAlarmRate(data1, data2, data3), the result indicates that the more signs could be recognized if we set maxFalseAlarmRate as 0.6.Higher or lower value does not necessarily ensure a better or worse result.

When keeping maxFalseAlarmRate the same and changing minHitRate(data2, data4, data5), the result indicates that the higher minHitRate is, more signs could be detected. In all, 0.998 for minHitRate and 0.6 for maxFalseAlarmRate is

	minHitRate	maxFalseAlarmRate	detect	miss
data1	**0.995**	**0.5**	**22**	**25**
data2	**0.995**	**0.6**	**28**	**19**
data3	**0.995**	**0.7**	**12**	**35**
data4	**0.996**	**0.6**	**30**	**17**
data5	**0.998**	**0.6**	**40**	**7**

Fig. 9: speed limit sign trial 2

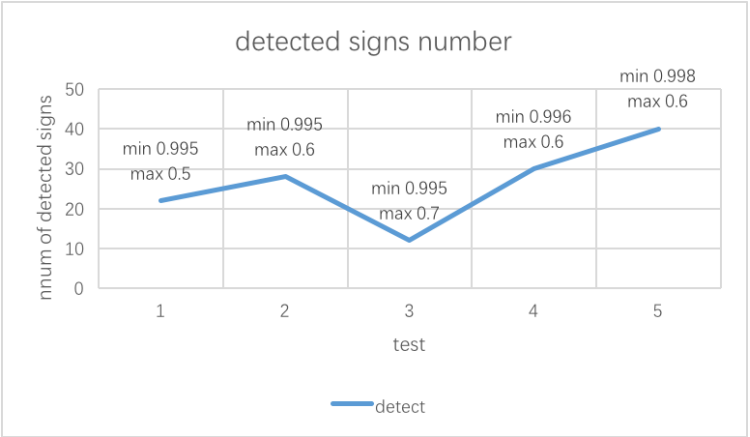


Fig. 10: results from different parameters

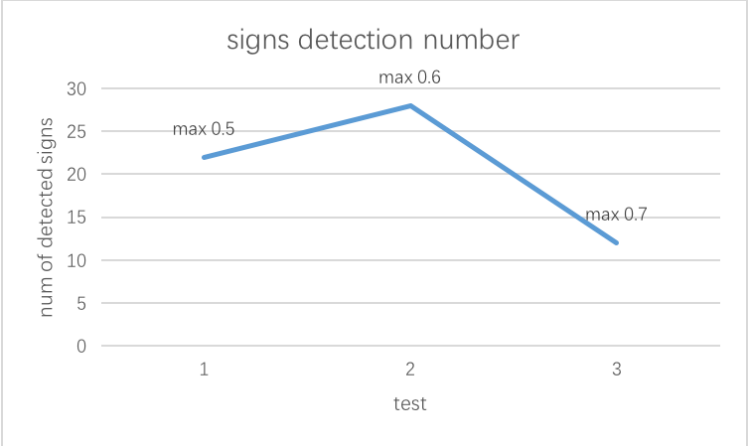


Fig. 11: altering maxFalseAlarmRate

recommended if you want more signs to be detected.

The samples we use for training above share similar features. However, traffic signs in America are very different in size, shape and color. We are not sure whether the classifier trained by different signs could perform well. Hence, to verify the performance, we conducted a trial in which training samples are selected both from speed limit signs and stop signs. We choose 1315 speed limit sign samples and 1315 stop signs

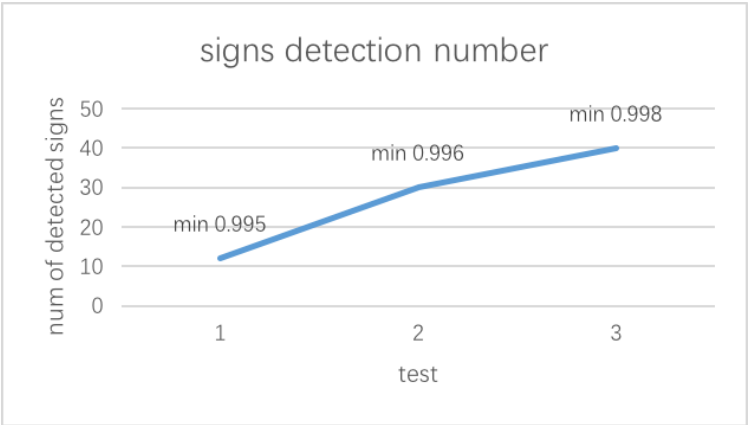


Fig. 12: altering minHitRate

Fig. 13: different signs in the US

	stop signs		speed limit signs	
	detected	missed	detected	missed
stop signs only	27	2	x	x
speed limit signs only	x	x	52	1
both(16×16)	27	2	34	19
both(16×20)	28	1	48	5

Fig. 14: speed limit sign and stop sign mixed trial

and train them separately and together. Below is the result:

Considering speed limit signs and stop signs have different width-to-height ratio, we use 16*16 and 16*20 in our test, which are more similar to that of stop signs and speed limit signs separately. When we set the ratio of samples as 16 to 16, the result of speed limit sign is not very good, but it improved when the ratio is set as 16 to 20, which is similar to the actual shape of a speed limit sign. A classifier with different kinds of samples could have a reasonable performance toward two different signs. However, we did not conducted more rigorous trial to see if the performance of classifier would be worse if more different feature-based samples are selected into the training set.

VI. Creating samples from synthetic images

Some traffic signs are very rare, so we cannot find enough samples. In this case, we should create samples on our own. . We start from an ideal synthetic image of the traffic sign. From this image many more images are created by random roation around all three axes. Then, white noise is added to the intensities of the foreground. Finally, the obtained images are placed onto a randomly selected background from the background description file, resized to the desired size specified by -w and -h and stored to a data base. (the vec-file, specified by the -vec command line option).

A. alter the color property of images

In an image processing context, the histogram of an image normally refers to a histogram of the pixel intensity values. This histogram is a graph showing the number of pixels

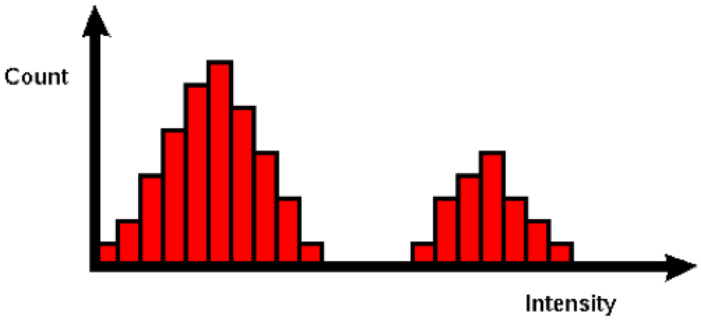


Fig. 15: piexl intensity histogram

in an image at each different intensity value found in that image. Histograms can also be taken of color images — either individual histograms of red, green and blue channels can be taken.

By adjusting its properties like exposure, contrast and saturation, we create more images and they look like those in different environment. These can be observed in the histogram of the images. We can achieve this with photoshop, to make more images, you can also use openCV to create your own program.

B. trials and results

The picture below showed ten different image samples created from one synthetic image by employing the method mentioned above.

In our test, we choose five kinds of synthetic images: “no right turn”, “signal ahead”, “do not enter”, “added lane” and “speed limit 25” as positive samples. In the test1, we create 5000 positive samples only from 5 original synthetic images by employing opencv_createsamples. In contrast, we also conducted tset2. By changing the color property of each original image, we create 10 pictures for one kind. Then 1000 positive samples are generated from every category. Combine samples in five categories into one dataset, we could obtain positive samples whose amount number could be up to 5000.

The trial result indicates that, after applying color property-altering method, basically detection rate for each kind of traffic sign has been improved more or less. Considering another fact that testing dataset contains some images in which traffic signs are somewhat tiny and vague, the real detection result could be better than the statistics given above. Hence, we can use synthetic images to solve the problem of lack of samples.

VII. Bootstrap training

In order to further improve the performance of the classifier, more real samples need to be added to the training set. By comparing the intensity histogram of synthetic images and real images, we can find that most value of synthetic image intensity count is zero, while most of real images are not. Since “Haar-like” feature is based on the sum of pixels in a certain area, we can conclude that the features of synthetic images

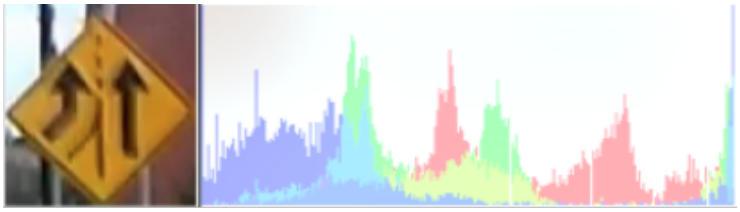


Fig. 16: original

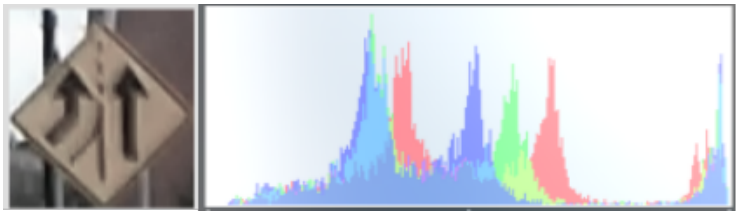


Fig. 17: altering saturation

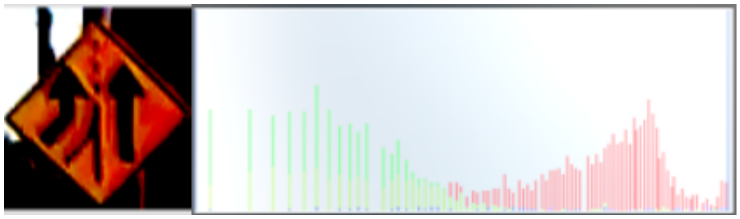


Fig. 18: altering contrast

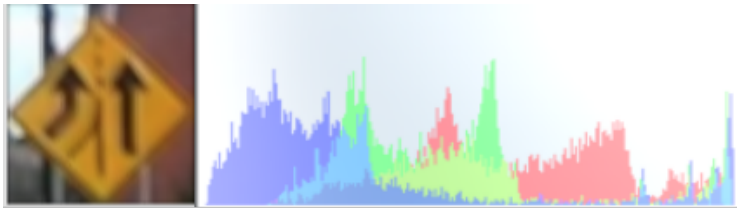


Fig. 19: altering sharpness

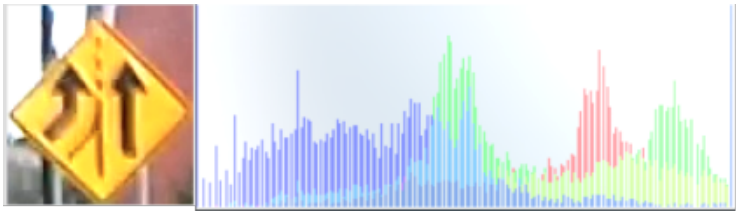


Fig. 20: altering exposure

Fig. 21: orgiginal image

Fig. 22: 5 images creating by changing color property

Fig. 23: 5 images creating by changing color property

Fig. 24: samples on the negative background

Fig. 25: cropped samples

are limited, while real images contain more. So if we add some real pictures to the training dataset created with synthetic images, more information will the classifier contains. We use synthetic images with boost training in the following series of test. Firstly, 3000 samples are created from 3 synthetic speed limit signs to train a classifier. Then, we use the classifier to detect 3213 speed limit sign testing samples. Afterwards, we crop out all testing results and classify them into true positive samples(traffic signs which are detected by the classifier) and false positive samples(samples which are considered traffic signs by classifier even though they are not. Finally, we added these samples to the training set separatly and retrain a new classifier. The training process will be stopped If the result could meet our requirement, or we will repeat this process.

The types and parameters of each test are listed below:
When we use synthetic images only, the recall rate is 93.088% while precision rate is 46.785%. After we add all true positive samples to the positive training dataset, the recall rate increased to 98.973% while precision rate decreased to 26.452%.
Since the precision rate is too low, which means there are many mistakes when we test an image. So we add some false positive samples to lower the number of mistakes. Considering the value of minHitRate has a great influence on precision rate based on our prior test. We set it 0.5 and 0.6 while keep other conditions the same to have a test. In test3, precision rate rise to 97.85%, which is very high. And recall rate is 93.50%, which does not improve too much. And in test4, when minHitRate is set 0.6, we find that the recall rate rise to 97.14%, but precision

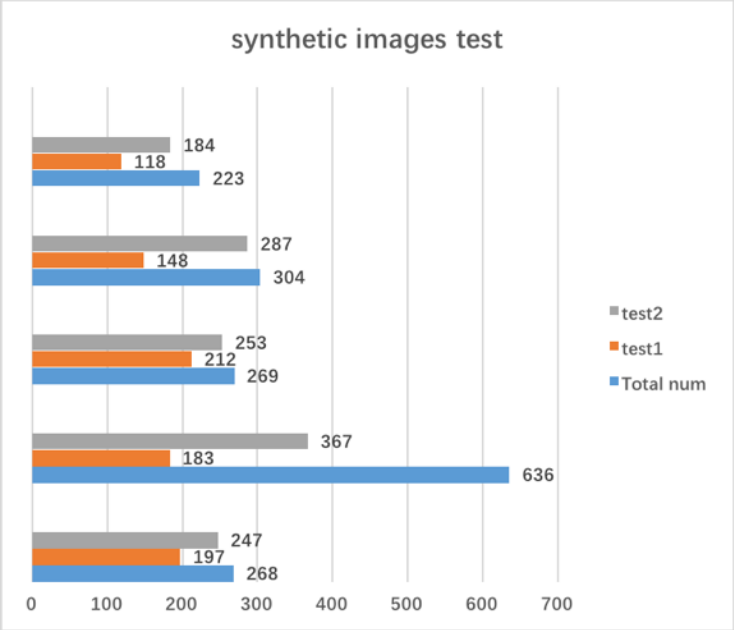


Fig. 26: synthetic image test1

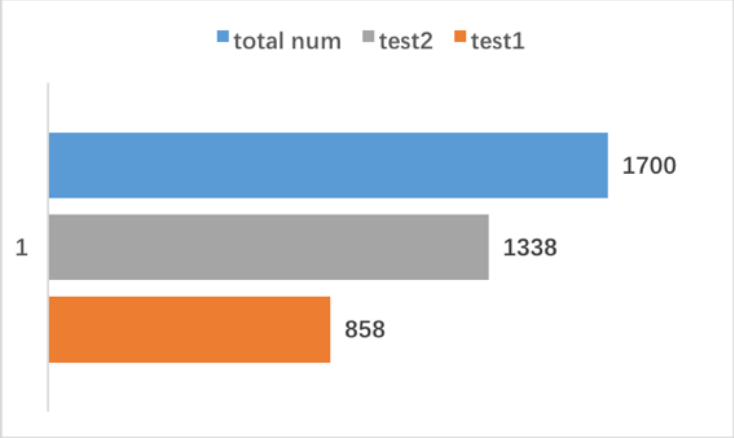


Fig. 27: synthetic image test2

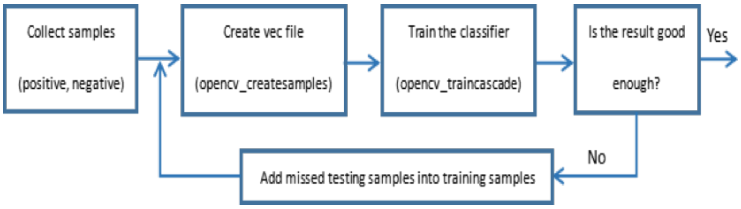


Fig. 28: bootstrapping block diagram

Fig. 29: test exmple

Fig. 30: test example

Positive samples	detected	precision	recall
Test1	2990	46.785%	93.088%
Test2	3179	26.452%	98.973%
Test3	3004	97.850%	93.524%
Test4	3121	39.122%	97.167%

Fig. 31: bootstrapping trial result

rate is 39.12%, lower compared with that in test1.
In conclusion, we can improve the performance of the classifier by boost training. In detail, more true positive samples added to the training dataset, higher will recall rate be. And with more false positive samples added to negative samples, some mistakes in detection will be reduced. And by adjusting the value of minHitRate between 0.5 and 0.6, we can get a result which get a balance between precision rate and recall rate as we want.

VIII. Yellow Sign Detection

Besides, we also conducted yellow sign detection based on the similar method applied in speed limit sign detection. In these trials, we select 6 kinds of yellow signs and utilize color property alternation strategy to generate original positive samples. Then 2400 positive samples are created, together with

	pos	neg	minhitrate
test1	3000	500	0.6
test2	5990	500	0.6
test3	5000	1800	0.5
test4	5000	1800	0.6
test1	pos:syn;neg:neg		
test2	pos:syn+real;neg:neg		
test3	pos:syn+real;neg:neg+false pos		
test4	pos:syn+real;neg:neg+false pos		
annotation:syn=synthetic img;real=real img;false pos=false positive img			

Fig. 32: parameters of bootstrapping trial

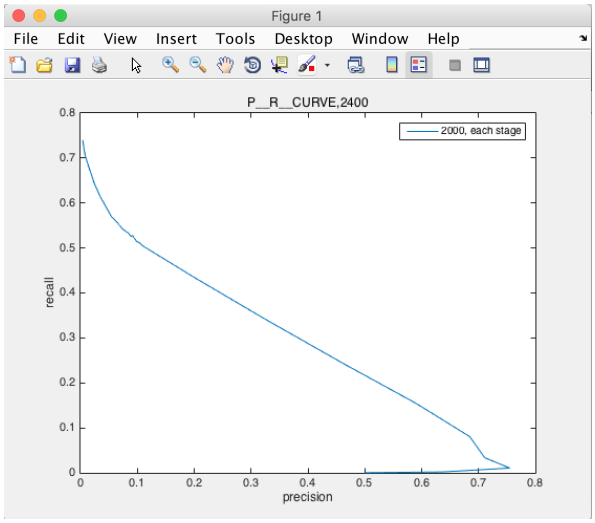


Fig. 33: the results of yellow sign before bootstrapping

training and bootstrapping. In the bootstrapping process, we both select true positive signs and false positive signs into positive sample and negative sample training sets. In order to decrease calculation time and avoid overfitting, we try to select some of signs which are different and contain more features(no matter positive or negative).

This time we use the famous "Pascal VOC challenge" to evaluate our test results instead of calculating manually. The results before and after bootstrapping are shown as P-R curve plot.

P means precision,which is the ratio of true positive samples to all the positive samples detected by the classifier.

R means recall,which is the ratio of true positive samples to all the positive samples in the test.

The results show that before bootstrapping, the highest recall rate is nearly 80 percent, and precision rate can be up to 75 percent. After bootstrapping, the highest recall rate rise up to over 90 percent, while the highest precision rate arrived at 1. Both overall and the highest recall rate are both improved. It can be deduced that bootstrapping could improve the recall

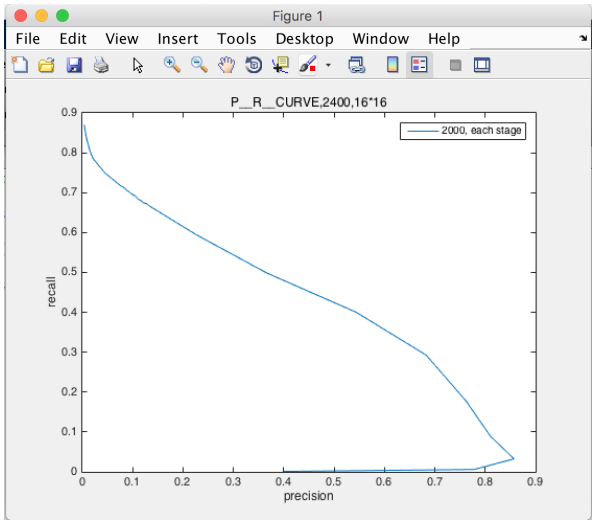


Fig. 34: the results of yellow sign after bootstrapping

rate of a classifier in an efficient way as long as the signs, which are added into the training set, are representative to the samples.

IX. Conclusion

The classifier's performance depends on the shape of test object, sample quality, training parameters. Based on "Haar-like" feature extraction and AdaBoost algorithm, an open source application, provided on openCV platform, could be utilized to train a classifier to detect particular objects. Upon small number trial, "Haar+AdaBoost" method could work properly toward traffic sign recognition. By altering the color property of samples and applying boost training, the performance of the classifier could be improved. A method that a huge classifier generated by different small well-performed classifier is gradually formed. For the future work, several more rigorous trials would be conducted through "HOG+SVM" method to verify the further performance of the classifier.

Acknowledgments

Special Thanks to our mentor Christoph Mertz who provide generous support and academic guidance to our program. Gratitude to Robotics Institute, Carnegie Mellon University for holding this wonderful research program.

References

[1] Andreas mogelmose, Dongran Liu, Mohan Trivedi, Life Fellow. *Detection of US Traffic Signs*, IEEE Transactions on Intelligent Transportation system, 2015.

[2] Christoph Mertz, Srivastan Varadharajan, Sobhagya Jose,Karan Sharma and Jina Wang. *road distress monitoring with smartphones*, Proceeding of ITS world Congress, September, 2014.

[3] Radu Timofte. Karel Zimmermann. Luc Van Gool. *Multi-view traffic sign detection,recognition, and 3D locoalisation*, Machine Vision and Application manuscript. November 2011.

RISS | ROBOTICS INSTITUTE SUMMER SCHOLARS

Volume 4 Fall 2016