Wire Detection, Reconstruction, and Avoidance for Unmanned Aerial Vehicles

Ratnesh Madaan

CMU-RI-TR-18-61
August 2018

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Abstract

Thin objects, such as wires and power lines are one of the most challenging obstacles to detect and avoid for UAVs, and are a cause of numerous accidents each year. This thesis makes contributions in three areas of this domain: wire segmentation, reconstruction, and avoidance.

Pixelwise wire detection can be framed as a *binary semantic segmentation* task. Due to the lack of a labeled dataset for this task, we generate one by rendering photorealistic wires from synthetic models and compositing them into frames from publicly available flight videos. We show that dilated convolutional networks trained on this synthetic dataset in conjunction with a few real images perform with reasonable accuracy and speed on real-world data on a portable GPU.

Given the pixel-wise segmentations, we develop a method for 3D wire reconstruction. Our method is a *model-based multi-view algorithm*, which employs a minimal parameterization of wires as catenary curves. We propose a bundle adjustment-style framework to recover the model parameters using non-linear least squares optimization, while obviating the need to find pixel correspondences by using the distance transform as our loss function. In addition, we propose a *model-free voxel grid* method to reconstruct wires via a pose graph of disparity images, and briefly discuss the pros and cons of each method.

To close the sensing-planning loop for wire avoidance, we demonstrate a reactive, trajectory library-based planner coupled with our *model-free* reconstruction method in experiments with a real UAV.

Acknowledgments

I would like to begin by thanking my advisor, Basti, for giving me an opportunity to work on an important and interesting problem. Thanks for allowing to me to explore new ideas, while always ensuring I am always on track for my thesis. A big thanks goes out to Daniel Maturana, Geetesh Dubey, and Michael Kaess, for their critical technical contributions, feedback, and advice in the relevant chapters. I am grateful for Michael for the frequent meetings during the last semester here, I'll miss them. Also thanks to Sankalp Arora and Sanjiban Choudhury, for helpful advice and feedback during my time here. Thanks again to Daniel and Geetesh for the same.

Thanks to John Keller, Silvio Maeta, and Greg Armstrong for building the systems and help with experimentation over the two years. Thanks to Jaskaran Singh Grover for jumping in for a critical review of this thesis a night before its due. Thanks to Eric Westman for reviewing my approach now and then. Thanks to Nora Kazour, Rachel Burcin, and Sanae Minick for bringing me to CMU multiple times. Thanks to Robbie Paolini, Erol Sahin, Matt Mason, and Rachel again for bringing me to CMU via RISS 2015, and great mentorship over the summer. Thanks to my friends Tanmay Shankar and Sam Zeng for life outside work. Thanks Rohit Garg for the expensive coffee grabs and awesome conversation. Thanks Rogerio Bonatti for all things McKinsey.

We gratefully acknowledge Autel Robotics for supporting this work, under award number A018532.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

We are at an interesting inflection point in the trajectory of Unmanned Aerial Vehicle (UAV) research and industry. The essential and critical modules - state estimation, control, and motion planning, have been shown to work reliably in multiple publications and products. With the fundamentals of autonomous flight figured out, the focus of research is now shifting to making these algorithms and systems more reliable and robust, and also on integrating all the components together moving towards higher levels of autonomy. The industry is developing drone applications for various B2B verticals on platforms by manufacturers like DJI, and the promise of autonomous drone selfie camera is becoming a reality [Skydio, 2018]. This also means that the hard problems, the edge cases, which albeit did receive some amount of attention in the past, are now again being looked closely upon by both academia and industry. A few examples are :

- Thin obstacles [Zhou et al., 2017b]
- Translucent obstacles [Berger et al., 2017, Liu et al., 2017]
- Unexpected, dynamic obstacles flying towards the UAV [Poiesi and Cavallaro, 2016]

This thesis addresses an infamous, long-standing problem in the UAV community: detection, reconstruction and avoidance of wires and power lines, and corresponding application areas of automated powerline corridor inspection and management Tuomela and Brennan [1980a,b], Fontana et al. [1998], Kasturi and Camps [2002], Australian Transport Safety Bureau [2006], McLaughlin [2006], Rubin and Zank [2007], Yan et al. [2007], Hrabar [2008], Nagaraj and Chopra [2008], Scherer et al. [2008], Candamo et al. [2009], De Voogt et al. [2009], Moore and Tedrake [2009], Sanders-Reed et al. [2009], Law et al. [2010], Li et al. [2010], Jwa and Sohn [2012], Lau [2012], Pagnano et al. [2013], Song and Li [2014], Pouliot et al. [2015], Ramasamy et al. [2016], Hirons [2017], Flight Safety Australia [2017], McNally [2018]. Thin wires and similar objects like power lines, cables, ropes and fences are one of the toughest obstacles to detect for autonomous flying vehicles, and are a cause of numerous accidents each year Australian Transport Safety Bureau [2006], Nagaraj and Chopra [2008], De Voogt et al. [2009], Lau [2012], Tuomela and Brennan [1980a,b], Flight Safety Australia [2017]. They can be especially hard to detect in cases where the background is cluttered with similar looking edges, when the contrast is low, or when they are of barely visible thickness [Flight Safety Australia, 2017, Kasturi and Camps, 2002, Candamo et al., 2009, Yan et al., 2007, Li et al., 2010, Sanders-Reed et al., 2009, Song and Li, 2014]. Power line corridor inspection is another area of potentially widespread application of wire detection capabilities, and leveraging UAVs for this task can save a lot of money, time, and help avoid dangerous manual labor done by linemen. Apart from UAVs, there have been recent reports which document fatal injuries and deaths caused by hitting barbed wire fences while riding ATVs, and dirt and mountain bikes [The Telegraph, 2007, The Local, 2011, Daily Mail, 2014, Metro News, 2016].

Wire strike accidents have been a constant problem through the years. Apart from work focusing on detecting and avoiding wires, there have also been patents on a more heads-on approach, such as installing a wire cutter aboard helicopters and fixed wing aircrafts [Law et al., 2010, Hirons, 2017], and attaching visible markers on wires with UAVs [McNally, 2018]. To provide a more realistic angle to the issue of wire strikes, we now review a few key points from two reports Flight Safety Australia [2017], Australian Transport Safety Bureau [2006] which document statistics and interesting anecdotes shared by pilots.

- Flight Safety Australia [2017]

  Pilot error: Interestingly, roughly 52 per cent of wire strike accidents are caused by human error by experienced pilots with more than 5000 hours on their belt. Roughly 40 per cent of accidents involved hitting a wire which the crew was already aware of. 60 per cent of wire strikes result in a fatality.

  Visibility issues: Atmospheric conditions, cockpit ergonomics, dirt or scratches on cockpit windows, viewing angle, sun position, visual illusions, pilot scanning abilities and visual acuity, flight deck workload, and camouflaging effect of nearby vegetation were noted down as the main casues.
  Wire are also known to camouflage with the background. Older wires change in color, which causes a pilot to not register them immediately. For example, copper wires oxidize to a greenish colour and camouflage with vegetation, and also with electricity transmission towers which are deliberately painted green in certain areas to blend in with the environment.

  Due to lighting, a wire which is perfectly visible from a particular direction can be completely invisible from the opposite. They also note a few optical illusions like (1) *High-wire illusion:* When looking at two parallel wires from 200 meters away or more, the wire with the greater altitude appears to be located further away from the drone than it is in reality; (2)*Phantom-line illusion:* A wire running parallel to multiple other wires can become camouflaged with the environment causing a pilot to lose their vigilance.

- Australian Transport Safety Bureau [2006]

  This report reviews Australian Transport Safety Bureau's accident and incident database over a decade, from 1994 to 2004. They identify 119 wire-strike accidents, and 98 wire-strike incidents between 1994 and 2004. The rate of wire-strike accidents per 100,000 hours flown ranged from around 0.9 in 1997 and 1998, to 0.1 in 2003. The numbers a downward indicated trend from 1998 to 2003, but in 2004 the rate

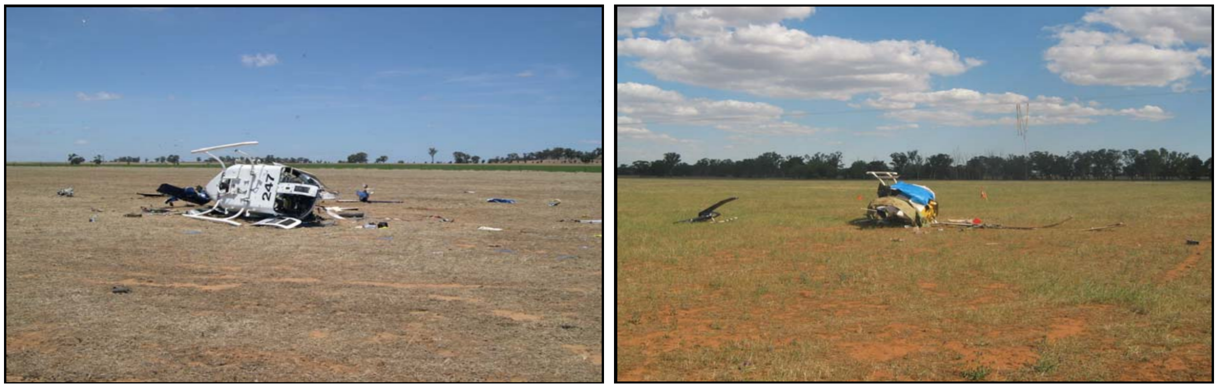
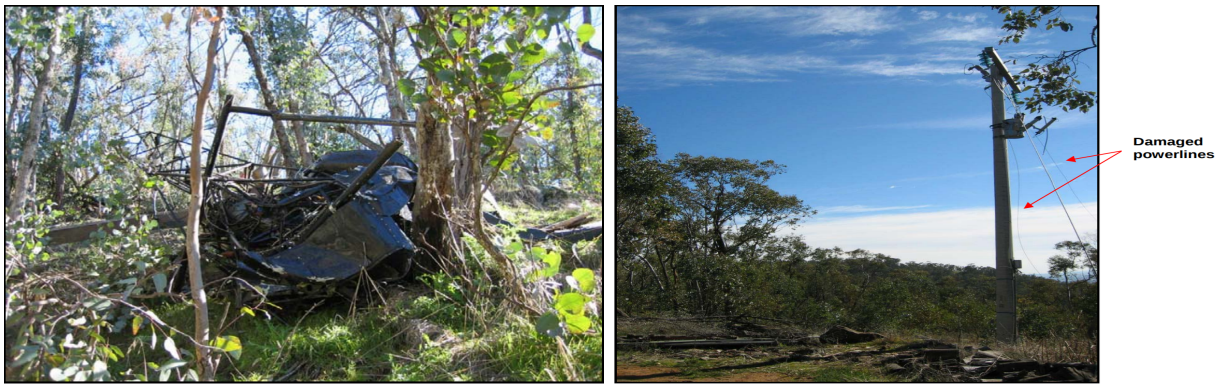

(a) Dunedoo on 22 November 2004 (b) Forbes on 31 October 2004

Figure 1.1: Bell 206B helicopter after striking powerlines during locust control campaigns at different locations and date. Image from Australian Transport Safety Bureau [2006].



(a) The remains of the helicopter (b) Damaged powerlines

Figure 1.2: Bell 47G-3B-1 Soloy helicopter after striking powerlines near Wodonga on 19 June 2004. Image from Australian Transport Safety Bureau [2006].

increased to 0.7. There report 169 people involved in the 119 wire-strike accidents, where the occupant received varying degrees of injury in almost 67 percent of the cases. There were 45 people fatally injured, 22 seriously injured, and 42 who received minor injuries.

Aerial agriculture operations accounted for 62 per cent, that is 74 wire-strike accidents. Fixed-wings were involved in 57 per cent of wire-strike accidents and rotary-wings were involved in the remainer of 43 per cent. Given that fixed-wing aircraft out number rotary-wing aircraft by seven to one in the Australian aviation industry, rotary-wing aircraft were over-represented in the data. This data imbalance indicates that rotary-wing vehicles are involved in higher risk applications.

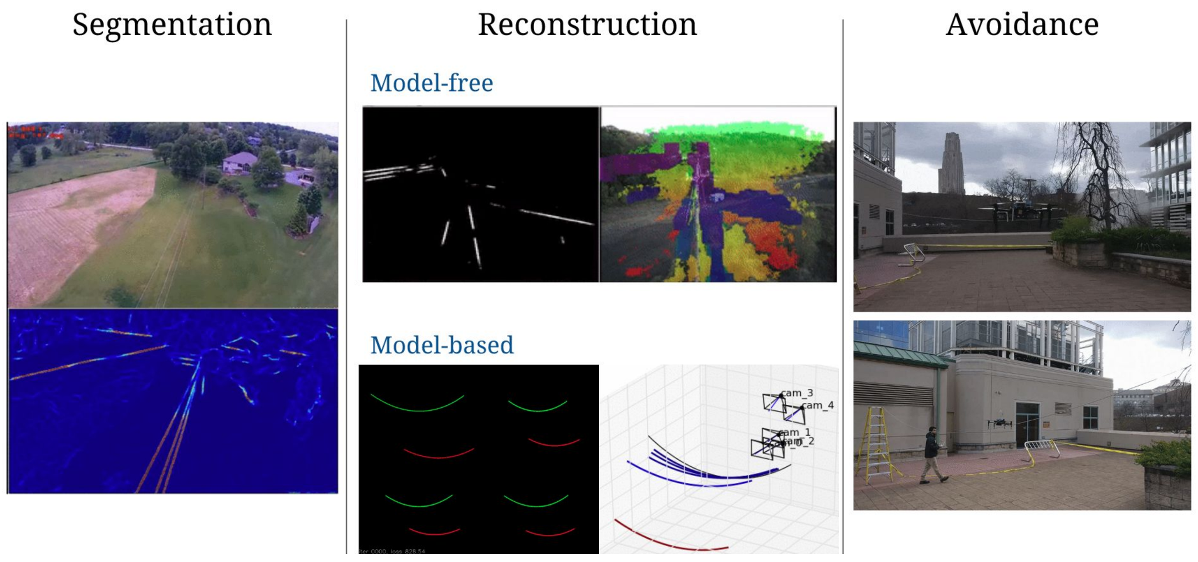


Figure 1.3: Thesis outline

1.2 Thesis Overview

Generally, one could use lidar, radar, infrared, electromagnetic sensors or cameras to detect wires and power lines McLaughlin [2006], Sanders-Reed et al. [2009], Luque-Vega et al. [2014], Matikainen et al. [2016]. While lidar has been shown to work in multiple publications and products [McLaughlin, 2006, Scherer et al., 2008, Jwa and Sohn, 2012, Cheng et al., 2014, Ramasamy et al., 2016], for small UAVs it is infeasible due to the high payload and cost of such sensors. To see thin wires which are more than a few meters early enough so that one has enough time to avoid them, which means that the drone should be moving slowly enough to obtain a really dense point cloud. Cameras on the other hand are lightweight, cheap, and can see wires upto long distances.

We propose an approach to detect, reconstruct, and avoid wires using a monocular camera as our sensor, under the assumption of a known state estimate. Another assumption is that the RGB image from the camera itself has enough information about the wire that they can be segmented from the image.

We now delineate our contributions, along with the organization of this thesis

- Wire Segmentation from Monocular Images (Chapter 2)
  In this chapter, we present an approach to segment wires from monocular images in a per-pixel wise fashion. Pixelwise wire detection can be framed as a binary semantic segmentationtask. Due to the lack of a labeled dataset for this task, we generate one by rendering photorealistic wires from synthetic models and compositing them into frames from publicly available flight videos. We show that dilated convolutional networks trained on this synthetic dataset in conjunction with a few real images

perform with reasonable accuracy and speed on real-world data on a portable GPU. This work was published in Madaan et al. [2017].

- Multi-view Reconstruction of Wires using a Catenary Model (Chapter 3)
  Given the pixel-wise segmentations, we develop a method for 3D wire reconstruction. Our method is a *model-based multi-view algorithm*, which employs a minimal parameterization of wires as catenary curves. We propose a bundle adjustment-style framework to recover the model parameters using non-linear least squares optimization, while obviating the need to find pixel correspondences by using the distance transform as our loss function.
- Model-free Mapping and Reactive Avoidance (Chapter 4)
  In this section, propose a model-free, implicit voxel-grid mapping method which uses a pose graphof disparity images to reconstruct both wires from a monocular segmentations from Chapter 2, and generic obstacles detected from stereo images. To close the sensing-planning loop for wire avoidance, we demonstrate a reactive, trajectory library-based planner coupled with the mapping framework in real-world experiments. This work will be published in [Dubey et al., 2018], and relies heavily on previous work by Dubey et al. [2017], Dubey [2017].

To get the most out of this thesis, we encourage the reader to visit the corresponding thesis webpage, which has links to the thesis defence, associated publications and code, and animation and videos which supplement the figures provided in this document.

Chapter 2

Wire Segmentation from Monocular Images

2.1 Introduction

Thin wires and similar objects like power lines, cables, ropes and fences are one of the toughest obstacles to detect for autonomous flying vehicles, and are a cause of numerous accidents each year. They can be especially hard to detect in cases where the background is cluttered with similar looking edges, when the contrast is low, or when they are of barely visible thickness. Power line corridor inspection is another area of potentially widespread application of wire detection capabilities, and leveraging UAVs for this task can save a lot of money, time, and help avoid dangerous manual labor done by linemen. Apart from UAVs, there have been recent reports which document fatal injuries and deaths caused by hitting barbed wire fences while riding ATVs, and dirt and mountain bikes [**?**]. Generally, one could use lidar, infrared, electromagnetic sensors or cameras to detect wires and power lines. Out of these, a monocular camera is the cheapest and most lightweight sensor, and considering the recent advances in deep learning on images, we use it as our perception sensor. Apart from a good detection rate, real time performance on a portable GPU like the NVIDIA Jetson TX2 with a decent resolution like 480x640 is critical to ensure that wires are still visible in the input image, and that the drone has enough time for potentially avoiding them.

Previous work uses strong priors on the nature of power lines - assuming they are straight lines, have the highest or lowest intensity, appear with a fixed number in an image, have a gaussian intensity distribution, are the longest line segments, are parallel to each other, or can be approximated by quadratic polynomials [Kasturi and Camps, 2002, Candamo et al., 2009, Yan et al., 2007, Li et al., 2010, Sanders-Reed et al., 2009, Song and Li, 2014]. These works first gather local criteria for potential wire pixels by using an edge detection algorithm, filter them using the heuristics mentioned above, and then gather global criteria via variants of the Hough or Radon transform [Kasturi and Camps, 2002, Candamo et al., 2009, Yan et al., 2007], clustering in space of orientations [Li et al., 2010] and graph cut models [Song and Li, 2014]. These approaches demand a lot of parameter tuning, work well only in specific scenarios, and are prone to false positives and negatives since the distinction between wires and other line-like objects in the image does not always obey the appearance based assumptions and priors.

Another issue with wire detection is the unavailability of a sufficiently large public dataset with pixel wise annotations. The only existing dataset with a decent number of labeled images is provided by [Candamo et al., 2009], which we refer to as the USF dataset. We use it for evaluation of our models, as done by [Song and Li, 2014] as well.

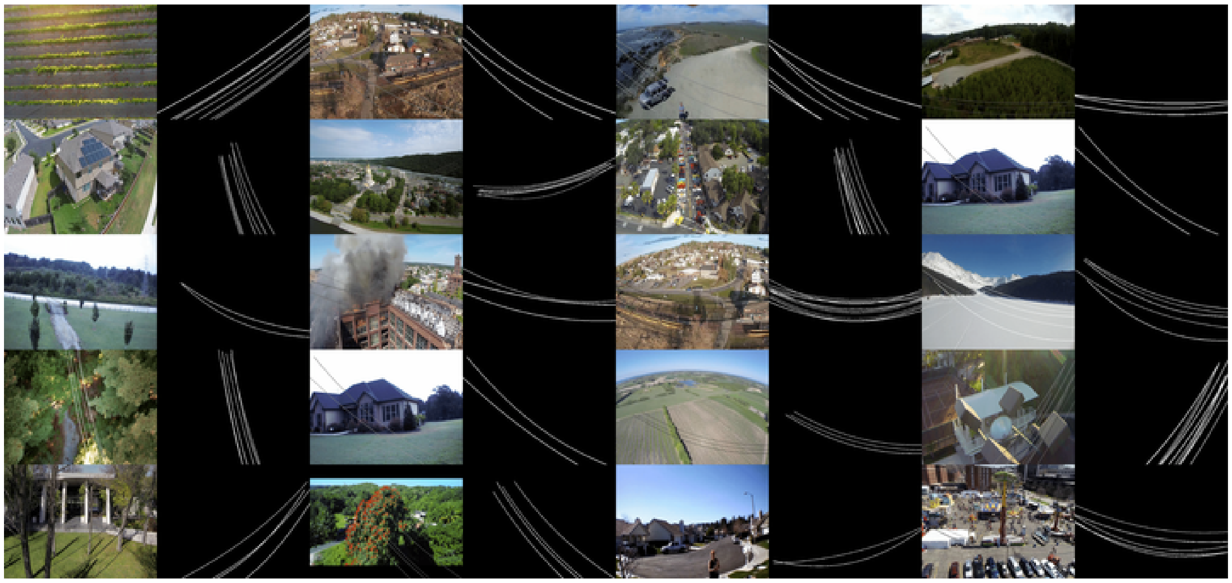


Figure 2.1: A few samples from our synthetically generated dataset along with ground truth labels of wires. Video examples available at this link and our project page.

We address the former of the aforementioned issues by investigating the effectiveness of using convolutional neural networks (convnets) for our task. Recent advances in semantic segmentation and edge detection using convnets are an indication of an end to end wire detection system omitting hand-engineered features and parameter tuning, which can also potentially generalize to different weather and lighting conditions depending on the amounts of annotated data available [Long et al., 2015, Bertasius et al., 2015, Xie and Tu, 2015]. To meet our objectives of real time performance on the TX2 and reasonable detection accuracy, we perform a grid search over 30 architectures chosen based on our intuition. We investigate the effect of filter rarefaction [Yu and Koltun, 2015] by systematically adding dilated convolutional layers with increasing amounts of sparsity as shown in Table 2.1 and 2.2. We show that the top 5 models from our grid search outperform various baselines like E-Net [Paszke et al., 2016], FCN-8 and FCN-16s [Long et al., 2015], SegNet [Badrinarayanan et al., 2017], and non-deep baseline of [Candamo et al., 2009] as shown in Fig. 2.4 and Table 2.4 across multiple accuracy metrics and inference speeds on the Jetson TX2.

To address the latter issue of the unavailability of a large dataset with pixel wise annotations, we generate synthetic data by rendering wires in 3D as both straight lines and in their natural catenary curve shape, using the POV-Ray[POV-Ray, 2004] ray-tracing engine with varying textures, numbers, orientations, positions, lengths, and light source, and superimpose them on 67702 frames obtained from 154 flight videos collected from the internet. A few samples from our dataset can be seen in Figure 2.1 and in the video here. Although training on synthetic data alone is not enough to match the performance of models which are finetuned or trained from scratch on the USF dataset, we observe that synthetic data alone gave pretty accurate qualitative detections on a few unlabeled flight videos we tested on as shown in Figure 2.5. More results are available at our project page.

Further, for wire detection and avoidance, false negatives (undetected wires) are much more harmful than false positives. We adapt the line of thought of finding local evidence and then refining it via heuristics or global criteria used by previous works, to convnets by explicitly providing local evidence of 'wiryness' in the form of extra channels to the network input. One would expect that doing this would make the network's task easier as now it has to filter out the non-wire edges and line segments and return pixels that were actually wires. However, in the case of training from scratch, or finetuning on the USF dataset after pretraining on synthetic data, we find that the local primitive information does not have much effect in the performance of the network. On the other hand, we find out that adding this information of local criteria does help in the case of models trained on only synthetic data, as shown in Table 2.2.

In summary, the contributions of this chapter are:

- Systematic investigation of the effectiveness of convnets for wire detection by a grid search.
- Large scale generation of a synthetic dataset of wires and demonstrating its effectiveness.
- Investigation of the effect of explicitly providing local evidence of wiryness.
- Demonstrating real time performance on the NVIDIA Jetson TX2 while exceeding previous approaches both in terms of detection accuracy and inference speed.

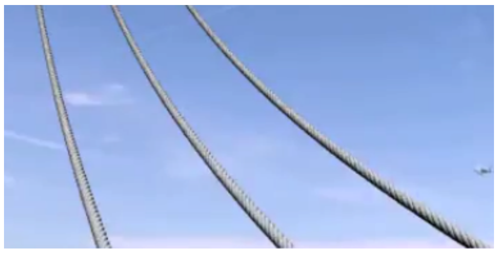


Figure 2.2: Close up of synthetic wires rendered using POV-Ray [POV-Ray, 2004].

2.2 Related Work

2.2.1 Wire detection using traditional computer vision

One of the earliest works in wire detection is from [Kasturi and Camps, 2002], who extract an edgemap using Steger's algorithm [Steger, 1998], followed by a thresholded Hough

Table 2.1: The parameter space over which we perform a grid search. Each context module from the 2nd column was stacked over each front-end module from the first column to get a total of 30 models. All layers use 3*3 convolutions. Each layer is separted by a '-' sign. 'k' refers to the number of channels in each layer. 'p' refers to pooling layers, 'd' refers to the dilation factor.

Front-end Modules		**Context Modules**	
Key	Architecture	Key	Architecture
f1	k64-k64-p2-k128-k128	c1	k2(none)
f2	k32-k32-k64-k64	c2	d1-d2-d1-k2
f3	k32-k32-k32-k32	c3	d1-d2-d4-d1-k2
f4	k32-k32-k64-k64-k64-k64	c4	d1-d2-d4-d8-d1-k2
f5	k32-k32-k32-k32-k32-k32	c5	d1-d2-d4-d8-d16-d1-k2
f6	k32-k32		

transform which rejects lines of short length. However, due to unavailability of ground truth, they evaluate their approach only on synthetically generated wires superimposed on real images.

[Candamo et al., 2009] find edges using the Canny detector and then weigh them proportionally according to their estimated motion found using optical flow, followed by morphological filtering and the windowed Hough transform. The resulting parameter space is then tracked using a line motion model. They also introduce the USF dataset and show that their approach performs better than [Kasturi and Camps, 2002]. We modify their temporal algorithm to a per-frame algorithm as explained later, and use it as a baseline.

[Song and Li, 2014] proposed a sequential local-to-global power line detection algorithm which can detect both straight and curved wires. In the local phase, a line segment pool is detected using Gaussian and first-order derivative of Gaussian filters, following which the line segments are grouped into whole lines using graph-cut models. They compare their method explicitly to [Kasturi and Camps, 2002] and their results indicate similar performance as [Candamo et al., 2009], which is one of our baselines.

2.2.2 Semantic segmentation and edge detection using deep learning

Fully Convolutional Networks(FCNs) [Long et al., 2015] proposed learned upsampling and skip layers for the task of semantic segmentation, while SegNet [Badrinarayanan et al., 2017] proposed an encoder-decoder modeled using pooling indices. For thin wires, FCNs and SegNet are intuitively suboptimal as crucial information is lost in pooling layers which becomes difficult to localize in the upsampling layers. E-Net [Paszke et al., 2016] develop a novel architecture by combining tricks from multiple works for real time semantic segmentation performance, but one of their guiding principles is aggresive downsampling again.

Table 2.2: Results of our grid search experiments. The strings in bold, blue text represent the front end architecture as explained in text. Each front end is appended with five different context modules as shown in the Col. 1. We conduct six experiments, all of which are evaluated on our test split of the USF dataset. Each experiment category is grouped with the same hue (Col. 2-3, 4-5, 6-7 respectively). Darker color implies better performance. Choice of colors is arbitrary. Col. 8-11 list the performance speeds on the NVIDIA TitanX Pascal and the Jetson TX2, with batch size 1 and input resolution of 480x640.

	Average Precision Scores (evaluation on USF test split)						**Performance**			
Trained On →	**Synthetic Data**		**Scratch on USF**		**Finetuned on USF**		**TX2**		**TitanX**	
Context Module ↓	**RGB**	**RGBLE**	**RGB**	**RGBLE**	**RGB**	**RGBLE**	**Time (ms)**	**Speed (fps)**	**Time (ms)**	**Speed (fps)**
Column 1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Front End Module →	**k32-k32**									
k2	0.23	0.30	0.45	0.45	0.50	0.47	37.7	26.6	3.2	309.6
d1-d2-d1-k2	0.33	0.35	0.63	0.58	0.61	0.59	115.7	8.7	8.0	124.7
d1-d2-d4-d1-k2	0.28	0.42	0.65	0.61	0.64	0.62	159.3	6.3	10.2	98.0
d1-d2-d4-d8-d1-k2	0.35	0.45	0.65	0.62	0.66	0.65	203.7	4.9	12.4	80.6
d1-d2-d4-d8-d16-d1-k2	0.36	0.49	0.70	0.64	0.70	0.69	249.4	4.0	14.6	68.4
	k32-k32-k32-k32									
k2	0.24	0.32	0.57	0.50	0.57	0.54	70.7	14.2	5.7	176.1
d1-d2-d1-k2	0.25	0.39	0.62	0.60	0.64	0.59	148.5	6.7	10.1	99.1
d1-d2-d4-d1-k2	0.28	0.40	0.63	0.62	0.66	0.63	192.7	5.2	11.9	83.9
d1-d2-d4-d8-d1-k2	0.33	0.42	0.69	0.62	0.68	0.66	236.6	4.2	13.8	72.6
d1-d2-d4-d8-d16-d1-k2	0.43	0.48	0.70	0.64	0.72	0.64	282.6	3.5	15.7	63.9
	k32-k32-k32-k32-k32-k32									
k2	0.18	0.35	0.63	0.53	0.61	0.57	104.0	9.6	8.2	121.7
d1-d2-d1-k2	0.30	0.42	0.61	0.58	0.64	0.59	181.5	5.5	13.2	76.1
d1-d2-d4-d1-k2	0.25	0.43	0.62	0.59	0.67	0.62	225.4	4.4	15.3	65.2
d1-d2-d4-d8-d1-k2	0.32	0.47	0.66	0.64	0.70	0.65	270.1	3.7	17.6	57.0
d1-d2-d4-d8-d16-d1-k2	0.03	0.47	0.68	0.66	0.47	0.65	315.0	3.2	19.8	50.6
	k32-k32-k64-k64									
k2	0.20	0.28	0.59	0.50	0.60	0.55	118.7	8.4	8.3	120.9
d1-d2-d1-k2	0.25	0.45	0.62	0.60	0.62	0.60	350.7	2.9	19.9	50.4
d1-d2-d4-d1-k2	0.29	0.38	0.65	0.62	0.68	0.62	496.2	2.0	26.3	38.1
d1-d2-d4-d8-d1-k2	0.28	0.45	0.66	0.64	0.66	0.64	641.7	1.6	32.8	30.5
d1-d2-d4-d8-d16-d1-k2	0.40	0.48	0.71	0.62	0.71	0.67	787.6	1.3	38.4	26.1
	k32-k32-k64-k64-k64-k64									
k2	0.27	0.39	0.63	0.57	0.61	0.58	206.6	4.8	13.4	74.8
d1-d2-d1-k2	0.29	0.42	0.62	0.60	0.67	0.59	439.2	2.3	24.7	40.5
d1-d2-d4-d1-k2	0.28	0.38	0.64	0.60	0.65	0.62	582.7	1.7	31.0	32.3
d1-d2-d4-d8-d1-k2	0.37	0.45	0.68	0.62	0.67	0.63	729.8	1.4	37.2	26.9
d1-d2-d4-d8-d16-d1-k2	0.42	0.49	0.68	0.63	0.70	0.63	877.7	1.1	43.9	22.8
	k64-k64-p2-k128-k128									
k2	0.26	0.35	0.65	0.60	0.66	0.60	136.0	7.4	8.1	124.1
d1-d2-d1-k2	0.30	0.34	0.66	0.63	0.72	0.66	279.8	3.6	17.1	58.5
d1-d2-d4-d1-k2	0.41	0.45	0.70	0.67	0.73	0.67	350.8	2.9	21.8	45.8
d1-d2-d4-d8-d1-k2	0.34	0.49	0.71	0.66	0.73	0.70	421.8	2.4	26.5	37.7
d1-d2-d4-d8-d16-d1-k2	0.36	0.47	0.72	0.64	0.75	0.68	493.2	2.0	31.2	32.0

2.3 Approach

2.3.1 The search for the right architecture

For a wire detection network running on a platform like the NVIDIA TX2 atop a small UAV, we desire low memory consumption and a fast inference time. Further, as wires are barely a few pixels wide, we do not want to lose relevant information by the encoder-decoder approaches [Long et al., 2015, Paszke et al., 2016, Badrinarayanan et al., 2017] in which one first downsamples to aggregate global evidence and then upsamples back up to localize the target object.

Dilated convolutional layers [Yu and Koltun, 2015] are a simple and effective way to gather context without reducing feature map size. Each layer is defined by a dilation factor of d, which correspond of $d - 1$ number of alternating zeros between the learnt elements, as visualized in [Yu and Koltun, 2015, van den Oord et al., 2016]. [Yu and Koltun, 2015] proposed appending a 'context module' comprising of 7 layers with increasing dilation factors (d for each layer following the series $\{1,1,2,4,8,16,1\}$) to existing segmentation networks (front-end modules), which boosted their performance significantly.

In order to investigate the effect of dilation on our task, we run a grid search over a finite set of front-end and context modules as summarized in Table 2.1. We now introduce a simple naming scheme to refer our models with. Each layer is separated by a '-' the number succeeding 'k' is equal to the number of channels, 'p' refers to a pooling layer and 'd' to a dilation layer. For all layers, we use 3*3 convolutions.

For front-end modules, we begin with trimming down VGG [Simonyan and Zisserman, 2014] to its first two blocks, which is the model f1 in Table 2.1. Next, we halve the number of channels across f1 and remove the pooling layer to get f2. Then, we halve the number of channels in the last two layers of f2 of this model to get f3. Further, to investigate the effect of depth we append 2 layers each to the f2 and f3, while maintaining the number of channels to get f4 and f5. We stop at 6 layers as previous work [Xie and Tu, 2015] found it to be enough to detect edges. Finally to check how well a simple two layer front-end performs, we add f6 to our list. The second column shows the range of the context modules we evaluated, starting with an empty context module (c1) and building up to c5 as in [Yu and Koltun, 2015]. Our first model is d1+d2+d1 due to the fact that a dilation factor of 1 is equivalent to standard convolution without holes. Note that there is always a k2 in the end as in the last layer, we do a softmax operation over two output classes (wire and not-wire).

By choosing a front-end (f1-f6) and appending it with a context module (c1-c5), we do a grid search over 30 architectures and do multiple experiments - training each model on synthetic data, real data, finetuning models trained on synthetic data, and finally evaluation on real data as shown in Table 2.2. We use a 50-50 train-test split of the USF dataset for the grid search. We then choose the top 5 models based on the accuracy and inference speed on the NVIDIA TX2 (bold text in Table 2.3) for comparison with various baselines.

2.3.2 Generating synthetic data

Due to the unavailaibility of a large dataset with annotations of wires, we render synthetic wires using the POV-Ray ray tracing engine [POV-Ray, 2004], and superimpose them on 67702 frames sampled from 154 flight videos collected from the internet. Figure 2.1 and this video shows some examples from our synthetic dataset.

Our dataset consists of wires occuring as both straight lines and catenary curves (the natural shape of a wire hanged at its ends under uniform gravity). We vary the wire sag, material properties, light source location, reflection parameters, camera angle, number of wires and the distance between them across the dataset to obtain a wide variety of configurations. A close up of the rendered wires can be seen in Figure 2.2.

2.3.3 Class Balancing Loss Function

As wire pixels occupy only a minisicule fraction of the total number of pixels in an image, it is important to account for class imbalance in the loss function, as well as in the evaluation metrics used. To that end, we use the image-level, class-balanced cross entropy loss function defined in [Xie and Tu, 2015]. For the USF dataset, we find that wire pixels account for only 4.9% of the total number of pixels, and the rest 95.1% are background pixels, and weigh the loss accordingly.

2.3.4 Explicitly providing local evidence of wires

Previous works extract local evidence of wiryness via edge maps obtained from standard edge detection algorithms, which is followed by filtering using domain knowledge, and finally gathering global evidence. We adapt this line of thought to convnets by appending the results of line segment and edge detection using the recently proposed CannyLines and CannyPF detectors [Lu et al., 2015] in the form of two additional channels to the network's input.

2.4 Experiments and Results

2.4.1 Evaluation Metrics

We first define and justify the metrics used. Average Precision(AP) and Optimal Dataset Scale(ODS) F1 score are standard metrics used for edge detection tasks Arbelaez et al. [2011]. AP is defined as the mean precision taken over all recall values, and is equal to the area under the Precision-Recall(PR) curve. ODS F-score is the optimal F1 score obtained by choosing the threshold of a probabilistic classifier which gives the best performance across the training dataset. Previous works on wire detection use the Receiver Operating Characteristics(ROC) at a *per-wire* basis. This entails fitting a line or a curve to thresholded predictions and then counting instances of the same, which leads to ambiguity due to issues like minimum length to register a detection, choosing polynomials to fit the thresholded predictions for curved wires, and coming up with tolerances which relax the detection criteria. Therefore, we calculate both ROC and PRC on a *per-pixel* basis and report Area under the Curve of the ROC diagram(AUC), which like AP, summarizes the curve with a single number.

It is also worth mentioning that for the case of severely class-imbalanced problems such as ours, AP is a better performance metric than ROC. In ROC analysis. false detections are evaluated via False Positive Rate, which compares false positives to true negatives (i.e. background pixels which account for roughly 95% of the USF dataset). Whereas in PR curves, the measure of false detections is done by Precision, which compares false positives to true positives. Therefore, in problems where the negative class outnumbers the positive class by a huge margin, AP is a better measure for classifier evaluation. We refer the reader to Davis and Goadrich [2006] for more details.

2.4.2 Grid Search

We do extensive evaluation of the models listed in Table 2.1 on the USF dataset. First, we randomly select half of the USF videos (21 out of 42 available) for the training set and kept the rest for testing, leading to 2035 frames for training and 1421 frames for testing. Meanwhile, we also have a synthetic dataset consisting of 67702 frames from 154 publicly available flight videos as mentioned before. For all cases, the input resolution used is 480x640. For each model from Table 2.1, we conduct three kind of experiments:

- **Synthetic**: Train on our synthetic dataset
- **Scratch**: Train on train split of USF dataset
- **Finetune**: Finetune on the USF train split after training on synthetic dataset

For each category above, we conduct runs with two different inputs to investigate the benefits of explicitly providing local evidence:

- **RGB**: training on monocular images only
- **RGBLE**: training on monocular images, concatenated with lines(L) and edges(E) detected by Lu et al. [2015].

Table 2.2 shows the results of all the six experiments on each model, along with the AP scores and performance on the NVIDIA Titan X Pascal and the TX2. Each experiment category is assigned an arbitrary hue, and darker color emphasizes better performance. Same is true for the last 4 columns which depict the inference performance with a batch size of 1, and input resolution of 480x640. Each class of models with the front-end is demarked by the bold blue text, while context modules of increasing amount of dilation are added to them in 5 consecutive rows. From Table 2.2, we can draw the following conclusions:

- *Dilated kernels with increasing sparsity help*:

  This trend can be seen throughout in Col 2-7 for each front-end. For each column, for the same frontend module, we can observe increasing AP scores as we add more dilation in the context modules. As the AP numbers can be close and hard to judge, we verify the same with Precision Recall curves for each front-end with varying context modules as shown in Figure 2.3. At the same time, the run-time performance of the models decrease as we can keep on adding more layers (Col 8-11).

- *Explicitly providing local information only helped in the case of synthetic data*:

  This can be seen by comparing the RGB and RGBLE columns in Table 2.2 under each category. In case of training (Col 4-5) or finetuning on real data (Col 6-7), we actually observe slightly poorer performance by providing results of line and edge detection. However, for the case of training on synthetic data (Col 2-3), providing local evidence helps to boost the networks' performance on real data by fairly big margins. We believe this is due to the nature of the synthetic dataset which doesn't have the same distribution as the USF test split, and hence, giving the networks all lines and edges explicitly helps to improve their performance. This also suggests that an RGBLE input might generalize better to a test set having a significantly different distribution from the training set, but more experimentation is needed to ascertain that.

- *Pretraining on synthetic data helps slightly*:

  This can be seen by comparing Col 4 with Col 6 (RGB inputs). We believe this is another artifact of the USF dataset, which is small in size and a lot of images are relatively simplistic. The previous statement is supported with evidence of results presented in Fig. 2.5. Here, we train only on synthetic data with 720x1280 resolution, and tested on a few publicly available videos. In this case, the value of synthetic data is clearly evident.

For our use case, we desire fast inference speed on the TX2 and decent precision at high recalls for wires. To pick the best of the lot, the top 15 models from our grid search experiments on finetuning on the USF dataset with RGB input can be seen in Table 2.3, along with inference speed with batch size 1 and resolution of 480x640, and number of parameters (although number of parameters isn't that much of a burden at inference time due to enough memory on the TX2). We cherry pick the 5 models in bold text by looking at these three metrics, the precision recall scores (Fig. 2.4), and qualitative comparisons (Fig. 2.6). Table 2.4 compares these top-5 models with various baselines.

Implementation details: For training on synthetic data, we use a minibatch size of 4 and train for 2000 iterations, while for the USF dataset, we train for 1000 iterations. All input images are of 480x640 resolution. We use AdaDelta Zeiler [2012] we found that stochastic gradient descent with momentum lead to slower convergence and a few architectures getting stuck in local minimas. We implement our models in both Lasagne Dieleman et al. [2015] and Pytorch, as we found that the latter is faster and consumes less memory on the TX2.

2.4.3 Baselines

We consider four baselines to compare our approach with - Candamo et al.Candamo et al. [2009], FCNsLong et al. [2015], SegNetBadrinarayanan et al. [2017] and E-NetPaszke et al. [2016]. We adapt the temporal approach of Candamo et al. [2009] described in the related work section to a per-frame method: first we perform edge detection using Lu et al. [2015], followed by morphological filtering of 8-connected components with connectivity of less

Table 2.3: Top models from our grid search. All models were trained on synthetic data first, then finetuned on our USF dataset test split using RGB input. Darker color in each colum implies higher accuracy, higher speed (frames per second) on the TX2, and smaller number of model parameters. Hues for each column are chosen arbitrarily.

Model	**AP Score (Finetune RGB)**	**TX2 (fps)**	**Number of params**
k64-k64-p2-k128-k128-d1-d2-d4-d8-d16-d1-k2	0.75	2.03	1,147,970
k64-k64-p2-k128-k128-d1-d2-d4-d1-k2	**0.73**	**2.85**	**852,802**
k64-k64-p2-k128-k128-d1-d2-d4-d8-d1-k2	0.73	2.37	1,000,386
k32-k32-k32-k32-d1-d2-d4-d8-d16-d1-k2	**0.72**	**3.54**	**84,706**
k64-k64-p2-k128-k128-d1-d2-d1-k2	0.72	3.57	705,218
k32-k32-k64-k64-d1-d2-d4-d8-d16-d1-k2	0.71	1.27	288,290
k32-k32-d1-d2-d4-d8-d16-d1-k2	**0.70**	**4.01**	**66,210**
k32-k32-k32-k32-k32-k32-d1-d2-d4-d8-d1-k2	**0.70**	**3.7**	**93,954**
k32-k32-k64-k64-k64-k64-d1-d2-d4-d8-d16-d1-k2	0.70	1.14	362,146
k32-k32-k32-k32-d1-d2-d4-d8-d1-k2	0.68	4.23	75,458
k32-k32-k64-k64-d1-d2-d4-d1-k2	0.68	2.02	214,434
k32-k32-k32-k32-k32-k32-d1-d2-d4-d1-k2	**0.67**	**4.44**	**84,706**
k32-k32-k64-k64-k64-k64-d1-d2-d1-k2	0.67	2.28	251,362
k32-k32-k64-k64-k64-k64-d1-d2-d4-d8-d1-k2	0.67	1.37	325,218

Table 2.4: Our top models compared with various baselines. The metrics used are Average Precision(AP), Area under the Curve of ROC diagram(AUC), Optimal Dataset Scale F1-Score(ODS F-Score)

Model	AP	AUC	ODS F-Score	TX2 (fps)
FCN-8 Long et al. [2015]	0.581	0.960	0.598	1.37
FCN-16 Long et al. [2015]	0.639	0.972	0.663	1.39
Segnet Badrinarayanan et al. [2017]	0.571	0.939	0.567	2.19
Candamo Candamo et al. [2009]	0.408	-	0.382	-
E-Net Paszke et al. [2016]	0.580	0.945	0.595	-
k64-k64-p2-k128-k128-d1-d2-d4-d1-k2	**0.729**	0.972	**0.688**	2.85
k32-k32-k32-k32-d1-d2-d4-d8-d16-d1-k2	0.717	**0.976**	0.678	3.54
k32-k32-d1-d2-d4-d8-d16-d1-k2	0.703	0.969	0.673	4.01
k32-k32-k32-k32-k32-k32-d1-d2-d4-d8-d1-k2	0.696	0.973	0.656	3.70
k32-k32-k32-k32-k32-k32-d1-d2-d4-d1-k2	0.667	0.970	0.647	**4.44**

than 30 pixels, followed by a windowed Hough transform by breaking up the edgemap image into 16 (4*4) subwindows.

Table 2.4 and Figure 2.4 shows the performance of our top 5 models along with the baselines. We report AP, AUC, ODS-F1 scores, inference speed for a 480x640 image with batch size 1 on the NVIDIA TX2, and the PR curves. For Candamo et al. [2009], as the output is a binary prediction (detected lines via windowed Hough transform) and not a confidence value as in the case of CNNs, we report the F1 score and the Precision value in the ODS F-score and the AP fields respectively. We don't report AUC as that is applicable to probabilistic classifiers only. To calculate the metrics in this case, we rasterize the results of the Hough transform with varying thickness of the plotted lines from 1 to 5 pixels and report the best F1 and Precision scores obtained for a fair comparison.

2.5 Conclusion and Future Work

In this paper, we presented a method to detect wires using dilated convnets to facilitate autonomous UAVs. We generate a large synthetic dataset by rendering wires and superimposing them on frames of publicly available flight videos, and demonstrate its effectiveness by showing qualitative and quantitative results. Our experiments systematically find the best architectures over a finite space of model parameters and outperform various baselines across multiple detection metrics and inference speeds on the NVIDIA TX2.

We are currently working on multi-view stereo methods to find distance of the wires and coming up with methods to avoid them robustly at high speeds. For improving the perception pipeline, we are looking into eliminating false positives and considering temporal information into account for consistent detection of wires.

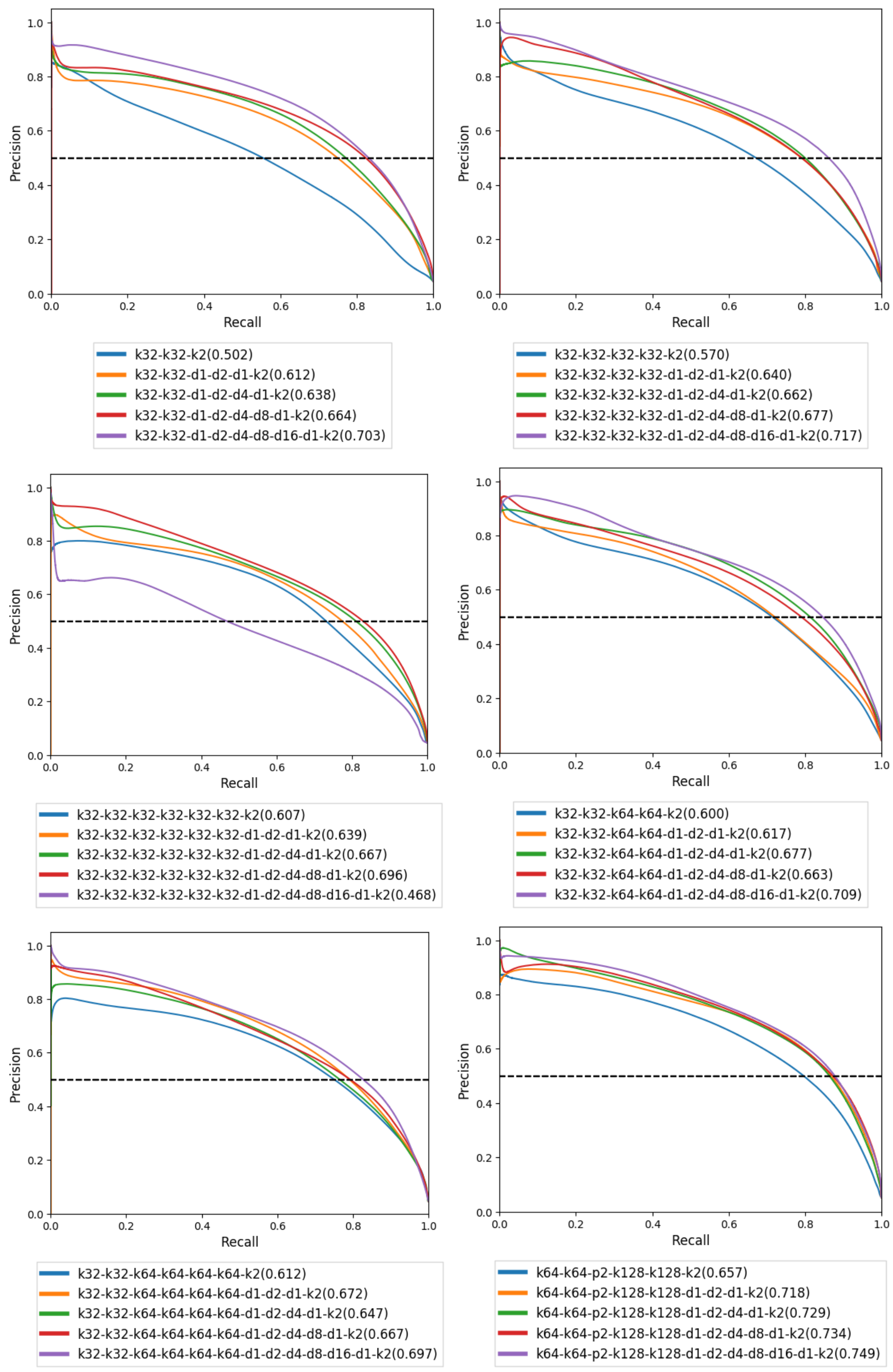


Figure 2.3: Precision recall curves obtained by evaluating our models on USF test split with RGB input, after finetuning our models on USF train split after pretraining on synthetid data. Each subfigure has the same front end module, while the context module changes as shown in the legends. Numbers in paranthesis are the Average Precision scores for the respective model. In all cases we can see that adding increasing dilation increases the precision of the classifier at various recall levels (except for one case in k32-k32-k32-k32-k32-k32's series where the model with the maximum dilation didn't converge.

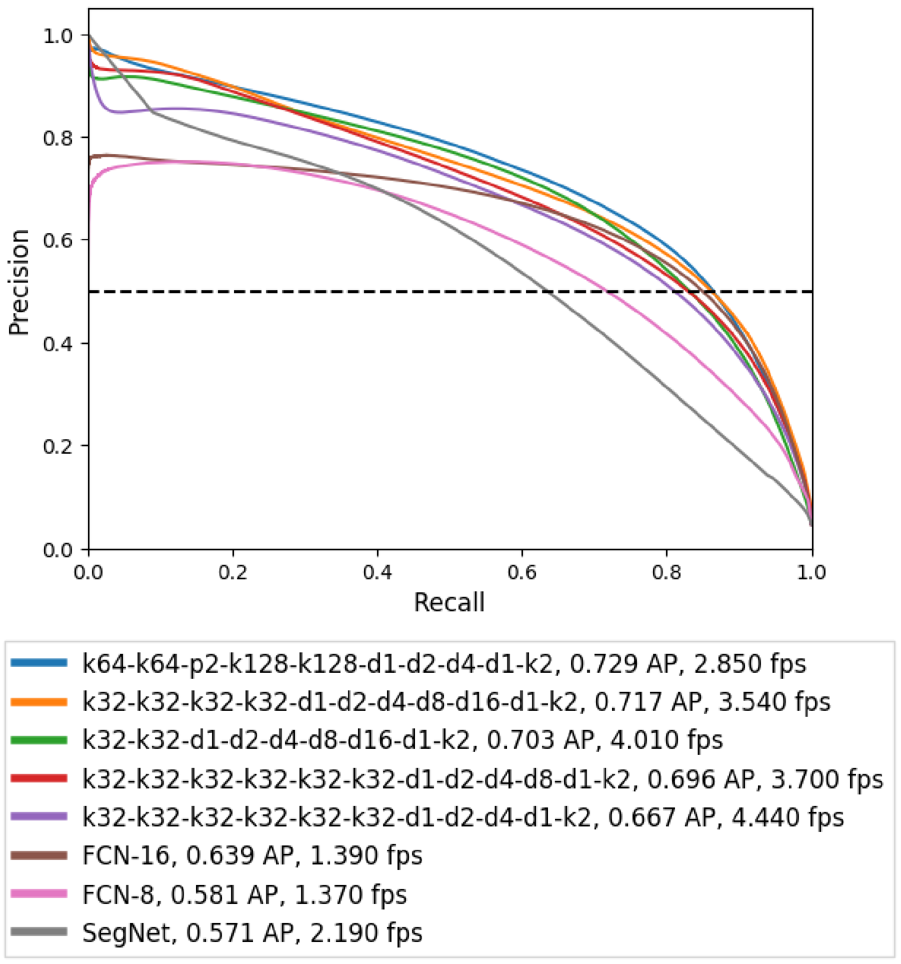


Figure 2.4: Precision Recall curves of the top models from our grid search experiment (in bold text in Table 2.3). Legend shows Average Precision scores on the USF test split, and speed on the NVIDIA TX2 in frames per second.

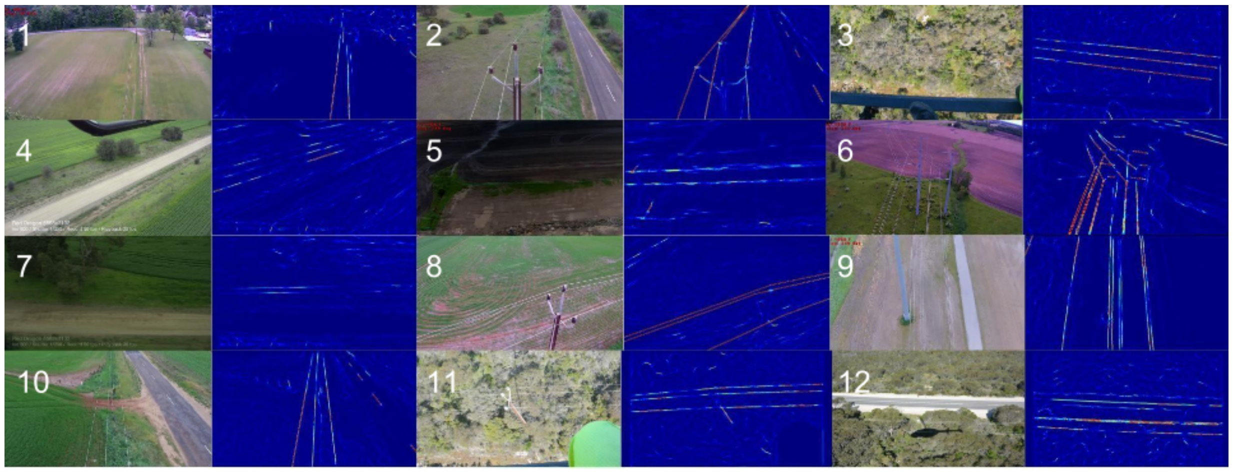


Figure 2.5: Qualitative results on real images with model k32-k32-k32-k32-d1-d2-d4-d8-d16-d1-k2 trained on our synthetic dataset with input resolution of 720x1280. High resolution helps the network pick far off (Image 6), very thin wires (Images 3 and 4), those in clutter (Image 11) and taken when in high speed (Images 5 and 7). Video available at this link and our project page.

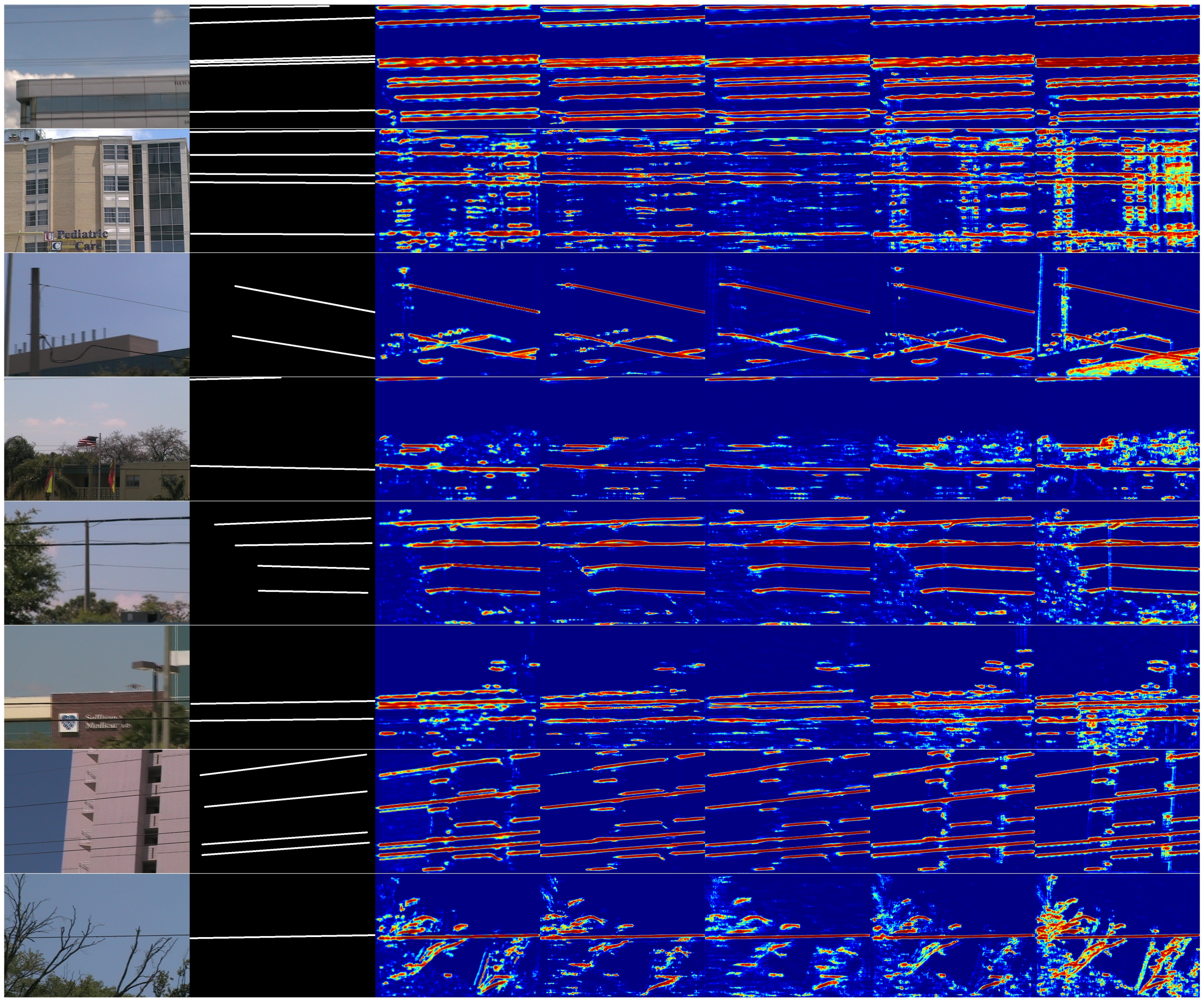


Figure 2.6: Qualitative results of our top 5 models after finetuning on the USF dataset with RGB input. Left to right : Input image, Ground Truth, predictions with architecture k64-k64-p2-k128-k128-d1-d2-d4-d1-k2, k32-k32-k32-k32-d1-d2-d4-d8-d16-d1-k2, k32-k32-d1-d2-d4-d8-d16-d1-k2, k32-k32-k32-k32-k32-k32-d1-d2-d4-d8-d1-k2, k32-k32-k32-k32-k32-k32-d1-d2-d4-d1-k2.

Chapter 3

Multi-view Reconstruction of Wires using a Catenary Model

3.1 Introduction

This chapter goes into the details of our model-based multiview reconstruction algorithm for wires. Wires, and in general any thin cable with a uniform mass distribution suspended by two points under its own weight under uniform gravitational force, follows a specific shape, given by the catenary curve. The key motivation here is to leverage this structure in the problem. Using a model will allow us to reconstruct wires with more fidelity than a model-free method such as voxel grid based mapping (proposed in the next chapter). This helps us in recovering the full curve in case of occlusion in the input images, and any missed detections (false negative pixels) along the wire. Estimation of the catenary is also highly important for inspection tasks such as monitoring the sag of wires and estimating distance of powerlines from vegetation [Sun et al., 2006, Ahmad et al., 2013].

In this work, we develop a multiview algorithm for wire reconstruction using a catenary model, for a monocular camera with known camera poses. While catenary models have been used before for wire detection and reconstruction from airborne lidar data [McLaughlin, 2006, Jwa and Sohn, 2012], this is the first work proposing a multiview algorithm. Given a set of images with known camera poses containing a single wire, we first perform pixel-wise wire segmentation with a convolutional neural network(CNN) using our previous work [Madaan et al., 2017]. Then, we estimate the catenary model parameters via non-linear least squares optimization, using cheap-to-compute distance transform as our loss function, thereby obviating the need to find wire pixel correspondences. We evaluate our method by simulating random camera poses, as well as avoidance scenarios. We also demonstrate our method on real data gathered with a small UAV.

Our key contributions are a minimal parameterization of the catenary, and an objective function which helps us reconstruct the wire without defining pixel correspondences.

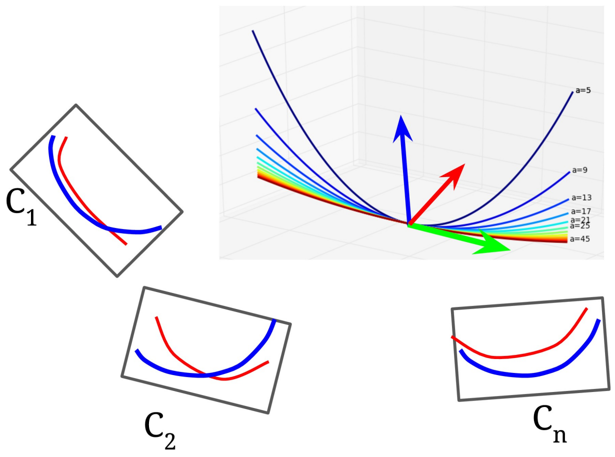


Figure 3.1: Schematic of the problem formulation. We are given N binary images of wire segmentations (red), along with their corresponding camera poses $\{C_1, C_2, ..., C_n\}$. The multiple curves and the transform are indicate various possible hypotheses. The blue curves denote the projection of the current hypothesis in each keyframe.

3.2 Related Work

While catenary models have been used before for wire extraction from airborne lidar data [McLaughlin, 2006, Jwa and Sohn, 2012], this is the first work proposing a multiview algorithm. [Jwa and Sohn, 2012] use a catenary model for powerline reconstruction from lidar data using non-linear least squares, while [McLaughlin, 2006] first proposed to use catenary models and used numerical methods for optimization. Interestingly, catenaries are also found in human biology and models have been used previously for analysing dental arches [Hasse Pepe, 1975, Burdi and Lillie, 1966] and breast curvature [Lee et al., 2012b,a].

In the realm of general thin obstacle reconstruction, [Hofer et al., 2013] propose an incremental line based 3D reconstruction algorithm use by first using Structure from Motion to recover camera poses and then perform line based modeling, however their method is not real-time as they solve for a combanatorial number of possible matches of line features using epipolar geometry. Similar combanatorial matching is observed in curve matching in [Liu et al., 2017]'s work on image based reconstruction of wire art. A solution to this issue of hard to define features for thin obstacles (lines and curves) and subsequent data association is provided by using the distance transform as the objective function, as done by recent works on edge based visual odometry [Kuse and Shen, 2016, Ling et al., 2018], subdivision curve reconstruction [Kaess et al., 2004], and first introduced in literature by [Fitzgibbon, 2003].

Quite recently, [Zhou et al., 2017a] developed a promising generic thin obstacle mapping approach building on edge based visual odometry work Jose Tarrio and Pedre [2015].

3.3 Approach

Let us begin this section by outlining the high level problem statement:
Given N views observing a single wire with known camera poses, recover the model parameters Θ, minimizing the reprojection error $f(\Theta)$.
Figure 3.1 provides a high level description of the problem statement. We are given N binary images of wire segmentations (given in red), along with their corresponding camera poses $\{C_1, C_2, ..., C_N\}$. The multiple curves and the transform are indicate various possible hypotheses. The blue curves in each image denote the projection of the current hypothesis in each frame.

This brings us to the two main **research questions** we address in this chapter:

- What is the model Θ?
  - What is the minimal parameterization of this model?
    We care about the minimal number of parameters so that we can scale our current formulation for reconstructing a single wire to multiple wires, in an efficient manner. Avoiding redundant parameters makes leads to faster convergence as well.
- What is the objective, the reprojection error $f(\Theta)$?
  - How can we evaluate a catenary hypothesis? This is achieved by computing the distance between the wire detections (red) and the catenary hypothesis' projections (blue), as depicted in Figure 3.1. How can we define pixel correspondences for such a thin obstacle, detected by a per-pixel CNN? Defining visual features and then establishing correspondences is known to be a problem for thin objects like edges, lines, and curves. One thing we want to investigate is then can we somehow escape the problem of defining them in the first place?

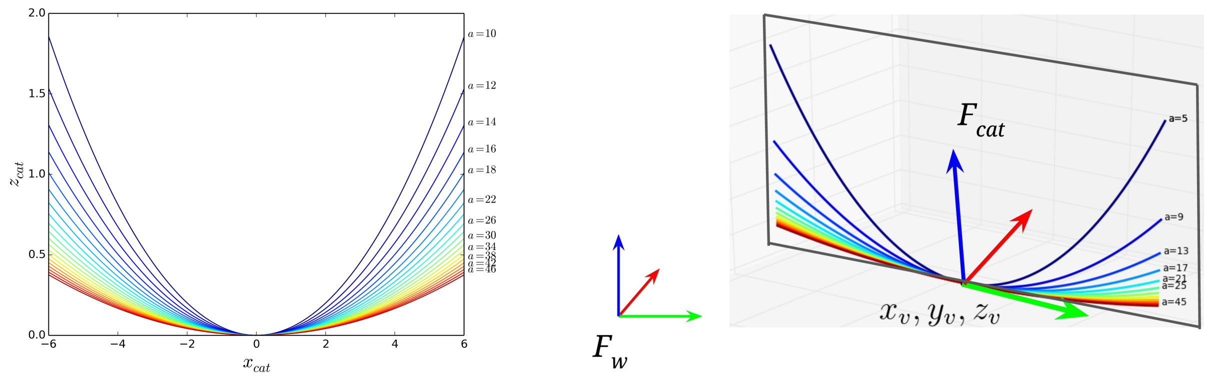
We design our approach keeping in mind the issue of establishing pixel correspondences and the goal of a real time algorithm. We present a catenary model based multi-view algorithm given a set of images and the corresponding camera poses. In the upcoming sections, we first introduce the catenary model in 3.3.1, followed by our objective function in 3.3.2, and finally the optimization problem in 3.3.3.

3.3.1 Catenary Model

A wire with uniform density and thickness when suspended at its two ends under uniform gravity follows the shape of the single parameter catenary curve given by:

$$z_{cat} = a\left(cosh\left(\frac{x_{cat}}{a}\right) - 1\right), \; y_{cat} = 0 \tag{3.1}$$

where a is the catenary sag parameter defining how fast or slow the catenary opens up as can be seen in Fig 3.2a, and $P_{cat} = \{x_{cat}, y_{cat}, z_{cat}, 1\}$ are homogeneous coordinates in the local catenary frame, F_{cat}. Note that we have brought the origin of F_{cat} to the lowest point of the catenary by subtracting a from z_{cat}, unlike the more popular form of the curve,



(a) (b)

Figure 3.2: (a) Catenary curves from Equation 3.1 in the local catenary frame, F_{cat} with varying sag parameter (a) values.
(b) Our 3D catenary model is obtained by defining the local catenary frame, F_{cat} with respect to a world frame, F_W. Note that because the curve is planar and sags under the effect of gravity, we only need to define a relative yaw instead of a full rotation transform. The catenary parameters are then given by $\{x_v, y_v, z_v, \psi, a\}$. Here, $\{x_v, y_v, z_v\}$ are the coordinates of the lowermost of the curve which we also refer to as the vertex of the catenary, ψ refers to the relative yaw of F_{cat} w.r.t. F_W

, and finally a refers to the sag parameter of the curve.

$z_{cat} = a\ cosh(x_{cat}/a)$. We are going to refer to the lowermost point of the catenary which is also the origin of F_{cat} as the *catenary vertex*. This planar catenary can easily be extended to 3D by defining F_{cat}, or the catenary vertext w.r.t. an arbitrary world frame F_w, by a relative translation and rotation. Let the 3D coordinates of the catenary vertex in F_w be given by the tuple (x_v, y_v, z_v). Note that the gravity vector is parallel to the plane of the catenary. Thus if we choose an F_w s.t. its $+Z$ axis is anti-parallel to the negative gravity vector, there is only one degree of rotational freedom we need to consider while defining F_{cat}. Let ψ be the parameter denoting the relative yaw between F_{cat} and F_W. Finally, we can define the parameters of our catenary model as the 5-tuple $\Theta = \{x_v, y_v, z_v, \psi, a\}$. It is a 3D transform in addition to the sag parameter, except the roll and pitch in the rotation part. We can transform P_{cat} to F_w to get points in world frame, P_W by constructing appropriate rotation and translation matrices, R and T from Θ:

$$\Theta = \{x_v, y_v, z_v, \psi, a\} \tag{3.2}$$
$$P_W(\Theta) = R\,(\psi)\,P_{cat} + T(x_v, y_v, z_v) \tag{3.3}$$

At this point, one might wonder that wires of a finite width, so either we should have the width as another parameter, or we should model two endpoints. For the purposes of this document, we assume we know the width of the curve, and use a fixed resolution for the same. This fixed width and resolution leads to a fixed number of points on the curve, given by M. Thus, P_{cat} and P_W above are both matrices of size $M \times 3$. We propose to relax this assumption with a method to reconstuct a catenary by computing an upper bound for its width in a section 3.7.1, as part of future work. Note that with this extension, one does not need to model the end points of the catenary, and can still use our minimal parameterization. With our model, the optimization variables are limited to 5 in number,

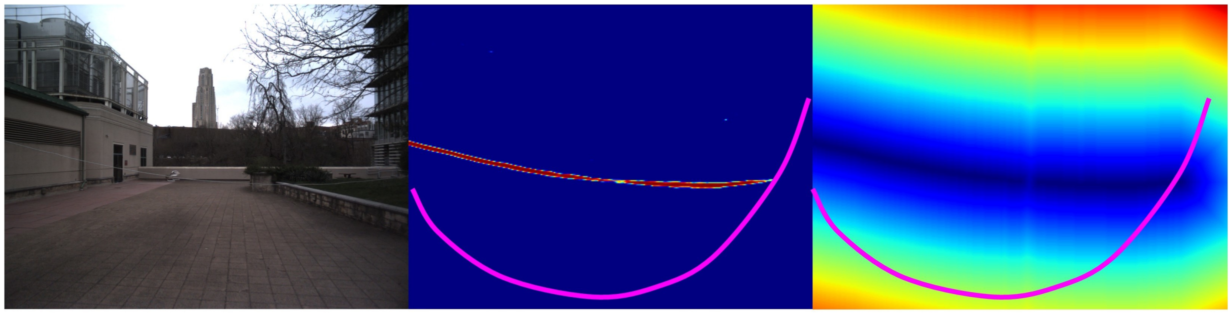


Figure 3.3: Example demonstrating our objective function for valid projections of the catenary hypothesis. Left: RGB image with thin wire running across the image. This image was taken in the patio behind Newell Simon Hall at CMU. Middle: Confidence map of per-pixel wire segmentation from our CNN. Colormap used is Jet. Given this confidence map, the goal is to establish the distance from the projection of a catenary hypothesis (shown in pink). Right: We binarize the segmentation map, and use the distance transform of the binarized segmentations as our objective function.

instead of tens to hundreds of points in $\mathbb{R}^3$ in bundle adjustment like approaches.

3.3.2 Objective Function

In order to recover the model parameters Θ, we want the projections of our resultant catenary model on each view to be as close as possible with the wire detections. The challenge here lies in defining the pixel correspondences, which is a known issue with thin features like lines, edges and curves. To combat this problem, we take inspiration from previous works [Kuse and Shen, 2016, Ling et al., 2018, Kaess et al., 2004, Fitzgibbon, 2003] and find the *normalized* distance transform, D_i for each keyframe.

We use the euclidean distance with a $3a \times 3$ mask to define the distance. This means that the distance is defined by the shortest path to the nearest wire pixel, where horizontal, vertical, or diagonal moves are allowed. We use OpenCV's distance transform function which implements Borgefors [1986]. The details are available in the documentation here.

We can then define a *residual vector*, $\mathbf{r} \in \mathbb{R}^{MN}$ by projecting each of the M points on the catenary onto each of the N keyframes, and stacking the distance transform values for every j^{th} reprojected pixel in the i^{th} view:, $p_i^j = (u_i^j, v_i^j)$. The residual for pixel p_i^j is then given by $r_i^j = D_i(p_i^j)$. This is visualized in Figure 3.3 with a help of an example RGB image with a wire (left)m and the corresponding wire segmentation confidence map (center). For each pink projected pixel of the catenary model, the corresponding residual vector is simply the distance transform value of the binarized segmented image (right).

Now, another thing to take care of is defining a loss value for any **invalid pixels**, defined as pixels whose coordinate values fall outside the image height and width limits (h, w). A naive option is to run the distance transform on arbitrary large images for each view by padding them, but then it is hard to define what is the optimal padding size.

We propose an **on-demand, approximate extrapolation** of D_i for such invalid pixels, which is continuous and differentiable at the image boundaries, thus aiding the non-linear optimization.

For any j^{th} invalid pixel in the i^{th} view given by p_i^j, we first find a corresponding

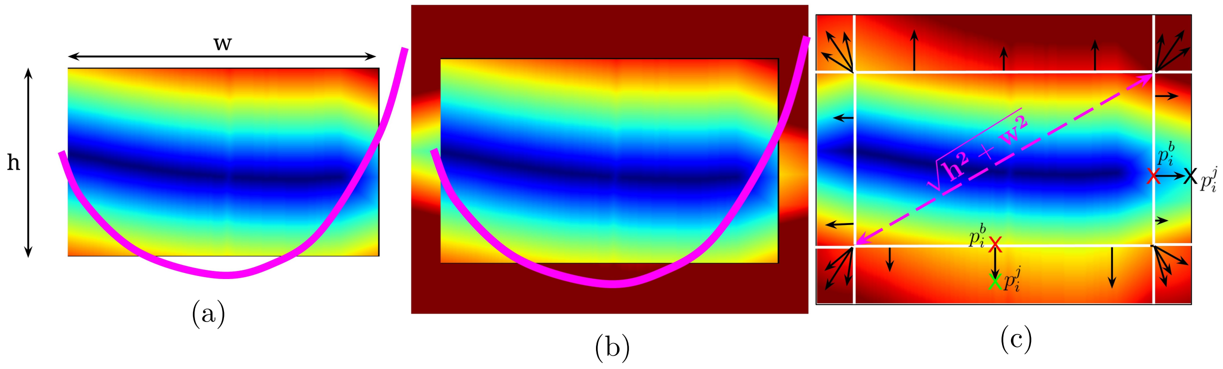


Figure 3.4: (a) This highlights the issue with invalid projections, falling outside the image coordinates. The question then is, how do we quantify the distance of such invalid pixels from the segmented wire.
(b) If we extrapolate the loss function from the image center, it becomes discontinuous and non-differentiable at the boundaries of the image.
(c) The fix for this is then intuitive. By extrapolating from the boundary pixels, we mitigate the problem. The invalid area outside the image is divided into 8 regions. For any point p_i^j denoted by the arrow heads, we find the corresponding pixel at the boundary p_i^b given by the base of the arrow, and extrapolate the value as explained in the text.

boundary pixel p_i^b by defining 8 regions around the image as shown in Figure 3.4 and Equation 3.15. Then, we find the euclidean distance between the pixels p_i^b and p_i^j, $d(p_i^b, p_i^j)$ and normalize it by the maximum possible value of the *unnormalized* distance transform, the image diagonal $\sqrt{h^2+w^2}$. Then, the loss value at any invalid pixel is obtained by adding this value to the normalized distance transform (D_i) value at the corresponding boundary pixel p_i^b, given by $D_i(p_i^b)$ as shown in Equation 3.16 and Fig 3.4. Recall that D_i is normalized itself, which means its maximum value can be 1.

The above can be expressed formally as

$$r_i^j(p_i^j) = \begin{cases} D_i(p_i^j), & \text{if } u_i^j \in [0,h), v_i^j \in [0,w) \\ D_i(p_i^b) + \left(d(p_i^b, p_i^j)/\sqrt{h^2+w^2}\right), & \text{otherwise} \end{cases} \tag{3.4}$$

$$\text{where } p_i^b = (u_i^b, v_i^b),\ u_i^b = \begin{Bmatrix} 0, & \text{if } u_i^j < 0 \\ h-1, & \text{if } u_i^j >= h \end{Bmatrix},\ \text{and } v_i^b = \begin{Bmatrix} 0, & \text{if } v_i^j < 0 \\ w-1, & \text{if } v_i^j >= w \end{Bmatrix} \tag{3.5}$$

Also note that we compute D_i for each of the N image once, and reuse it for each iteration of the optimization. This is pretty fast, and takes less than 1 ms. For any pixel that fall outside the image, we can compute the residual by extrapolating D_i in an on-demand fashion.

3.3.3 Optimization Problem

Let us begin by recapping how we obtain the residual vector from the model parameters. We begin with the model parameters Θ, and using Equation 3.3, we can get points on the

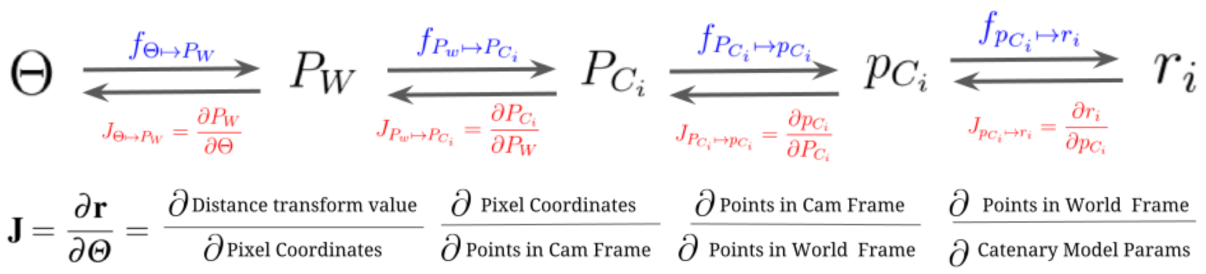

$\Theta \xrightarrow{f_{\Theta \mapsto P_W}} P_W \xrightarrow{f_{P_w \mapsto P_{C_i}}} P_{C_i} \xrightarrow{f_{P_{C_i} \mapsto p_{C_i}}} p_{C_i} \xrightarrow{f_{p_{C_i} \mapsto r_i}} r_i$
$J_{\Theta \mapsto P_W} = \frac{\partial P_W}{\partial \Theta}$
$J_{P_w \mapsto P_{C_i}} = \frac{\partial P_{C_i}}{\partial P_W}$
$J_{P_{C_i} \mapsto p_{C_i}} = \frac{\partial p_{C_i}}{\partial P_{C_i}}$
$J_{p_{C_i} \mapsto r_i} = \frac{\partial r_i}{\partial p_{C_i}}$
$\mathbf{J} = \frac{\partial \mathbf{r}}{\partial \Theta} = \frac{\partial \text{Distance transform value}}{\partial \text{Pixel Coordinates}} \frac{\partial \text{Pixel Coordinates}}{\partial \text{Points in Cam Frame}} \frac{\partial \text{Points in Cam Frame}}{\partial \text{Points in World Frame}} \frac{\partial \text{Points in World Frame}}{\partial \text{Catenary Model Params}}$

Figure 3.5: Chain of transformations and jacobians.

curve in the world frame, $P_W \in \mathbb{R}^{3MN}$. Then, each point in P_W is transformed to each camera's local frame, to get P_{C_i} via the respective extrinsics matrices. Then, the camera intrisics are applied to get a vector of projected pixels $p_i \in \mathbb{R}^{2MN}$ from . Finally, we can use Equations 3.15 and 3.16 to get a vector of residuals, $\mathbf{r} \in \mathbb{R}^{MN}$.

The above chain of mappings can be expressed as:

$$\Theta \mapsto P_W \mapsto P_{C_i} \mapsto p_i \mapsto r_i \ , \ i \in \{1, 2, ..., N\} \tag{3.6}$$

Each mapping is differentiable and lends us a corresponding analytical jacobian via the chain rule, as shown in Figure 3.5.

Our objective function is then given by the squared sum of the vector of residuals. We can also obtain an analytical Jacobian matrix of the residual vector w.r.t. the model parameters, $\mathbf{J} = \frac{\partial \mathbf{r}}{\partial \Theta}$ via the chain rule:

$$f(\Theta) = \frac{1}{2}\mathbf{r}^\top \mathbf{r} \tag{3.7}$$

$$\mathbf{J} = \frac{\partial \mathbf{r}}{\partial \Theta} = \frac{\partial r_i}{\partial p_i} \frac{\partial p_i}{\partial P_i} \frac{\partial P_i}{\partial P_W} \frac{\partial P_W}{\partial \Theta} \tag{3.8}$$

The analytical form of the Jacobian matrix is derived in Section 3.5. We optimize for this non-linear least squares objective using the Trust Region Reflective algorithm [Branch et al., 1999, scikitlearn: least squares, 2011] to recover the optimal model paramters Θ^*:

$$\Theta^* = \arg\min_{\Theta} f(\Theta) \tag{3.9}$$

We use the analytical jacobian as it is faster than computing it numerically. Another reason for doing that is numerical stability. Looking at Section 3.5, due to the equation of the catenary, for some combinations of the sag (a) and yaw (ψ) values, the numerical gradient can be instable.

3.4 Experiments and Results

3.4.1 Simulation

We simulate two types of scenarios: (1) random camera placement and (2) wire avoidance scenario. For random camera placement, we first sample a *look-at* point on the catenary where the camera will be pointing towards. Then, we sample a random viewing direction relative to this point, via cylindrical coordinates (equal area projection of sphere onto the lateral surface of a cylinder). One we have a viewing direction and a look-at point, we sample a distance along this vector which tells us where the camera will be placed relative to the wire. For avoidance scenarios, we limit the scope of the viewing direction, such that the cameras are all within a small steradian, thereby mimicking the kind of keyframes one might expect while flying a UAV head on towards a wire.

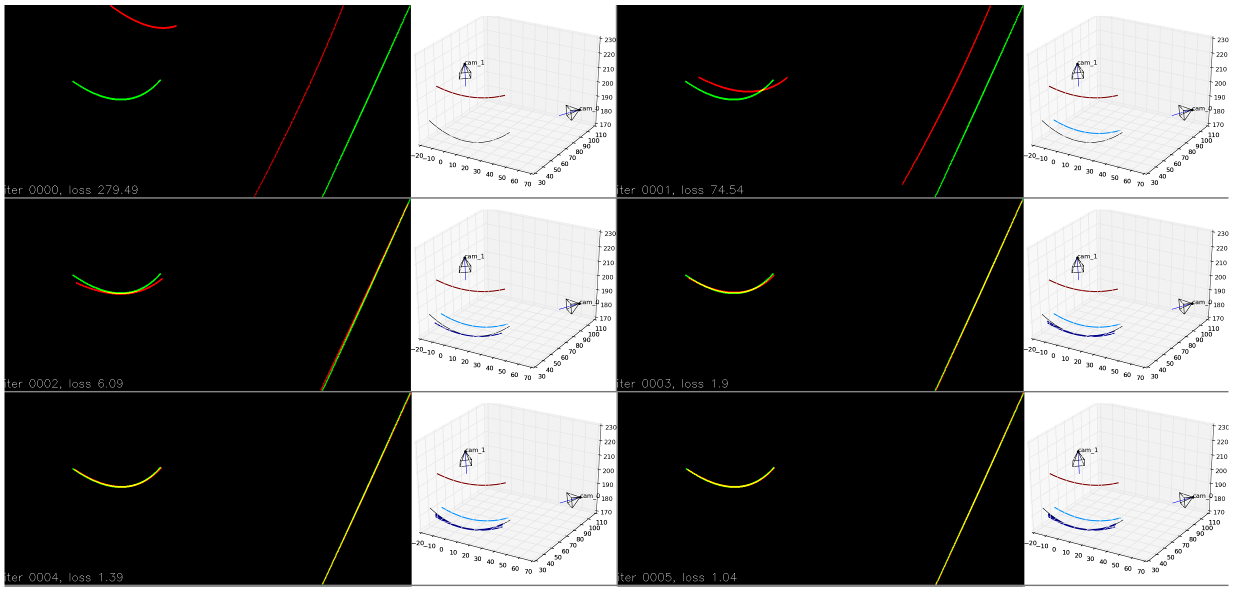
A few qualitative results are shown in Figures 3.6 and 3.7. For the purpose of this document, we demonstrate three main results for each of the two scenarios:

- Curve, Two cameras:
  - Random camera placement: Figure 3.6a
  - Avoidance scenario: Figure 3.7a
- Straight Line, Two cameras:
  - Random camera placement: Figure 3.6b
  - Avoidance scenario: Figure 3.7b
- Curve, Multiple cameras:
  - Random camera placement: Figure 3.6c
  - Avoidance scenario: Figure 3.7c

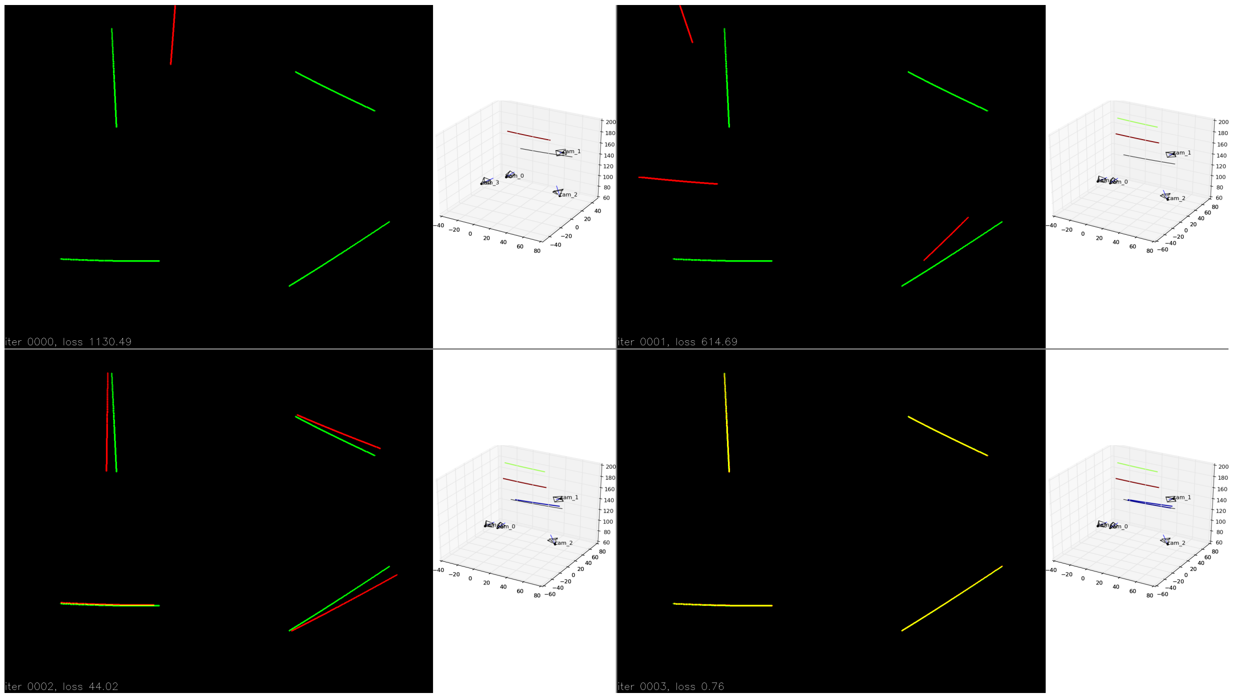
3.4.2 Real Data

We use the DJI Matrice M100, retrofitted with in-house developed sensor suite consisting of a stereo- camera, an FPGA stereo processor [Schauwecker], a monocular color camera, an IMU, and a Nvidia TX2 ARM computer as shown in Figure 4.9. For this section, we are using only the monocular camera for detection. Then we extract and analyse 16 chronological keyframes from the dataset with the criteria of a delta translation of at least 0.5 m or a delta orientation of 18°(quaternion geodesic distance) between consecutive keyframes. The RGB images are shown in 3.9a, and the corresponding segmentations are visualized in 3.9b. Figure 3.8 shows two handpicked keyframes to help examine the segmentation result.

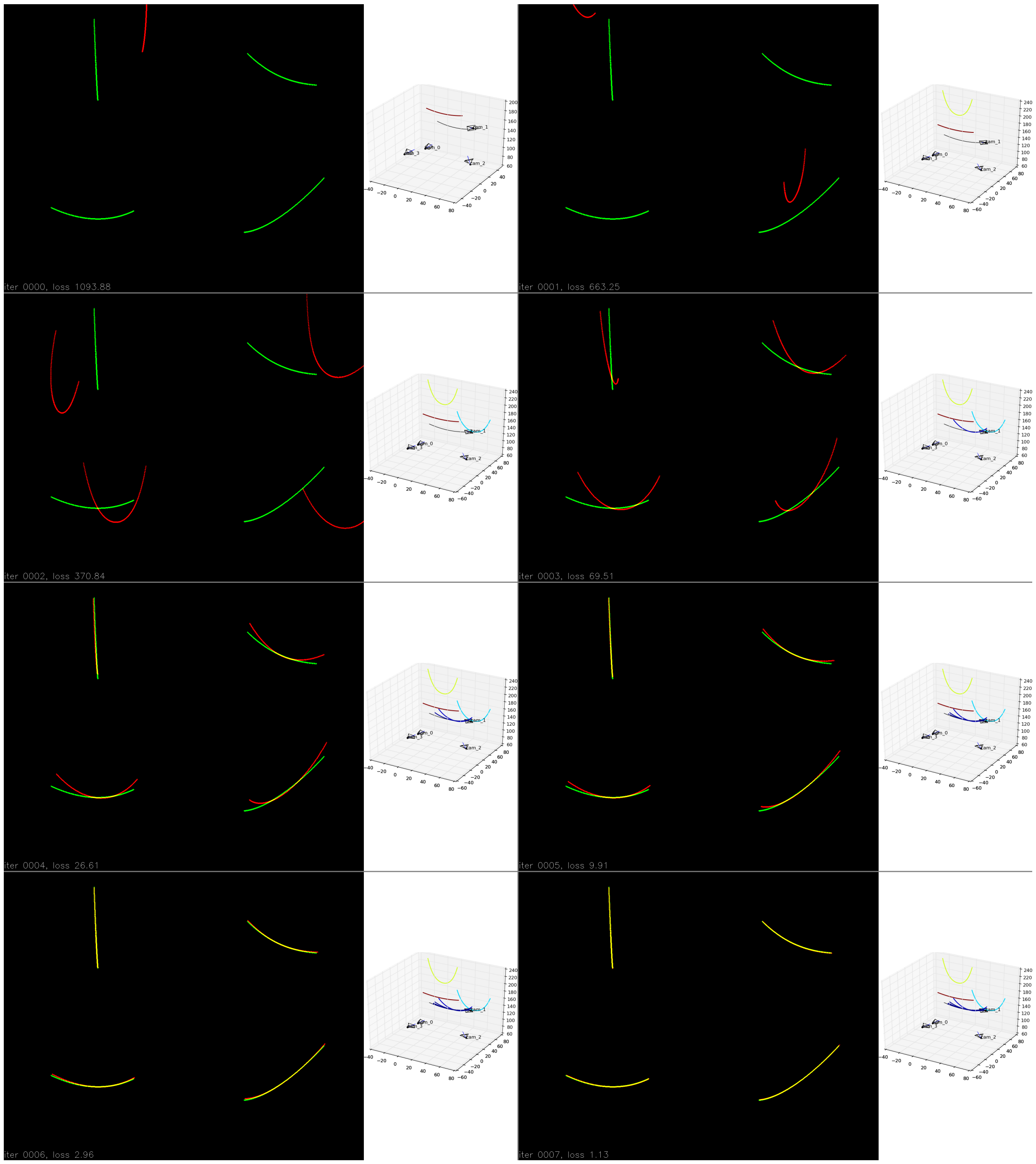
The results of the optimization are shown in Figures 3.9c and 3.9d. Figure 3.9c shows the projection of the converged catenary hypothesis match the observed segmentations closely. Figure 3.9d shows the odometry of the real data, along with initialized and the resultant catenary.



(a) Random camera placement, Curve, 2 cameras.

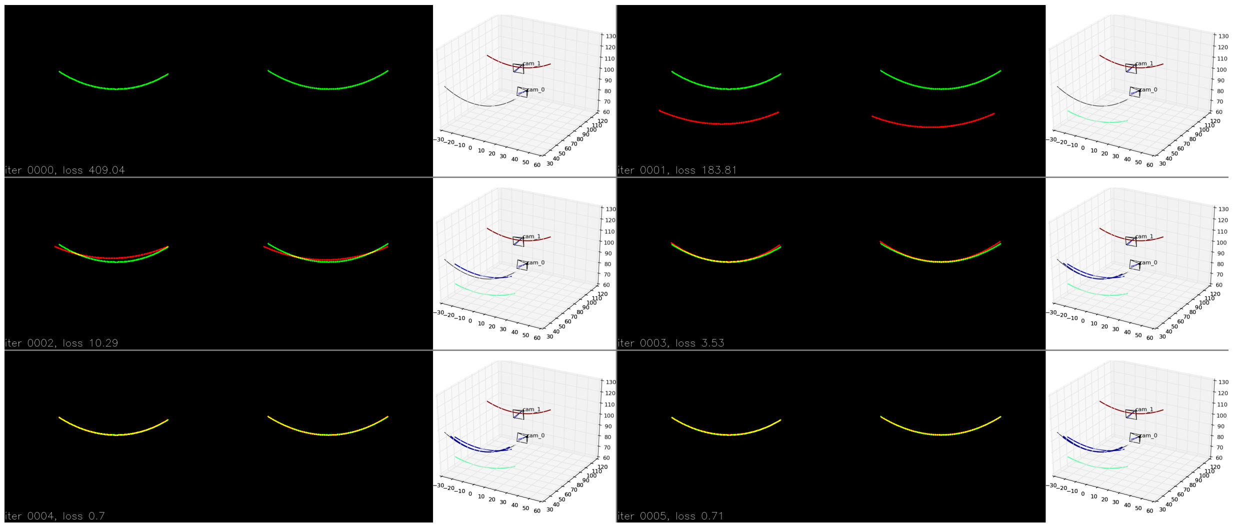


(b) Random camera placement, Straight line, 2 cameras.



(c) Random camera placement, Curve, 4 cameras.

Figure 3.6: Simulation of random camera placement scenarios. We encourage the reader to examine the corresponding animations available at the thesis webpage.



(a) Avoidance scenario, Curve, 2 cameras.

3.5 Analytical Jacobian Derivation

Let's first recall the 3D catenary model:

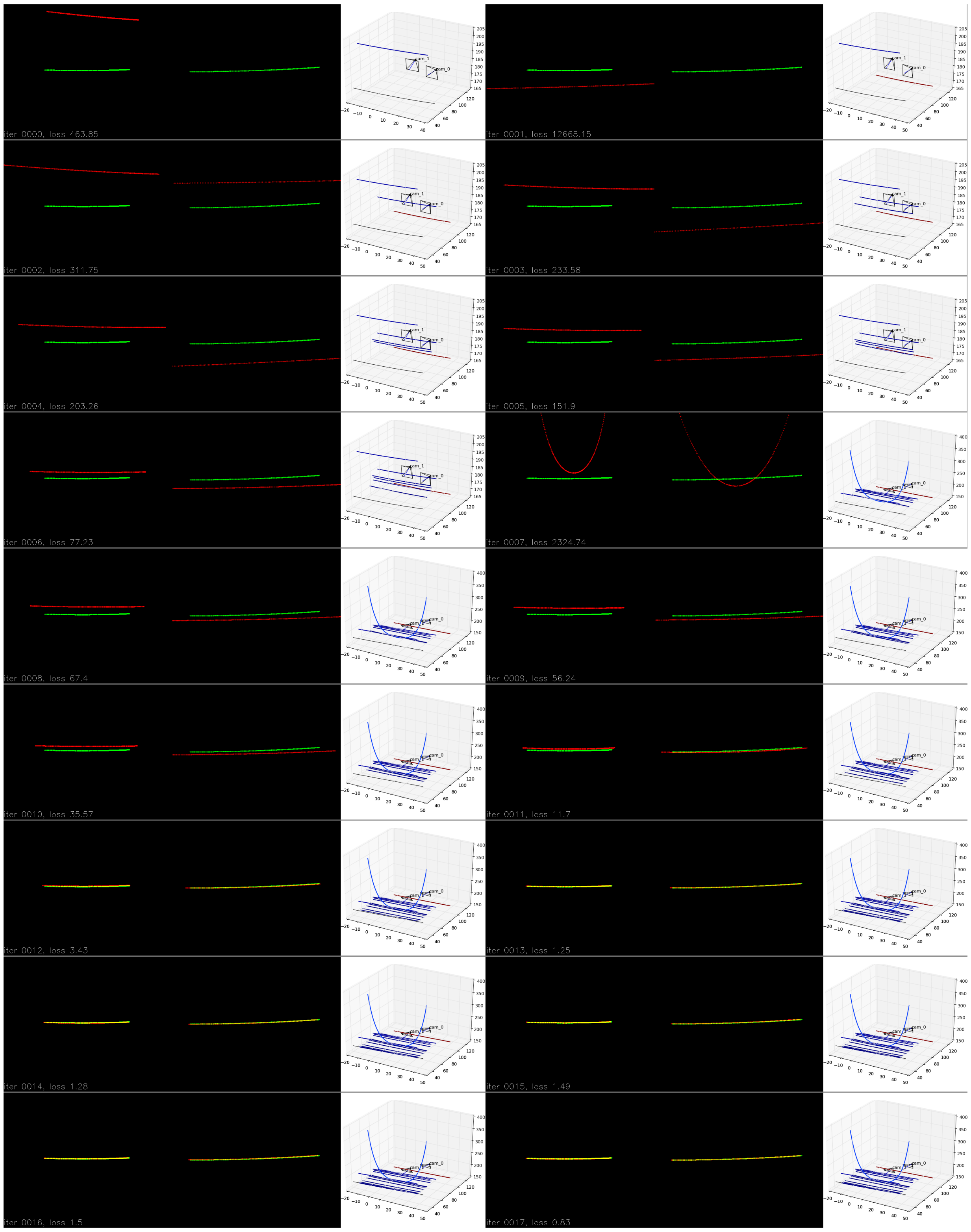
$$\Theta = \{x_v, y_v, z_v, \psi_v, a\} \tag{3.10}$$

$$z_{cat} = a\left(cosh\left(\frac{x_{cat}}{a}\right) - 1\right),\ \ y_{cat} = 0 \tag{3.11}$$

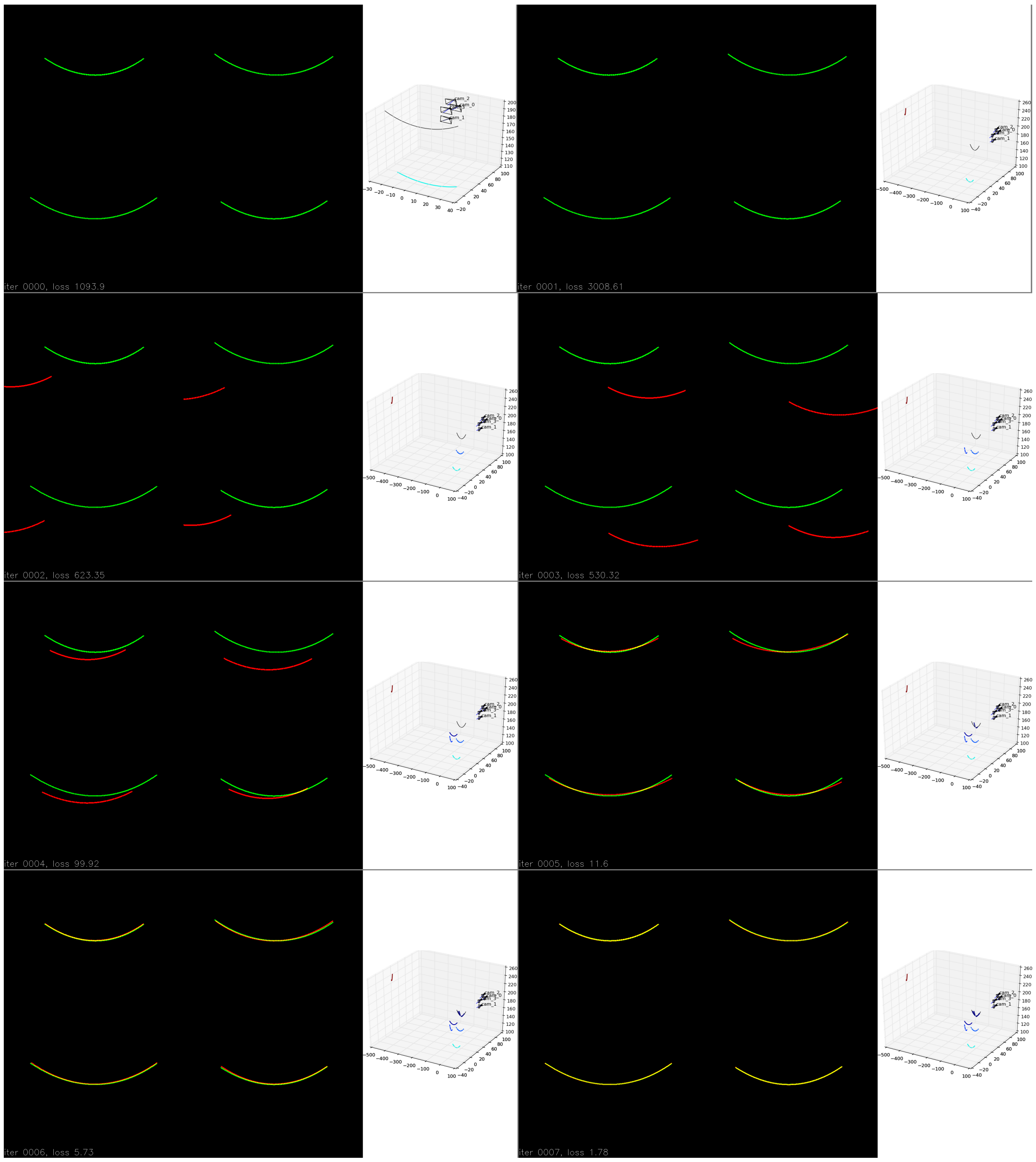
$$P_W(\Theta) = R\left(\psi_v\right)P_{cat} + T(x_v, y_v, z_v) \tag{3.12}$$

From equation 3.11, we can get a vector of points in the catenary frame, P_{cat} from the model parameters Θ, which are then transformed to the world frame using equation 3.12 to get P_W using $\{x_v, y_v, z_v, \psi_v\}$ from Θ, which are then transformed to each camera's local frame to get P_i via the respective extrinsics matrices, and finally the camera intrisics are applied to get a vector of projected pixels, p_i. Finally, we apply our objective function $f_i^j(\Theta)$ to each reprojected pixel get a vector of residuals, $\mathbf{r}$. We compute for a fixed discretization and width of the catenary.

Our objective function is given by the squared sum of the vector of residuals. We can also obtain an analytical Jacobian matrix of the residual vector w.r.t. the model parameters, $\mathbf{J} = \frac{\partial \mathbf{r}}{\partial \Theta}$ via the chain rule:



(b) Avoidance scenario, Straight line, 2 cameras. With a high sag parameter, the catenary approaches a straight line. This results demonstrates that our model still works for such cases.



(c) Avoidance scenario, Curve, 4 cameras. This is a really bad initialization, which becomes worse in the second iteration (examine the 3D views and the loss value closely). However, our algorithm still convergences due to the way we handle invalid projections, as explained in the text.

Figure 3.7: Simulation of avoidance scenarios. We encourage the reader to examine the corresponding animations available at the thesis webpage.

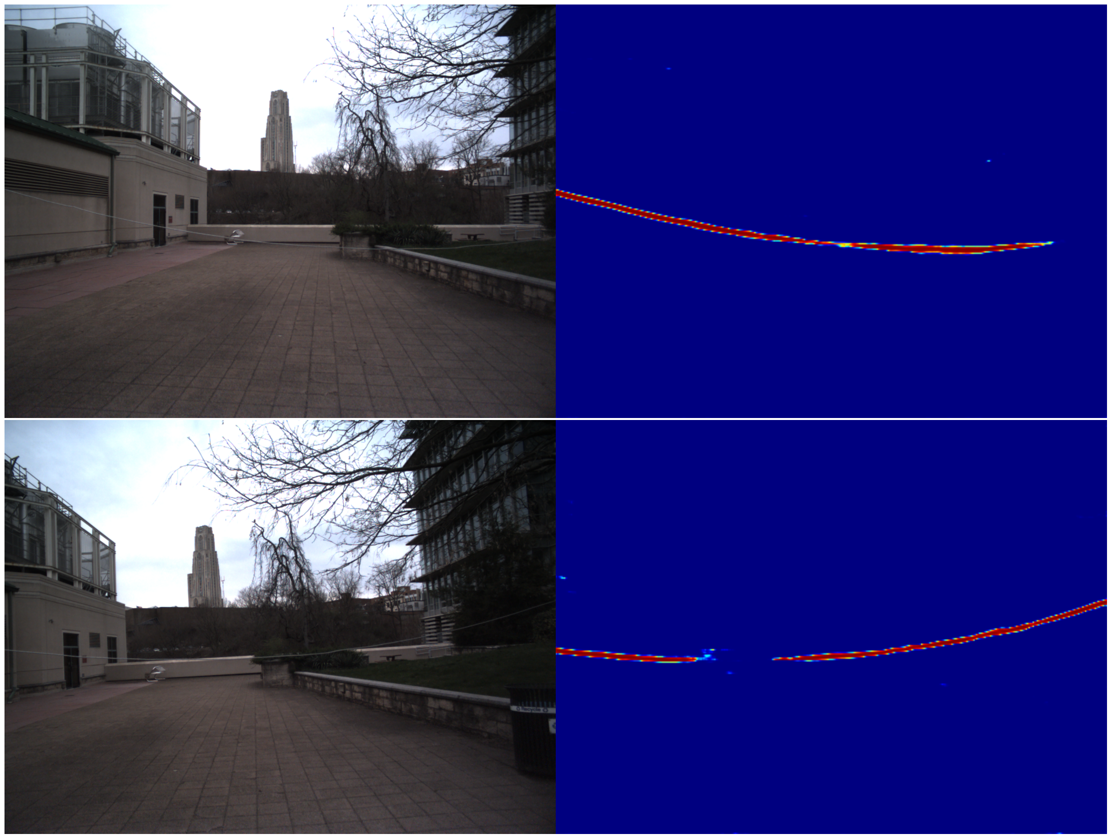
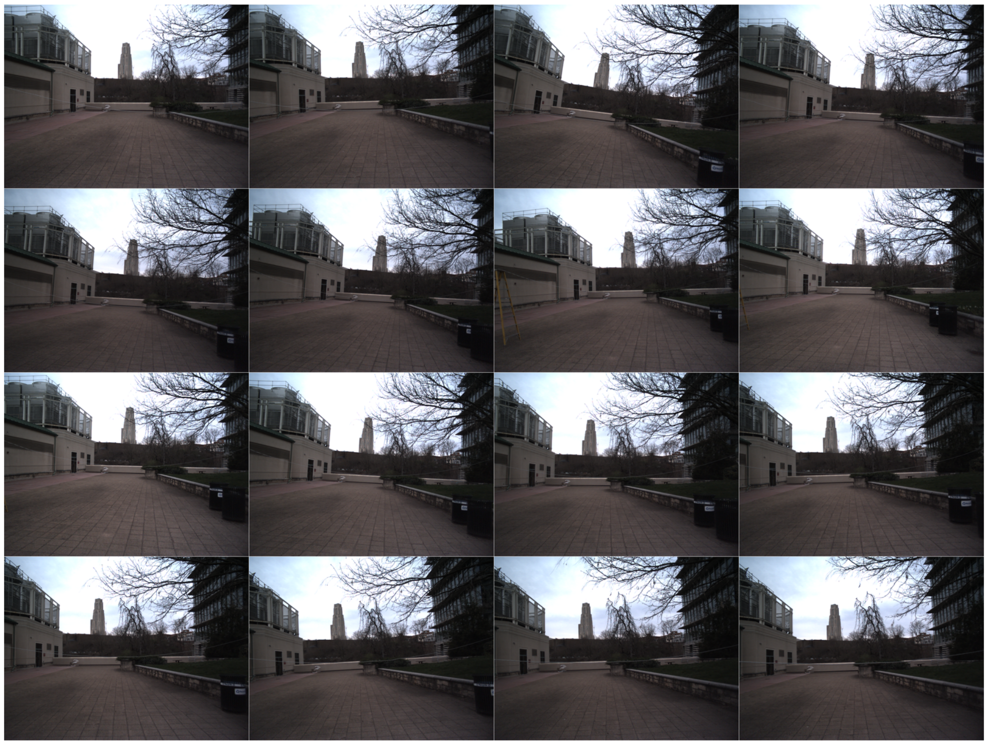
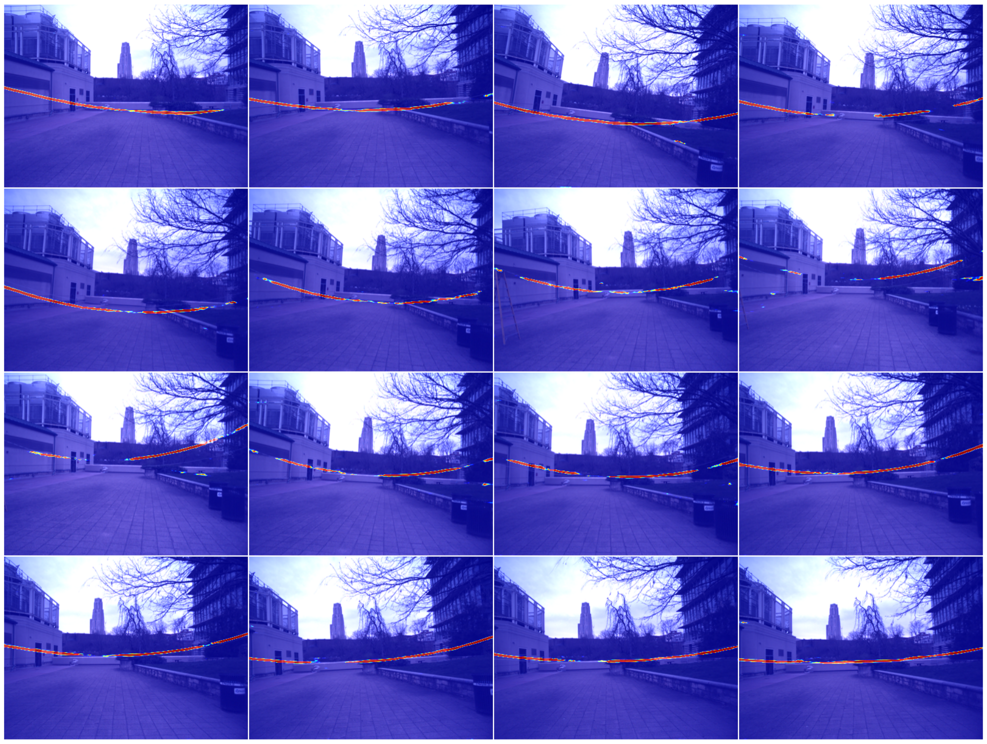
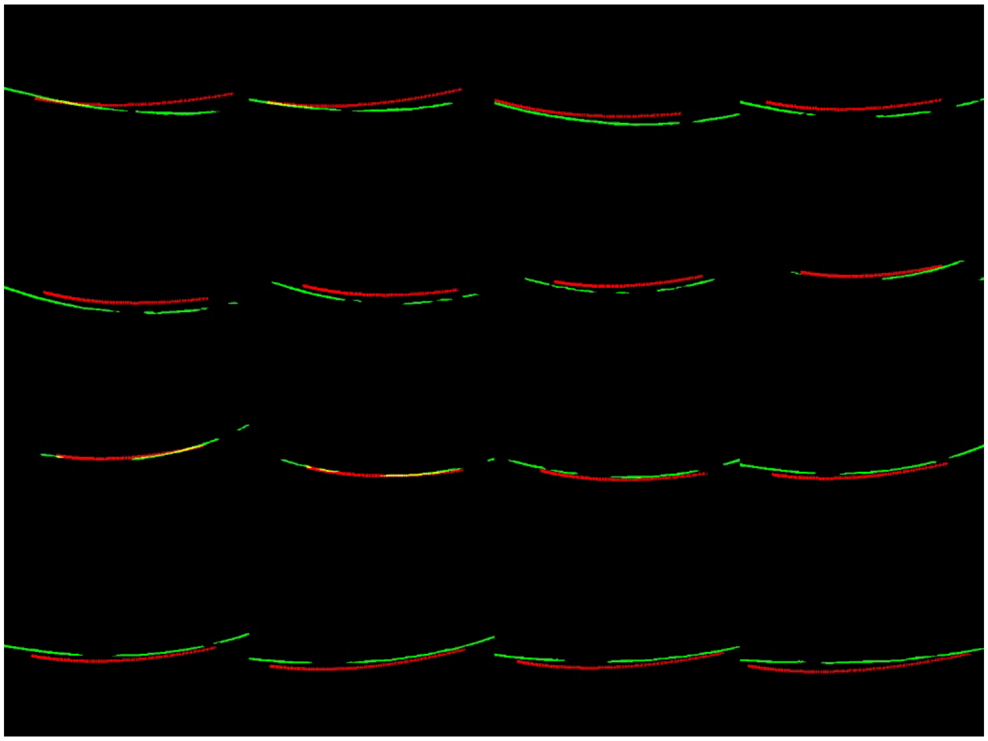


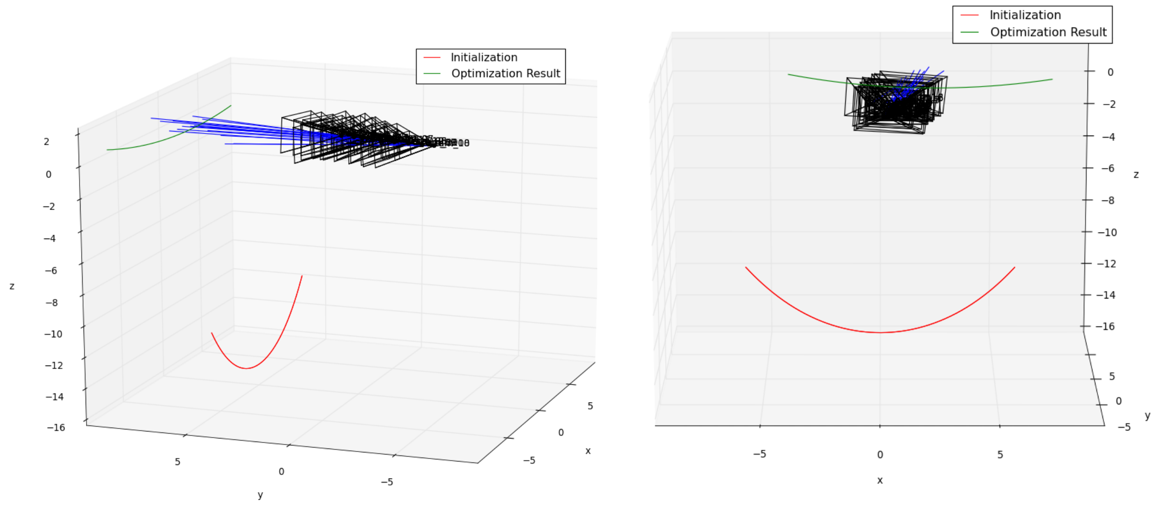
Figure 3.8: Wire detection in 2 handpicked keyframes of total 16 frames we perform multiview reconstruction on. We perform wire detection on a 300X400 image and our CNN runs at greater than 3 Hz on an Nvidia TX-2.


(a) Montage of RGB images. We use 300x400 resolution as input to our CNN.


(b) Wire segmentation confidence maps overlayed on the RGB images



(c) Result of our optimization on each keyframe. Green shows detected wires and red shows the projection of our resultant catenary model.



(d) Here we show the camera poses and the random catenary used for initialization in red. The fitted catenary is showed in green.

Figure 3.9: Experiments on real data. We extract 16 keyframes looking head on at a wire.

$$f(\Theta) = \frac{1}{2}\mathbf{r}^\top \mathbf{r} \tag{3.13}$$

$$\mathbf{J} = \frac{\partial \mathbf{r}}{\partial \Theta} = \frac{\partial f_i(p_i)}{\partial p_i}\,\frac{\partial p_i}{\partial P_i}\,\frac{\partial P_i}{\partial P_W}\,\frac{\partial P_W}{\partial \Theta} \tag{3.14}$$

$$r_i^j(\Theta) = \begin{cases} D_i(p_i^j), & \text{if } u_i^j \in [0,h), v_i^j \in [0,w) \\ D_i(p_i^b) + \left(d(p_i^b, p_i^j)/\sqrt{h^2+w^2}\right), & \text{otherwise} \end{cases} \tag{3.15}$$

where $p_i^b = (u_i^b, v_i^b)$, $u_i^b = \begin{cases} 0, & \text{if } u_i^j < 0 \\ h-1, & \text{if } u_i^j > h-1 \end{cases}$, and $v_i^b = \begin{cases} 0, & \text{if } v_i^j < 0 \\ w-1, & \text{if } v_i^j > w-1 \end{cases}$ (3.16)

Partial derivative of loss value f_i, w.r.t. pixel coordinates $p_i = \{u_i, v_i\}$: Within the image limit, it is simply the gradient of the distance transform. Outside, the image limits, it can be calculated analytically by differentiating the loss function expression above w.r.t. image coordinates, u_i and v_i.

For each j^{th} reprojected pixel, we can write

$$\frac{\partial f_i^j}{\partial p_i^j} = \begin{cases} \begin{bmatrix} \dfrac{\partial D_i}{\partial u_i^j} & \dfrac{\partial D_i}{\partial v_i^j} \end{bmatrix}, & \text{if } u_i^j \in [0,h), v_i^j \in [0,w) \\ \dfrac{1}{\sqrt{h^2+w^2}} \begin{bmatrix} \dfrac{u_i^j - p_b^j}{d(u_i^b, p_i^j)} & \dfrac{v_i^j - u_b^j}{d(p_i^b, p_i^j)} \end{bmatrix}, & \text{otherwise} \end{cases} \tag{3.17}$$

Then we can stack all $\dfrac{\partial f_i^j}{\partial p_i^j}$ s together to get $\dfrac{\partial f_i}{\partial p_i}$ in equation 3.14

Partial derivative of image coordinates p_i w.r.t. 3D points P_i in camera frame:
$p_i = \{u_i, v_i\}$ is obtained from $P_i = \{X_i, Y_i, Z_i\}$ by applying the camera intrinsics matrix.

$$p_i = \begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} f_x \dfrac{X_i}{Z_i} + u_0 \\ f_y \dfrac{Y_i}{Z_i} + v_0 \end{bmatrix} \tag{3.18}$$

where u_0, v_0 is the principal point and f_x and f_y are focal lengths.

We can differentiate the above to get:

$$\frac{\partial p_i}{\partial P_i} = \begin{bmatrix} \frac{\partial u_i}{\partial X_i} & \frac{\partial u_i}{\partial Y_i} & \frac{\partial u_i}{\partial Z_i} \\ \frac{\partial v_i}{\partial X_i} & \frac{\partial v_i}{\partial Y_i} & \frac{\partial v_i}{\partial Z_i} \end{bmatrix} \tag{3.19}$$

$$\tag{3.20}$$

$$= \begin{bmatrix} \frac{f_x}{Z_i} & 0 & -f_x \frac{X_i}{Z_i^2} \\ 0 & \frac{f_y}{Z_i} & -f_y \frac{Y_i}{Z_i^2} \end{bmatrix} \tag{3.21}$$

Partial derivative of points P_i in camera frame w.r.t. points P_W in the world frame:
In our case, the camera extrinsics are fixed, so the derivative $\frac{\partial P_i}{\partial P_W}$ is just the respective extrinsics rotation matrix:

$$P_C = R_{ext} * P_W + T_{ext} \tag{3.22}$$

$$\frac{\partial\ P_C}{\partial\ P_W} = R_{ext} \tag{3.23}$$

Partial derivative of points P_W in the world frame w.r.t. Catenary parameters Θ:
From equations 3.11 and 3.12, we can write P_W in terms of the catenary parameters, Θ. Below, $X_{cat}, Y_{cat}, Z_{cat}$ refer to coordinates of the curve in the catenary frame.

$$P_W(\Theta) = R(\psi_v)P_{cat} + T(x_v, y_v, z_v) \tag{3.24}$$

$$P_W(\Theta) = \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} = \begin{bmatrix} cos(\psi_v) & -sin(\psi_v) & 0 & x_v \\ sin(\psi_v) & cos(\psi_v) & 0 & y_v \\ 0 & 0 & 1 & z_v \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_{cat} \\ Y_{cat} \\ Z_{cat} \\ 1 \end{bmatrix} \tag{3.25}$$

$$P_W(\Theta) = \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} = \begin{bmatrix} cos(\psi_v) & -sin(\psi_v) & 0 & x_v \\ sin(\psi_v) & cos(\psi_v) & 0 & y_v \\ 0 & 0 & 1 & z_v \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_{cat} \\ 0 \\ a\left(cosh\left(\frac{X_{cat}}{a}\right) - 1\right) \\ 1 \end{bmatrix} \tag{3.26}$$

$$P_W(\Theta) = \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} = \begin{bmatrix} X_{cat}cos(\psi_v) + x_v \\ X_{cat}sin(\psi_v) + y_v \\ a\left(cosh\left(\frac{X_{cat}}{a}\right) - 1\right) + z_v \\ 1 \end{bmatrix} \tag{3.27}$$

The above lends as the required Jacobian matrix.

$$\frac{\partial P_W}{\partial \Theta} = \begin{bmatrix} \frac{\partial X_W}{\partial x_v} & \frac{\partial X_W}{\partial y_v} & \frac{\partial X_W}{\partial z_v} & \frac{\partial X_W}{\partial \psi_v} & \frac{\partial X_W}{\partial a} \\ \frac{\partial Y_W}{\partial x_v} & \frac{\partial Y_W}{\partial y_{cat}} & \frac{\partial Y_W}{\partial z_{cat}} & \frac{\partial Y_W}{\partial \psi_v} & \frac{\partial Y_W}{\partial a} \\ \frac{\partial Z_W}{\partial x_v} & \frac{\partial Z_W}{\partial y_{cat}} & \frac{\partial Z_W}{\partial z_{cat}} & \frac{\partial Z_W}{\partial \psi_v} & \frac{\partial Z_W}{\partial a} \end{bmatrix} \tag{3.28}$$

$$\tag{3.29}$$

$$= \begin{bmatrix} 1 & 0 & 0 & -X_{cat} sin(\psi_v) & 0 \\ 0 & 1 & 0 & X_{cat} cos(\psi_v) & 0 \\ 0 & 0 & 1 & 0 & cosh\left(\frac{X_{cat}}{a}\right) - \frac{X_{cat}}{a} sinh\left(\frac{X_{cat}}{a}\right) - 1 \end{bmatrix} \tag{3.30}$$

Now writing X_{cat} in terms of output X_W and input variable x_v, we can get:

$$X_{cat} = \frac{X_W - x_v}{cos(\psi_v)} \tag{3.31}$$

Back-substitution yields:

$$= \begin{bmatrix} 1 & 0 & 0 & (X_W - x_v)\, tan(\psi_v) & 0 \\ 0 & 1 & 0 & (X_W - x_v) & 0 \\ 0 & 0 & 1 & 0 & cosh\left(\frac{X_W - x_v}{a\, cos(\psi_v)}\right) - \frac{1}{a} sinh\left(\frac{X_W - x_v}{a\, cos(\psi_v)}\right) - 1 \end{bmatrix} \tag{3.32}$$

3.6 Conclusion

We proposed a multiview algorithm for wire reconstruction using a 5 parameter catenary model. Our algorithm is robust to bad initializations, which we validate on both real data, and simulated data with random camera poses and realistic scenarios.

3.7 Future Work

3.7.1 Computing the width of the catenary

We use a hard-coded width for the catenary, which works well in non-degenerate cases when we have good multiple views providing enough information, as long as our width estimate is somewhat reasonable. However, this is non-ideal and can fail in cases like those shown in Figure 3.10.

This can be solved by finding an upper bound on the extent of the catenary via 3D geometry. Each keyframe tells us the rays along which the curve may lie in, which together form surface of the feasible catenary hypotheses. We can relax the above, by founding

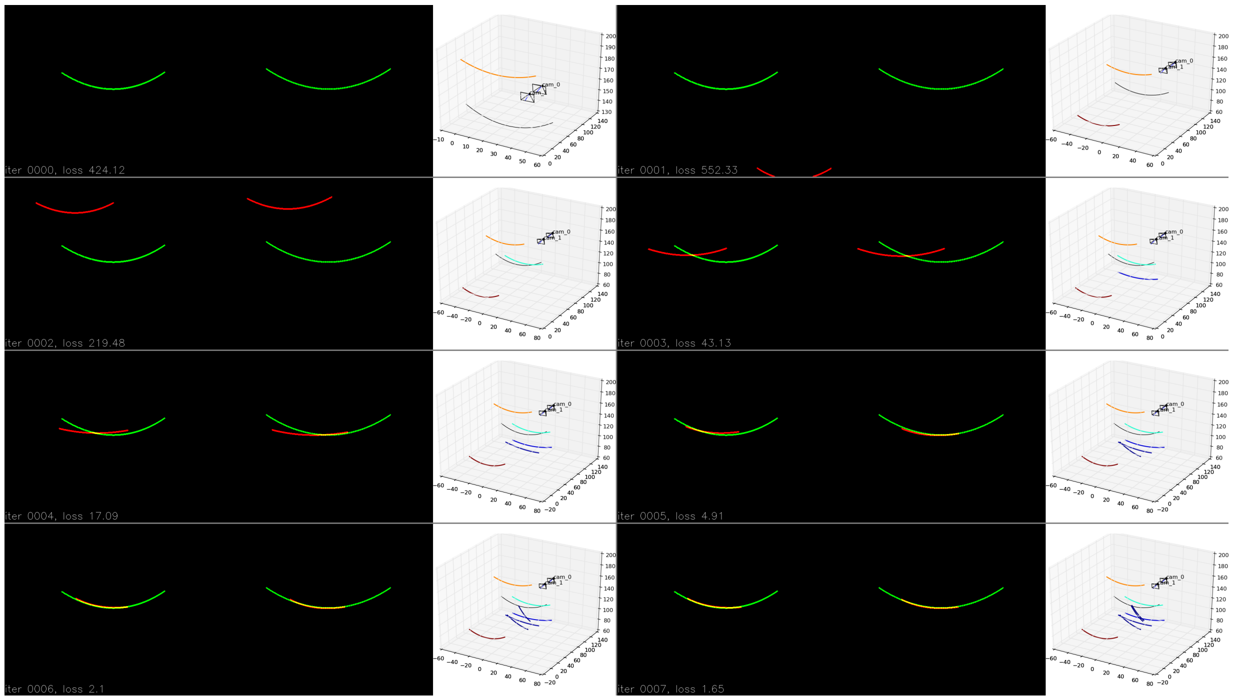


Figure 3.10: Failure case for an avoidance scenario with two cameras. Notice that even though the loss is going down, we obtain the wrong result due to degeneracy. We encourage the reader to examine the corresponding animations available at the thesis webpage.

a bounding viewing frustum in which the catenary solution may lie. As we get more keyframes, we can compute the maximum possible region where the catenary may lie. From this region, we can compute the maximum possible width of the catenary, which will also change with the depth. Factoring this into our framework would make it make it much more robust.

3.7.2 Closing the loop by modeling uncertainty

To close the loop with a motion planner for tasks like wire avoidance and active perception of wires in order to reconstruct them accurately while flying in the most efficient manner possible, the first thing that needs to be done is modeling the uncertainty of the wire reconstruction itself. The traditional way of doing this in SLAM literature has been by modeling landmarks (in this case catenary curves) in the inverse-depth coordinates of the first keyframe in which they are observed. The reason for doing this is that in inverse depth coordinates, we can model uncertainties as Gaussians. One may imagine an uncertainty in the bearing measurement can be accomplished by defining a 2D Gaussian in the image space. And as inverse-depth is related inversely to depth, a Gaussian in inverse-depth maps to a long tail distribution in the depth coordinates. This is explained in the next chapter on model-free mapping, in Figures 4.2a and 4.2b. The covariance in the inverse depth coordinates can be obtained from the solution of the non-linear optimization itself. And this covariance in the inverse depth of the catenary vertex can then be mapped to 3D coordinates of the vertex in the world frame, by linearizing the mapping involved. Finally we can also linearize the equation of the catenary itself, to get a covariance of each point of the catenary from the covariance estimate of the catenary vertext. We elucidate these concepts over the upcoming paragraphs.

Catenary model in inverse-depth coordinates

For us, this means that we need to change three parameters in our model which define the translation of the plane of catenary with respect to the world frame. Hence, we need to change the sub-tuple $\{x_v, y_v, z_v\}$ from our parameterization Θ to $\{u_v^{C_1}, v_v^{C_1}, d_v^{C_1}\}$, where $(u_v^{C_1}, v_v^{C_1})$ are the projected pixel coordinates of the catenary vertex and $d_v^{C_1}$ is the corresponding inverse depth, all defined w.r.t. the first keyframe in which the wire is observed, C_1.

We can define this new model as Ψ. This is related with our previous model Θ by the intrinsics and the matrix matrices, which are again differentiable. Essentially, inverting the camera matrices defines a new mapping $f_{\Psi \mapsto \Theta}$ and the corresponding jacobian $J_{\Psi \mapsto \Theta}$, which when chained together with our transformations in Equation 3.6 and Figure 3.5, is nothing but a reparameterization of our model introduced before.

$$\Psi = \{u_v^{C_1}, v_v^{C_1}, d_v^{C_1}, \psi_v^W, a\} \tag{3.33}$$

First, let us define the catenary vertex coordinates in the camera frame C_1, $P_v^{C_1} = \{x_v^{C_1}, y_v^{C_1}, z_v^{C_1}\}$, by undoing the camera projection:

$$P_v^{C_1} = \begin{bmatrix} x_v^{C_1} \\ y_v^{C_1} \\ z_v^{C_1} \end{bmatrix} = \begin{bmatrix} \dfrac{u_v^{C_1} - u_0}{f_x d_v^{C_1}} \\ \dfrac{v_v^{C_1} - v_0}{f_y d_v^{C_1}} \\ \dfrac{1}{d_v^{C_1}} \end{bmatrix} \tag{3.34}$$

$$\frac{\partial P_v^{C_1}}{\partial \{u_v^{C_1}, v_v^{C_1}, d_v^{C_1}\}} = \begin{bmatrix} \dfrac{1}{f_x d_v^{C_1}} & 0 & -\dfrac{u_v^{C_1} - u_0}{(d_v^{C_1})^2 f_x} \\ 0 & \dfrac{1}{f_y d_v^{C_1}} & -\dfrac{v_v^{C_1} - v_0}{(d_v^{C_1})^2 f_y} \\ 0 & 0 & -\dfrac{1}{(d_v^{C_1})^2} \end{bmatrix} \tag{3.35}$$

Now, we can transform the catenary vertex $P_v^{C_1}$ to the world frame using the *inverse* of the extrinsics matrix and get $P_v^W = \{x_v^W, y_v^W, z_v^W\}$. Note that the following equation is inverting the homogeneous transform of the extrinsics matrix. If $\{R_{ext}^{C_1}, T_{ext}^{C_1}\}$ were the rotation and translational components of the extrinsics, then $\{{R_{ext}^{C_1}}^\top, {R_{ext}^{C_1}}^\top T_{ext}\}$ will be the corresponding components of the inverse transform.

$$P_v^W = {R_{ext}^{C_1}}^\top P_v^{C_1} + {R_{ext}^{C_1}}^\top T_{ext}^{C_1} \tag{3.36}$$

$$\frac{\partial P_v^W}{\partial P_v^{C_1}} = {R_{ext}^{C_1}}^\top \tag{3.37}$$

Note that we can extend the above partial derivatives in Equations 3.37 and 3.35 by $I_{2\times2}$ identity matrix for the two parameters that are the same in both models, ψ and a: to obtain $J_{\Psi\mapsto\Theta} = \frac{\partial\Theta}{\partial\Psi}$.

Projecting covariances

We will be solving for Ψ, instead of Θ. The previous subsection introduced the mapping from the model parameters ψ to the residual vector $\mathbf{r}$, $f_{\Psi\mapsto r}$, and the corresponding jacobian $J_{\Psi\mapsto r}$:

$$\Psi^* = \arg\min_{\Psi} f(\Psi) \tag{3.38}$$

The covariance matrix of the solution Ψ^*, following the non-linear least squares theory is then given by

$$\Sigma_\Psi = (J_{\Psi\mapsto r} J_{\Psi\mapsto r}^\top)^{-1} \tag{3.39}$$

And we can get a first order estimate of the uncertainty in the 3D coordinates of the catenary by linearizing the mapping from inverse depth model parameters to the original model parameters:

$$\Sigma_\Theta = J_{\Psi\mapsto\Theta} \Sigma_\Psi J_{\Psi\mapsto\Theta}^\top \tag{3.40}$$

Similarly, we can linearize the catenary equation to get an estimate of the covariance for each point on the curve.

Finally collision checks, we can introduce a standard 6σ threshold to infer occupancy. For closing the loop, we can then use our reactive planner, which will be introduced in Chapter 4.

3.7.3 Handling multiple wires

While our contributions lay the foundations of a model-based reconstruction algorithm, they are still lacking in the context of pragmatism. Real world scenarios have multiple wires, and their projected curves in the images would generally be intersecting. This is highly problematic for our distance transform based residual vector. One can imagine that running our algorithm for even a single key frames with roughly parallel projections in the images can lead to the wrong result, depending on the initialization. This is because the distance transform will have a local minima right in the middle of the projections!

The issue highlighted above can be resolved if we assume we have instance segmentations of the wire detections, as well ass wire-to-wire correspondences across all keyframes. Then we can simply isolate each wire, and run our proposed algorithm W times, where W is the number of wires we saw.

This begets two questions. The first question we need to answer is how do we get a per-wire instance segmentations in each keyframe. For that, B-splines or something simpler like a third order polynomial can be fitted to the segmentations of the catenaries, depending on the use case.

The second question we have to think about is that assuming we have per-wire instance segmentations in each keyframe, how can we establish wire-to-wire correspondences. If we have a wire avoidance scenario, we will need to run our algorithm by reprojecting the wire-to-wire to association from the previous keyframe.

Intersecting curves in the random camera placement scenario is the hardest case. For this, we refer the reader to the recent work by [Liu et al., 2017]. If instance segmentations in each view are known, wire-to-wire correspondence is a combinatorial problem, over the feasible associations found from epipolar geometry, as can be seen in Figures 4 and 6 of [Liu et al., 2017]. This would need an establishment of a global constraint or cost function, that the reconstructions are sufficiently long catenary curves for example.

Chapter 4

Model-free Mapping and Reactive Avoidance

4.1 Acknowledgements

This chapter presents work done with Geetesh Dubey (MSR 2017). The majority of this chapter is from our IROS 2018 publication [Dubey et al., 2018], which itself builds on Geetesh's work in IROS 2017 [Dubey et al., 2017] and his Masters thesis [Dubey, 2017].

4.2 Introduction

In this chapter, we present an approach and a system which can tackle the three challenges of wire detection, reconstruction and avoidance, along with a generic stereo-based obstacle avoidance pipeline. First, we perform monocular wire detection using a convolutional neural network under the semantic segmentation paradigm, and obtain a confidence map of wire pixels. Along with this, we also use a binocular stereo pair to detect other generic obstacles. We represent wires and generic obstacles using a disparity space representation and do a C-space expansion by using a non-linear sensor model we develop. Occupancy inference for collision checking is performed by maintaining a pose graph over multiple disparity images. For avoidance of wire and generic obstacles, we use a precomputed trajectory library, which is evaluated in an online fashion in accordance to a cost function over proximity to the goal. We follow this trajectory with a path tracking controller. Finally, we demonstrate the effectiveness of our proposed method in simulation for wire mapping, and on hardware by multiple runs for both wire and generic obstacle avoidance.

In this chapter, we extend our previous works, DROAN (Disparity-space Representation for Obstacle AvoidaNce) [Dubey et al., 2017] and monocular wire detection using synthetic data and dilated convolutional networks [Madaan et al., 2017], by stitching monocular and binocular stereo observations in a seamless way to enable detection and avoidance of generic obstacles and especially hard to perceive thin obstacles such as wires and power lines, in a novel fashion.

For monocular wire detection, we use a semantic segmentation paradigm using fully convolutional neural networks [Long et al., 2015] with multiple dilation layers [Yu and Koltun, 2015] and no downsampling, trained on synthetically generated wires superimposed on real flight images as explained in [Madaan et al., 2017]. For a complete obstacle avoidance system however, a monocular camera is not enough. Hence, we use a binocular stereo pair as a generic obstacle perception sensor. The standard paradigm for doing this involves using the point cloud obtained from the stereo images to maintain and update a 3D occupancy map, where obstacles are then expanded according to the robot's radius to do motion planning by treating the robot as a point-sized obstacle. However, such an occupancy map is expensive to maintain and store in practice. Hence, we use the inverse depth or disparity space as our configuration space [Matthies et al., 2014, Dubey et al., 2017]. We model the sensor uncertainty in order to do a C-space expansion of the observed data associated with generic obstacles. For estimating the depth of wires, we first threshold and filter the confidence map obtained from the CNN, and make a synthetic disparity image for the wire obstacle with a conservative depth estimate. Then, a pose graph maintained over multiple disparity images is used to infer occupancy for both wire and generic obstacle in order to perform collision checking in the planning stage. For planning, we use a receding horizon planner deploying a precomputed trajectory library composed of dynamically feasible segments to a pre-specified goal location. The precomputed candidate trajectories are then evaluated online to assign a cost based on how close they get to the goal in position, heading and remain free from collisions. Finally a path tracking controller from [Dubey et al., 2017] is used to follow the selected candidate trajectory. This process continues at the rate of the fastest observation sensor i.e. the disparity image frequency of 10 Hz. We test the effectiveness of wire depth estimation in simulation as shown in Figure 4.5. Consequently, we demonstrate our pipeline on multiple runs in both an outdoor environment with wire obstacles and in an indoor environment with generic obstacles.

4.3 Related Work

4.3.1 Depth or Disparity map based Obstacle Detection

Most approaches generate point clouds from disparity images and generate 3D evidence or occupancy grids to infer occupancy for collision checking [Heng et al., 2011, Andert and Adolf, 2009]. Gohl et. al [Gohl et al., 2015] proposed to use a spherical coordinate based gridmap suitable for stereo sensors, but this requires that each disparity map has to be converted to a 3D point cloud before being injected into the 3D gridmap.

Working with 3D gridmaps is both memory intensive for large occupancy maps, and computationally expensive for registration and book keeping when scrolling or moving the grid along with the robot. OctoMaps [Hornung et al., 2013] have recently become popular due to their efficient structure for occupancy mapping. However, due to excess noise in stereo sensor generated data at long ranges, often a smaller map is maintained and full stereo sensor data is not used. A pushbroom stereo scanning method is proposed in [Barry and Tedrake, 2015] for obstacle detection for MAVs flying at high speeds, however it is capable of only detecting a prefixed disparity which it accumulates as the robot moves in

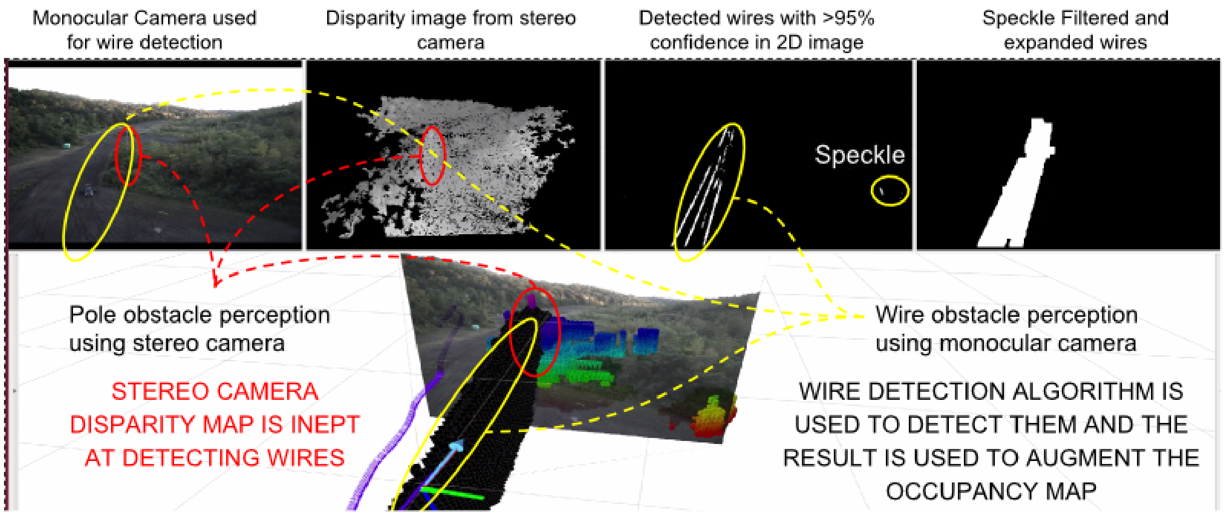

Monocular Camera used for wire detection
Disparity image from stereo camera
Detected wires with >95% confidence in 2D image
Speckle Filtered and expanded wires
Speckle
Pole obstacle perception using stereo camera
STEREO CAMERA DISPARITY MAP IS INEPT AT DETECTING WIRES
Wire obstacle perception using monocular camera
WIRE DETECTION ALGORITHM IS USED TO DETECT THEM AND THE RESULT IS USED TO AUGMENT THE OCCUPANCY MAP

Figure 4.1: Wire Mapping using Monocular Detection and Obstacle Perception using Disparity Expansion. Obstacles perceived using a stereo-camera are shown in red ellipses, and wire obstacles detected from monocular camera are shown with yellow ellipses. The bottom half shows the obstacle grid represented by our pose of graph of disparity images. Wires are shown as black boxes, and stereo camera perceived obstacles are colored by height.

the environment, and therefore is limited to myopic sensing.

We base our work on [Matthies et al., 2014] which proposed a Configuration-Space (C-Space) expansion step to apply an extra padding around disparities based on robot size, but it also fails to use complete sensor data as only the closer occurring obstacles are represented by their method without considering any sensor uncertainty.

4.4 Approach

We now explain each module in our pipeline.

4.4.1 Monocular Wire Detection

We treat wire detection as a semantic segmentation problem in monocular images and leverage our work in [Madaan et al., 2017] for the same. Due to unavailability of a large dataset of synthetic wires, we generate a large number of synthetic wires using a ray-tracing engine [POV-Ray, 2004, **?**], and superimpose them on publicly available videos' frames, in order to make an ImageNet analogue for pre-training the network. We then do a grid search over multiple dilated convolutional networks we designed in [Madaan et al., 2017], and pick the best in terms of accuracy and performance on the Nvidia TX-2, specifically the architecture k32-k32-k32-k32-d1-d2-d4-d8-d16-d1-k2 as explained in Table III of [Madaan et al., 2017]. For good performance on our real world experiments, we fine-tuned the network weights on an assortment of a few manually labelled images with real wires we collected ourselves.

The output of the CNN is a confidence map, where each pixel maps to a confidence value $\in [0, 1]$ of that pixel belonging to the wire class. First, we threshold this map and remove pixels with confidence < 0.95. The generated mask at times has false positives in terms of speckles or disconnected regions similar to stereo-sensor generated disparity maps. Since, our current approach is very conservative about avoiding wire obstacles, such speckles are detrimental when planning for avoidance of obstacles, resulting in erratic robot motion, or in the worst case, complete failure to find a plan to avoid the wire. Hence, we run it through a speckle filter to get rid of small non-wire patches. The filtered mask is then treated as the wire-obstacle map as shown in Figure 4.1. In Section 4.4.3, we discuss how we convert this map to a disparity space map for seamless integration into the occupancy inference pipeline.

4.4.2 Characterizing Disparity Space Uncertainty

Modeling perception error

In order to detect generic obstacles, we use a binocular stereo camera, and in this section we explain the need for using a sensor model and how we develop the same. We use the disparity or inverse depth image for obstacle representation as it naturally captures spatial volume according to the sensor resolution [Gohl et al., 2015], which means that implicitly nearby obstacles have a denser representation, whereas obstacles which are further away are represented with a sparser resolution.

In stereo vision, the depth z of a pixel (u, v) and the corresponding disparity value d are related as:

$$z = \frac{bf}{d} \tag{4.1}$$

where b is stereo baseline and f is the focal length in pixels. The 3D coordinates of the corresponding point can be expressed as

$$P(x, y, z) = (uz/f, vz/f, z) \tag{4.2}$$

The accuracy of the stereo setup is drastically affected as the disparity decreases. The error in depth increases quadratically with depth, as can be seen by differentiating equation (4.1) w.r.t d and then back-substituting for z as explained in [Dubey et al., 2017], i.e. $\partial z \sim z^2$.

It is evident that the sensor model is non-linear in the depth space and has a long tail distribution, as can be seen in Figure 4.2a. However, in the disparity space, it can be modelled as a Gaussian distribution, $\mathcal{N}(d, \sigma^2)$, where d is the observed disparity, and σ^2 is the covariance parameter of the sensor model which we determine empirically in the next section. The reasoning behind using a Gaussian in disparity space is that disparity error is primarily caused due to correspondence error while matching pixels along the epipolar line. Figure 4.2a shows that a Gaussian pdf in disparity space captures a difficult to model pdf in depth which has an elongated tail on one side and a compressed tail on the other. The blue curve shows how depth is related to disparity. We can see that a Gaussian pdf centred around a low disparity value (red on X-axis) maps to a long tail distribution (red on Y-axis) in the depth space. Similarly, a Gaussian around a larger disparity value (green

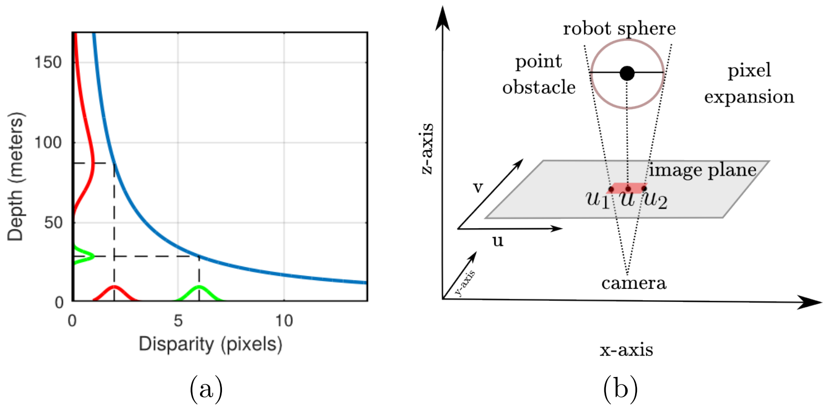


Figure 4.2: (a) Disparity and corresponding depth values are shown in blue. A Gaussian pdf centred around a low disparity value (red on X-axis) maps to a long tail distribution (red on Y-axis) in the depth space, whereas a Gaussian for larger disparity value maps to less uncertainty in depth space. This explains how we capture difficult to model distributions in the depth space with a Gaussian counterpoint in the disparity space. (b) Shows the pixel-wise area expansion of a point obstacle according to robot size in the image plane.

on X-axis) maps to less uncertainty in depth (green on Y-axis). This fits well with the fact that disparity naturally captures spatial volume according to the sensor resolution, and establishes our motivation to use disparity image space domain directly for occupancy inference rather than resorting to depth or 3D spatial domain.

Finding the sensor model parameters empirically

In this section, we explain how we find the σ value from the previous section, given a disparity image and corresponding ground truth value of depth. To this end, we generate disparity using an FPGA based solution [Schauwecker, 2015], and obtain the ground truth depth value by placing a known chequered board pattern in front of the sensor. Then, we generate a histogram of disparity errors by collecting several samples of disparity values corresponding to each corner pixel. Figure **??** shows our setup at a distance of 2.5 m. Data was sampled at multiple distances to model the error, and we fit a Gaussian distribution to the obtained histogram to obtain a value of $\sigma = 0.23$, as depicted in Figure 3 of Dubey et al. [2018].

4.4.3 Configuration-Space Expansion in Disparity Space

Now that we have established a sensor model in disparity space, in this section we explain how we do configuration space expansion for collision checking. We capture the volume occupied by an obstacle using two virtual *limit surfaces*, one each for front and back by generating two corresponding disparity images via the sensor model we developed in the previous section. Each pixel in these two images effectively captures the range of disparity based on the robot size and the sensor error model as shown in the Figure 4.2a. We do the expansion in two steps : the first one expands disparities along the image XY axes as shown in Figure 4.2b, and the second step expands along the depth dimension (image Z axis) as shown in Figure 4.3.

The first step does area expansion of the obstacle pixel (u, v) in the disparity image, to occupy a set of pixels ranging from $[u_1, u_2]$ to $[v_1, v_2]$ after inflation. This is similar to [Matthies et al., 2014] which introduced a procedure to generate a look-up table (LUT) to obtain the expansion mappings across both image dimensions, $u \rightarrow [u_1, u_2]$ and $v \rightarrow [v_1, v_2]$, given disparity value d. However, we differ from [Matthies et al., 2014] in that we also incorporate the sensor error modelled in the previous section. To ensure robot safety while flying, we introduce another parameter, λ which is a sigma multiplier in the expansion step depending on how far the obstacle is from the robot. The intuition here is that the nearby obstacles are expanded more, whereas the obstacles further away are expanded less, to enable long-range deliberate planning for exploration tasks. Thus, instead of looking up for the raw disparity value d from the LUT as done in [Matthies et al., 2014], we rather look up for $(d + \lambda\sigma)$. By varying λ we ensure safe planning at short range and a more optimistic planning at long range.

The second step in C-space expansion expands disparities in the depth dimension to get values for front and back images using equation (4.3), as shown in Figure 4.3. These images represent the maximum and minimum disparities for every pixel respectively:

$$d_{\mathrm{f}} = \frac{bf}{z - r_v} + \lambda\sigma \quad , \quad d_{\mathrm{b}} = \frac{bf}{z + r_v} - \lambda\sigma \tag{4.3}$$

where r_v is the expansion radius based on robot size, d_{f} and d_{b} are the computed front and back disparities which encompass the obstacle. As shown in illustration on left side of Figure 4.3, the red area around the original disparity of obstacle is the padding generated in the expansion step. This padding is based on the robot size and sensor error model. The reader is advised to refer to our previous work [Dubey et al., 2017] for further details on the algorithm used for C-Space expansion. **Note:** For the experiments in this paper we set $\lambda = 1$.

C-Space expansion of detected wire pixels

As explained in Section 4.4.1, we obtained a thresholded and filtered confidence map of detected wires from monocular images. However, we have no estimate of depth to generate a disparity map, so we assume the wire to be present between 4 m and 20 m distance. Using the 4 m assumption we apply this prior to all wire-pixels and obtain a synthetic disparity image. Using this synthetic disparity we apply the steps from the previous subsection to generate a frontal expansion. For back expansion however, we apply a fixed padding using second assumption of wire being as far as 20 m as shown in Figure 4.4. These assumptions can be attributed to generation of fixed size solid planes in 3D space and by using multiple such observations we can triangulate the real location of wire within those ranges. The motivation of using these close proximity depths is to avoid wires as early as possible or move wire obstacles out of robot view by changing course of motion. We test the effectiveness of depth estimation in simulation as shown in Figure 4.5.

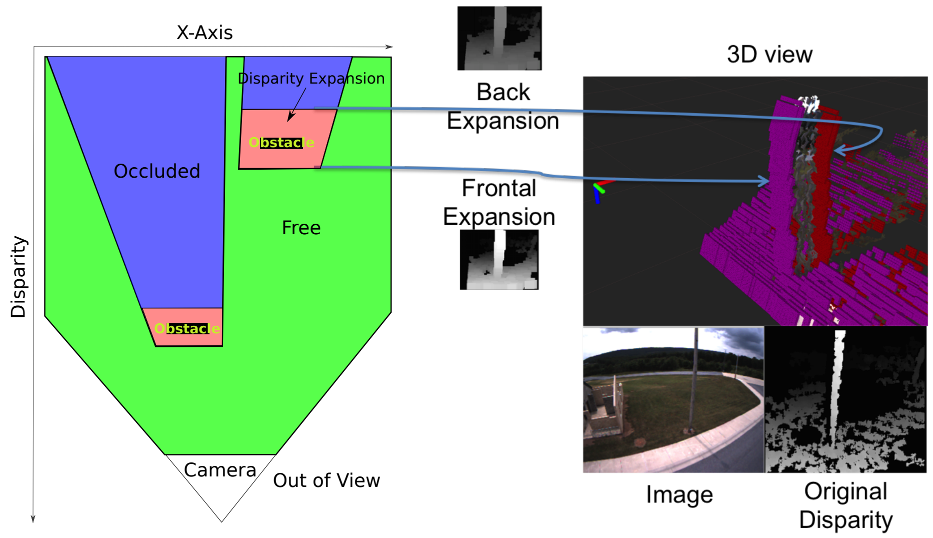


Figure 4.3: Disparity expansion shown as point cloud. The pink and red point cloud represent the foreground and background disparity limits.

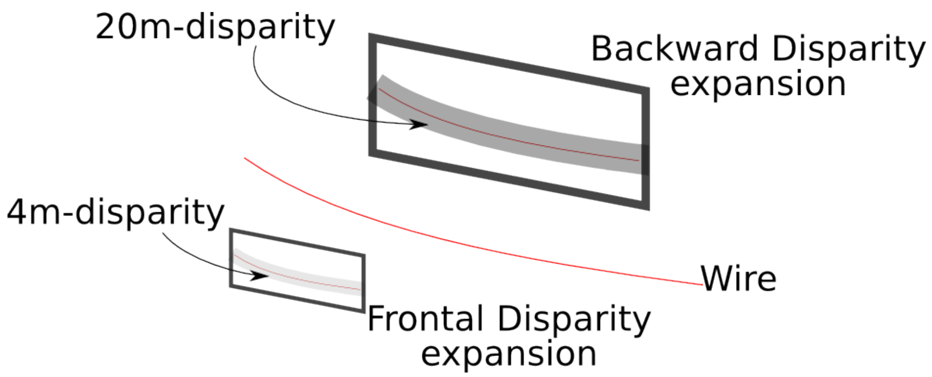
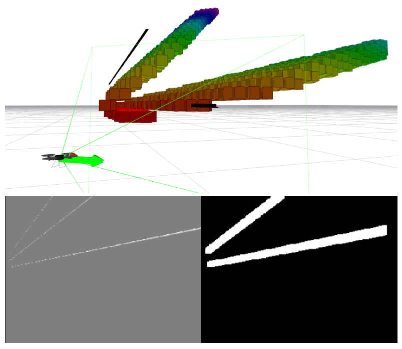
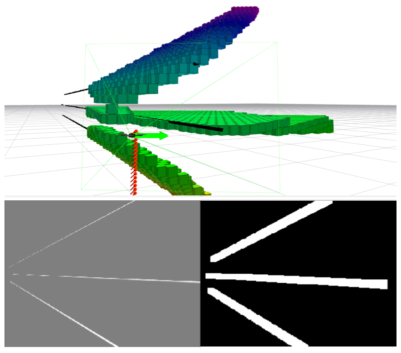
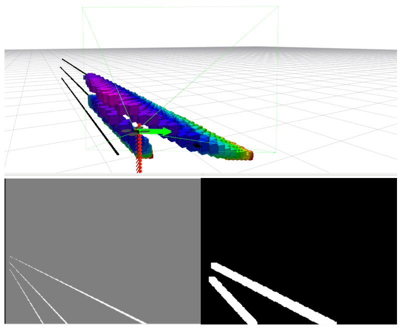
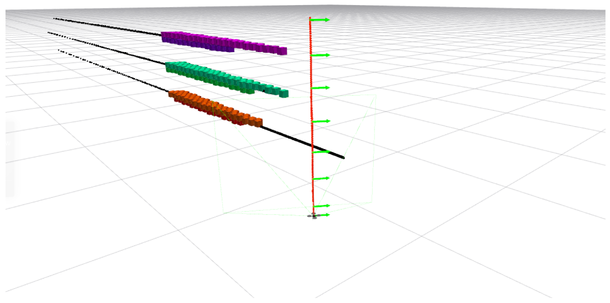


Figure 4.4: Synthetic disparity generation and expansion of wire. The frontal and backward image planes show the thin wire projection along with the result of applied expansion steps.


(a) Low elevation


(b) Medium elevation


(c) High elevation


(d) Reconstruction result: Oblique view

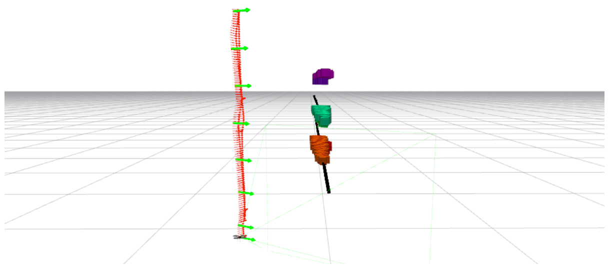

(e) Reconstruction result: Side view

Figure 4.5: (a)-(c) show single observations at different heights using our representation. For each subfigurem, the bottom row shows the simulated camera RGB observation on the left (white on gray background), and the binary wire detection post *expansion* on the bottom right. The top row shows the corresponding assumption derived from the binary detection: we assume the wire is uniformly distributed between a range of depth (4m to 20m from the camera), as is depicted by the voxels colored by their height. The ground truth wires are shown as black linear point cloud. (d)-(e) show the oblique and side views of the result of our fusion and occupancy inference. As can be inferred qualitatively, our approach is successful in mapping the three wires. Again, in (d)-(e) the voxels are colored by their heights, and the ground truth wires are shown as black lines. The robot odometry is shown in red, and the individual nodes of the pose-graph are depicted by the green arrows.

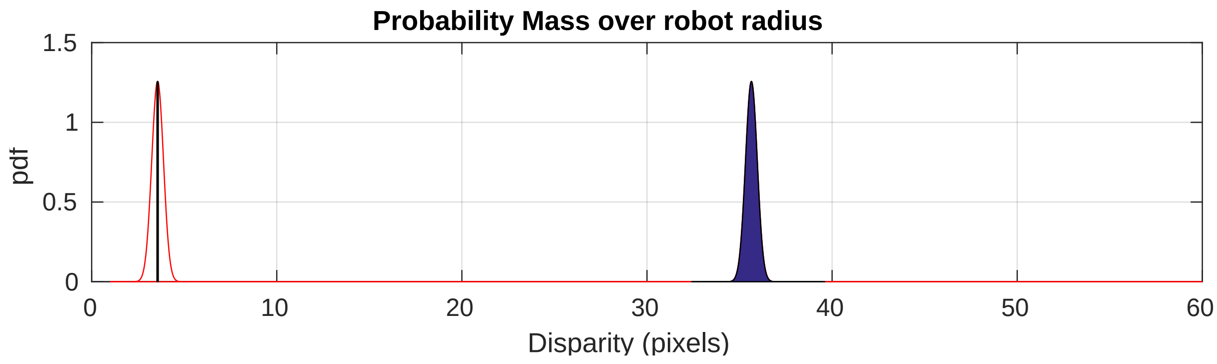


Figure 4.6: Probability mass (shown in blue area) occupied by a robot of radius $0.5m$ at a distance of $50m(3.5px)$ and $5m(35.6px)$. As the distance increases or disparity decreases the probability mass occupied by the robot varies.

4.4.4 Probabalistic Occupancy Inference

Occupancy inference is the method to derive occupancy of a volume using all the observations or evidence we have. This is a widely studied topic and is often not a trivial problem. Chapter 9 of [Thrun et al., 2005] explains this in detail. 3D Evidence grids or occupancy grid maps are practical methods for fusion of different measurements taken over time. The proposed pose-graph, explained later in Section 4.4.6, enables similar fusion of multiple observations. In [Dubey, 2017] we showed how an inverse sensor model for stereo-cameras can be used for occupancy inference in a probabilistic manner, i.e. similar to log-odds used in evidence grids. We also compared it to a easy to compute *Confidence Function.* Following is a brief discussion, for details please refer [Dubey, 2017].

Confidence Function for inference in Disparity Space

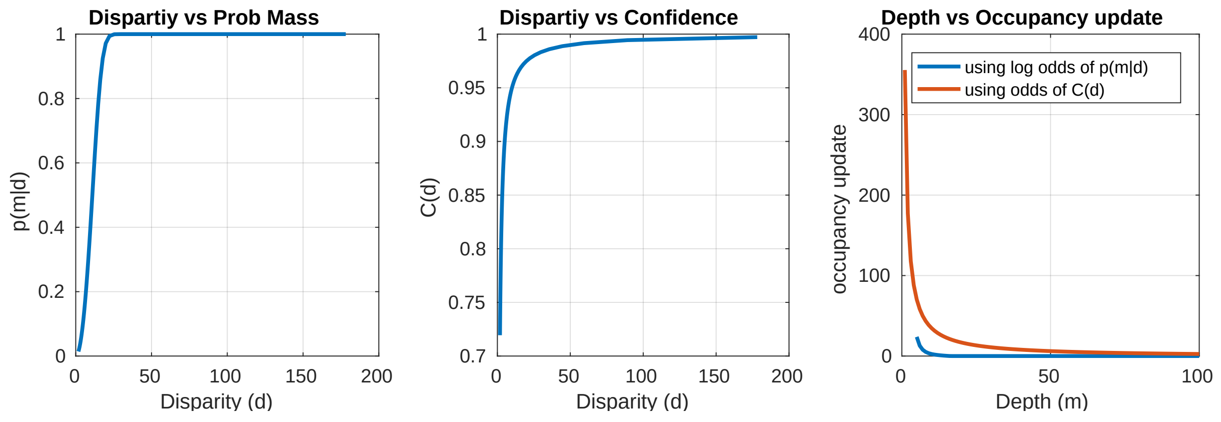
Using the Gaussian sensor error model as explained earlier we make the following analysis. Figure 4.6 shows how the occupied volume changes in disparity space given a fixed robot size. Figure 4.7a shows the probability mass as a function of inverse depth or disparity and shows that the same Gaussian distribution at different disparities the actual range of disparity that the robot would occupy falls drastically.

However, it is difficult to compute the probability mass online as it requires integration of inverse-sensor-model, hence we propose a new confidence function which is inexpensive to compute online. Given the standard deviation of correspondence error σ, we compute confidence of a disparity state d in the following manner.

$$C(d) = \frac{(d - \sigma)}{d} \tag{4.4}$$

Confidence measure from equation(4.4) gives us a measure of how much can we trust a given disparity for occupancy inference. Figure 4.7b shows how this measure relates to disparity. For details please refer [Dubey, 2017].

We compare the occupancy update using the log-odds probabilistic inference and the proposed confidence inference method. Figure 4.7c shows the plot of occupancy update



(a) (b) (c)

Figure 4.7: (a) Probability Mass of occupancy, given a robot at disparity d pixels. This curve is used to compute the log-odds to infer occupancy. (b) Disparity vs Confidence plot obtained from equation (4.4). This is inexpensive to compute online compared to computation of probability mass which involves integration of inverse-sensor-model over robot radius. (c) Occupancy update comparison between log-odds and proposed confidence inference. Confidence based occupancy update is more conservative in nature and will lead to detection of an obstacle using fewer observations. Hence, this function is pessimistic about free-space and ensures an obstacle will be detected in all cases when log-odds also detects an obstacle.

Table 4.1: Occupancy update

Condition	Occupancy status	Ocupancy cost $occ(d_s)$
$d_s > d_{\mathrm{f}}(u, v)$	Safe	$-0.5\frac{C(d_s)}{1-C(d_s)}$
$d_s < d_{\mathrm{f}}(u, v)$ and $d_s > d_{\mathrm{b}}(u, v)$	Occupied	$\frac{C(d_s)}{1-C(d_s)}$
$d_s < d_{\mathrm{b}}(u, v)$	Potentially safe	$-0.5\frac{C(d_s)}{1-C(d_s)}$

using the two methods. The confidence function is more conservative in nature when doing occupancy update at longer ranges. Hence, it is guaranteed that the proposed confidence function will mark a state as occupied if the probabilistic inference also evaluates to the same. We further discount measurements that mark an area safe or potentially safe(occluded) by 0.5 to be more conservative about clearing areas previously marked occupied.

4.4.5 Collision Checking

Collision checking is used to plan a new path and to validate if an existing path is safe to follow. Collision checking is performed by projecting the 3D world point $P(x, y, z)$ to image pixel $I(u.v)$ with disparity d_s using equation (4.1) and equation (4.2). The point P is projected in all the images that constitute the nodes of the pose-graph and checked for collision as follows.

A state is in collision if the total occupancy measure $\mathcal{M}$ as shown in equation (4.5) crosses a pre-defined threshold γ.

$$\mathcal{M} = max(\sum_{nodes} occ(d_s), 0) \tag{4.5}$$

$$Collision = \begin{cases} 1, & \text{if } \mathcal{M} \geq \gamma \\ 0, & \text{otherwise} \end{cases} \tag{4.6}$$

where $occ(d_s)$ is computed according to Table 4.1. If the occupancy for a state is below the threshold, we consider that state as not occupied by an obstacle. We also clamp $\mathcal{M}$ to be not negative to prevent over confidence for free volume.

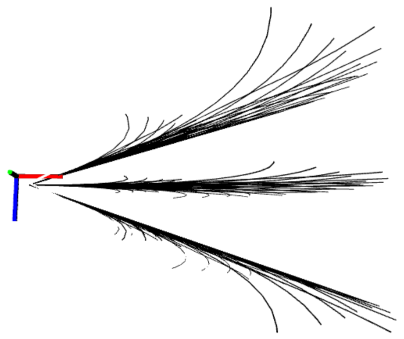
Note: For wire occupancy inference, since the wires are assumed to be uniformly distributed over a region per observation (see Figure 4.4), we directly use the confidence function $C(d)$ with $\sigma = 0$ instead of its odds to compute the occupancy cost $occ(d)$. This simplifies to summation of observations with wire over a given volume.

4.4.6 Pose Graph of Disparity Images

A single observation is often not enough to construct a reliable occupancy map, hence several observations are fused into a local map enabling local spatial memory. Moreover, stereo cameras only observe the environment in the overlapping field of view. Hence, a spatial memory is required to create a local map of the environment as the robot moves in it. We propose to use a pose-graph of our disparity image based representation to maintain a spatial memory. A pose-graph can further benefit from a simultaneous localization and mapping solution to correctly register the observations [Florence et al.], and can be later used to generate a global occupancy map. By maintaining a pose graph of expanded disparity images, we can do fusion without building a spatial grid which are not suited for stereo data as discussed previously. Details are provided in [Dubey et al., 2017]

4.5 Planning & Motion Control

We implemented a receding horizon planner based on a trajectory library. Trajectory or manoeuvre libraries have been widely used in the robotics community to solve high dimensional control problems such as graph selection or for trajectory set generation for mobile robot navigation [Frazzoli et al., 2000, Stolle and Atkeson, 2006, Berenson et al., Goldfeder et al., Dey et al., 2012]. The motivation for using a prior set of libraries is that they effectively discretize a large control or planning space and enable good performance within possible computational limits. All candidates in the library are evaluated at runtime, and the one with least cost and free from collision is chosen and executed. However, the performance is hugely affected by the size and content quality or coverage of the library. Size refers to the number of candidates that can be evaluated during runtime and quality or coverage refers to dispersion of the candidates [Green and Kelly, 2010]. The main advantage of using such libraries is that they are guaranteed to be dynamically feasible and hence allow smooth motion, or manipulation of the original library.


(a)

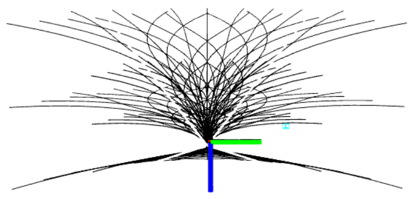

(b)

Figure 4.8: (a) Side view (b) back view of 3D Trajectory Library used on the robot. The Axes represent the robot pose, red is forward x-axis and blue is down z-axis.

Figure 4.8 shows a set of 3D trajectories used on the robot. To evaluate a trajectory, it is split into m waypoints and the following cost function is used to prune the original trajectory:

$$i = \arg\min_i \alpha\Delta x_i + (1-\alpha)\Delta\theta_i \quad , \text{where } i \in [1, m] \tag{4.7}$$

$$J(\tau) = \sum_{n=1}^{i} \alpha\Delta x_n + (1-\alpha)\Delta\theta_n \tag{4.8}$$

where Δx_i is the distance to goal and $\Delta\theta_i$ is the angle offset from goal for the waypoint τ_i. The parameter α is hand tuned to prefer proximity v/s heading offset to goal. The trajectories are checked for collision in ascending order of their traversal cost, by checking each waypoint on the trajectory using the method explained in Section 4.4.5, until a valid trajectory is found. If no valid trajectory is found, the robot is commanded to brake and rotate in place as a recovery manoeuvre to find alternate paths.

4.6 System and Experiments

4.6.1 System

We developed two MAV systems to conduct the experiments. Both are similar in capabilities but with different sizes as explained below.

Robot-1: For wire avoidance tests, we used a COTS (Commercial Off-The-Shelf) quadrotor (DJI Matrice M100), retrofitted with in-house developed sensor suite consisting of a stereo-camera, an FPGA stereo processor [Schauwecker, 2015], a monocular color camera, an IMU, and a Nvidia TX2 ARM computer as shown in Figure 4.9. Figure 4.10 shows the system diagram with the algorithms that run onboard. The FPGA board computes the stereo disparity while the rest of computation is done on the TX2 board. A ground station computer is used to set the goal or to provide a set of intermediate goal locations.

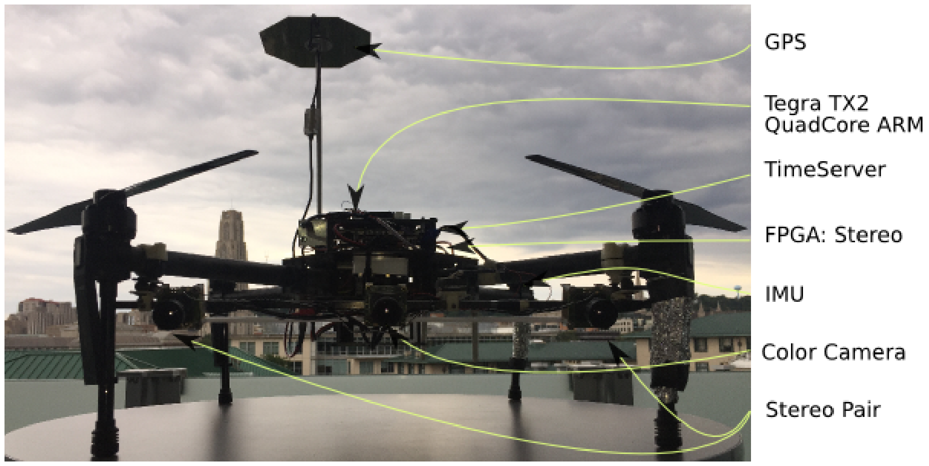


Figure 4.9: Robot-1: Quadrotor platform used for experiments: equipped with stereo camera + color (middle) camera sensor suite and onboard ARM computer

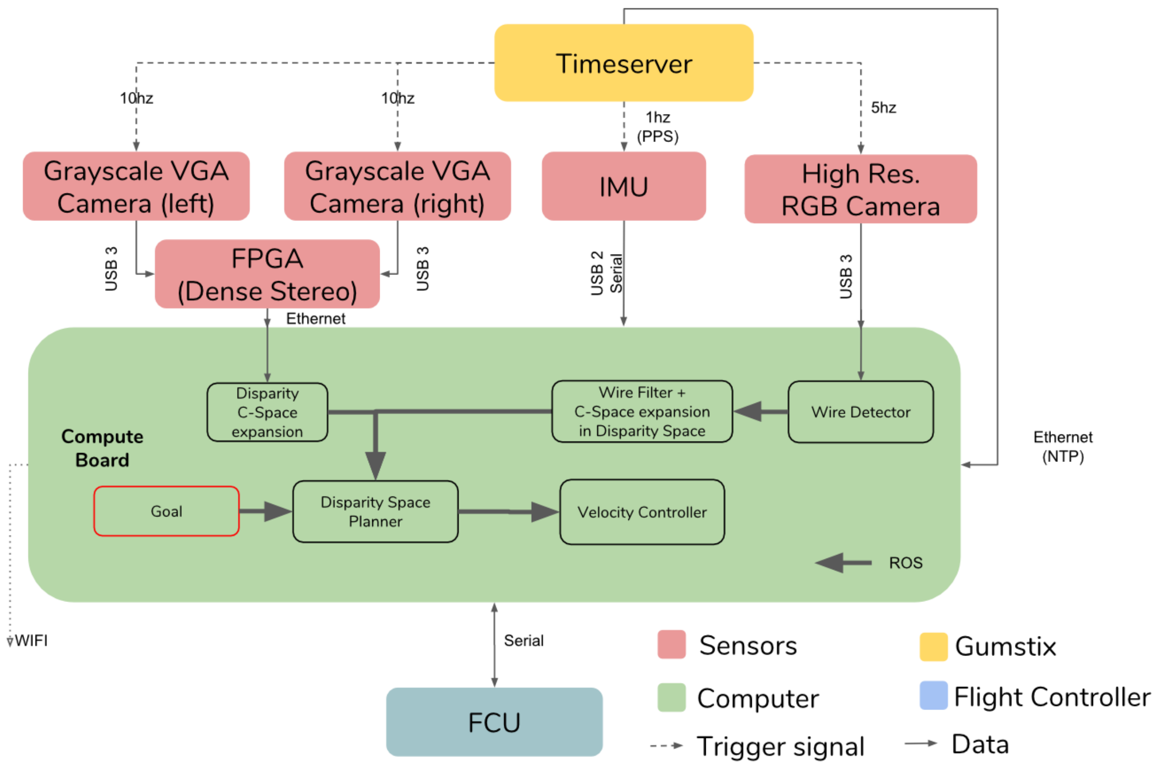


Figure 4.10: System diagram of the robot. Shows the hardware and main software components and their data exchange. ROS is used to interchange data between various onboard software components and to provide goal from the ground station.

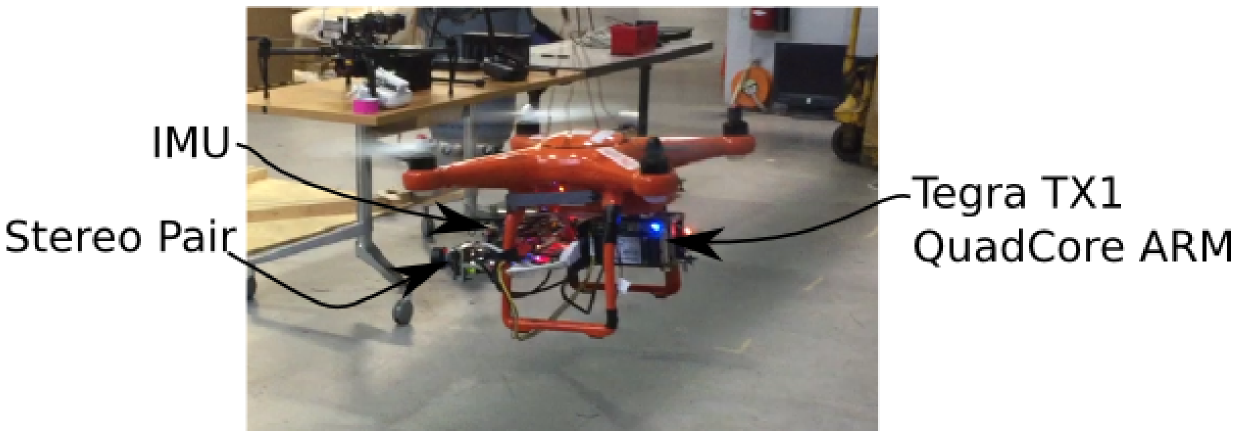


Figure 4.11: Robot-2: A smaller quadrotor platform used for experiments: equipped with stereo camera sensor suite and onboard ARM computer

Robot-2: For regular obstacle avoidance using stereo-camera disparity we used a small COTS base platform (Autel X-Star quadrotor) retrofitted with in-house developed sensing and computing suite consisting of a stereo camera pair, an integrated GPS/INS unit, and an Nvidia TX1, as shown in Figure 4.11. The stereo camera pair provides 640×512 resolution image which is used to compute a disparity image at 10 FPS on the embedded GPU. All computation for autonomous operation is performed onboard.

4.6.2 Experiments

We conducted separate experiments for wire and regular obstacle avoidance using trajectory libraries. Two different systems as explained in previous section were used.

We then tested our DROAN mapping approach on regular and wire obstacles combined as shown in Figure 4.1.

Wire Avoidance

We suspended a metal wire between two ladders such that it lies directly between the start and goal locations. We first conducted some baseline experiments using COTS DJI-Phantom4 PRO quadrotor. It has a onboard obstacle detection system that uses a stereo-camera sensor.

To validate our approach for wire avoidance, we configured Robot-1 system to only detect and avoid wires. We conducted 10 runs where the goal was set roughly 10 m behind the wire with varying start distances from the wire. The robot was configured to fly at a maximum speed of 2 m/s.

Regular Obstacle Avoidance using Stereo-Camera Disparity

We conducted indoor and outdoor experiments using a set of trajectory library. Tests involved autonomous take off, navigate to pre-fixed well separated global waypoints and finally land. For state estimation the EKF based methods from [Song et al., 2017] are used.

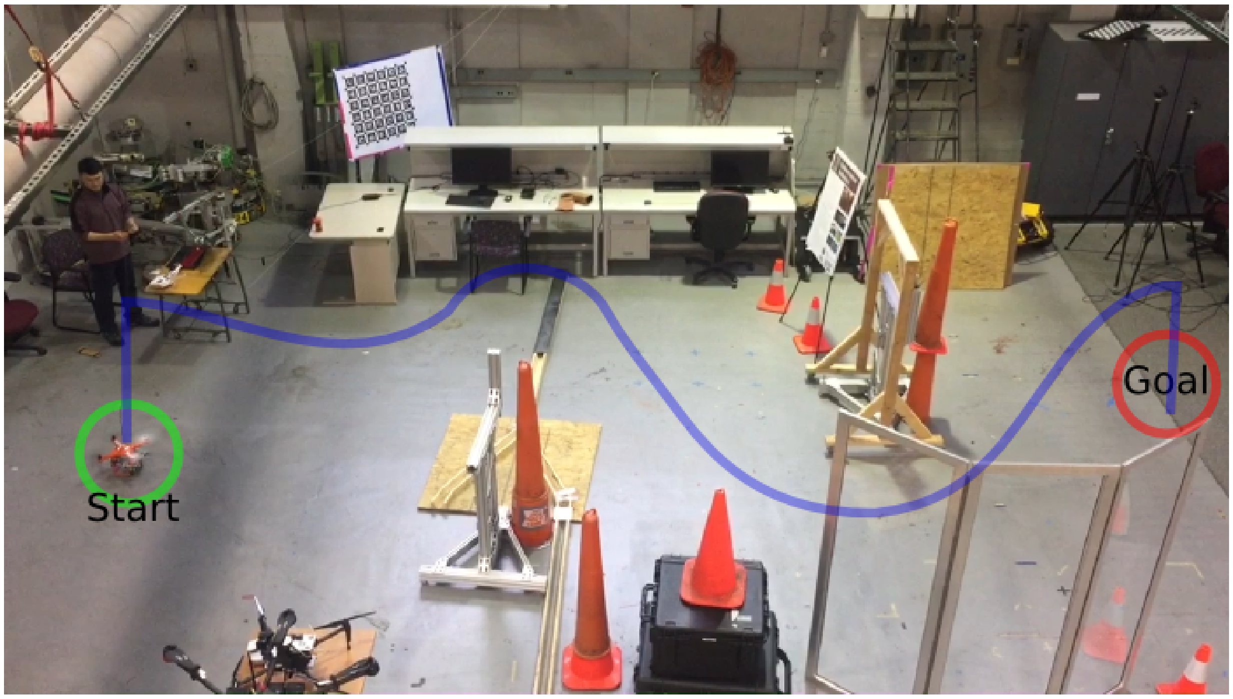


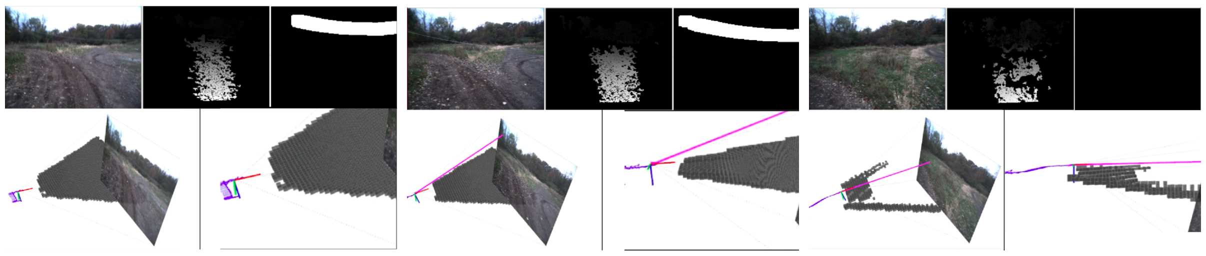
Figure 4.12: Demo setup for Robotics Week, 2017 at RI-CMU. The start is on left and the goal is 10 m away on right with several obstacles restricting a direct path, hence forming a curvy corridor (in blue) to follow. We did more than 100 runs with no failures at 2.5 m/s

In outdoor experiments, GPS was fused with IMU data to obtain robot state information while for indoor experiments stereo-camera based visual-inertial-odometry was used. Figure 4.12 shows the setup. The Start and Goal are separated by 10m with a curvy path obstructed with multiple obstacles.

4.7 Results

4.7.1 Simulation Results

To test our wire mapping approach we setup a simulation environment in Gazebo [Koenig and Howard, 2004] with three horizontal wires stacked 3 m away as shown in Figure 4.5. The first three snapshots show single observations at different heights using our representation. The wire (ground truth shown as black linear point cloud) is assumed to be uniformly distributed in this space as depicted by colored voxels. The last two snapshots show the final result of our fusion and occupancy inference from oblique and along the wire direction perspectives. It is evident that our approach is successful in mapping the three wires. The voxels are only generated for visualization purpose by creating a 0.5 m resolution, ego-centric gridmap ($60 \times 40 \times 40$ m^3) with respect to the robot. Hence, there are no voxels beyond a certain point displayed in Figure 4.5. Real world experiments are explained in the next section.



(a) (b) (c)

Figure 4.13: This figure shows three timestamps from run 8 of our real world wire avoidance experiment. Top row shows the input color image used for wire detection, disparity map, and (c-space) expanded wire obstacle map. Bottom row shows corresponding scenarios in different 3D perspectives (oblique and side). The black voxels are shown for visualization of wire collision map. (a) Robot observes the wire. (b) Executes upward trajectory. (c) After ascending upwards enough to avoid the wire, the robot executes a straight trajectory towards a goal 15 m ahead from the start point of the mission. Wire collision map is updated with new observations void of wire.

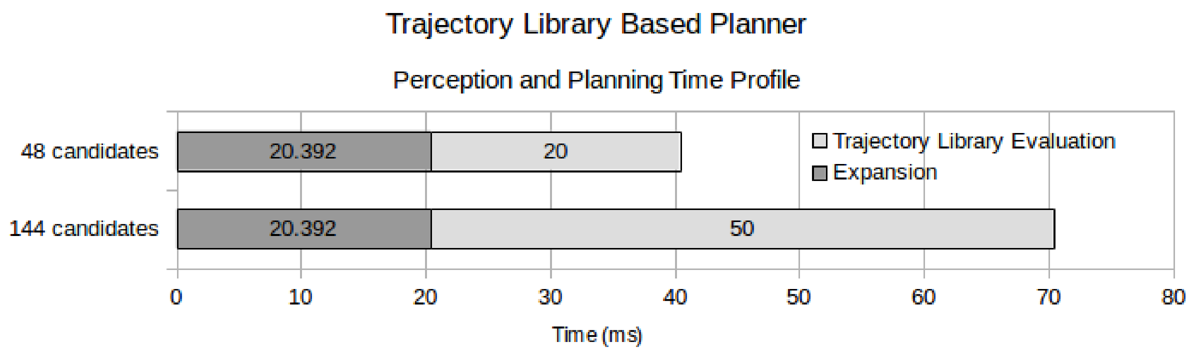


Figure 4.14: Time Profile for trajectory library based planner approach on Jetson TX2.

4.7.2 Wire Avoidance Results

We first tested with a COTS DJI Phantom 4 Pro drone. It successfully detected regular obstacles but fails to detect wires as an obstacle. Figure 4.13 shows the process of wire avoidance in Run-8 using our approach. We show the black voxel collision map of wire obstacle only for visualization. For collision avoidance, we only need to perform a collision check for the states traversed by the trajectories. In Figure 4.13a the robot observed the wire and we show the current collision map. The goal is set approximately 15 m in front of the robot. Figure 4.13b shows the robot executing the upward trajectory to avoid the wire in its path. Figure 4.13c, after the robot has ascended enough to fly over the wire, the straight trajectory going towards the goal is selected, and the wire occupancy map gets updated.We obtained 90% success rate as summarized in Table 4.2.

4.7.3 Disparity Map based Obstacle Avoidance Results

Figure 4.14 shows the effect on compute times on varying the size of candidates in the trajectory library. In either case we are able to do real-time planning i.e. within the sensor

Table 4.2: Details of our 10 runs of wire avoidance at Gascola, a test site half an hour away from CMU. Videos available at the thesis webpage.

Run	Success	Remark
1	True	Goal sent towards right. Avoids wire and moves rightwards
2	True	Pilot took over but the plan was successfully avoiding the wire
3	True	Robot moved straight and up to avoid the wire
4	True	Robot moved straight and up to avoid the wire
5	True	Goal sent towards left. Robot flew towards left and over the wires.
6	True	Avoids wire by going left. Ladder obstacle was in the way.
7	True	Avoided wire by going around it from left direction.
8	True	Robot moved straight and up to avoid the wire
9	True	Robot moved straight and up to avoid the wire
10	False	Avoids the wire by going under it, GPS boom got stuck in the wire.

frame rate. Most runs were conducted between the speed of $2m/s$ to $3m/s$. During the Robotics Week, 2017 at RI-CMU more than 100 runs were conducted in front of public with no failures.

4.8 Conclusion and Future Work

We have presented a novel method and a system which can detect wires from monocular images, estimate their depth and navigate around them. We proposed an efficient method to represent wires and generic obstacles in 2.5D image-space (disparity) suitable for cameras, while accounting for perception uncertainty, and demonstrated avoidance using a trajectory library. In the future, we aim to propose a mapping framework which accounts for both state estimate and perception uncertainty explicitly.

Chapter 5

Conclusion

This thesis makes the following contributions towards wire detection, reconstruction and avoidance for UAVs using a monocular camera, under the assumption of a known state estimate

- Wire Segmentation from Monocular Images
  We proposed using fully convolutional networks which do no downsampling, have a fewer number of layers than a generic semantic segmentation network, and use a context module composed of dilated convolutional layers to gather global evidence. We demonstrated the feasibility of generating a synthetic dataset, by rendering straight lines and catenaries in a randomized fashion, and superimposing them on a large number of frames taken from publicly available FPV flight videos on the internet. Our top networks run at more than 3 Hz on an embedded GPU (Nvidia TX2).
- Multi-view Reconstruction of Wires using a Catenary Model
  We proposed a model-based multiview reconstruction algorithm for a single catenary curve. We use a minimal parameterization of the curve, and an objective function based on the distance transform of the binary segmentations which makes our algorithm robust to arbitrary initializations and obviates the need to establish pixel or feature correspondences.
- Model-free Mapping and Reactive Avoidance
  We proposed a method which can map both wires and generic stereo obstacles in 2.5D disparity space suitable for a monocular camera and binocular stereo pair, Our method accounts for perception uncertainty and is efficient than a standard voxel-grid based method as it uses a pose-graph of disparity images. We closed the loop and demonstrated wire avoidance using a trajectory library.

5.1 Associated Publications

- *Wire Detection using Synthetic Data and Dilated Convolutional Networks for Unmanned Aerial Vehicles.*
  Ratnesh Madaan, Daniel Maturana, Sebastian Scherer. IROS 2017
  Finalist, Best Application Paper Award

- *DROAN - Disparity-space Representation for Obstacle AvoidaNce: Enabling Wire Mapping Avoidance.*
  Geetesh Dubey, Ratnesh Madaan, Sebastian Scherer. IROS 2018
- *Multi-view Reconstruction of Wires using a Catenary Model.*
  Ratnesh Madaan, Michael Kaess, Sebastian Scherer.
  to be submitted, ICRA 2019
- *Learning Adaptive Sampling Distributions for Motion Planning via Imitation.*
  Ratnesh Madaan*, Sam Zeng*, Brian Okorn, Sebastian Scherer. *submitted*, CoRL 2018
  *Equal Contribution

Bibliography

Junaid Ahmad, Aamir Saeed Malik, Likun Xia, and Nadia Ashikin. Vegetation encroachment monitoring for transmission lines right-of-ways: A survey. *Electric Power Systems Research*, 95:339–352, 2013. 3.1

Franz Andert and Florian Adolf. Online world modeling and path planning for an unmanned helicopter. *Autonomous Robots*, 27(3):147–164, 2009. 4.3.1

Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2011. 2.4.1

Australian Transport Safety Bureau. Wire-strike Accidents in General Aviation, Data Analysis 1994 to 2004. https://www.atsb.gov.au/media/32640/wirestrikes_20050055.pdf, 2006. September 2006. (document), 1.1, 1.1, 1.2

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for scene segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 2.1, 2.2.2, 2.3.1, 2.4.3, 2.4

A. J. Barry and R. Tedrake. Pushbroom stereo for high-speed navigation in cluttered environments. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3046–3052, May 2015. doi: 10.1109/ICRA.2015.7139617. 4.3.1

Dmitry Berenson, Rosen Diankov, Koichi Nishiwaki, Satoshi Kagami, and James Kuffner. Grasp planning in complex scenes. 4.5

Kai Berger, Randolph Voorhies, and Larry H Matthies. Depth from stereo polarization in specular scenes for urban robotics. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1966–1973. IEEE, 2017. 1.1

Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4380–4389, 2015. 2.1

Gunilla Borgefors. Distance transformations in digital images. *Computer vision, graphics, and image processing*, 34(3):344–371, 1986. 3.3.2

Mary Ann Branch, Thomas F Coleman, and Yuying Li. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing*, 21(1):1–23, 1999. 3.3.3

Alphonse R Burdi and John H Lillie. A catenary analysis of the maxillary dental arch during human embryogenesis. *The Anatomical Record*, 154(1):13–20, 1966. 3.2

Joshua Candamo, Rangachar Kasturi, Dmitry Goldgof, and Sudeep Sarkar. Detection of thin lines using low-quality video from low-altitude aircraft in urban settings. *IEEE Transactions on aerospace and electronic systems*, 45(3), 2009. 1.1, 2.1, 2.1, 2.2.1, 2.4.3, 2.4

Liang Cheng, Lihua Tong, Yu Wang, and Manchun Li. Extraction of urban power lines from vehicle-borne lidar data. *Remote Sensing*, 6(4):3302–3320, 2014. 1.2

Daily Mail. 'I can't believe someone would do that': Father of four almost decapitated while riding his dirt bike by a barbed wire booby trap strung at neck height. http://www.dailymail.co.uk/news/article-2794848/i-believe-father-four-decapitated-riding-dirt-bike-barbed-wire-booby-trap-strung-nec html, 2014. Online; Oct 15, 2014. 1.1

Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006. 2.4.1

Alexander J De Voogt, Sjir Uitdewilligen, and Nick Eremenko. Safety in high-risk helicopter operations: The role of additional crew in accident prevention. *Safety Science*, 47(5): 717–721, 2009. 1.1

D Dey, TY Liu, B Sofman, and JA Bagnell. Efficient Optimization of Control Libraries. *Aaai*, pages 1983–1989, 2012. 4.5

Sander Dieleman, Jan SchlÃijter, Colin Raffel, Eben Olson, SÃÿren Kaae SÃÿnderby, Daniel Nouri, et al. Lasagne: First release., August 2015. URL http://dx.doi.org/10.5281/zenodo.27878. 2.4.2

Geetesh Dubey. Droan: Disparity space representation for obstacle avoidance. Master's thesis, Carnegie Mellon University, Pittsburgh, PA, August 2017. 1.2, 4.1, 4.4.4, 4.4.4

Geetesh Dubey, Sankalp Arora, and Sebastian Scherer. DroanâĂŤdisparity-space representation for obstacle avoidance. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 1324–1330. IEEE, 2017. 1.2, 4.1, 4.2, 4.4.2, 4.4.3, 4.4.6

Geetesh Dubey, Ratnesh Madaan, and Sebastian Scherer. Droan - disparity-space representation for obstacle avoidance: Enabling wire mapping avoidance. In *Intelligent Robots and Systems (IROS), 2018 IEEE/RSJ International Conference on*. IEEE, 2018. 1.2, 4.1, 4.4.2

Andrew W Fitzgibbon. Robust registration of 2d and 3d point sets. *Image and Vision Computing*, 21(13-14):1145–1153, 2003. 3.2, 3.3.2

Flight Safety Australia. Wire, the invisible enemy. http://www.flightsafetyaustralia.com/2017/11/wire-the-invisible-enemy/, 2017. Online; Nov 20, 2017. 1.1

Peter R Florence, John Carter, Jake Ware, and Russ Tedrake. Nanomap: Fast, uncertainty-aware proximity queries with lazy search over local 3d data. 4.4.6

Robert J Fontana, J Frederick Larrick, Jeffrey E Cade, and Eugene P Rivers. Ultrawideband synthetic vision sensor for airborne wire detection. In *Enhanced and Synthetic Vision 1998*, volume 3364, pages 2–11. International Society for Optics and Photonics, 1998. 1.1

E. Frazzoli, M. A. Dahleh, and E. Feron. Robust hybrid control for autonomous vehicle motion planning. In *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No.00CH37187)*, volume 1, pages 821–826 vol.1, 2000. doi: 10.1109/CDC.2000.912871. 4.5

Pascal Gohl, Dominik Honegger, Sammy Omari, Markus Achtelik, Marc Pollefeys, and Roland Siegwart. Omnidirectional visual obstacle detection using embedded FPGA. *IEEE International Conference on Intelligent Robots and Systems*, 2015-Decem:3938–3943, 2015. ISSN 21530866. doi: 10.1109/IROS.2015.7353931. 4.3.1, 4.4.2

Corey Goldfeder, Matei Ciocarlie, Jaime Peretzman, Hao Dang, and Peter K. Allen. Data-driven grasping with partial sensor data. 4.5

Colin J. Green and Alonzo Kelly. Toward optimal sampling in the space of paths. *Springer Tracts in Advanced Robotics*, 66(STAR):281–292, 2010. ISSN 16107438. doi: 10.1007/978-3-642-14743-2_24. 4.5

Susan Hasse Pepe. Polynomial and catenary curve fits to human dental arches. *Journal of Dental Research*, 54(6):1124–1132, 1975. 3.2

Lionel Heng, Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. Autonomous obstacle avoidance and maneuvering on a vision-guided mav using on-board processing. In *Robotics and automation (ICRA), 2011 IEEE international conference on*, pages 2472–2477. IEEE, 2011. 4.3.1

Paul Hirons. Cable cutter antenna for aircraft, July 4 2017. US Patent 9,694,904. 1.1

Manuel Hofer, Andreas Wendel, and Horst Bischof. Incremental line-based 3d reconstruction using geometric constraints. In *BMVC*, 2013. 3.2

Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: an efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013. ISSN 1573-7527. doi: 10.1007/s10514-012-9321-0. URL http://dx.doi.org/10.1007/s10514-012-9321-0. 4.3.1

Stefan Hrabar. 3d path planning and stereo-based obstacle avoidance for rotorcraft uavs. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 807–814. IEEE, 2008. 1.1

Juan Jose Tarrio and Sol Pedre. Realtime edge-based visual odometry for a monocular camera. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 702–710, 2015. 3.2

Yoonseok Jwa and Gunho Sohn. A piecewise catenary curve model growing for 3d power line reconstruction. *Photogrammetric Engineering & Remote Sensing*, 78(12):1227–1240, 2012. 1.1, 1.2, 3.1, 3.2

Michael Kaess, Rafal Zboinski, and Frank Dellaert. Mcmc-based multiview reconstruction of piecewise smooth subdivision curves with a variable number of control points. In

European Conference on Computer Vision, pages 329–341, 2004. 3.2, 3.3.2

Rangachar Kasturi and Octavia I Camps. Wire detection algorithms for navigation. *NASA Technical Report*, 2002. 1.1, 2.1, 2.2.1

Nathan P Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IROS*, volume 4, pages 2149–2154. Citeseer, 2004. 4.7.1

Manohar Kuse and Shaojie Shen. Robust camera motion estimation using direct edge alignment and sub-gradient method. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 573–579. IEEE, 2016. 3.2, 3.3.2

Stuart Lau. A plan for reducing wire strike accidents. *Helo Air Safety, Professional Pilot*, pages 84–87, 2012. 1.1

James Donald Law, Robert Mark Law, and Dean Charles Smith. Wire-strike defence system, May 25 2010. US Patent 7,721,993. 1.1

J Lee, Gregory P Reece, and Mia K Markey. Breast curvature of the upper and lower breast mound: 3d analysis of patients who underwent breast reconstruction. In *3rd International Conference on 3D Body Scanning Technologies*, pages 171–179, 2012a. 3.2

Juhun Lee, Si Chen, Gregory P Reece, Melissa A Crosby, Elisabeth K Beahm, and Mia K Markey. A novel quantitative measure of breast curvature based on catenary. *IEEE Transactions on Biomedical Engineering*, 59(4):1115–1124, 2012b. 3.2

Zhengrong Li, Yuee Liu, Rodney Walker, Ross Hayward, and Jinglan Zhang. Towards automatic power line detection for a uav surveillance system using pulse coupled neural filter and an improved hough transform. *Machine Vision and Applications*, 21(5):677–686, 2010. 1.1, 2.1

Yonggen Ling, Manohar Kuse, and Shaojie Shen. Edge alignment-based visual–inertial fusion for tracking of aggressive motions. *Autonomous Robots*, 42(3):513–528, 2018. 3.2, 3.3.2

Lingjie Liu, Duygu Ceylan, Cheng Lin, Wenping Wang, and Niloy J Mitra. Image-based reconstruction of wire art. *ACM Transactions on Graphics (TOG)*, 36(4):63, 2017. 1.1, 3.2, 3.7.3

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 2.1, 2.2.2, 2.3.1, 2.4.3, 2.4, 4.2

Xiaohu Lu, Jian Yao, Kai Li, and Li Li. Cannylines: A parameter-free line segment detector. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 507–511. IEEE, 2015. 2.3.4, 2.4.2, 2.4.3

Luis F Luque-Vega, Bernardino Castillo-Toledo, Alexander Loukianov, and Luis E González-Jiménez. Power line inspection via an unmanned aerial system based on the quadrotor helicopter. 2014. 1.2

Ratnesh Madaan, Daniel Maturana, and Sebastian Scherer. Wire detection using synthetic data and dilated convolutional networks for unmanned aerial vehicles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, September 2017. 1.2, 3.1, 4.2, 4.4.1

Leena Matikainen, Matti Lehtomäki, Eero Ahokas, Juha Hyyppä, Mika Karjalainen, Anttoni Jaakkola, Antero Kukko, and Tero Heinonen. Remote sensing methods for power line corridor surveys. *ISPRS Journal of Photogrammetry and Remote Sensing*, 119: 10–31, 2016. 1.2

L. Matthies, R. Brockers, Y. Kuwata, and S. Weiss. Stereo vision-based obstacle avoidance for micro air vehicles using disparity space. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3242–3249, May 2014. doi: 10.1109/ICRA.2014.6907325. 4.2, 4.3.1, 4.4.3

Robert A McLaughlin. Extracting transmission lines from airborne lidar data. *IEEE Geoscience and Remote Sensing Letters*, 3(2):222–226, 2006. 1.1, 1.2, 3.1, 3.2

Jonathan McNally. Method for installing an object using an unmanned aerial vehicle, April 3 2018. US Patent 9,932,110. 1.1

Metro News. Cyclist âĂŸalmost decapitatedâĂŹ after barbed wire is strung across bike lane. `https://metro.co.uk/2016/11/14/cyclist-almost-decapitated-after-barbed-wire-is-strung-across-bike-lane-6257899/`, 2016. Online; Nov 14, 2016. 1.1

Joseph Moore and Russ Tedrake. Powerline perching with a fixed-wing uav. In *AIAA Infotech@ Aerospace Conference and AIAA Unmanned... Unlimited Conference*, page 1959, 2009. 1.1

Vengalattore T Nagaraj and Inderjit Chopra. Safety study of wire strike devices installed on civil and military helicopters. 2008. 1.1

Angelo Pagnano, Michael Höpf, and Roberto Teti. A roadmap for automated power line inspection. maintenance and repair. *Procedia Cirp*, 12:234–239, 2013. 1.1

Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016. 2.1, 2.2.2, 2.3.1, 2.4.3, 2.4

Fabio Poiesi and Andrea Cavallaro. Detection of fast incoming objects with a moving camera. In *BMVC*, 2016. 1.1

Nicolas Pouliot, Pierre-Luc Richard, and Serge Montambault. Linescout technology opens the way to robotic inspection and maintenance of high-voltage power lines. *IEEE Power and Energy Technology Systems Journal*, 2(1):1–11, 2015. 1.1

POV-Ray. Persistence of vision raytracer (version 3.7), 2004. URL `http://www.povray.org/download/`. (document), 2.1, 2.2, 2.3.2, 4.4.1

Subramanian Ramasamy, Roberto Sabatini, Alessandro Gardi, and Jing Liu. Lidar obstacle warning and avoidance system for unmanned aerial vehicle sense-and-avoid. *Aerospace Science and Technology*, 55:344–358, 2016. 1.1, 1.2

Eugene S Rubin and Paul A Zank. Method and apparatus for avoidance of power lines or trip wires by fixed and rotary winged aircraft, October 23 2007. US Patent 7,286,912. 1.1

John N Sanders-Reed, Dennis J Yelton, Christian C Witt, and Ralph R Galetti. Passive obstacle detection system (pods) for wire detection. In *SPIE Defense, Security, and Sensing*, pages 732804–732804. International Society for Optics and Photonics, 2009. 1.1, 1.2, 2.1

Konstantin Schauwecker. Sp1: Stereo vision in real time. 3.4.2

Konstantin Schauwecker. Sp1: Stereo vision in real time. In *MuSRobS@ IROS*, pages 40–41, 2015. 4.4.2, 4.6.1

Sebastian Scherer, Sanjiv Singh, Lyle Chamberlain, and Mike Elgersma. Flying fast and low among obstacles: Methodology and experiments. *The International Journal of Robotics Research*, 27(5):549–574, 2008. 1.1, 1.2

scikitlearn: least squares. Scikit-learn: Trust-region-reflective least squares, 2011. URL http://www.mathworks.com/help/optim/ug/least-squares-model-fittinghttps://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.least_squares.html. 3.3.3

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2.3.1

Skydio. Skydio. 2018. 1.1

Biqin Song and Xuelong Li. Power line detection from optical images. *Neurocomputing*, 129:350–361, 2014. 1.1, 2.1, 2.2.1

Yu Song, Stephen Nuske, and Sebastian Scherer. A multi-sensor fusion mav state estimation from long-range stereo, imu, gps and barometric sensors. *Sensors*, 17, 2017. 4.6.2

Carsten Steger. An unbiased detector of curvilinear structures. *IEEE Transactions on pattern analysis and machine intelligence*, 20(2):113–125, 1998. 2.2.1

Martin Stolle and Chris Atkeson. Policies based on trajectory libraries. In *In Proceedings of the International Conference on Robotics and Automation (ICRA*, 2006. 4.5

Changming Sun, Ronald Jones, Hugues Talbot, Xiaoliang Wu, Kevin Cheong, Richard Beare, Michael Buckley, and Mark Berman. Measuring the distance of vegetation from powerlines using stereo vision. *ISPRS journal of photogrammetry and remote sensing*, 60(4):269–283, 2006. 3.1

The Local. Barbed wire attacks leave bikers scared of decapitation. https://www.thelocal.de/20111024/38411, 2011. Online; Oct 24, 2011. 1.1

The Telegraph. Motorcyclist decapitated by wire across path . https://www.telegraph.co.uk/news/worldnews/1548300/Motorcyclist-decapitated-by-wire-across-path.html, 2007. Online; Apr 11, 2007. 1.1

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN 0262201623. 4.4.4

Clyde H Tuomela and Mark F Brennan. Civil helicopter wire strike assessment study. volume 1: Findings and recommendations. 1980a. 1.1

Clyde H Tuomela and Mark F Brennan. Civil helicopter wire strike assessment study. volume 2: Accident analysis briefs. 1980b. 1.1

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR abs/1609.03499*, 2016. 2.3.1

Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1395–1403, 2015. 2.1, 2.3.1, 2.3.3

Guangjian Yan, Chaoyang Li, Guoqing Zhou, Wuming Zhang, and Xiaowen Li. Automatic extraction of power lines from aerial images. *IEEE Geoscience and Remote Sensing Letters*, 4(3):387–391, 2007. 1.1, 2.1

Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 2.1, 2.3.1, 4.2

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. 2.4.2

C. Zhou, J. Yang, C. Zhao, and G. Hua. Fast, accurate thin-structure obstacle detection for autonomous mobile robots. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 318–327, July 2017a. doi: 10.1109/CVPRW.2017.45. 3.2

Chen Zhou, Jiaolong Yang, Chunshui Zhao, and Gang Hua. Fast, accurate thin-structure obstacle detection for autonomous mobile robots. *arXiv preprint arXiv:1708.04006*, 2017b. 1.1