Towards Scalable Visual Navigation of Micro Aerial Vehicles

Shreyansh Daftry
daftry@cmu.edu

Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

April 2016

Thesis Supervisors:
Prof. Dr. Martial Hebert
Prof. Dr. J. Andrew Bagnell

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

CMU-RI-TR-16-07

Keywords: Computer vision, Mobile Robots, Visual Navigation, Micro Aerial Vehicle (MAV), Transfer Learning, Domain Adaptation, Direct SLAM, Dynamic Wind-Estimation, Vision-based control, Receding Horizon, Imitation Learning, Deep Learning, Introspection, Failure Prediction.

The only true wisdom is in knowing you know nothing.

Socrates

Abstract

Micro Aerial Vehicles (MAVs) have built a formidable résumé by making themselves useful in a number of important applications, from disaster scene surveillance and package delivery to robots used in aerial imaging, architecture and construction. The most important benefit of using such lightweight MAVs is that it allows the capability to fly at high speeds in space-constrained environments. While autonomous operations in structured motion-capture systems has has been well studied in general, enabling resource-efficient, persistent navigation methods for long-term autonomy in unstructured and dynamic environments is still an open problem in current robotics research. In this thesis, we take a small step in this direction and present a scalable framework for robust visual navigation of MAVs in the wild.

Our first contribution is a toolbox of approaches for perception, planning and control of agile, low-cost MAVs in cluttered urban and natural outdoor environments, based on a monocular camera as the main exteroceptive sensor. In particular, we present novel geometry and data-driven depth estimation methods for non-translational camera motion and dynamic scenes, where traditional structure from motion (SfM) and Simultaneous Localization and Mapping (SLAM) techniques do not apply, with accuracy comparable to that of a stereo setup.

Second, we propose a generic framework for introspection in autonomous robots. As robots aspire for complete autonomy in human-centric environments, accurate situational awareness becomes a critical requirement for verifiable safety standards. We call this self-evaluating capability, to assess how qualified they are at that moment to make a decision, as introspection. Inspired by this, we advocate the need to build systems with the ability to take mission-critical decisions in ambiguous situations and present a failure prediction and recovery framework for perception systems.

Finally, our third contribution towards developing a scalable autonomous behavior is a framework for knowledge transfer across robots and environments. We argue that for many learning based robot tasks, it is not possible to obtain training data. The ability to transfer knowledge gained in previous tasks into new contexts is one of the most important mechanisms of human learning. In this work, we develop a similar technique for learning transferable motion policies i.e. solve a learning problem in a target domain by utilizing the training data in a different but related source domain.

The proposed algorithms are quantitatively and qualitatively evaluated on a variety of datasets and validated through real-world field experiments. Our experiments demonstrate that our contributions help to build a scalable, accurate, and computationally feasible visual navigation system for micro aerial vehicles in the wild.

Acknowledgments

This thesis would have been impossible without the guidance and help of several people who have contributed their valuable time in one way or another.

First of all, I would like to thank my advisor Martial Hebert for his unconditional support. I feel fortunate to have been advised by him for my studies at Carnegie Mellon. I am grateful for the confidence he has placed in me when working on the BIRD project, and I appreciate that he gave me the opportunity to develop and explore my own ideas. Martial's wisdom, insight and experience made working with him incredibly inspiring and enlightening. I strive to uphold his lofty standards throughout my life.

Next, I would like to express my gratitude to my second supervisor Drew Bagnell. Not many people have the fortune to be co-advised by Drew. I appreciate his efforts to take out time to advise me, even during his sabbatical. His initiatives to keep all the members of the LAIRLab focused, motivated and always inspired to think in novel ways has been very rewarding.

I am very thankful to the former and current members of the BIRD project, Debadeepta Dey, Sam Zeng, Harsimrat Sandhawalia, Narek Melik-Barkhudarov, Dhruv Saxena and Arbaaz Khan for helping get the Angry Bird to fly. I would also like to thank the amazing group of people in LAIRLab: Allie Del Giorno, Wen Sun, Arun Venkataraman, Jiaji Zhou, Hanzhang Hu, Shervin Javdani and Roberto Capobianco. I will miss our lab scrum and socials.

Further, my sincere thanks goes to the entire cohort of the RI Masters program and all my former colleagues at ICG, Andreas Wendel, Christof Hoppe, Markus Rumpler, Manuel Hofer, Michael Maurer and my advisor Horst Bischof. I have been lucky to learn great things from each of you.

Finally, I feel indebted to my parents, sister, friends and family for always being there when I needed them. Thank you!

My research has been funded by the BIRD Multidisciplinary University Research Initiatives (MURI) project as part of the Office of Naval Research (ONR) grant on "Provably-Stable Vision-based Control of High-speed Flights through Forest and Urban Environments".

Contents

1 Introduction **1**
- 1.1 Micro Aerial Vehicles: An Emerging Robotic Platform . . . 1
- 1.2 Biological Inspiration . . . 2
- 1.3 Current State of Research . . . 2
- 1.4 Contribution of the Thesis . . . 4
  - 1.4.1 Accurate, Robust and Computationally feasible Visual Navigation . . . 4
  - 1.4.2 Safe, Reliable and Verifiable Autonomy . . . 4
  - 1.4.3 Scalable, Adaptive Learning of Policies . . . 4
- 1.5 Outline . . . 5

2 Robust Monocular Visual Navigation **7**
- 2.1 System Overview . . . 7
- 2.2 Monocular Depth Estimation . . . 9
  - 2.2.1 Deep Learning based Depth Prediction . . . 9
  - 2.2.2 Direct Visual Odometry based Depth Estimation . . . 12
  - 2.2.3 IMU Pre-Integration on Lie-Manifolds . . . 14
- 2.3 Multiple Predictions . . . 15
- 2.4 Planning and Control . . . 17
  - 2.4.1 Receding Horizon Planner . . . 17
  - 2.4.2 State Estimation . . . 18
  - 2.4.3 Wind-Resistant LQR Control . . . 19
- 2.5 Experimental and Results . . . 21
  - 2.5.1 Performance Evaluation of Monocular Depth Estimation . . . 21
  - 2.5.2 Performance Evaluation of Wind Estimation and Resistance . . . 24
  - 2.5.3 System-wide Performance Evaluation . . . 24
- 2.6 Summary . . . 26

3 Introspection for Failure Recovery **29**
- 3.1 Risk-Aware Perception . . . 29
  - 3.1.1 Need for Failure Prediction in Vision Systems . . . 29
  - 3.1.2 Uncertainty Propagation . . . 31
- 3.2 Introspection Perception . . . 31
  - 3.2.1 Introspection Paradigm . . . 31
  - 3.2.2 Deep Introspection . . . 32

3.2.3 Quantifying Introspection . . . 34
3.3 Introspection in Navigation . . . 34
3.3.1 Learning the Introspection Model . . . 35
3.3.2 Emergency Maneuvers . . . 35
3.4 Experiments and Results . . . 35
3.4.1 BIRD Dataset . . . 36
3.4.2 Quantitative Evaluation . . . 37
3.4.3 Introspection Benefits to Autonomous Navigation . . . 38
3.4.4 Qualitative Evaluation . . . 39
3.5 Summary . . . 40

4 Adaptive Learning for MAV Control **43**
4.1 Domain Adaptation in Robot Learning . . . 43
4.2 Learning Transferable Policies . . . 45
4.2.1 Monocular Reactive Control using Imitation Learning . . . 45
4.2.2 Domain Adaptive Neural Networks . . . 47
4.3 Experiments and Results . . . 48
4.3.1 Performance Evaluation . . . 48
4.3.2 Experimental Insights . . . 49
4.3.3 Scheduled Experiments . . . 51
4.4 Summary . . . 51

5 Conclusion **53**
5.1 Summary . . . 53
5.2 Future Work . . . 54

Bibliography **57**

List of Figures

1.1 Inspiration from Nature: Learning from Birds 2

2.1 Hardware Platforms: MAV and Mobile Ground Robot 8
2.2 Software Architecture 9
2.3 CNN Architecture for Depth Prediction 11
2.4 Results of Deep Depth Prediction 11
2.5 Semi-dense Depth Map Estimation. 13
2.6 Illustration of Need for Multiple Prediction 15
2.7 Advantage of Multiple World Prediction 16
2.8 Receding Horizon Control on MAV 18
2.9 On-board Control Module 19
2.10 Testing Area for Flight Tests 22
2.11 Performance Evaluation of Depth Prediction 23
2.12 Qualitative Analysis of Depth Prediction 24
2.13 Analysis of Wind modeling, Detection and Correction 25
2.14 Performance Evaluation of State Estimation 25
2.15 System Performance Evaluation 26

3.1 Robot Introspection: Having a self-evaluating Capability 32
3.2 Spatio-temporal CNN Architecture for Introspection 33
3.3 Receding Horizon Control for Self-supervised Data 34
3.4 Examples from BIRD Dataset 36
3.5 Error vs. Failure Rate Curves 37
3.6 System Performance Evaluation 39
3.7 Qualitative Analysis of Failure Prediction 40

4.1 DAgger: Learning from Demonstrations 45
4.2 Imitation Learning Interface 46
4.3 Domain Adaptive Learning of MAV Control 47
4.4 Transfer across Physical Systems 49
4.5 Transfer across Weather Conditions 50
4.6 Transfer across Environments 51

Chapter 1

Introduction

Contents

1.1 Micro Aerial Vehicles: An Emerging Robotic Platform 1
1.2 Biological Inspiration 2
1.3 Current State of Research 2
1.4 Contribution of the Thesis 4
1.5 Outline 5

1.1 Micro Aerial Vehicles: An Emerging Robotic Platform

In recent years, autonomous aerial robots are making themselves useful in a number of important applications, from disaster scene surveillance and package delivery to robots used in aerial imaging, architecture and construction. A variety of remotely piloted aerial vehicles is used for these applications, including airplanes as well as helicopters, quad-rotors, and octo-rotors. While fixed-wing vehicles offer the advantage of fast rectilinear flight to cover an extended area quickly and systematically, rotary aircraft provide the flexibility to explore a smaller area from multiple vantage points. Micro Aerial Vehicles offer a significant advantage over traditional methods in terms of costs, as not only the vehicle itself is comparably cheap but also the deployment is fast and simple.

However, most MAVs are limited to flight at altitudes well above ground-level obstacles, foliage and buildings, where basic path planning techniques and GPS are sufficient for navigation. The most important benefit of using such lightweight MAVs is that it allows the capability to fly at high speeds in space constrained environments. Their practical use would expand considerably if they were capable of flying at high speeds and among obstacles at low altitudes, enabling them to take detailed measurements of the street level of a city or the terrain in wilderness areas and cover significantly more ground than the conservative motions of many current aerial robots. In this thesis, our work is primarily concerned with navigating MAVs that have very low payload capabilities, and operate close to the ground where they cannot avoid dense obstacle fields.

(a) (b)

Figure 1.1: (a) Goshawk flying through dense forest (b) An MAV flying through dense clutter in a disaster rescue scenario.

1.2 Biological Inspiration

It is natural to look to nature for inspiration when approaching such design challenges. Nature provides many examples of graceful and yet purposeful flight. In particular, several species of birds have mastered the art of maneuverability, swiftly maneuvering among trees as they are encountered, during high-speed flights as shown in Figure 1.1a. Inspired by this, flying robots that can similarly navigate through cluttered environments, such as urban canyons and dense forests, have long been an objective of robotics research [57, 58, 82, 84, 103]. An autonomous robot with such capabilities can be envisioned to be extremely useful in applications of search and rescue, disaster scene surveillance, etc. Although birds differ from aerial robots in a number of fundamental ways, we still have the opportunity to incorporate avian behaviors in modern day drones. However, till date, very little have been established so far in realizing the same for autonomous drones [50]. Motivated by this, we take a small step in the direction of building a robust system that allows Micro Aerial Vehicles (MAVs) to autonomously fly at high speeds of up to 1.5 m/s through a cluttered forest environment, as shown in Figure 1.1b.

1.3 Current State of Research

In recent years, the development of autonomous flying robots has been an area of increasing research interest. The research in such autonomous MAVs is relatively young but advancing very fast. This research has produced a number of systems with a wide range of capabilities when operating in outdoor environments. For example vehicles have been developed that can perform high-speed flight through cluttered environments [81], or even acrobatics [8]. Other researchers have developed systems capable of autonomous landing and terrain mapping [94], as well as a host of high level capabilities such as coordinated tracking and planning of ground vehicles [40, 44], or multi-vehicle coordination [29]. While these are all challenging research areas in their own right, and pieces of the previous work (such as the modeling and control techniques)

carry over to the development of vehicles operating without GPS, these previous systems rely on external systems such as GPS, or external cameras [65] for localization. Similarly, a number of researchers have flown indoors using position information from motion capture systems, or external cameras. In discussing further related work, we focus on flying robots that are able to operate autonomously while carrying all sensors used for localization, control and navigation onboard.

Visual Navigation

In general, we consider two approaches using on-board cameras: the first one is tracking a known, fixed object (like artificial markers, or user-specified points), which implicitly solves the matching and relocalization problem because the markers are already known as well as their relative 3D position; the second approach is extracting distinctive natural features whose 3D position is not known a priori and use them for both motion and structure estimation, and re-localization (i.e. visual SLAM, visual odometry). Accurate depth estimation and localization is also possible purely based on passive visual sensors, and several state-of-the-art approaches exist. An important technique is stereo imaging, which has been successfully used in realtime, outdoor SLAM systems [61, 142]. Stereo systems use two cameras with a fixed, known distance (baseline) between them. This has the essential benefit that the depth measurements are retrieved in metrical scale. When using only one camera, the relative transform between individual camera poses can be estimated as well, but the overall scale factor remains unobservable unless at least one metrical distance between poses or map points is known. However, on the one hand the baseline available to most micro aerial vehicles is quite small and thus there is little benefit for outdoor usage, and on the other hand the required processing power for stereo image acquisition and processing is typically higher than for monocular vision.

Early visual SLAM systems using a single camera as the main perceptual sensor were inspired by traditional SLAM methods using recursive filtering approaches. The Bayesian network allows handling of uncertainties in position and scale in a way that robust localization is possible. Davison [37] fused measurements from a sequence of images by updating probability distributions over features and extrinsic camera parameters using an Extended Kalman Filter (EKF). This requires tracking and mapping to be closely linked, as the camera pose and feature locations are updated together at every single frame. Based on this approach Davison et al. [38] proposed MonoSLAM. They extend the previously proposed method of monocular feature initialization by a feature orientation estimation. This successfully increased the range in which landmarks are detected and hence improved tracking. However, MonoSLAM only worked for slow camera motions or in combination with additional sensors.

Outdoor Visual Control

While outdoor vehicles can usually rely on GPS, there are many situation where relying on GPS would be unsafe, since the GPS signal can be lost due to multi-path, satellites being occluded by buildings and foliage, or even intentional jamming. In response to these concerns, a number of researchers have developed systems that rely on vision for control of the vehicle. Early work in this area by [7] used a stereo camera to enable position hold capabilities. Other researchers have

developed capabilities such as visual servoing relative to a designated target [66], landing on a moving target, and even navigation through urban canyons [42]. While the systems developed by these researchers share many of the challenges faced by indoor or urban MAVs, they operate on vehicles that are orders of magnitude larger, with much greater sensing and computation payloads. In addition, the outdoor environments tend to be much less cluttered, which gives greater leeway for errors in the state estimation and control.

1.4 Contribution of the Thesis

Visual navigation for MAVs based on a monocular camera is a challenging task, and several problems have to be tackled to enable autonomous flight. In this thesis, we propose techniques to build a computationally feasible, robust and scalable visual navigation system. In the following, we summarize our key contributions towards this goal.

1.4.1 Accurate, Robust and Computationally feasible Visual Navigation

We present a toolbox of approaches for perception, planning and control of agile, low-cost MAVs navigation in cluttered urban and natural outdoor environments, based on a monocular camera as the main exteroceptive sensor. In particular, we present two novel approaches for depth estimation for non-translational camera motion and dynamic scenes: a Deep learning based data-driven approach and a Direct Visual Odometry based geometry-driven approach. Furthermore, we augment our depth estimation techniques with state-of-the-art visual inertial fusion, multiple list prediction techniques and a novel method for wind-resistant control to demonstrate robust, accurate and computationally feasible autonomous flight in real-forest environments.

1.4.2 Safe, Reliable and Verifiable Autonomy

We propose a generic framework for introspection in autonomous MAVs to enable them with the ability to assess how qualified they are at any given moment to make a decision. Using a spatio-temporal Convolutional Neural Network, we present a failure prediction and recovery strategy for perception systems. We demonstrate through extensive experiments that we are reliably able to predict failures reliably, and better than confidence based systems. This helps us build safe, reliable robot behaviors that can carry out emergency verifiable behaviors when uncertain.

1.4.3 Scalable, Adaptive Learning of Policies

In order for robots to have long-term autonomy and scale to dynamic environments, it is necessary to be able to transfer knowledge across tasks and domains. In this work, we present a technique to learn domain-adaptive policies using deep neural networks and imitation learning. We demonstrate the effective learning of policies for a target domain by using only labeled data from the source domain. We study the transfer of policies in the context of MAV control policies across physical systems, dynamic environments and also weather conditions.

1.5 Outline

This thesis is organized as follows. Chapter 2 introduces the monocular visual navigation system. we explain how robust, accurate depth can be obtained from single image. We also discuss optimized strategies for wind-resistant control. In Chapter 3, we introduce our novel introspection system to predict failures in visual systems and in Chapter 5, we show how to learn transferable policies across domains. Finally, we show a direction for future work and conclude.

Chapter 2

Robust Monocular Visual Navigation

Contents

2.1 System Overview 7
2.2 Monocular Depth Estimation 9
2.3 Multiple Predictions 15
2.4 Planning and Control 17
2.5 Experimental and Results 21
2.6 Summary 26

Recently, there have been numerous advances in the development of biologically inspired lightweight Micro Aerial Vehicles (MAVs). While autonomous navigation is fairly straightforward for large UAVs as expensive sensors and monitoring devices can be employed, robust methods for obstacle avoidance remains a challenging task for MAVs which operate at low altitude in cluttered unstructured environments. Due to payload and power constraints, it is necessary for such systems to have autonomous navigation and flight capabilities using mostly passive sensors such as cameras. In this chapter, we describe a robust system that enables autonomous navigation of small agile quad-rotors at low altitude through natural forest environments. We present a direct depth estimation approach that is capable of producing accurate, semi-dense depth-maps in real time. Furthermore, a novel wind-resistant control scheme is presented that enables stable way-point tracking even in the presence of strong winds. We demonstrate the performance of our system through extensive experiments on real images and field tests in a cluttered outdoor environment.

2.1 System Overview

In this section, we describe the hardware platforms and overall software architecture. Developing and testing all the integrated modules of an MAV "in the wild" is very challenging. Therefore, in addition to a MAV (Figure 2.1a), we also assembled a rover (Figure 2.1b) to be able to test

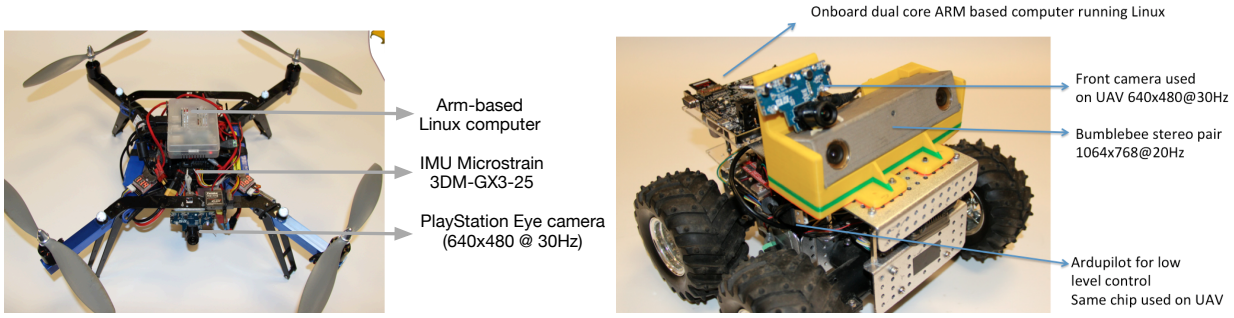


Figure 2.1: Hardware Platforms. (Left) Quadrotor used as our development platform. (Right) Rover assembled with the same hardware and software configuration as the MAV for rapid tandem development and validation of modules.

various modules separately. Along with facilitate parallel development and testing of modules, the rover was also used for data collection purposes.

Quadrotor Platform

We have used a modified version of the 3DR ArduCopter 2.1 with Odroid XU-3, a quad-core ARM processor that runs Ubuntu 14.04 and ROS Groovy. The base chassis, motors and autopilot are assembled using the Arducopter kit. Due to drift and noise of the IMU integrated in the Ardupilot unit, we added a Microstrain 3DM-GX3-25 IMU, that aids with real-time pose estimation. Onboard, there are two monocular cameras: one PlayStation Eye camera (320×240 at 60 fps) facing downward for state estimation and one high-dynamic range PointGrey Chameleon camera (640×480 at 30 fps) for monocular navigation. A single beam lidar based sensor is used to resolve the metric altitude.

Mobile Ground Robot

We built a skid-steered ground robot that has all the software and hardware aspects exactly as the MAV, other than the low-level controllers, to allow seamless transfer of modules. In addition, for data collection purposes, a Bumblebee color stereo camera pair (1024×768 at 20 fps) is rigidly mounted with respect to the front camera using a custom 3D printed fiber plastic encasing. Even though the validation images are from a slightly different perspective than encountered by the MAV during flight, we found in practice that modules generalized well.

Software Architecture

We have a distributed processing framework for the high-level planning as shown in Figure 2.2, where an image stream from the front facing camera is streamed to the base station (at 15 Hz) where the perception module produces a local 3D scene map; the planning module uses these maps to find the best trajectory to follow and transmits it back to the on-board computer where the control module does trajectory tracking. The resulting desired velocity control commands are sent to the Ardupilot which sends low level control commands to the motor controllers to

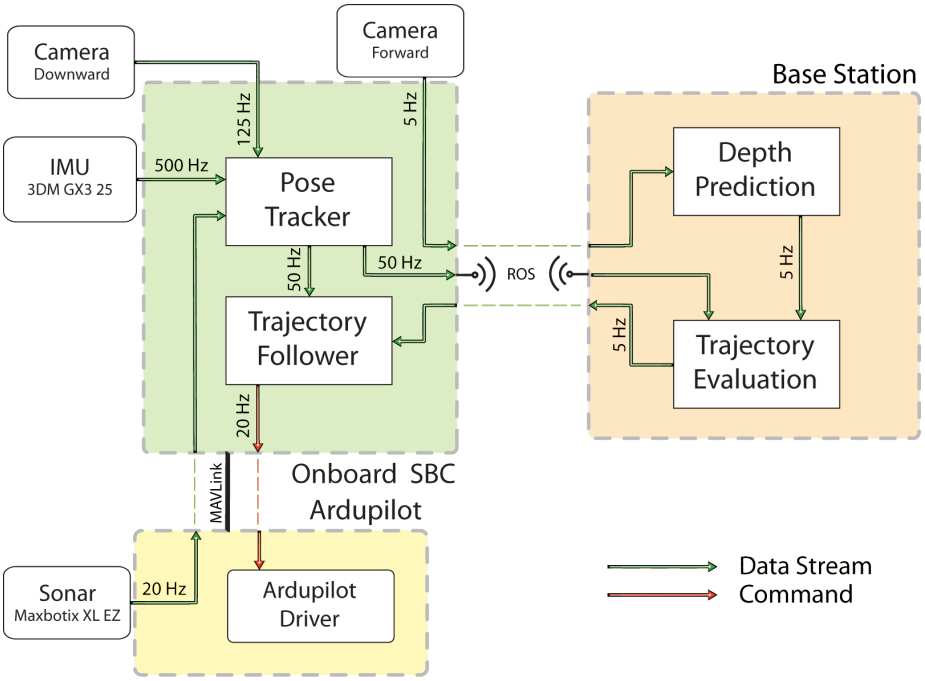


Figure 2.2: Software Architecture. The flow of data and commands across our distributed processing framework. Pose estimation and trajectory tracking is done online, whereas the high-level planning is done off-board on a base-station. The communication is done over WiFi at 15 Hz.

achieve the desired motion. The pose estimation and trajectory following modules is run on-board to minimize the effects of latency.

2.2 Monocular Depth Estimation

The perception system runs on the base station and is responsible for providing a 3D scene structure which can be used for motion planning and control strategies. Depth estimation from a monocular video sequence is a fundamental problem in computer vision. Most existing methods, such as structure from motion (SFM) and motion stereo, are only applicable to static scenes and require the camera motion to introduce parallax. In contrast, in this section, we present two monocular depth estimation techniques for video sequences acquired by an MAV, that involves non-translational cameras.

2.2.1 Deep Learning based Depth Prediction

In recent years, deep learning approaches have achieved significant success on a large variety of computer vision tasks like image classification [39, 53, 85, 90], object detection [33], semantic segmentation [62]. Such methods have recently outperformed even humans in image classification tasks [39, 90]. Most of these methods use various neural network (NN) architectures

for learning complicated non-linear models from large amount of supervized data. Neural networks have been used in the past for learning autonomous driving policies from camera streams [70, 74]. Fouhey et al. [28] have used deep networks for surface normal prediction from single images, Eigen et al. [23] have used CNN's for both surface normal and depth prediction from single images. In this section we leverage recent advances in deep learning to train a convolutional neural network [53] for predicting depth at every pixel for a video sequence from the front-facing camera stream of the MAV.

Convolutional neural networks (CNN) are a particular kind of neural network particularly suited for vision tasks. A typical CNN architecture has many successive layers of filters which are convolved over the output of previous layers, and their outputs are passed on to subsequent layers. The input to the network is the image and the output in our case is depth in meters at every pixel of the image. These networks are trained by backpropagation [76] on a differentiable loss function appropriately designed for the task at hand. To handle large amounts of data stochastic gradient descent (SGD) [6] is usually used as the solver. SGD only computes the gradient needed for backpropagation with respect to small batches of the data instead of with respect to the entire dataset (as is the case in regular batch gradient descent). Modern implementations like Caffe [47] and Theano [5] can use modern graphical processing units (GPUs) for fast training and inference. Furthermore they can seamlessly switch to CPU for both training and inference if a suitable GPU is not available. This is very convenient since once can train these networks on expensive but fast GPUs and then do inference on regular CPUs mounted on a robot or in a laptop serving as the base station.

Network Architecture

Figure 2.3 shows the CNN architecture we have used for predicting depth at every pixel of an incoming image. The network architecture is the same as that used by Krizhevsky et al. [53] for classifying images in the ImageNet classification challenge in 2012. This network is also known informally as "AlexNet". The difference is that our loss function is the squared loss for regressing to the ground truth depth value as opposed to the softmax loss function used for multi-class classification in the ImageNet task. In Figure 2.3, the image is passed in as the first layer which is the data layer. This is followed by two sets of convolution, rectified linear unit, local response normalization and pooling layers. These are indicated in the diagram as `conv1`, `relu1`, `norm1`, `pool1` and `conv2`, `relu2` etc. These are further followed by another three sets of convolution and rectified linear unit layers followed by a final pooling layer (`pool5`). The output of `pool5` is given to two fully connected layers `fc6` and `fc7` which finally feed their outputs to the last layer which is also fully connected `fc8`. During training phase, the output of `fc8` layer goes to the squared loss layer which determines how off we are from the ground truth depth data obtained from stereo. This squared error is then used to compute the gradients needed for backpropagation which tweaks the weights of each neuron such that the output will in turn lower the error. During inference time, the loss layer is not used anymore and the output of `fc8` constitutes the depth image which is then used for trajectory evaluation by the planning layer.

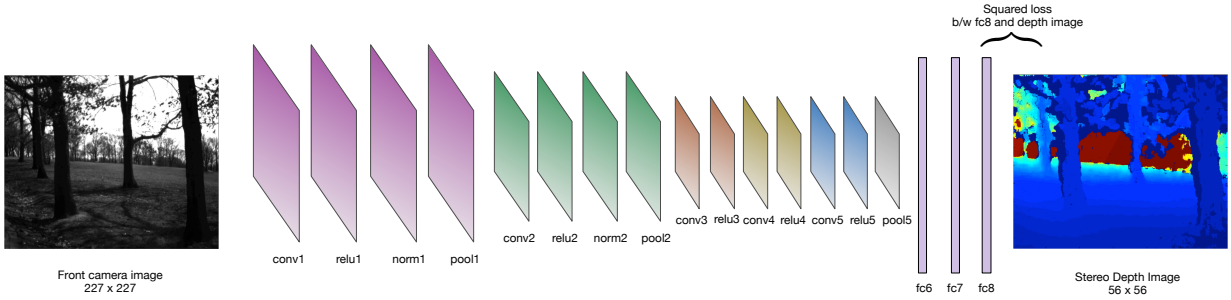


Figure 2.3: We use the AlexNet architecture from [53] to train a CNN for depth prediction. The input is an image from the front camera of the MAV and the output at the last layer (`fc8`) is a 56×56 depth image. The network is trained by using backpropagation and stochastic gradient descent as the solver. Depth images obtained from stereo image processing are used as the ground truth values for computing the squared error and gradient used in backpropagation.

Figure 2.4: Example results from our deep network based depth regression scheme. (Left) Image, (Middle) Ground truth (Right) Predicted depth map.

Implementation Details

The front camera image stream is resized to 227×227 pixels for input to the CNN. For training, ground truth depth values are obtained from the Bumblebee stereo camera registered to the front camera of the MAV. We collected a dataset of 60k images and their corresponding stereo depth images. 50k images were used for training while 10k images were used for validation. We take the logarithm of depth values for training the network to keep errors scale-independent. During inference time the predicted depth values are exponentiated back. The depth images are also binned into 4×4 windows to produce a 56×56 depth image. All depth values which are less than 1 m are filtered out. This is because if the MAV gets less than 1 m to any obstacle then it is inevitably going to crash and the CNN should concentrate on accurately predicting obstacles further than 1 m away. The parameters of stereo processing are set conservatively so that only high confidence regions are kept and others are rejected and marked invalid with the special value 0 in the resulting depth image. During backpropagation, these invalid pixel values are ignored for error calculation. We use the Caffe [47] toolbox and a NVIDIA Titan Z graphics card [72] with 5760 parallel GPU cores and 12 gb of gpu ram for training the depth prediction model. Inference

is done off-board but online on the ground station laptop on a quad core Intel i7 processor which is able to do depth prediction at 13Hz on 640×480 pixel images from the front camera stream. Figure 2.4 shows example image, and predicted depth images.

2.2.2 Direct Visual Odometry based Depth Estimation

In the last few years, direct approaches [31, 89, 93, 101] for scene geometry reconstruction have become increasingly popular. Instead of operating solely on visual features, these methods directly work on the image intensities for both mapping and tracking: The world is modeled as a dense surface while in turn new frames are tracked using whole-image alignment. In addition to higher accuracy and robustness, in environments with few features, this provides substantially more information about the geometry of the environment. In this section, we build upon recently published work on monocular visual odometry [12, 25] to present a complex system focusing on depth map estimation for fast obstacle avoidance and then introduces our contributions for robust behavior in cluttered environment as explained below.

Mapping:

The depth estimates are obtained by a probabilistic approach for adaptive-baseline stereo [24]. This method explicitly takes into account the knowledge that in video, small baseline frames occurs before large baseline frames. A subset of pixels is selected for which the disparity is sufficiently large, and for each selected pixel a suitable reference frame is selected and a one dimensional disparity search is performed. The obtained disparity is converted to an inverse-depth representation, where the inverse depth is directly proportional to the disparity. The map is then updated using this inverse depth estimate. The inverse depth map is propagated to subsequent frames, once the pose of the following frames has been determined and refined with new stereo depth measurements. Based on the inverse depth estimate d_0 for the pixel, the corresponding 3D point is calculated and projected onto the new frame and assigned to the closest integer pixel position providing the new inverse depth estimate d_1. Now, for each frame, after the depth map has been updated, a regularization step is performed by assigning each inverse depth value the average of the surrounding inverse depths, weighted by their respective inverse variance (σ^2). An example of the obtained depth estimates has been shown in Figure 2.5. Note: In order to retain sharp edges, which can be critical for detecting trees, we only perform this step if the two adjacent depth values are statistically similar i.e. their variance is within 2σ.

Tracking:

Given an image $I_M : \Omega \rightarrow \mathbf{R}$, we represent the inverse depth map as $D_M : \Omega_D \rightarrow \mathbf{R}^+$, where Ω_D contains all pixels which have a valid depth. The camera pose of the new frame is estimated using direct image alignment. The relative pose $\xi \in SE(3)$ of a new frame I, is obtained by directly minimizing the photometric error:

$$E(\xi) := \sum_{x \in \Omega_{D_M}} \|I_M(x) - I(w(x, D_m(x), \xi))\|_\delta \tag{2.1}$$

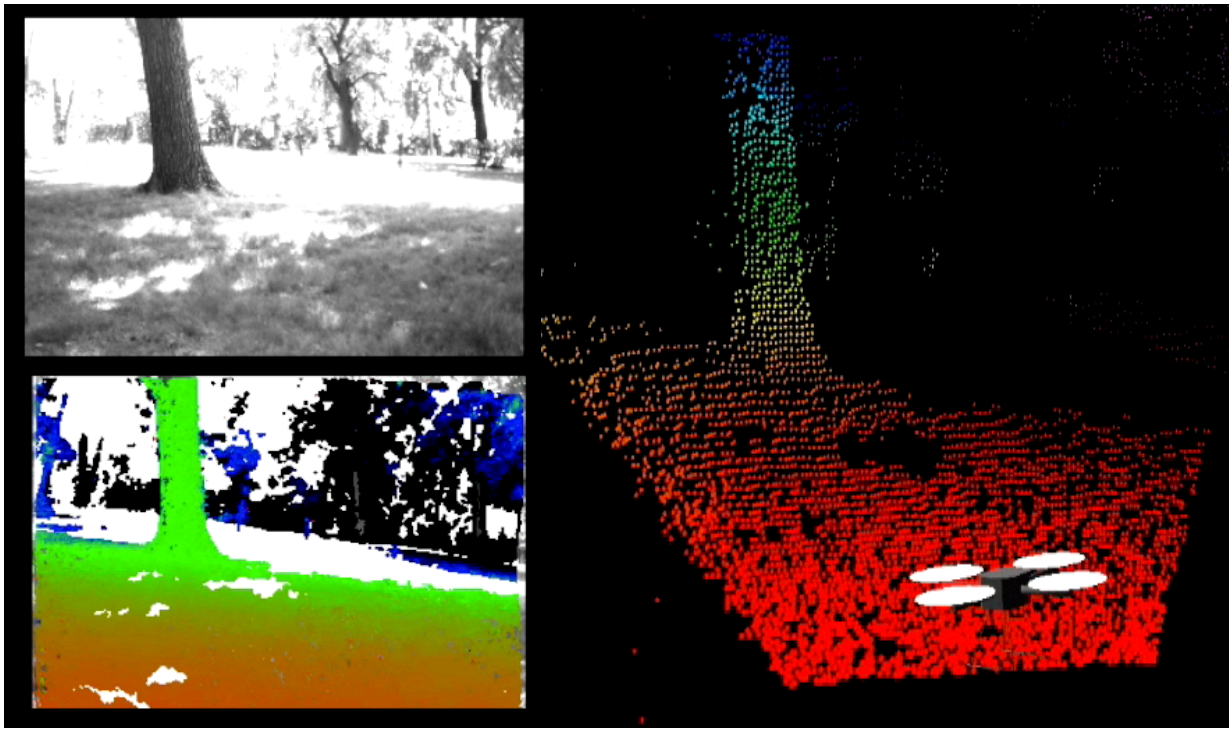


Figure 2.5: Semi-dense Depth Map Estimation. (Top left) Example image frame (Bottom left) Depth map. Colormap: Inverted Jet, Black pixels are for invalid pixels (Right) Corresponding 3D point cloud. Color based on height

where $w : \Omega_{D_M} \times \mathbf{R} \times SE(3) \rightarrow \omega$ projects a point x from the reference frame image into the new frame and $\| \cdot \|_\delta$ is the Huber norm to account for outliers. The minimum is computed using iteratively re-weighted Levenberg-Marquardt minimization [25].

Scale Estimation:

Scale ambiguity is inherent to all monocular visual odometry based methods. This is not critical in visual mapping tasks, where the external scale can be obtained using fiducial markers [13, 78, 79]. However, for obstacle avoidance in real-time, it is required to accurately recover the current scale so that the distance to the object is known in real world units. We resolve the absolute scale $\lambda \in \mathbf{R}^+$ by leveraging motion estimation from a highly accurate single beam laser lite sensor onboard. We measure, at regular intervals (operating at 15 Hz), the estimated distance traveled according to the visual odometry $\mathbf{x}_i \in \mathbf{R}^3$ and the metric sensors $\mathbf{y}_i \in \mathbf{R}^3$. Given such sample pairs $(\mathbf{x}_i, \mathbf{y}_i)$, we obtain a scale $\lambda(t_i) \in \mathbf{R}$ as the running arithmetic average of the quotients $\frac{\|\mathbf{x}_i\|}{\|\mathbf{y}_i\|}$ over a small window size. We further pass the obtained set of scale measurements through a low-pass filter in order to avoid erroneous measurements due to sensor noise. The true scale λ thus obtained is used to scale the depth map to real world units.

2.2.3 IMU Pre-Integration on Lie-Manifolds

Visual Odometry (VO) based approaches, like the one described above, are inherently susceptible to strong inter-frame rotations. One possible solution is to use a high frame rate camera. However, this has its own limitations in terms of power and computational complexity. Rather we build upon the recent advancements in visual-inertial navigation methods [48, 83, 99] that use an Inertial Measurement Unit (IMU) to provide robust and accurate inter-frame motion estimates.

We denote with $\mathcal{I}_{ij}$ the set of IMU measurements acquired between two consecutive keyframes i and j. We denote with $\mathcal{C}_i$ the camera measurements at keyframe i and $\mathcal{K}_k$ denotes the set of all keyframes up to time k. Usually, each set $\mathcal{I}_{ij}$ contains hundreds of IMU measurements. The set of measurements collected up to time k is

$$\mathcal{Z}_k = \{\mathcal{C}_i, \mathcal{I}_{ij}\}_{i,j \in \mathcal{K}_k} \tag{2.2}$$

We use this to infer the motion of the system from IMU measurements. All measurements between two keyframes at times $k = i$ and $k = j$ can be summarized in a single compound measurement, named *preintegrated IMU measurement*, which constrains the motion between consecutive keyframes. This concept was first proposed by Lupton and Sukkarieh [64] using Euler angles and Forster et al. [27] extended it, by developing a suitable theory for preintegration on Lie-manifolds. The resulting *preintegrated rotation increment* is given as:

$$\Delta\tilde{R}_{ij} \doteq \sum_{k=i}^{j-1} \text{Exp}((\tilde{w}_k - b_i)\Delta t) \tag{2.3}$$

where $\tilde{w}_k \in \mathrm{R}^3$ is the instantaneous angular velocity of the k^{th} frame, relative the world frame. b_i is a constant noise bias at time t_i and the shorthand $\Delta t \doteq \sum_{k=i}^{j} \Delta t$. Equation 2.3 provides an estimate of the rotational motion between time t_i and t_j, as estimated from inertial measurements.

Now, the warp function w as described in Eq. 1 can be written as:

$$w(x, D_M, T) = \pi(T\pi^{-1}(x, D_M(x))). \tag{2.4}$$

where $\pi^{-1}(x, D_M)$ represents the inverse projection function and $\mathrm{T} \in SE(3)$ represents the 3D rigid body transformation. The estimated motion transform is updated using the calculated preintegrated inertial rotation from the previous step such that

$$T_{imu} = \begin{bmatrix} \Delta\tilde{R}_{imu} & \mathrm{t} \\ 0 & 1 \end{bmatrix} \tag{2.5}$$

where $\Delta\tilde{R}_{imu} \in SO(3)$ and $\mathrm{t} \in \mathrm{R}^3$. During optimization, a minimal representation for the pose is required, which is given by the corresponding element $\xi \in SE(3)$ of the associated Lie-algebra such that $\xi_{imu}^{(n)} = \log_{SE(3)}(T_{imu})$. In the prediction step, the new estimate is obtained by multiplication with the computed update:

$$\xi^{(n+1)} = \delta\xi^{(n)} \circ \xi_{imu}^{(n)} \tag{2.6}$$

where $\delta(\xi)^n$ is computed by solving for the minimum of a Gauss-Newton second order approximation of the photometric error *E*. As the optimizer is fed with more robust rotational estimates, the resulting pose after visual-inertial fusion is less susceptible to strong inter-frame rotations.

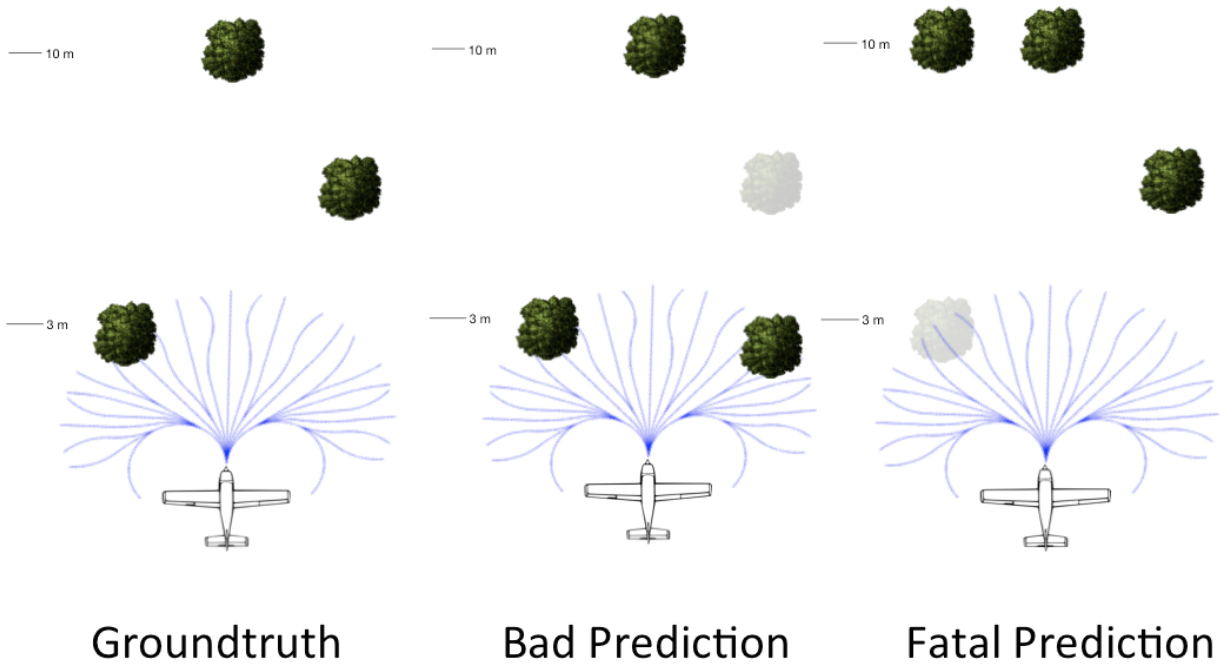


Figure 2.6: Illustration of the complicated nature of the loss function for collision avoidance. (Left) Groundtruth tree locations (Middle) Bad prediction where a tree is predicted closer than it actually is located (Right) Fatal prediction where a tree close by is mispredicted further away.

2.3 Multiple Predictions

The monocular depth estimates are often noisy and inaccurate due to the challenging nature of the problem. A planning system must incorporate this uncertainty to achieve safe flight. Figure 2.6 illustrates the difficulty of trying to train a predictive method for building a perception system for collision avoidance. Figure 2.6 (left) shows a ground truth location of trees in the vicinity of an autonomous MAV. Figure 2.6 (middle) shows the location of the trees as predicted by the perception system. In this prediction the trees on the left and far away in front are predicted correctly but the tree on the right is predicted close to the MAV. This will cause the MAV to dodge a ghost obstacle. While this is bad, it is not fatal because the MAV will not crash but make some extraneous motions. But the prediction of trees in Figure 2.6 (right) is potentially fatal. Here the trees far away in front and on the right are correctly predicted whereas the tree on the left originally close to the MAV, is mispredicted to be far away. This type of mistake will cause the MAV to crash into an obstacle it does not know is there.

Ideally, a vision-based perception system should be trained to minimize loss functions which will penalize such fatal predictions more than other kind of predictions. But even writing down such a loss function is difficult. Therefore most monocular depth perception systems try to minimize easy to optimize surrogate loss functions like regularized L_1 or L_2 loss [80]. We try to reduce the probability of collision by generating multiple interpretations of the scene to hedge against the risk of committing to a single potentially fatal interpretation. Specifically we generate 3 interpretations of the scene and evaluate the trajectories in all of them. The trajectory which

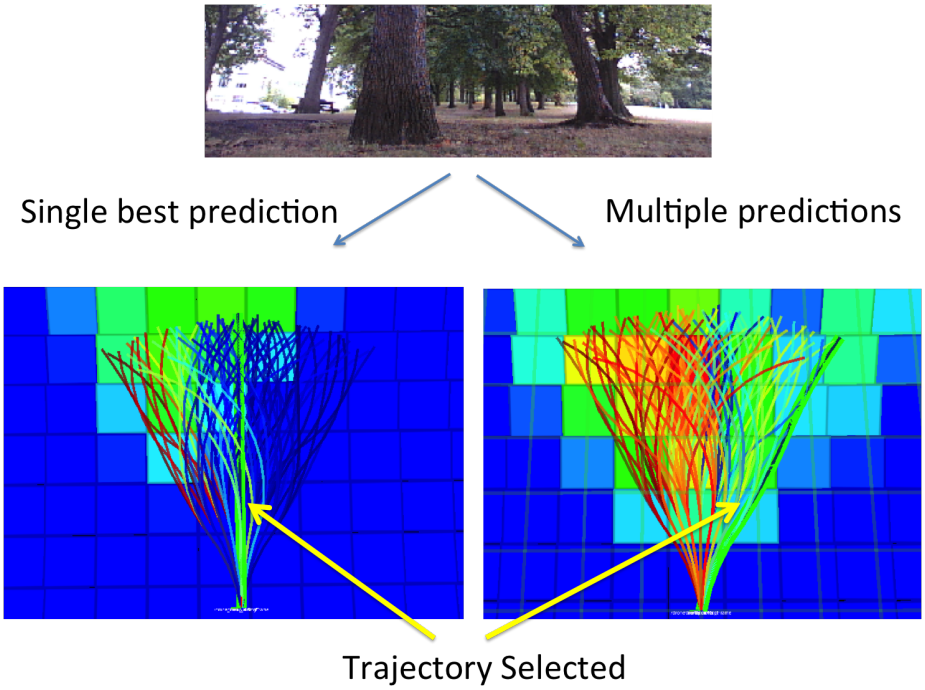


Figure 2.7: The scene at top is an example from the front camera of the MAV. On the left is shown the predicted traversability map (red is high cost, blue is low cost) resulting from a single interpretation of the scene. Here the MAV has selected the straight path (thick, green) which will make it collide with the tree right in front. While on the right the traversability map is constructed from multiple interpretations of the image, leading to the trajectory in the right being selected which will make the MAV avoid collision.

is least likely to collide on average in all interpretations is then chosen as the one to traverse. One way of making multiple predictions is to just sample the posterior distribution of a learnt predictor. In order to truly capture the uncertainty of the predictor, a lot of interpretations have to be sampled and trajectories evaluated on each of them. A large number of samples will be from around the peaks of this distribution leading to wasted samples. This is not feasible given the real time constraints of the problem

In previous work, Dey et al. [18] have developed techniques for predicting a budgeted number of interpretations of an environment with applications to manipulation, planning and control. These approaches try to come up with a small number of relevant but diverse interpretations of the scene so that at least one of them is correct. In this work, we adopt a similar philosophy and use the error profile of the deep learning based depth regressor described in the Section 2.2.1 to make two additional predictions: For each depth value predicted by it, the average over-prediction and under-prediction error is recorded. For example the predictor may say that an image patch is at 3 meters while it is actually either, on average, at 4 meters or at 2:5 meters. We round each prediction depth to the nearest integer, and record the average over and under-predictions as in the above example in a look-up table (LUT). At test time the predictor produces a depth map and the LUT is applied to this depth map, producing two additional depth maps: one for over-prediction error, and one for the under-prediction error.

In a similar spirit, we make multiple predictions by utilizing the variance of the estimated inverse depth which is already calculated in our direct visual odometry framework. At every pixel, the variance of the inverse depth is used to find the inverse depth value one standard deviation away from the mean (both lower than and higher than the mean value) and inverted to obtain a depth value. So, a total of 3 depth predictions are made: 1) mean depth estimate 2) depth estimate at one standard deviation lower than the mean depth at every pixel and 3) depth estimate at one standard deviation greater than the mean depth at every pixel. Figure 2.7 shows an example in which making multiple predictions is clearly beneficial compared to the single best interpretation (using the deep learning based depth estimation method).

2.4 Planning and Control

2.4.1 Receding Horizon Planner

We use a semi-global planning approach in a receding horizon planning scheme [51]. To the best of our knowledge, this is the first demonstration of receding horizon control with monocular vision implementation on a MAV. Once the planner module receives a scaled-depth map from the perception module, it projects it to a 3D point cloud representation, and the local map is updated using the current pose of the MAV. A trajectory library of 78 trajectories of length 5 meters is budgeted and picked from a much larger library of 2401 trajectories using the maximum dispersion algorithm by Green et al. [35]. This is a greedy procedure for selecting trajectories, one at a time, so that each subsequent trajectory spans maximum area between it and the rest of the trajectories. Figure 2.8 shows our quadrotor evaluating a set of trajectories on the projected depth image obtained from monocular depth prediction and traversing the chosen one.

Our system accepts a goal direction as input and ensures that the vehicle makes progress towards the goal while avoiding obstacles along the way. For each of the budgeted trajectories a score value for every point in the point cloud, is calculated by taking into account several factors, including the distance to goal, cost of collision, etc., and the optimal trajectory to follow is selected. Further, the receding horizon module maintains a score for every point in the point cloud. The score of a point decays exponentially the longer it exists. After some time when it drops below a user set threshold, the point is deleted. The decay rate is specified by setting the time constant of the decaying function. This fading memory representation of the local scene layout has two advantages: Firstly, it prevents collisions caused by narrow field-of-view issues where the MAV forgets that it has just avoided a tree, sees the next tree and dodges sideways, crashing into the just avoided tree. Secondly, it also allows emergency backtracking maneuvers to be safely executed if required, since there is some local memory of the obstacles it has just passed.

The control module takes as input the selected trajectory to follow and generates waypoints to track using a pure pursuit strategy [10]. Specifically, this involves finding the closest point on the trajectory from the robot's current estimated position and setting the target waypoint to be a certain fixed lookahead distance further along the trajectory. The lookahead distance can be tuned to obtain the desired smoothness while following the trajectory; a larger lookahead distance leads to smoother motions, at the cost of not following the trajectory exactly. Using the

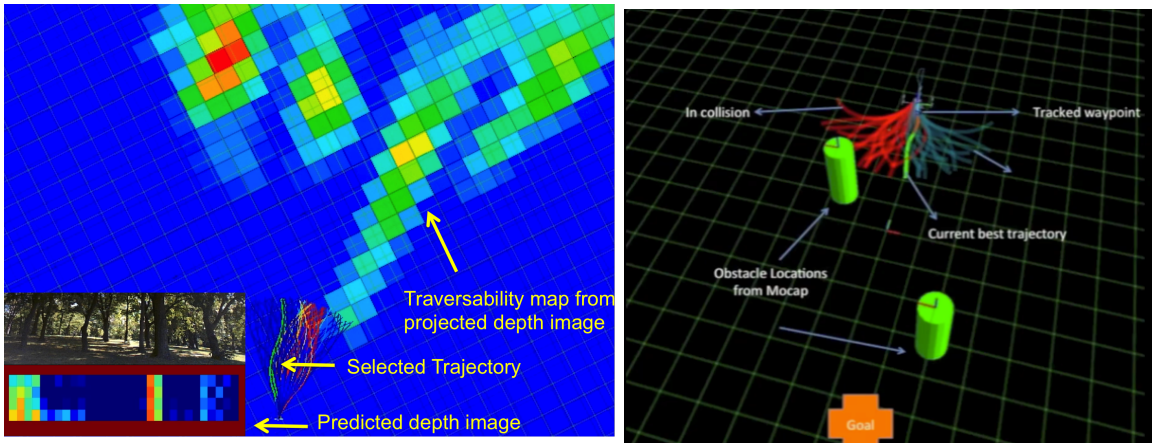


Figure 2.8: (Left) Example of receding horizon with a quadrotor using monocular vision. The lower left images show the view from the front camera and the corresponding depth images from the monocular depth perception module. The rest of the figure shows the overhead view of the quadrotor and the traversability map (builtby projecting out the depth image) where red indicates higher obstacle density. The grid is $1 \times 1\ m^2$. The trajectories are evaluated on the projected depth image and the one with the least collision score (thick green) trajectory followed. (Right) Receding horizon control on MAV in motion capture. A library of 78 trajectories of length 5 m are evaluated to find the best collision-free trajectory. This is followed for some time and the process repeated.

pose estimates of the vehicle, the MAV moves towards the next waypoint using a LQR controller as described in Section 2.4.3. Figure 2.9 shows the overall flow of data and control commands between various modules. For further details on our planning approach, we direct the reader to previous work by Daftry et al. [14], Dey et al. [19].

2.4.2 State Estimation

In receding horizon, one only needs a relative, consistent pose estimation system as trajectories are followed only for a short duration (3 seconds in our implementation). As looking forward to determine pose is ill-conditioned due to a lack of parallax, we use a downward looking camera in conjunction with a single beam lidar for determining this relative pose. There are still significant challenges involved when looking down. Texture is often very self similar making it challenging for traditional feature based methods [52, 71] to be used. We used a variant of a Kanade-Lucas-Tomasi (KLT) tracker [95] to detect where each pixel in a grid of pixels moves over consecutive frames, and estimating the mean flow from these after rejecting outliers. We do the outlier detection step by comparing the variation of the flow vectors obtained for every pixel on the grid to a specific threshold. Whenever the variance of the flow is high, we do not calculate the mean flow velocity, and instead decay the previous velocity estimate by a constant factor. However, this estimate of flow however tries to find the best planar displacement between the two patches, and does not take into account out-of-plane rotations, due to motion of the camera. Out-of-plane camera ego-motion is compensated using motion information from the IMU and the metric scale

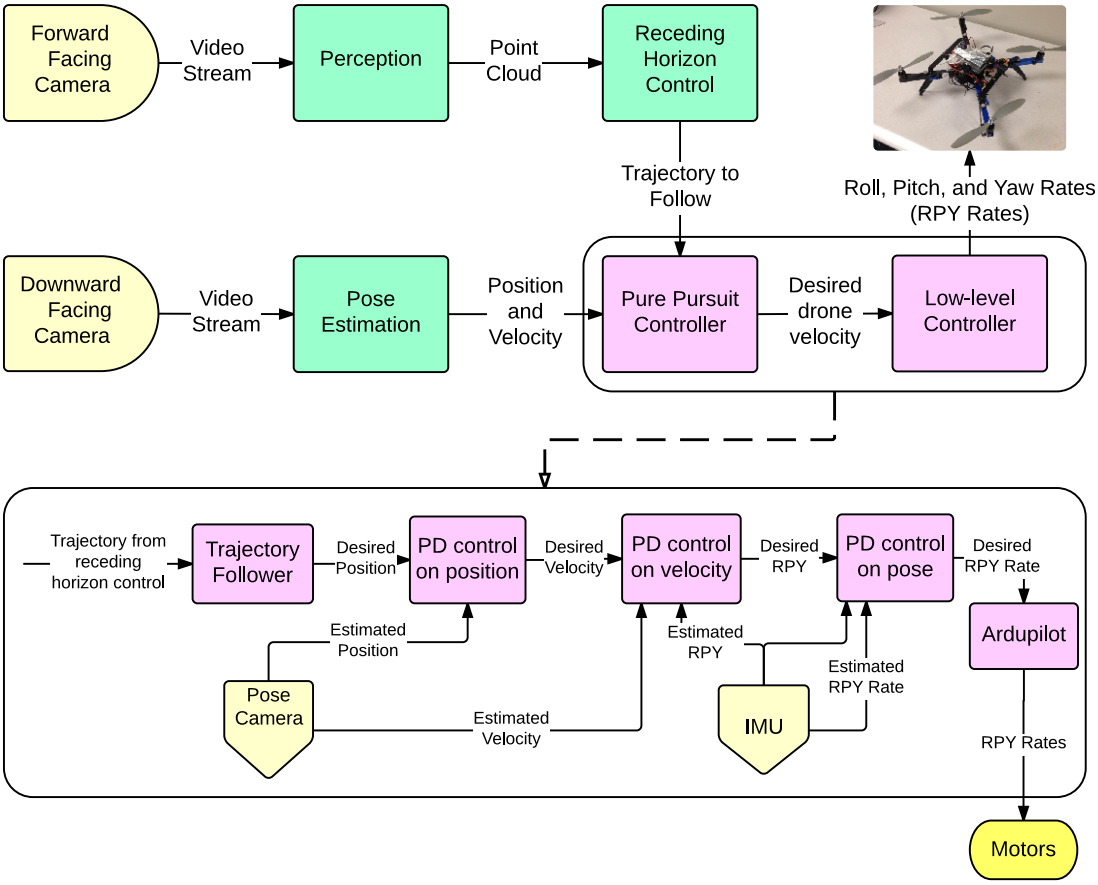


Figure 2.9: The overall flow of data and control commands between various modules. The pure pursuit trajectory follower and low-level controller (purple boxes) are shown in greater detail at the bottom.

is estimated from sonar. The computed instantaneous relative velocity between the camera and ground is integrated over time to get position. This process is computationally inexpensive, and can be run at very high frame rates. Higher frame rates lead to smaller displacements between pairs of images, which in turn makes tracking easier.

2.4.3 Wind-Resistant LQR Control

In this section, we describe our proposed approach for wind-resistant LQR control. The problem under consideration can be described as follows: Given the knowledge of the current and past states of the MAV, can we estimate the current wind conditions and determine appropriate control actions that would help recover from these expected disturbances? To this end, we propose to learn a dynamic model of our MAV and observe the errors in the model's predictions in windy conditions, and compare them to errors in calm conditions in order to determine the magnitude and direction of the wind. These observations can be built into a learned model for wind behavior, which allows for future control commands to be adjusted for the predicted effects of the wind.

System Identification:

Given the application and the requisite desired performance, we are only interested in the vehicle dynamics involving small angular deviations and angular velocities. Thus, we linearize the dynamics and approximate the MAV as a linear system. Now, while this is trivial for roll and pitch, yaw is not limited to small angles in our motion model, resulting in a non-linear behavior. This non-linearity can be removed by building our model in the world frame and transforming the results to the robot frame when necessary. Thus, the robot's motion model can be represented by the following state space equation:

$$x(t+1) = \mathcal{A}x(t) + \mathcal{B}u(t) \tag{2.7}$$

where the state vector $x = [\mathrm{x}, \mathrm{y}, \dot{\mathrm{x}}, \dot{\mathrm{y}}, \theta]$ consists of x and y positions, x and y velocities and the yaw angle and $u = [\tilde{\eta}, \tilde{\phi}, \tilde{\theta}]$ is the control input vector consisting of roll, pitch and yaw control commands. The matrices $\mathcal{A}$ and $\mathcal{B}$ can be obtained analytically by system parameters such as mass, moment of inertia, center of mass, etc. However. these parameters are difficult to obtain in practice. Moreover, this approach cannot model the effects of the time varying aerodynamic forces. Alternatively, we run a large number of experiments to record the state and command data from actual flight and then learn the matrices $\mathcal{A}$ and $\mathcal{B}$ that best fit the data. In this work, we recast the system identification problem into a least square solution to an overdetermined set of linear equations. Specifically, the state space equations can be condensed into the form

$$X(t+1) = [\mathcal{A}, \mathcal{B}] \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \tag{2.8}$$

where the matrix $[\mathcal{A}, \mathcal{B}]$ can be solved using linear regression.

Wind Modeling, Detection, and Resistance:

Traditional approaches for wind detection, model the gust forces assuming a static dominant direction derived from the Drysden model [88]. However, this model is only applicable to MAVs in open terrain and fails for MAV flights in cluttered environments like forests, as the gush of wind has a profile of varying velocity on a time-scale shorter than the scale of the vehicle flight time. Thus, we use an alternate approach for determining the wind disturbance, which relies on the use of the acceleration data. In our proposed approach we attempt to understand the behaviors of our learned system model in both windy and wind-free conditions. Any deviations that should occur, given the current state and inputs, can be attributed to wind disturbances. In particular, the effects of the wind on the system dynamics can be observed by analyzing the differences in model errors in both conditions (See Figure 2.13a). The magnitude of the difference at a given time represents the current wind impact on the MAV and the direction can be determined by x and y components of the velocity error since the wind should cause unexpected acceleration in the direction of the wind. Furthermore, the differences in model errors between the average error distribution and the current real time distribution at a given time can be broken down into each state's error. Additionally, if we assume that the current wind magnitude and direction is the

same as it was in the previous time step, the MAV model's state can be extended to include wind resulting in a new model:

$$x(t+1) = \mathcal{A}x(t) + \mathcal{B}u(t) + w(t) \tag{2.9}$$

where $w = [w_x, w_y, w_{\dot{x}}, w_{\dot{y}}, 0]$ are the wind-bias correction terms corresponding to x, y positions and x, y velocities. We assume the wind to have no impact on the yaw angles. The validity of these assumptions can be determined by observing the variance in the previous N errors. If the variance is low, it is likely that the next error will be the same as the past few and so the model should be updated to correct for this error since wind comes in low frequency bursts. If the variance is high, then the error is can be attributed to random noise or other modeling error and it is ignored in future predictions. This process can be executed in real time during flight allowing for current estimates for the wind force.

Controller Design:

The system dynamics model is used to implement an output feedback control law to track a given trajectory. An LQR controller [108], $u = -Kx$ was designed to minimize the following quadratic cost function

$$J = \lim_{N\to\infty} E(\sum_{k=1}^{N} x(t)^T \mathcal{Q}x(t) + u(t)^T \mathcal{R}u(t)) \tag{2.10}$$

where $\mathcal{Q} \geq 0$ and $\mathcal{R} \geq 0$ are the weighing matrices that reflect the tradeoff between regulation performance and control effort. The diagonal entries in the weighing matrices are iteratively tuned [17] to ensure a good transient response without saturating the control inputs.

2.5 Experimental and Results

In this section we analyze the performance of our proposed method for robust monocular MAV flight through outdoor cluttered environments. All the experiments were conducted in a densely cluttered forest area as shown in Figure 3.4, while restraining the MAV through a light-weight tether. It is to be noted that the tether is only for compliance to federal regulations and does not limit the feasibility of a free flight.

2.5.1 Performance Evaluation of Monocular Depth Estimation

Ground truth

In order to collect groundtruth depth values, a Bumblebee color stereo camera pair (1024×768 at 20 $\texttt{Hz}$) was rigidly mounted with respect to the front camera using a custom 3D printed fiber plastic encasing. We calibrate the rigid body transform between the front camera and the left camera of the stereo pair using the flexible camera calibration toolbox from Daftry et al.

Figure 2.10: (Top Left) Testing area near Carnegie Mellon University, Pittsburgh, USA and (Right) An example image illustrating the density of the forest environment during summer. (Below) Example images during winter conditions

[11]. Stereo depth images and front camera images are recorded simultaneously while flying the drone. The depth images are then transformed to the front camera's coordinate system to provide groundtruth depth values for every pixel. The corpus of train and test data consists of 50k and 10k images respectively, and was acquired from the same test site.

Accuracy compared to Ground truth

We evaluate performance of the perception module for depth map accuracy. Figure 2.11 shows the average depth error against ground truth depth images obtained from stereo processing as explained above. As baseline, we compare our method to state-of-the-art approaches for real-time monocular depth estimation using non-linear regression [19] and Depth CNN [23]. The error plots have been binned with respect to the ground-truth, in order to show the quality of depth maps at different working scale (For a receding horizon scheme, accurate reconstruction of objects only upto a 10 m range is relevant). It can be observed that direct visual odometry performs really well with low error values up to $[15, 20]$ m. Please note that both the baseline

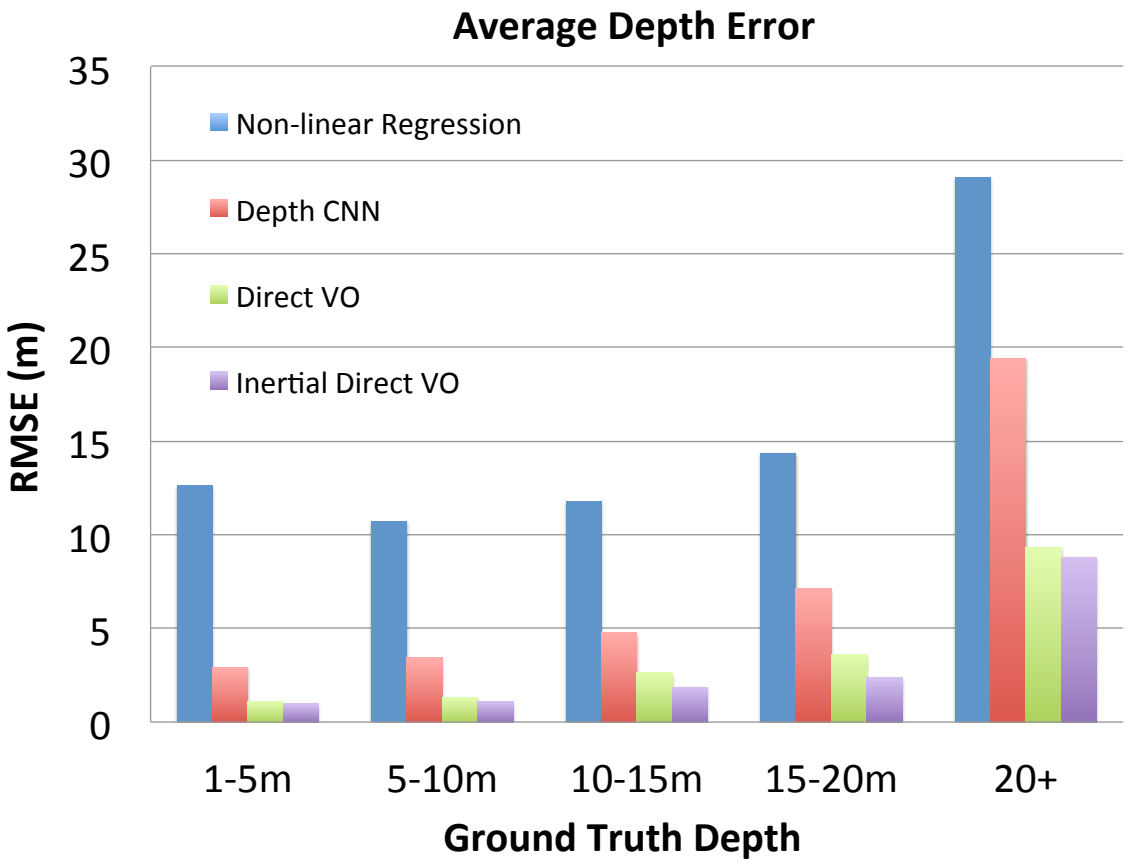


Figure 2.11: Average root-mean-squared-error (RMSE) binned over groundtruth depth buckets of [1, 5] m, [5, 10] m, etc. Groundtruth depth images are obtained from stereo image processing.

methods are learning-based approaches and thus have been trained on the training data collected. This graph nevertheless serves to show the accuracy of the our proposed perception method.

Benefits of Visual-Inertial Fusion

As discussed previously, tracking-based methods are highly susceptible to strong inter-frame rotations. We can observe from Figure. 2.11 that fusion of inertial data according to out proposed formulation improves the accuracy of the eventual depth maps. In particular, this can be directly attributed to two reasons: First, better initial conditions to the optimizer leads to faster convergence and prevents it from being stuck in a local optima. Secondly, lesser number of tracking losses; each time tracking is lost, the depth maps are initialized to random values and hence results in poor depth maps for the initial few frames till the VO recovers. Quantitatively, we observed the total tracking loss was reduced by 55% in average over a distance of 1 km. Qualitatively, the estimated depth maps can be observed to be within the uncertainty range of stereo, as shown in Figure 3.7. Its interesting to note from the qualitative results that our method is able to robustly handle even poor quality images that suffer from over- or under-exposure, etc.

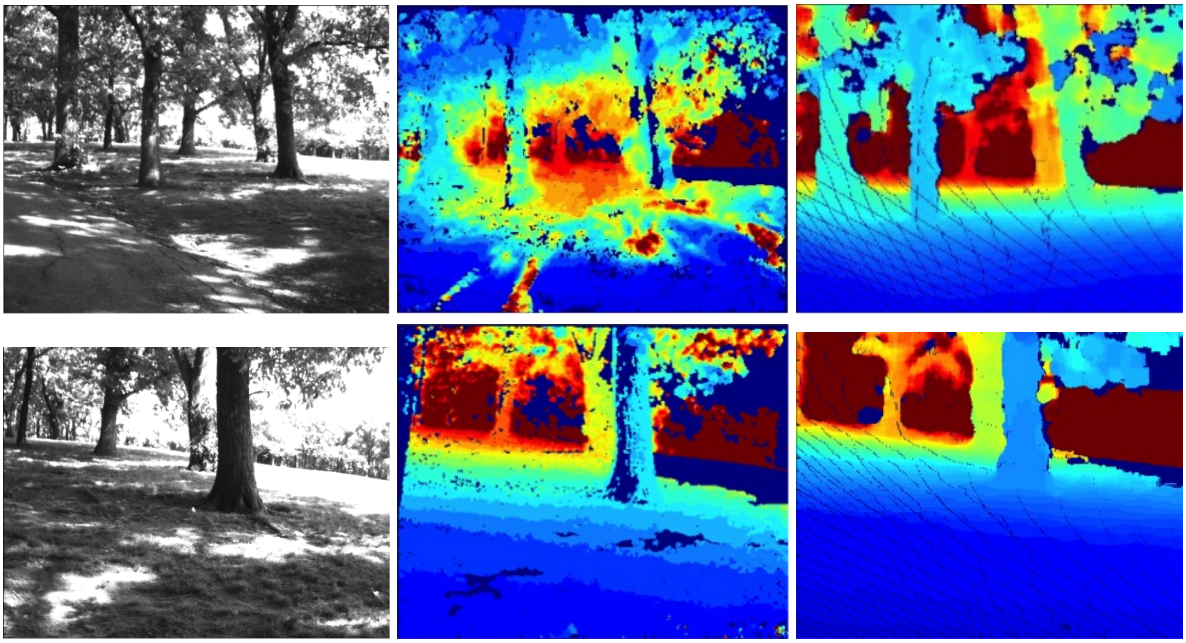


Figure 2.12: Qualitative analysis of depth maps produced by our proposed depth estimation approach with respect to stereo ground-truth. (Left) Image (Middle) estimated depth map and (Right) ground-truth depth map. Note: Jet colormap

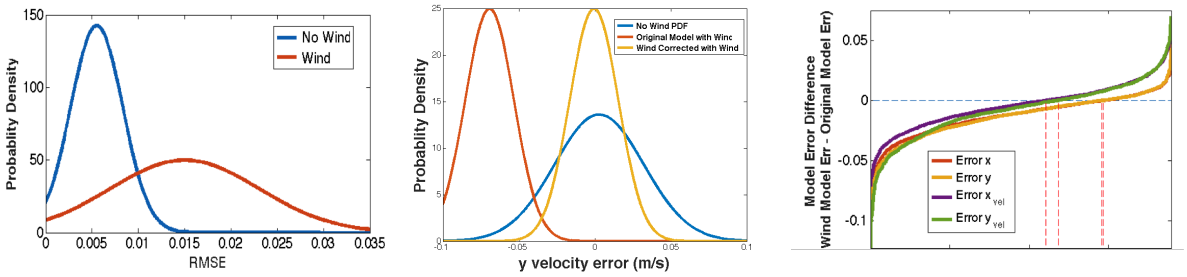
2.5.2 Performance Evaluation of Wind Estimation and Resistance

To evaluate the performance of the wind correction, we compared the accuracy of the original model and the adaptive wind correction on state and command data collected during flight under windy conditions. Figure 2.13b demonstrates how the wind correction can adjust the model's prediction during windy conditions. Specifically, the mean error in y velocity estimation during windy conditions was significantly shifted away from zero due to the force of the wind. Additionally, after wind correction, the mean modeling error returns to being centered around zero. Furthermore, our experiment demonstrated that overall, this wind correction performed significantly better than the original model during windy conditions. To elaborate, figure 2.13c shows the resulting difference between the wind corrected error and the original model's error during windy flight conditions sorted by their magnitude. As depicted by the magnitude and number of negative errors, the wind corrected model made better predictions then the original model on about 70% of the test data. Overall the wind corrected model demonstrated solid improvement across all elements of the state during flight in windy conditions.

2.5.3 System-wide Performance Evaluation

Accuracy of State Estimation:

We evaluated the performance of the flow based tracker in motion capture and compared the true motion capture tracks to the tracks returned by flow based tracker. The resulting tracks are as shown in figure 2.14



(a) (b) (c)

Figure 2.13: Wind Modeling, Detection and Correction. (a) Analysis of the differences in model prediction errors can be used to estimate the effects of the wind on the MAV. (b) Example of wind correction using our learned system model (c) Difference between the wind corrected error and the original model's error during windy flight show that wind corrected model made better predictions.

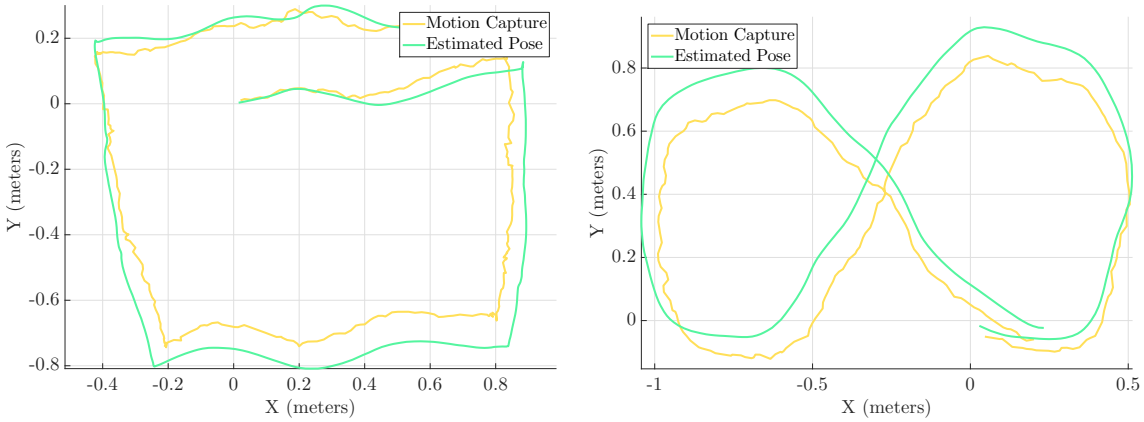


Figure 2.14: Comparison of the differential flow tracker performance vs ground truth in motion capture. Yellow tracks are the true trajectories as determined by the very accurate motion capture system, green are those determined by the algorithm. Note that due to constant replanning every 3 second, small drift in following a specific trajectory can be easily tolerated. So as long as the drift is not more than a few centimeters over a trajectory, collision avoidance is not compromised.

Autonomous Flight Tests

Quantitatively, we evaluate the performance of our system by observing the average distance flown autonomously by the MAV over several runs (at 1.5 m/s), before an intervention. An intervention, in this context, is defined as the point at which the pilot needs to overwrite the commands generated by our control system so as to prevent the drone from an inevitable crash. Experiments were performed using both the proposed multiple prediction approach and single best prediction, and results comparing to previous state-of-the-art approaches on monocular reactive control [77] and receding horizon control [19] has been shown in Figure 3.6. Tests were performed in regions of low and high clutter density (approx. 1 tree per $6 \times 6\ m^2$ and $12 \times 12\ m^2$, respectively).

It can be observed that using our perception approach MAVs can fly, at an average, 6 times

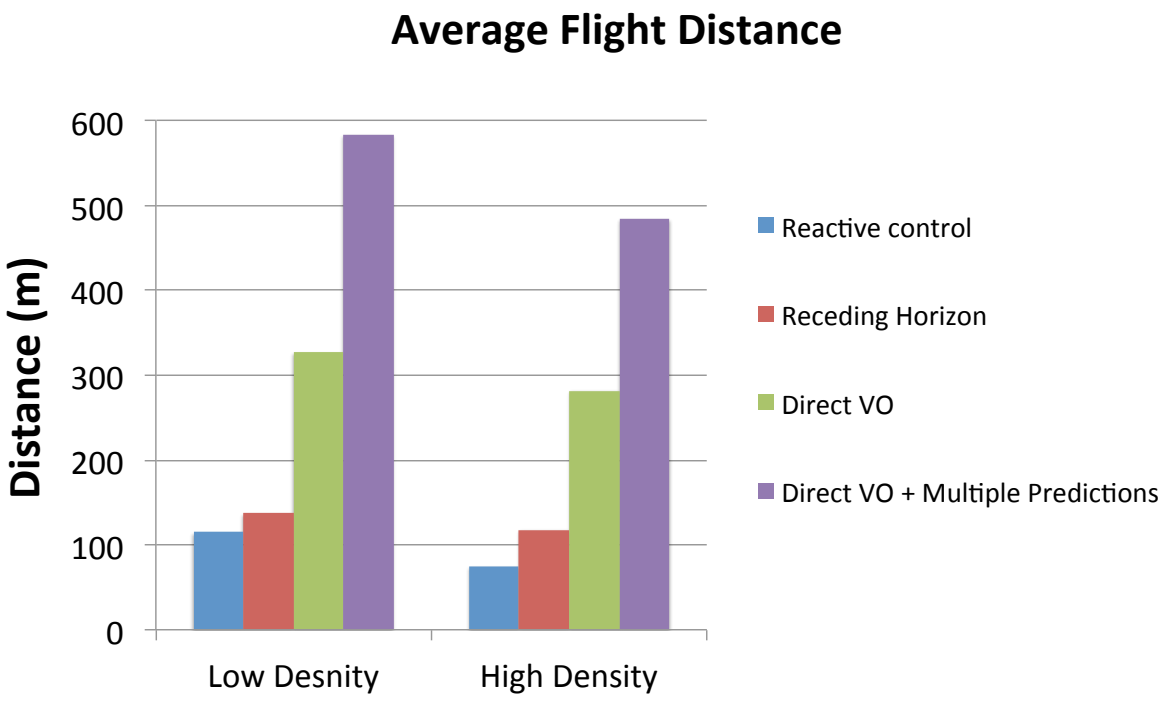


Figure 2.15: Average flight distance per intervention for (a) Low and (b) High density regions. For both, the corresponding multiple prediction variant performs significantly better.

further than pure reactive control in high density regions. While this is not surprising given the accuracy of the depth maps produced, it affirms our claim that even when we are moving forward, which is the direction of least parallax, good depth maps can be realized. Moreover, multiple predictions gives a significant boost (upto 78% improvement in low density regions) to the average flight distance over corresponding single prediction approach. In particular, the MAV was able to fly autonomously without crashing over a 580 m distance at average. This validates our intuition from Section **??**-C that by avoiding a small number of extra ghost obstacles, we can significantly reduce crashes due to uncertainty.

2.6 Summary

In this chapter, we have presented a robust approach for high-speed, autonomous MAV flight through dense forest environments. Our direct depth estimation approach enables real-time computation of accurate dense depth maps even when the MAV is flying forward, is robust to pure rotations and implicitly handles uncertainty through multiple predictions. Further, we also propose a novel formulation for wind-resistant control that allows stable waypoint tracking in the presence of strong gusts of wind. During a significant amount of outdoor experiments with flights over a distance of 2 km, our approach has avoided more than 530 trees in environments of varying density. In future work, we hope to move towards complete onboard computing of all modules to reduce latency. Another central future effort is to integrate the purely reactive

approach with the deliberative scheme detailed here, for better performance. This will allow us to perform even longer flights, in even denser forests and other cluttered environments.

Chapter 3

Introspection for Failure Recovery

Contents

3.1 Risk-Aware Perception 29
3.2 Introspection Perception 31
3.3 Introspection in Navigation 34
3.4 Experiments and Results 35
3.5 Summary 40

As autonomous robots aspire for long-term autonomous operations in complex dynamic environments, accurate situational awareness from perception becomes a pivotal requirement for verifiable safety standards. Thus, there has been a recent push towards building systems with the ability to mitigate potentially overconfident actions by an appropriate assessment of how qualified it is at that moment to make a decision. We call this self-evaluating capability as *introspection*[1]. In this chapter, we take a small step in this direction and propose a generic framework for introspective behavior in perception systems. Our goal is to learn a model to reliably predict failures in a given system, with respect to a task, directly from input sensor data. We present this in the context of vision-based autonomous MAV flight in outdoor natural environments, and show that it effectively handles uncertain situations.

3.1 Risk-Aware Perception

3.1.1 Need for Failure Prediction in Vision Systems

Perception is often the weak link in robotics systems. While considerable progress has been made in the computer vision community, the reliability of vision modules is often insufficient for long-term autonomous operation. Of course this is not surprising for two reasons. First, while vision algorithms are being systematically improved in particular by using common benchmark

[1] **Introspection.** The act or process of self-awareness; contemplation of one's own thoughts or feelings, and in the case of a robot, its current state.

datasets such as KITTI [30] or PASCAL [26], their accuracy on these datasets does not necessarily translate to the real-world situations encountered by robotics systems. We advocate that for robotics applications, which often involve mission-critical decision-making, evaluating perceptions systems based on these standard metrics is desirable but insufficient to comprehensively characterize system performance. The dimension missed is that spawned by a robot's ability to take action in ambiguous situations. Second, even if a vision algorithm had perfect performance on these off-line datasets, it is bound to encounter inputs that it cannot handle in the stream of sensor data continuously processed by a typical autonomous system. For example, over- or under-exposed images, images with massive motion blur would confound a vision-based guidance system for a mobile robot. As we aspire for long-term autonomous operation, autonomous robotics systems have to contend with vast amounts of continually evolving, unstructured sensor data from which information needs to be extracted. In particular, the problem aggravates for perception systems where the problems are often ambiguous, and algorithms are designed to work under strong assumptions that may not be valid in real-world scenario. Several factors - degraded image quality; uneven illumination; poor texture - can affect the quality of visual input, and consequentially affect the robots' action, possibly resulting in catastrophic failure. While it is possible to anticipate some of these conditions and hardcode ad-hoc test to reject such inputs, it is impossible to anticipate all of the conditions encountered in a real world environment.

This presents a challenge and an opportunity particularly to the robotics community as the real cost of failure can be significant. For example, an autonomous drone that misses to detect an obstacle can suffer disastrous consequences. They can cause catastrophic physical damage to the robot and its surroundings, which may also include people. We thus believe that it is essential to identify as early as possible, i.e., based on sensor input, situation in which a trustable decision cannot be reached because of degraded performance of the perception system. Crucially, and central to this paper, this requires the perception system to have the ability to evaluate its input based on a learned performance model of its internal algorithm. We call this capability *introspection*, a precursor to action selection. In this chapter, we explore the possibility of building introspective perception systems that can reliably predict their own failures and take remedial actions. We argue that while minimizing failures has been the primary focus of the community, embracing and effectively dealing with failures has been neglected.

We propose a generic framework for introspective behavior in perception systems that enables a robot to *know when it doesn't know* by measuring as as to how *qualified* the system it is to make a decision. This helps the system to mitigate the consequences of its failures by learning to predict the possibility of one in advance and take preventive measures. We explore these ideas in the context of our autonomous MAV flight, where mission-critical decisions equate to safety-critical consequences. This requires the MAV to successfully estimate a local map of its surrounding in real-time, at minimum the depth of objects relative to the drone, and based on this map to select an optical trajectory to reach its destination, while avoiding any obstacle. It is crucial to note that this kind of a system, like most other real-world robotics applications, involves pipelines, where the output of one system is fed into another as input. In such cases, decisions based on overconfident output from one subsystem, in this case a perception module, will lead to catastrophic failure. In contrast, anticipation of a possible failure based on previous experience allows for remedial action to be taken in the subsequent layers. Providing this more germane prediction is the introspective ability that we seek.

3.1.2 Uncertainty Propagation

Our work addresses an issue that has received attention in various communities. The concept of introspection as introduced here can be closely related to active learning [41, 49, 61], where uncertainty estimates and model selection steps are used to guide data acquisition and selection for an incremental learning algorithm. KWIK 'Knows What It Knows' frameworks [60] allow for active exploration which is beneficial in reinforcement learning problems. Further, reliably estimating the confidence of classifiers has received a lot of attention in the pattern recognition community [22, 56, 69]. Applications such as spam-filtering [15], natural language processing [2] and even computer vision [106] have used these ideas.

In robotics, the concept of introspection was first introduced by Morris *et al*. [68] and has recently been adopted more specifically for perception systems by Grimmett *et al*. [37, 38] and Triebel *et al*. [96]. They explore the introspective capacity of a classifier using a realistic assessment of the predictive variance. This is in contrast to other commonly used classification frameworks which often only rely on a one-shot (ML or MAP) solution. It is shown that the accountability of this introspection capacity is better for real world robotics applications where a realistic assessment of classification accuracy is required.

However, in all the above work, the analysis proceeds by examining the output of a system to assess its confidence on any given input. Thus, the confidence evaluation has to be designed specifically for each system. This is also particularly undesirable when the underlying system is computationally expensive. Instead, we explore the alternative approach of evaluating the input itself in order to assess the system's reliability. The main motivation for such an approach is that while perception system may not be able to reliably predict the correct output for a certain input, predicting the difficulty or ambiguity of the input may still be feasible. Moreover, this is applicable to any vision system because the reliability estimate is agnostic to the actual algorithm itself and based on the input alone.

3.2 Introspection Perception

Introspective robot perception is the ability to detect and respond to unreliable operation of the perception module. In this section, we describe a generic framework for introspective robot perception using Deep Convolutional Neural Nets, which we illustrate in the context of vision-based control of MAVs in the next section.

3.2.1 Introspection Paradigm

Introspection is distinct from conventional elements of robot architecture in that introspective processes do not reside in the path of architectural data flow as shown in **??**. Instead, the introspective layer envelops all other layers. This allows the introspective system to observe the robot's computational activity as a whole. Through observation, introspection produces estimates of the reliability of operational state and delivers system-level feedback to deliberative facilities. Our goal is to learn a model for the performance of a perception system relative to a specific task, e.g., trajectory evaluation for autonomous flying, solely based of the visual input; and use it to

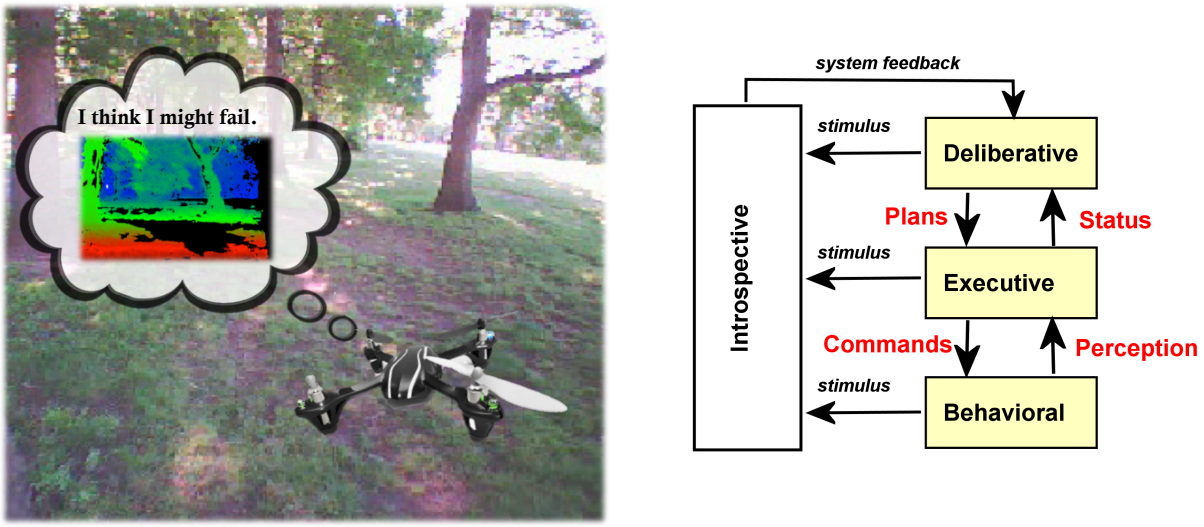


Figure 3.1: Robot Introspection: Having a self-evaluating facility to *know when it doesn't know* will augment a robot's ability to take reliable mission-critical decisions in ambiguous situations.

reliably predict whether the system is likely to fail on a given test instance.

In a similar spirit, Jammalamadaka *et al.* [45] recently introduced evaluator algorithms to predict failures for human pose estimators (HPE). They used features specific to this application; Welinder et al. [100] and Aghazadeh and Carslsson [1] predict the global performance of a classifier on a corpus of test instances by analyzing statistics of the training and test data. Another line of research involves the domain of 'rejection' - refusing to make a decision on an instance, is related to systems that identify an instance as belonging to none of the classes seen during training [97]. Zhang *et al.* [105] learn a SVM model to predict failures from visual input. Bansal *et al.* [4] learn a semantic characterization of failure modes. The biometrics community has been using image quality measures as predictors for matching performance [91]. In this context, the concept of input 'quality' is closely tied to the underlying perception system. Our work follows the same philosophy. Such methods, as do we, only analyze the input instance (e.g., fingerprint scan) to assess the likely quality of match. Our method is complementary to all the previous approaches: it uses generic image appearance features, provides an instance-specific reliability measure, and the scenario we consider is not only concerned with unfamiliar instances arising from discrepancies in the distributions of the training and test data but also addresses familiar but difficult instances. In addition, we show the effectiveness of such predictions in a mission-critical decision making scenario such as robot navigation. We argue that a similar level of self-evaluation should be part of all perception systems as they are involved in a variety of real-world applications.

3.2.2 Deep Introspection

We frame this as a multi-step supervised learning algorithm: A deep spatio-temporal convolutional net is trained to learn good invariant hidden latent representations. The corresponding hidden variables of data samples are then used as input to train a linear SVM [43], which generates

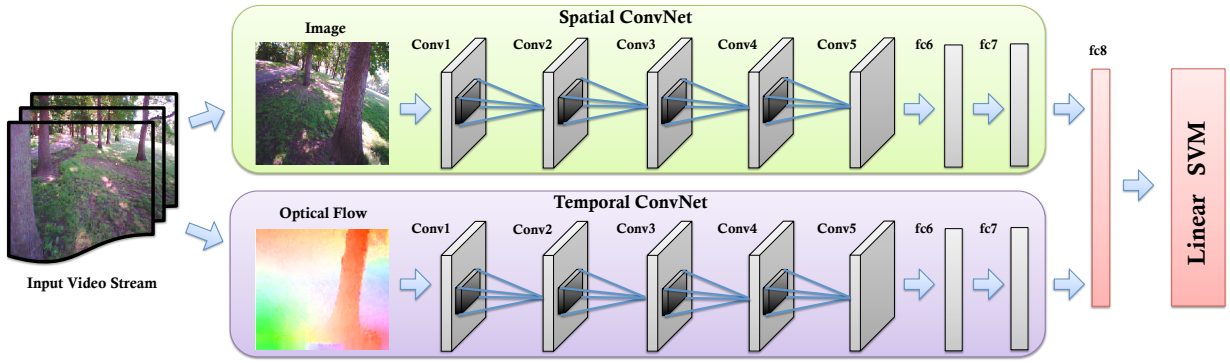


Figure 3.2: Spatio-temporal convolutional neural net architecture for deep introspective perception. The layer schematics are same as AlexNet [53]

a failure prediction score as output. SVMs in combination with CNN features are a widely used alternative to softmax for classification and regression tasks [92]. The output score is between 0 and 1 with higher scores indicating images for which the perception algorithm is predicted to be unreliable. The score can then be used by the rest of the system to decide whether to plan an action based on this input or to discard it and generate an alternate behavior.

Learning Spatio-Temporal ConvNets

In recent years, deep convolutional neural net (CNN) based models and features [53] have been proven to be more competitive than the traditional methods on solving complex learning problems in various vision tasks [75]. Now, while great progress have been made in the domain of single images for multimedia applications, the temporal component in the context of videos is generally more difficult to use. We argue that in robotic applications video data is readily available, and can provide an additional (and important) cue for any perception task.

We advocate that the use of this temporal component is crucial to introspective perception. The challenge, however, is to capture complementary information from still frames and motion between frames. To that end, we propose a two-stack CNN architecture as shown in Figure 3.2, which incorporates spatial and temporal networks. This architecture is reminiscent of the human visual system processes which uses the ventral pathway for processing the spatial information [54], such as shape and color, while the dorsal pathway is responsible for processing the motion information [46].

The layer configuration of our spatial and temporal ConvNet is similar to AlexNet [53]. Both the spatial and temporal stacks contain five feature extraction layers of convolution and max-pooling, followed by two fully connected layers. Dropout is applied to the two fully connected layers. The spatial stack operates on individual video frames, while the input to our temporal ConvNet stack is formed by trajectory stacking [98] of optical flow displacement fields between several consecutive frames. Simonyan *et al.* [86] have shown that such an input configuration, in the context of action recognition from videos, explicitly describes the motion between video frames. This makes the learning easier, as the network does not need to estimate motion implicitly. The outputs of the `fc7` layers from both the stacks are concatenated and fed to the `fc8`

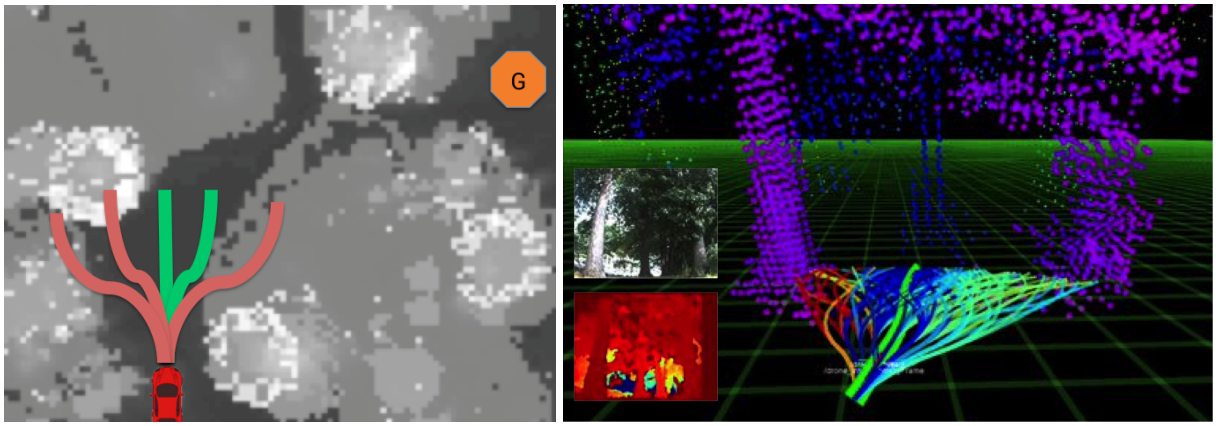


Figure 3.3: (a) Illustration of receding horizon control (b) Receding horizon control using depth images from stereo. Red trajectories indicate that they are more likely to be in collision. Note: Best seen in color.

layer, which builds a shared, high dimensional hidden representation.

The output of `fc8` layer goes to the softmax loss layer which estimates the deviation from the given target. This loss is then used to compute the gradients needed for back-propagation. During inference time, the loss layer is not used anymore and the output of `fc8`, represented as a d-dimensional feature vector: $x_i \in \mathcal{R}^d$, constitutes the deep features we seek. Responses from the higher-level layers of CNN have proven to be effective generic features with state-of-the-art performance on various image datasets [20, 75]. Thus, we use these responses as generic features for introspection.

3.2.3 Quantifying Introspection

In order to characterize the introspective capacity of a perception framework, a quantifiable measure is required. During training, we assume that we are given a dataset of N instances, $\mathcal{X} = \{x_1, ..., x_N\}$ and targets $\mathcal{Y} = \{y_1, ..., y_N\}$. The targets y_i are the task-dependent performance or accuracy score for the inputs x_i. This may be any metric that is representative of the failure probability of the system. For example, if the underlying perception system is a binary classifier, y_i can be the 0-1 loss. If the task is semantic segmentation, we can use proportion of correctly classified pixels as a measure of accuracy. SVM training then involves learning a set of parameters for regression task.

3.3 Introspection in Navigation

In this section we describe the introspective framework in the context of our autonomous flight through a dense, cluttered forest. This application is particularly interesting given the wide spectrum of challenges involved in fault tolerance control in perception for autonomous aerial systems navigating outdoor diverse environments based on a single sensor [107].

3.3.1 Learning the Introspection Model

To emulate the kind of introspective behavior we are looking for in navigation tasks, our proposed approach is to learn a introspection model to reliably predict when an image is going to give a poor estimate of the trajectory labels, using the generic learning approach described in Section 3.2.2. Hence, to train the introspection model for this application, the task-based performance score y_i is defined as the fraction of trajectories that are correctly predicted as collision-free or collision-prone, with respect to ground truth. Figure 3.3b illustrates an example of receding horizon control on an autonomous drone.

A pre-selected set of dynamically feasible trajectories of fixed length (the horizon), is evaluated on a cost map of the environment around the vehicle and the trajectory and the most optimal trajectory is selected. The challenge here, however, is to obtain ground-truth cost map, as it would require manual annotation of each frame over several hours of video. Rather, we use a self-supervised method to obtain ground truth by obtaining corresponding 3D map using stereo data as shown in Figure 3.3c. We collected a dataset - BIRD Dataset, as detailed in Section **??** for training and validation of our introspection model. During data collection flights, we run two pipelines in parallel: one using our described perception system, and one with the registered stereo depth images. Trajectory data from both pipelines are recorded simultaneously while flying the MAV, and we treat the one from stereo as the ground truth.

3.3.2 Emergency Maneuvers

During autonomous flight, inference is done on the learned model for each input frame to produce a failure probability score. If the score is above a given threshold (0.5 for all our experiments), the system is directed to execute a set of emergency maneuvers. Apart from preventing the drone from an inevitable crash, these emergency maneuvers need to additionally ensure that they help make the system more reliable by taking images under different conditions, than the ones under which the frame generating the failure was taken. Thus, the design of these maneuvers needs to take into account the failure modes specific to the system. A visual perception system, like ours, is strongly susceptible to sudden motions like pure rotation and image exposure. Thus, when *alerted*, a set of pure translational trajectories constrained in the image plane pointing in different directions to avoid exposure issues are executed for N seconds till the reliability is improved, after which normal deliberative planning is resumed.

3.4 Experiments and Results

In this section we analyze the performance of our proposed method for introspective robot perception. All the experiments were conducted in a densely cluttered forest area as shown in Fig 3.4, while restraining the drone through a light-weight tether. It is to be noted that the tether is only for compliance to federal regulations and does not limit the feasibility of a free flight.

Figure 3.4: BIRD dataset. Sample images depicting the diversity of data obtained over summer (Top) and winter (Bottom).

3.4.1 BIRD Dataset

A Bumblebee color stereo camera pair (1024×768 at 20 fps) is rigidly mounted with respect to the monocular camera, and registered using an accurate calibration technique [11]. We collected data at several locations with varying tree density, under varying illumination conditions and in both summer and winter conditions. Our corpus of imagery with stereo depth information consists of 60k images. The training set comprises of 50k images, while 10k images have been used for validation. The training/test split has been done so as preserve contiguous sequences as opposed to random sampling from the corpus of images. While the latter will lead to training data which is better represented by test data, we believe that the former is perhaps more realistic for a robotics application. The dataset will be made publicly available upon the acceptance of this paper.

We use the Caffe [47] toolbox and a NVIDIA Titan Z graphics card with 5760 parallel GPU cores and 12 gb of gpu ram for training our introspection model. The camera images are resized to 227×227 pixels for input to the spatial ConvNet. A multi-frame dense optical flow is computed for each frame using a total variational based approach [102], and fed to the temporal ConvNet. The convolutional layers (1-5) of the spatial network are pretrained on the ImageNet

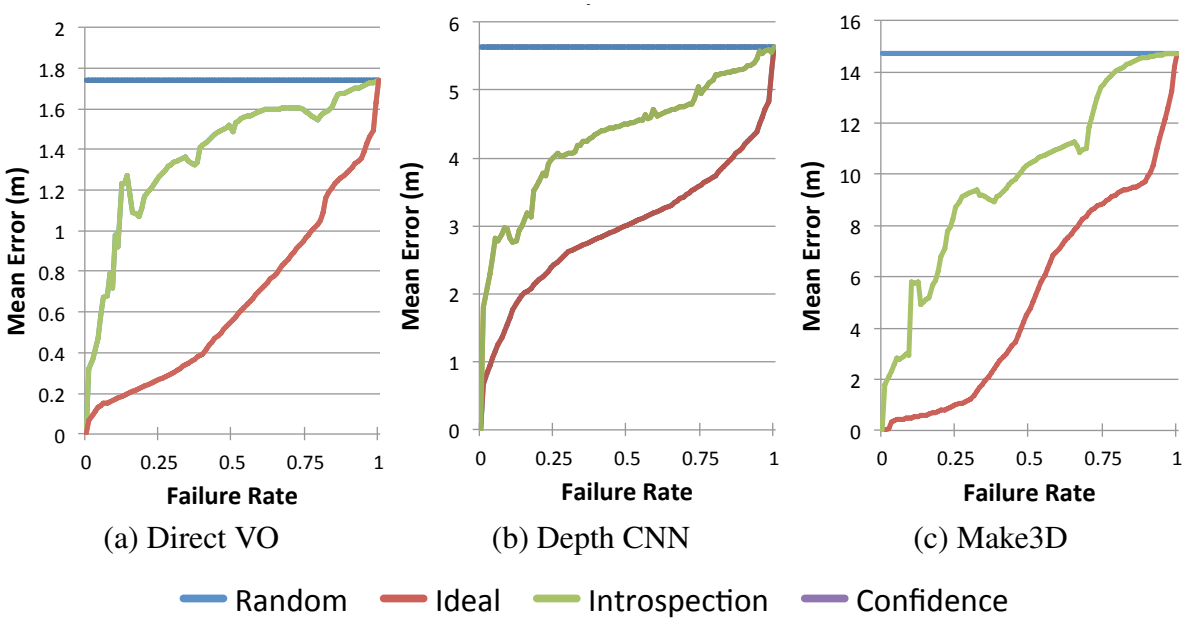


Figure 3.5: Error vs. Failure Rate curves for depth estimation using (a) Direct VO (b) Depth CNN and (c) Non-linear Regression

classification task [16] - while developing the model, we found pretraining on ImageNet worked better than initializing randomly. Unlike the spatial stream ConvNet, which can be pre-trained on a large still image classification dataset like ImageNet, the temporal ConvNet needs to be trained on video data - and the BIRD dataset is still rather small. To decrease over-fitting, we use ideas from multi-task learning to learn a (flow) representation by combining several datasets in a principled way [9]. We use two datasets from action recognition: UCF-101 [87] with 13K videos and HMDB-51 [55] with 6.8K videos to pre-train the temporal ConvNet using the multi-task learning formulation.

3.4.2 Quantitative Evaluation

Risk-Averse Metric: Quantitatively, we investigate the introspective capacity of the proposed model with its ability to trade-off the risk of making an incorrect decision with either taking remedial action or for the sake of this experiment, not making a decision at all. We believe such a metric is crucial when dealing with real applications and thus, present the Risk-Averse Metric (RAM). This metric was first proposed by Zhang *et al*. [105] and we adopt it to describes an Error vs Failure Rate (EFR) curve. Failure Rate is defined as the proportion of test images on which the perception system does not output a decision, in fear of providing an incorrect decision. This curve is computed by sorting the test images in descending order of their reliability as estimated by inference on our trained model. We then retain only a FR proportion of the test images (FR [0, 1]), and discard the rest. Since, the output of our perception system is a local 3D map, we compute the accuracy measure as mean reconstruction error with respect to stereo ground truth on these retained images and plot error vs. FR. For other applications like classification or

segmentation, it would be more natural to use accuracy instead of error. If introspection were perfect, it would discard the worst performing images. Error would be very low at low FR and then increase gracefully as FR approached 1. If introspection performed at chance level, on average, the accuracy would remain constant with varying FR.

Performance Benchmarking

We compare the performance of our method to these upper and lower bounds (Figure 3.5a). We see that even an approach as straightforward as introspection can perform significantly better than chance. As a baseline, we compare our approach to architectures with only Spatial ConvNet [53] and hand-crafted features [105]. The benefit of a deep hierarchical representation and temporal component is evident, as compared to only spatial approaches. Moreover, most vision systems often produce indicators of the confidence of their output e.g. energy of a conditional random field (CRF) for the task of semantic segmentation or the introspective capacity of a classifier [38]. Previous methods have used this confidence score as an estimate for failure probability. For our task, this is equivalent to classifying an image as failure if the mean per-pixel variance for depth estimates is higher than a given threshold. As baseline, we compare and empirically show an advantage over using such system-specific failure detection approach. We plot the results for RAM in Fig 3.5 along side that from our introspection model. While model-based confidence estimation can thought to be more reliable since it knows the inner workings of the underlying algorithm, it is interesting to note that a black-box approach, like ours, still performs better. This can be explained by the fact that analyzing specific error modes is not able to capture the entire spectrum of possible failures. For systems that can afford to evaluate output confidence, combining it with our approach would yield a better introspection model.

System-Agnostic Performance:

In order to show that our introspection model is generic and agnostic to the underlying perception system, we performed similar experiments with two other diverse approaches for depth map estimation - Non-linear Depth Regression [19] and Depth CNN [23] as can be see in Figure 3.5 (b-c). The results show that indeed the introspection model captures information orthogonal to the underlying perception system's beliefs. Moreover, our approach has an implicit computational advantage. It needs to run the underlying vision system to obtain the targets y_i only during training. During inference, we estimate the reliability of the system using the raw input image stream only. This makes the introspection model computationally better as any confidence based method would first need to run the base system to obtain its output. This also makes our system preemptive - allowing early rejection of possible failures. This is particularly desirable in real-world safety-critical applications where it is acceptable to have relatively lower accuracy and therefore deal with "ghost failures", but more expensive to miss a real failure.

3.4.3 Introspection Benefits to Autonomous Navigation

In this section, we study the benefits of our proposed introspection model to a downstream mission-critical application like autonomous navigation. Quantitatively, we evaluate the perfor-

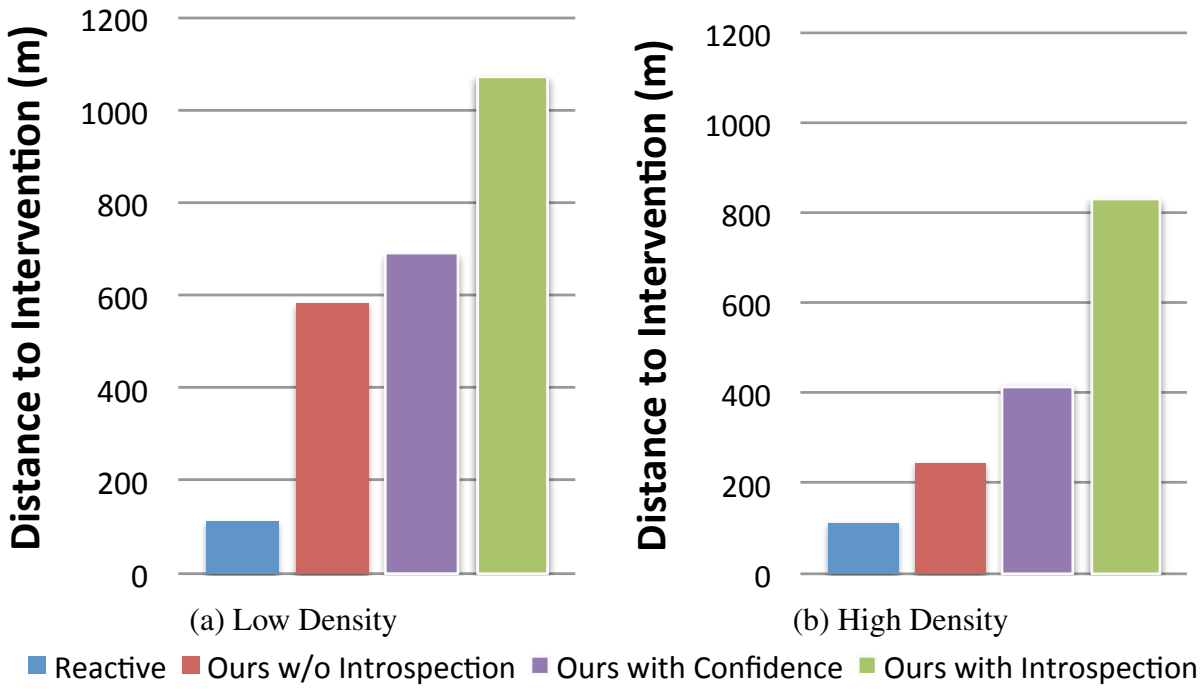


Figure 3.6: Average flight distance per intervention for (a) Low and (b) High density regions. For both, the corresponding introspection variant performs significantly better. We also plot the performance of the pure reactive approach from [77] as a baseline.

mance of the obstacle avoidance system by observing the average distance flown autonomously by the MAV over several runs (at 1.5 m/s), before an intervention. An intervention, in this context, is defined as the point at which the pilot needs to overwrite the commands generated by our control system so as to prevent the drone from an inevitable crash. Experiments were performed with and without using our proposed introspective perception approach. Results comparing to previous work on monocular reactive control by Ross *et al.* [77] as baseline has been shown in Figure 3.6. Tests were performed in regions of low and high clutter density (approx. 1 tree per $6 \times 6\ m^2$ and $12 \times 12\ m^2$, respectively). It can be observed that introspective navigation results in significantly better performance. In particular, the drone was able to fly autonomously without crashing over a 1000 m distance at average. The difference is higher in case of high-density regions where committing to an overconfident decision can be even more fatal.

3.4.4 Qualitative Evaluation

Further, we extend our evaluation to qualitatively assess our introspection model during flight tests. In Figure 3.7a, we show the flight path taken by the MAV during a test run. The points at which the introspection model *alerted* the system of a possible failure has been marked with an asterisk in blue. Now, neural networks are known to largely remain 'black-boxes' whose function is hard to understand. Additionally, since we do not have any ground-truth during system flight tests, it is very hard to semantically characterize predicted failures into failure modes. However,

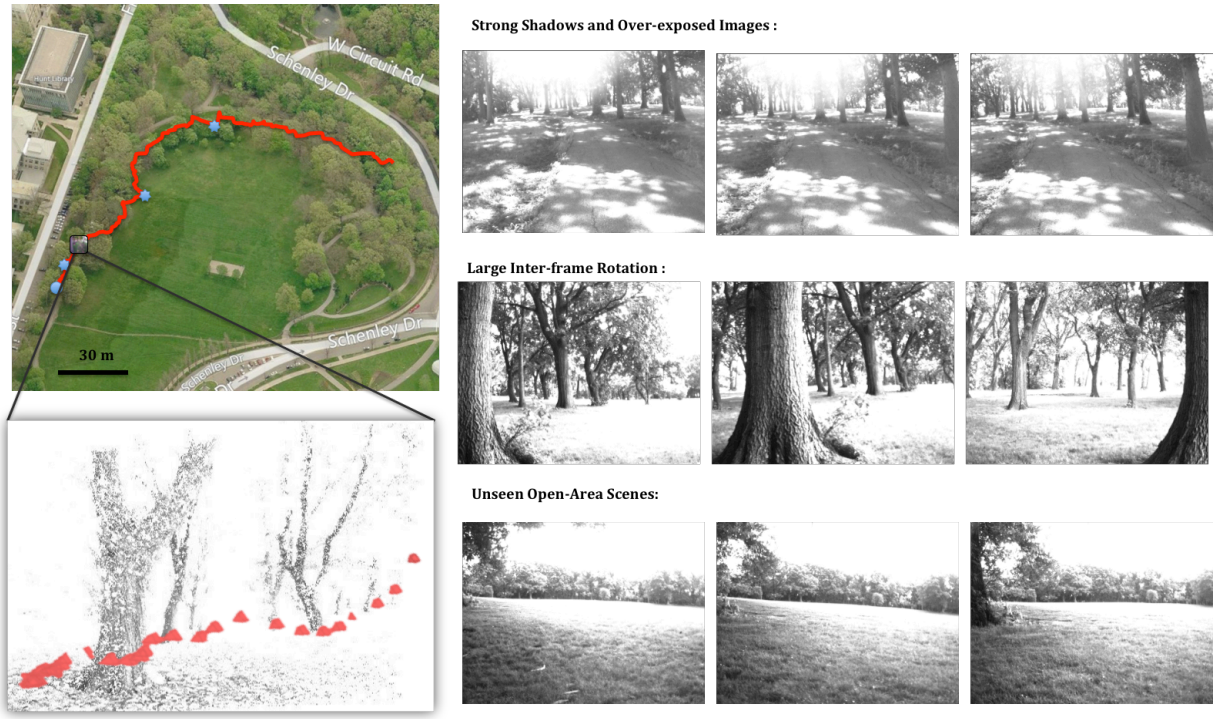


Figure 3.7: Qualitative results of our proposed introspective perception approach for autonomous navigation. (Top Left) Testing area near Carnegie Mellon University, Pittsburgh, USA. The flight path for one of the test runs has been overlayed on an aerial image of the test site. The instances where the system was alerted have been marked in blue asterisk. (b) A region of the path has been zoomed to show the quality of 3D map reconstructed and the corresponding flight path within the dense forest environment. (c) Row 1-3 illustrates the stream of images when system was alerted, along with the intuitive reasoning.

we try to analyze the predicted failures, with the aim to obtain some intuition towards the kind of failures our model is learning. Figure 3.7c shows the stream of images predicted to be unreliable. Each row shows a different scenario of failure that occurred during the run. The model correctly flagged images that are corrupted with strong shadows and over/under-exposure while retaining the images with good contrast (Row-1); the effectiveness of the temporal model is evident as the strong inter-frame rotations are flagged (Row-2); and finally, previously unseen scenarios like open grounds are again flagged as uncertain (Row-3). We only present the most interesting results here; See attached video for more qualitative results and observe introspective perception in action.

3.5 Summary

In this chapter, we introduced the concept of introspective perception - a generic framework for learning how to predict failures in vision systems. We have demonstrated our approach on a classical autonomous navigation task. However, our treatment and findings apply to any aspect

of robotics where action is required based on inference from sensor data. We present quantitative and qualitative results to support our claims that this can predict failure reliably, and can help improve the performance of a downstream application like autonomous navigation. Finally, with this work we hope to bridge the gap between vision and robotics communities by taking a step in the direction of building self-evaluating vision systems that fail gracefully, making them more usable in real-world robotics applications even with their existing imperfections.

Chapter 4

Adaptive Learning for MAV Control

Contents

4.1 Domain Adaptation in Robot Learning 43
4.2 Learning Transferable Policies 45
4.3 Experiments and Results 48
4.4 Summary 51

The ability to transfer knowledge gained in previous tasks into new contexts is one of the most important mechanisms of human learning. Despite this, adapting autonomous behavior to be reused in partially similar settings is still an open problem in current robotics research. In this chapter, we take a small step in this direction and propose a generic framework for learning transferable motion policies. Our goal is to solve a learning problem in a target domain by utilizing the training data in a different but related source domain. We present this in the context of an autonomous MAV flight using monocular reactive control, and demonstrate the efficacy of our proposed approach through extensive real-world flight experiments in outdoor cluttered environments.

4.1 Domain Adaptation in Robot Learning

As autonomous robots aspire for long-term autonomous operation in unstructured environments, learning based methods [59, 67, 74, 77] have become a very powerful alternative to designing hand-engineered perception and control software, even for basic tasks like collision avoidance. However, a major drawback with such data-driven approaches is that knowledge is usually built from scratch, and often involve complex data acquisition and training procedures. Seeking to reduce learning time in RL, many works have focused on transfer of knowledge between tasks. The determination of the knowledge to be transferred from one task to another has often been treated with techniques of transfer learning. Transfer learning allows knowledge to be achieved not only within tasks, but also across tasks. The goal is that the knowledge gained in source tasks helps in learning a new target task.

Furthermore, for autonomous navigation with mobile robots to be successful, it needs the ability to operate with partial information and in dynamic environments. These combine to produce what can be heavily stochastic observations and action outcomes. One solution is to try to produce as much domain knowledge as possible, typically at the cost of high resolution sensing and other resources. Producing such a high level of domain knowledge is useful to producing a large amount of detailed past experience, ideal in such model-based approaches. A more general approach, that requires much simpler modeling, is to provide or develop a direct mapping between sensory inputs to actions, avoiding the need for detailed domain knowledge.

In this work, we argue that for many robot tasks it is not even possible to obtain real training data for building reliable models. Even for tasks where training data can be obtained, the learned policies only apply to the physical system and environment the model was originally trained on, due to the limited variability of datasets. Moreover, in real-world applications we often encounter changes in dynamic conditions, such as weather and illumination, which change the characteristics of the domain. In all of the above scenarios, a good policy cannot be guaranteed if it is trained by using traditional learning techniques. Therefore, there is incentive in establishing techniques to reduce the labeling cost, typically by leveraging labeled data from relevant source domains such as off-the-shelf datasets or synthetic simulations.

Representation or feature learning provides a framework to reduce the dependency on manual feature engineering. Examples that can be considered as representation learning are Principal Component Analysis (PCA), Independent Component Analysis (ICA), Sparse Coding, Neural Net-works, and Deep Learning. In deep learning, the greedy layer-wise unsupervised training, which is known as the pretraining, has played an important role for the success of deep neural networks. Although representation learning-based techniques have brought some successes over many applications, methods to address the distribution mismatch have not yet been well studied.

Domain adaptation, a method to formally reduce the domain bias, has addressed this problem [3, 32, 73]. The domain discrepancy poses a major obstacle in adapting predictive models across domains. For example, an object recognition model trained on manually annotated images may not generalize well on testing images under substantial variations in the pose, occlusion, or illumination. Domainadaptation establishes knowledge transfer from the labeled source domain to the unlabeled target domain by exploring domain-invariant structures that bridge different domains of substantial distribution discrepancy. One of the main approaches to establishing knowledge transfer is to learn domain-invariant models from data, which can bridge the source and target domains in anisomorphic latent feature space.

However, to date there have been very few attempts to enhance the transferability of learned policies in the context of autonomous robotics. Even fewer have validated them experimentally through real-world experiments. In this chapter, we explore these ideas in the context of vision-based autonomous MAV flight in cluttered natural environments, and evaluate how a single policy learned from labeled data from source domain using domain adaptation methods could effectively enable and accelerate learning onto a new target domain.

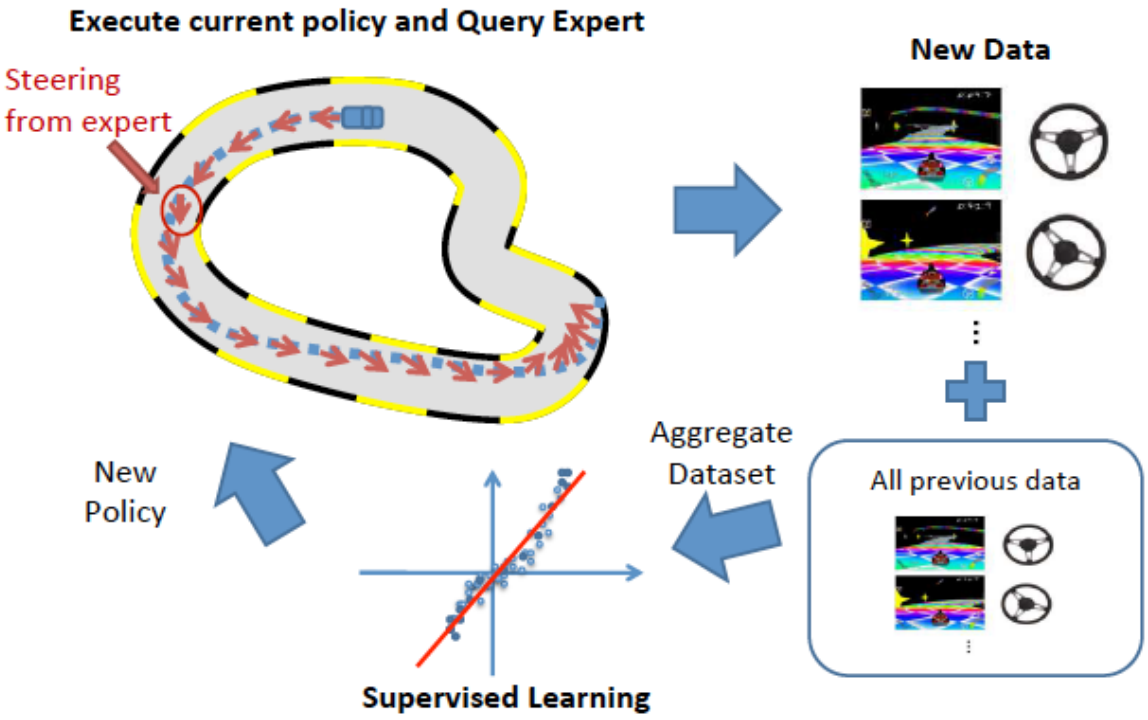


Figure 4.1: DAgger: A framework for Learnijng from Human Demonstrations.

4.2 Learning Transferable Policies

In this section, we describe our proposed approach for learning transferable policies for autonomous MAV flight. In previous work, we have proposed an imitation learning based technique [77] to directly learn a linear controller of drone's left-right velocity based on visual input.

4.2.1 Monocular Reactive Control using Imitation Learning

Imitation learning is concerned with the problem of learning to perform a control task(e.g. navigate safely and efficiently across a terrain from point A to point B) from demonstrations by an expert. These techniques allow programming autonomous behavior in robotic systems easily via demonstrations of the desired behavior, e.g. by human experts. This avoids time-consuming engineering required by traditional control methods that rely on accurate dynamic models of the system and well-engineered cost functions to synthesize controllers that produce the desired behavior. As the tasks performed by robotic systems continuously increase in complexity, it is becoming increasingly harder to provide good models and cost functions. As a result, learning by demonstration techniques are becoming more and more popular and have led to state-of-the-art performance in a number of recent applications, including, e.g. outdoor mobile robot navigation, legged locomotion, advanced manipulation, and electronic games. In Figure 4.1, we show the DAgger procedure proposed by [77] for imitation learning in a driving scenario.

In the same spirit, given a set of human pilot demonstrations in cluttered forest environments,

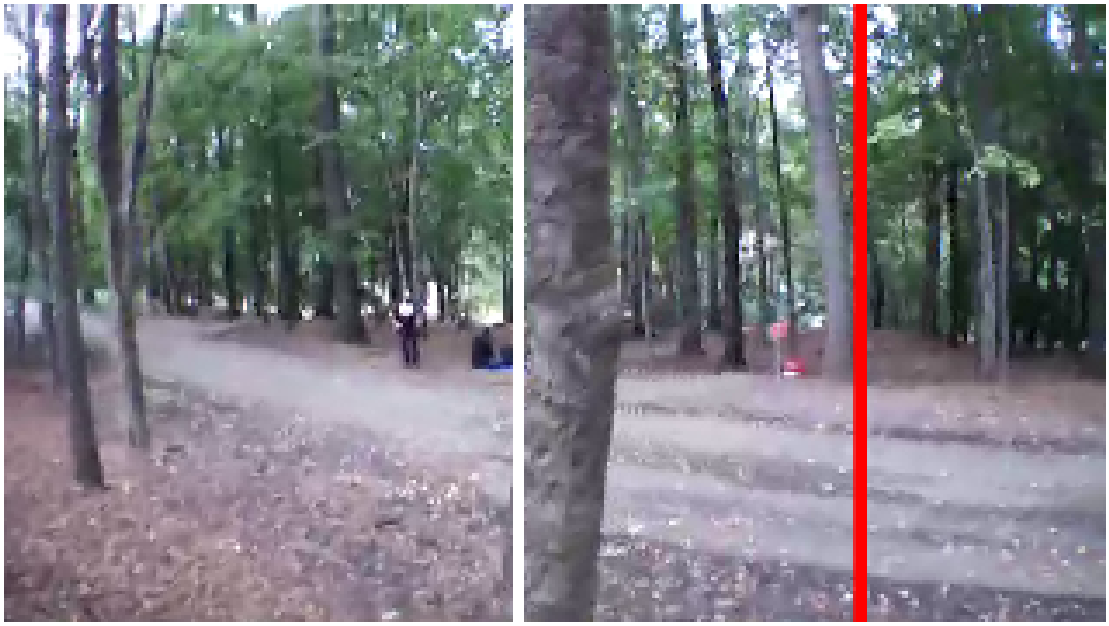


Figure 4.2: The interface for learning MAV control through human demonstrations. The pilot can provide virtual control commands using a joystick controller.

we train such a reactive policy that can avoid trees by adapting the MAV's heading as the drone moves forward. In this application, DAGGER is applied as follows. Initially, the pilot flies the quadrotor, via a joystick, through forest environments for several trajectories, to collect an initial dataset and train a first policy π_1. At following iterations, the drone is placed in various forest environments and flies autonomously using its current policy π_{n-1}. The pilot provides the correct controls he would perform in the situations encountered along the autonomous trajectories own by the drone, via the joystick. This allows the drone to collect data in new situations visited by the current policy, and learn the proper recovery behavior when these are encountered. The next policy π_{n+1} is obtained by training a policy on all the training data collected over all iterations (from iteration 1 to n). Figure 4.2 shows the DAGGER control interface used to provide correct actions to the drone. As mentioned, at iteration $n > 1$, the drone's current policy π_{n-1} is in control and the pilot just provides the correct controls for the scenes that the MAV visits.

Despite the great success with a relatively simple model, a major challenge is that DAGGER needs to collect data for all situations encountered by the current policy in later iterations. This would include situations where the drone crashes into obstacles if the current policy is not good enough. For safety reasons, we allow the pilot to take over or force an emergency landing to avoid crashes as much as possible. This implies that the training data used is not exactly what DAGGER would need, but instead a subset of training examples encountered by the current policy when it is within a "safe" region. This condition gets aggravated in cases where it is not possible to crash, for example in the case of a large scale autonomous helicopters. Thus it becomes a fundamental need for these policies be able to be transferred to new domains and across physical systems.

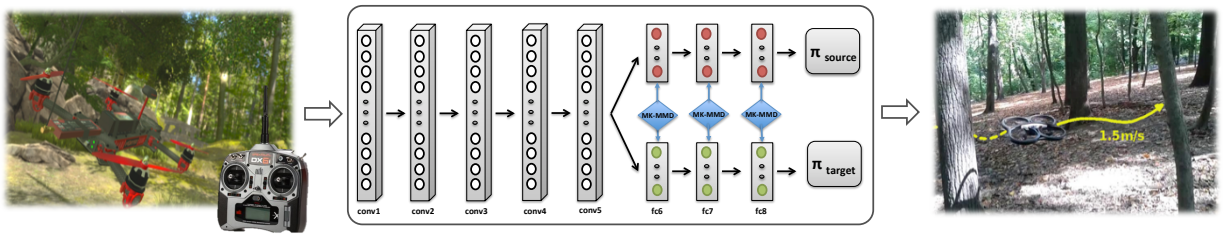


Figure 4.3: The framework for learning transferable policies from demonstrations in simulated source domain to real target domain using Deep Adaptation Network [63].

4.2.2 Domain Adaptive Neural Networks

In this work, we extend the above approach to learn domain-adaptive policies using labeled information from source domain and unlabeled information from target domain. Let the source domain $\mathcal{D}_s = \{(x_i, y_i)\}_{i=1}^{n_s}$ have n_s labeled examples drawn from a probability distribution p and target domain $\mathcal{D}_t = \{(x_j)\}_{j=1}^{n_t}$ have n_t unlabeled examples drawn from a probability distribution q. Then, the problem can be formulated as follows: Train a model to learn a set of features x that reduce the cross-domain discrepancy, and a policy $y = \pi_\theta(x)$, where y is the left-right velocity. Recently, deep convolutional neural network (CNN) based models and features [53] have been proven to be more competitive than the traditional methods on solving complex learning problems. While they have been shown to adapt to novel tasks [21], the main challenge here is that the target domain has no labeled information. Hence, directly adapting CNN to the target domain via fine-tuning is not possible. Thus, we build upon a recently proposed Deep Adaptation Network (DAN) architecture [63], which generalizes deep convolutional neural network to the domain adaptation scenario. The main idea is to enhance domain transferability in the task-specific layers of the deep neural network by explicitly minimizing the domain discrepancy.

We use a multiple kernel variant of the maximum mean discrepancies (MK-MMD) metric [36] as the measure of domain discrepancy. The MK-MMD of two distributions p and q is defined as the Reproducing Kernel Hilbert Space (RKHS) distance between mean embeddings of p and q:

$$d_k^2(p, q) = \|\mathrm{E}_p[\phi(x^s)] - \mathrm{E}_q[\phi(x^t)]\|_{\mathcal{H}_k}^2 \tag{4.1}$$

where ϕ is the characteristic kernel associated with the feature map. In order to minimize the domain discrepancy in the context of CNNs, we embed a MK-MMD based multi-layer adaptation regularizer (Eq. 4.1) to the CNN risk function:

$$\min_{\Theta} \frac{1}{n_s} \sum_{i=1}^{n_s} J(\theta(x_i^s), y_i^s) + \lambda \sum_{l=l_1}^{l_2} d_k^2(\mathcal{D}_s^l, \mathcal{D}_t^l) \tag{4.2}$$

where $\Theta = \{W^l, b^l\}_{l=1}^{l}$ denotes the set of all CNN parameters, $\lambda > 0$ is the penalty parameter, J is the cross-entropy loss function and $\theta(x_i^s)$ is the conditional probability that CNN assigns x_i^s to label y_i^s. $l_1(= 6)$ and $l_2(= 8)$ are the layer indices between which the regularizer is effective. The CNN architecture is based on AlexNet [53]. As the domain discrepancy increases in the higher layers [104], we fine-tune the convolutional layers of the CNN (`conv4-conv5`) using

source labeled examples and minimize the domain discrepamcy in the fully connected layers (`fc6-fc8`). The deep CNN is trained using a mini-batch supervised gradient descent (SGD) with the above optimization framework (Eq. 4.2).

4.3 Experiments and Results

In this section, we present experiments to analyze the performance of our proposed method of transferring policies for monocular reactive control of MAVs. All the experiments were conducted in a densely cluttered forest area using the commercially available ARDrone as our MAV platform. We use a distributed processing framework, where the image stream from the front facing camera is streamed to a base station over Wi-Fi at 15 `Hz`. The base station processes these images, and then sends back the desired control commands to the drone.

Methadology

Quantitatively, we evaluate the performance of our system by observing the average distance flown autonomously by the MAV over several runs (at 1.5 m/s), before a crash. Tests were performed in both regions of low and high tree density (approx. 1 tree per 6×6 m^2 and 3×3 m^2, respectively). For each scenario described below, training data with 1 `km` of human-piloted flight was collected in the source domain. Tests were then conducted on approximately 1 `km` of autonomous flights using policies learnt both with and without domain adaptation. As baseline, we compare our results to lower and upper bound: MAV flight using random policy and complete training data, respectively.

4.3.1 Performance Evaluation

Transfer across systems

In this experiment we try to answer the question: Can we transfer policies over different physical systems - from one configurations of sensors and dynamics to another? We collect training data using the ARDrone as the source domain, and test on a modified 3DR ArduCopter equipped with a high-dynamic range PointGrey Chameleon camera as the target domain (See Fig. 4.4a). The sensor system - global shutter vs rolling shutter, image resolution and camera intrinsics are different from that of the ARDrone. Hence, a policy learnt on one system cannot be expected to trivially generalize to the other.

Transfer across weather conditions

In this experiment we try to answer the question: Can we transfer policies over different weather conditions - from summer to winter? We collect training data during the summer season as our source domain and test during winter season as our target domain (See Fig. 3.4b). In this case the domain shift is induced by the difference in visual appearance. While the summer environment is cluttered with dense foliage, the winter conditions are often characterized with the presence of snow and absence of foliage.

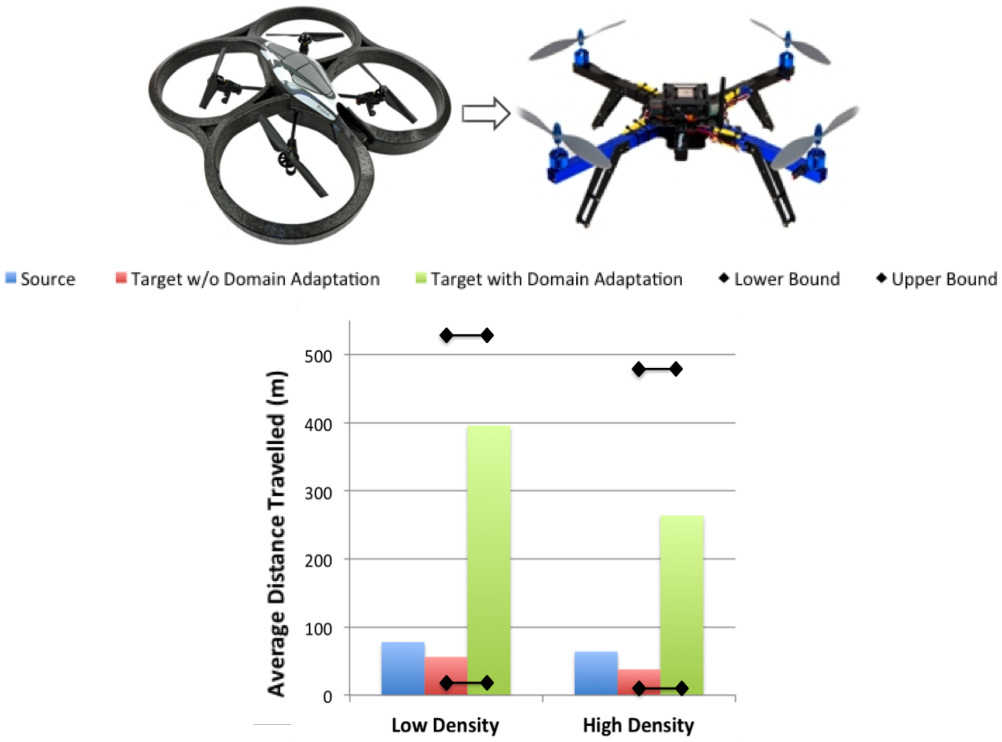


Figure 4.4: Experiments and Results for Transfer across physical systems from ARDrone to ArduCopter.

Transfer across environments

In this experiment we try to answer the question: Can we transfer policies over different environments - from one physical location to another? This is equivalent to using an off-the-shelf dataset as the source domain and testing in a seperate target domain. In particular, we use the ETH Zurich forest-trail dataset [34] as the source. The dataset provides a large-scale collection of images from a forest environment along with annotations for trail heading (left, center or right). Using these source labels, we train a policy for MAV reactive control and test at the forest environment near CMU as the target domain (See Fig. 3.4c). Here, the domain shift is induced by the implicit difference in physical location and nature of the task. Note: It is not possible to compare results to the source domain.

4.3.2 Experimental Insights

The main results obtained in this paper is that learning a transferable policy using the proposed approach can boost performance significantly in the target domain, as compared to simply re-using the learnt policy in new domains. Quantitatively, we show this through extensive outdoor flight experiments over a total distance of 6 km in environments of varying clutter density. Even without any training data, in the target domain, the MAV was able to successfully avoid more than 1900 trees, with an accuracy greater than 90%. We extend our evaluations to qualitatively assess

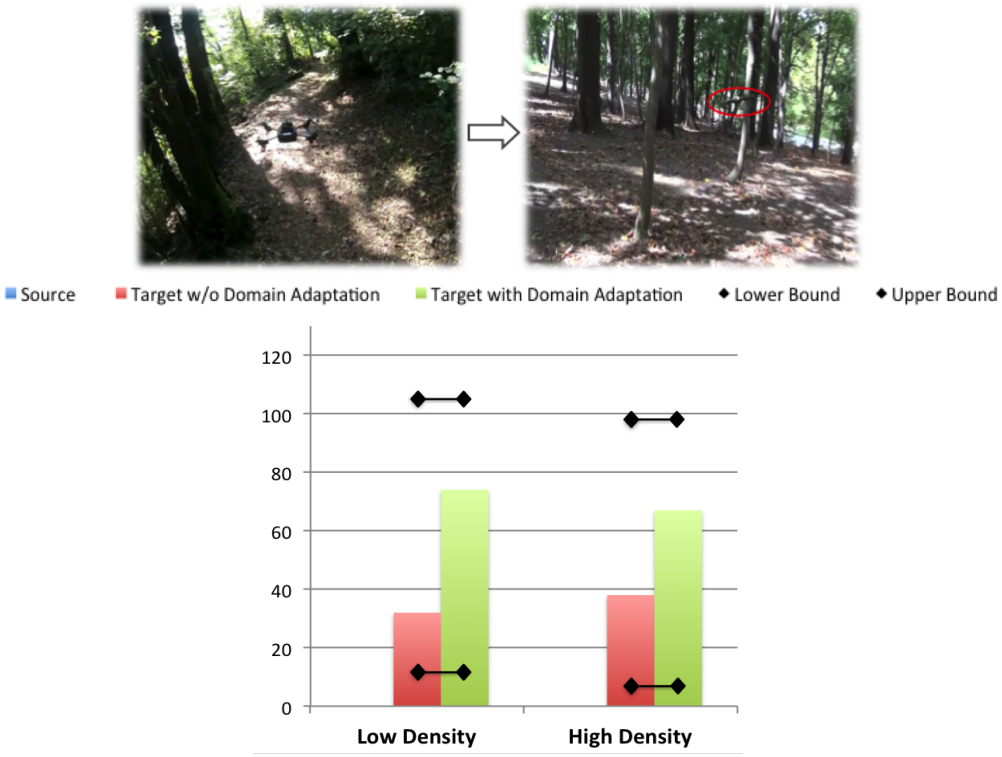


Figure 4.5: Experiments and Results for Transfer across weather conditions from summer to winter.

the learned policies during one of the runs from our flight test, as shown in Fig. 3.7. We show the nature of training data from summer, snapshots of predicted left-right velocity commands in the chronological order of the flight path taken by the MAV. Moreover, we also analyze the the policy learnt without domain adaptation by predicting control commands (offline) using the snapshot images as input. It can be observed that the domain adaptive policy performs better and is able to generalize to the new domain.

Furthermore, we observe that for the first two experiments the performance in the target domain is better than that in the source domain. For transfer across physical systems, this can be attributed to the underlying dynamics of the MAV. The ArduCopter has accurate and stable positioning system that allows it to be more resistive to strong winds, which is a major cause of crash for the less stable ARDrones. Moreover, the target domain has a better sensor suite. The increased resolution probably helped in detecting the thinner trees. For transfers across weather condition, we again observe a boost in performance in the target domain. Empirical analysis of the failure cases reveal that percentage of failures due to branches and leaves diminishes greatly in winter conditions resulting in better overall performance. In comparison to the above two experiments, the performance improves only slightly for transfer across environments. The reason for this is that the source labels are very coarse and were collected for a classification task (left, right or center). Hence, we learn that to improve the transferability of motion policies over physical systems, it is also important to explicitly address (1) The domain shift induced by discrepancy in dynamics, (2) The expected failure cases in the target domain, and (3) The

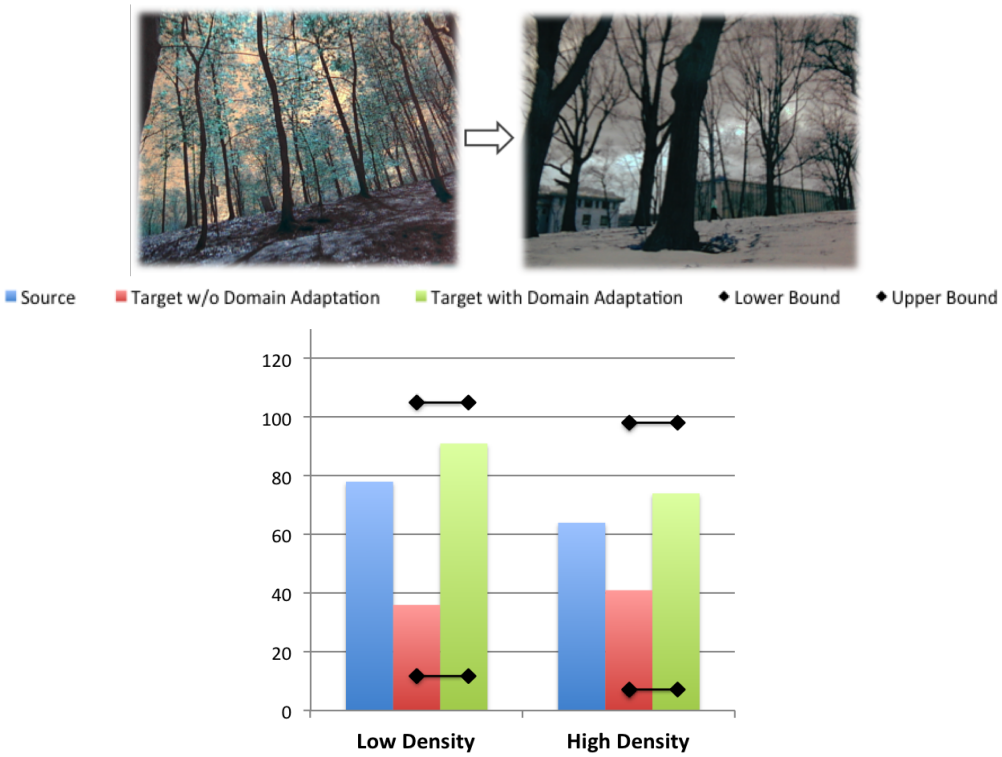


Figure 4.6: Experiments and Results for Transfer across environments from ETH Zurich to CMU.

discrepancy induced by not only the domain, but also the task.

4.3.3 Scheduled Experiments

The successful experimental results obtained so far motivate us to pursue more challenging goals. In scheduled experiments, we want to ask the question: Can we transfer policies from simulated to real environments? The idea is to use synthetic rendered environments such as drone simulators to train models that are effective for flying MAVs in the real world. Through such experimental analysis, we hope to gain better insights on design principles for domain adaptation techniques.

4.4 Summary

In this chapter, we have presented a generic framework for learning transferable motion policies. Our system learns to predict how a human pilot would control a MAV in a source domain, and is able to successfully transfer that behaviour to a new target domain. Quantitatively, we show significant boost in performance over simply re-using policies without any explicit transfer, through extensive real-world experiments. We have demonstrated our approach on an autonomous MAV navigation task using monocular reactive control. However, our treatment and findings apply

to any aspect of experimental robotics where a system needs to be trained for end-to-end autonomous behaviour based on sensor data.

Chapter 5

Conclusion

Contents

5.1 Summary 53

5.2 Future Work 54

In this thesis, we have presented a scalable framework for visual navigation of micro aerial vehicles in the wild. The presented algorithms facilitate robust, long-range autonomous flight in cluttered urban outdoor environments based on a monocular camera as the main exteroceptive sensor of the MAV. To conclude, we summarize our main contributions and give directions for possible future research.

5.1 Summary

Accurate, Robust and Computationally feasible Visual Navigation

Our first contribution is a toolbox of approaches for perception, planning and control of agile, low-cost MAVs navigation in cluttered urban and natural outdoor environments, based on a monocular camera as the main exteroceptive sensor. First, we present an end-to-end system consisting of a semi-dense direct visual odometry based depth estimation approach that is capable of producing accurate depth maps at 15 Hz, even while the MAV is flying forward; is able to track frames even with strong inter-frame rotations using a novel method for visual-inertial fusion on Lie-manifolds, and can robustly deal with uncertainty by making multiple, relevant yet diverse predictions. Secondly, we introduce a novel wind-resistant LQR control based on a learned system model for stable flights in varying wind conditions. Finally, we evaluate our system through extensive experiments and flight tests of more than 2 km in dense forest environments, and show the efficacy of our proposed perception and control modules as compared to the state-of-the-art methods.

Safe, Reliable and Verifiable Autonomy

Our second contribution is a generic framework for introspection in autonomous MAVs to enable them with the ability to assess how qualified they are at any given moment to make a decision. Using a novel spatio-temporal Convolutional Neural Network, trained using multi-task learning, we present a failure prediction and recovery strategy for perception systems. We demonstrate through extensive experiments that we are reliably able to predict failures reliably with high recall, and better than confidence based systems. This helps us build safe, reliable robot behaviors that can carry out emergency verifiable behaviors when uncertain.

Scalable, Adaptive Learning of Policies

Our third contribution is a model for knowledge transfer in autonomous robots. In order for robots to have long-term autonomy and scale to dynamic environments, it is necessary to be able to transfer knowledge across tasks and domains. In this work, we present a technique to learn domain-adaptive policies using deep neural networks and imitation learning. We demonstrate the effective learning of policies for a target domain by using only labeled data from the source domain. We study the transfer of policies in the context of MAV control policies across physical systems, dynamic environments and also weather conditions and show through real world experiments that we are able to successfully transfer policies from one domain to another in an unsupervised fashion.

5.2 Future Work

In this thesis, we have presented a variety of approaches to overcome problems of the current state-of-the-art in visual navigation, learning, and control. However, the challenging task of vision-based MAV navigation is by far not solved yet. In the following, we thus conclude our work with suggestions for future research directions.

Dealing with Illumination Changes

Outdoor environments are subject to severe changes in illumination and appearance, spanning from the regular variation in solar illumination during a day to weather-dependent changes, and further to seasonal variations. While navigation approaches should ideally be invariant to those changes in appearance, the underlying algorithms are only to a certain degree. The major problem is the correspondence computation between images, which is already a challenge when images of the exactly same scene are taken at a sunny day and at a cloudy day. In Section 4.3 we have proposed to domain-adaptive models to deal with such dynamic changes in the context of MAV control. While this approach works well, incorporating this feature in the context of vision based localization and mapping needs to be researched.

Active Perception

For successful visual navigation in dynamic, real-world outdoor environments, the contributions discussed before have been useful. However, navigation also involves a control component, and thus it is possible to define where a robot should ideally go to optimize its perception quality. We propose to that to enable fully autonomous behavior in robots, active perception is key and needs to be explored better in the future.

Bibliography

[1] Omid Aghazadeh and Stefan Carlsson. Properties of datasets predict the performance of classifiers. In *British Machine Vision Conference*, pages 22–31, 2013. 3.2.1

[2] Nguyen Bach, Fei Huang, and Yaser Al-Onaizan. Goodness: A method for measuring machine translation confidence. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 211–219. Association for Computational Linguistics, 2011. 3.1.2

[3] Mahsa Baktashmotlagh, Mehrtash Harandi, Brian Lovell, and Mathieu Salzmann. Unsupervised domain adaptation by domain invariant projection. In *Proceedings of the IEEE International Conference on Computer Vision*, 2013. 4.1

[4] Aayush Bansal, Ali Farhadi, and Devi Parikh. Towards transparent systems: Semantic characterization of failure modes. In *Computer Vision–ECCV 2014*, pages 366–381. Springer, 2014. 3.2.1

[5] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012. 2.2.1

[6] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010. 2.2.1

[7] Gregg Buskey, Jonathan M Roberts, Peter Corke, and Gordon Wyeth. Helicopter automation using a low-cost sensing system. In *Proceedings of the Australasian Conference on Robotics and Automation, 2003*. Australian Robotics and Automation Association Inc, 2003. 1.3

[8] Adam Coates, Pieter Abbeel, and Andrew Y Ng. Learning for control from multiple demonstrations. In *Proceedings of the 25th international conference on Machine learning*, pages 144–151. ACM, 2008. 1.3

[9] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008. 3.4.1

[10] R Craig Coulter. Implementation of the pure pursuit path tracking algorithm. Technical report, DTIC Document, 1992. 2.4.1

[11] Shreyansh Daftry, Michael Maurer, Andreas Wendel, and Horst Bischof. Flexible and

user-centric camera calibration using planar fiducial markers. In *British Machine Vision Conference (BMVC)*, 2013. 2.5.1, 3.4.1

[12] Shreyansh Daftry, Debadeepta Dey, Harsimrat Sandhawalia, Sam Zeng, J Andrew Bagnell, and Martial Hebert. Semi-dense visual odometry for monocular navigation in cluttered environment. In *IEEE International Conference on Robotics and Vision (ICRA) Workshops*, 2015. 2.2.2

[13] Shreyansh Daftry, Christof Hoppe, and Horst Bischof. Building with drones: Accurate 3d facade reconstruction using mavs. In *IEEE International Conference on Robotics and Vision (ICRA)*, 2015. 2.2.2

[14] Shreyansh Daftry, Sam Zeng, Arbaaz Khan, Debadeepta Dey, Narek Melik-Barkhudarov, J Andrew Bagnell, and Martial Hebert. Robust monocular flight in cluttered outdoor environments. *arXiv preprint arXiv:1604.04779*, 2016. 2.4.1

[15] Sarah Jane Delany, Pádraig Cunningham, Dónal Doyle, and Anton Zamolotskikh. Generating estimates of classification confidence for a case-based spam filter. In *Case-Based Reasoning Research and Development*, pages 177–190. Springer, 2005. 3.1.2

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. 3.4.1

[17] Xinyan Deng, Luca Schenato, and Shankar S Sastry. Model identification and attitude control for a micromechanical flying insect including thorax and sensor models. In *IEEE International Conference on Robotics and Vision (ICRA)*, 2003. 2.4.3

[18] Debadeepta Dey, Tian Yu Liu, Martial Hebert, and J Andrew Bagnell. Contextual sequence prediction with application to control library optimization. In *Robotics, Science and Systems (RSS)*, 2013. 2.3

[19] Debadeepta Dey, Kumar Shaurya Shankar, Sam Zeng, Rupesh Mehta, M Talha Agcayazi, Christopher Eriksen, Shreyansh Daftry, Martial Hebert, and J Andrew Bagnell. Vision and learning for deliberative monocular cluttered flight. In *In Proceedings of the International Conference on Field and Service Robotics (FSR)*, 2015. 2.4.1, 2.5.1, 2.5.3, 3.4.2

[20] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013. 3.2.2

[21] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Proceedings of The 31st International Conference on Machine Learning*, 2014. 4.2.2

[22] Robert PW Duin and David MJ Tax. Classifier conditional posterior probabilities. In *Advances in pattern recognition*, pages 611–619. Springer, 1998. 3.1.2

[23] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems*, pages 2366–2374, 2014. 2.2.1, 2.5.1, 3.4.2

[24] Jakob Engel, Jurgen Sturm, and Daniel Cremers. Semi-dense visual odometry for a monocular camera. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1449–1456. IEEE, 2013. 2.2.2

[25] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *Computer Vision–ECCV 2014*, pages 834–849. Springer, 2014. 2.2.2, 2.2.2

[26] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge 2012 (voc2012) results, 2012. 3.1.1

[27] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In *Robotics: Science and Systems (RSS)*, 2015. 2.2.3

[28] David Ford Fouhey, Abhinav Gupta, and Martial Hebert. Unfolding an indoor origami world. In *Computer Vision–ECCV 2014*, pages 687–702. Springer, 2014. 2.2.1

[29] Tomonari Furukawa, Frederic Bourgault, Benjamin Lavis, and Hugh F Durrant-Whyte. Recursive bayesian search-and-tracking using coordinated uavs for lost targets. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2521–2526. IEEE, 2006. 1.3

[30] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012. 3.1.1

[31] Christopher Geyer, Todd Templeton, Marci Meingast, and S Shankar Sastry. The recursive multi-frame planar parallax algorithm. In *3D Data Processing, Visualization, and Transmission, Third International Symposium on*, pages 17–24. IEEE, 2006. 2.2.2

[32] Muhammad Ghifary, W Bastiaan Kleijn, and Mengjie Zhang. Domain adaptive neural networks for object recognition. In *PRICAI 2014: Trends in Artificial Intelligence*. 2014. 4.1

[33] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014. 2.2.1

[34] Alessandro Giusti, Jerome Guzzi, Dan Ciresan, Fang-Lin He, Juan Pablo Rodriguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jurgen Schmidhuber, Gianni Di Caro, et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 2016. 4.3.1

[35] C. Green and A. Kelly. Optimal sampling in the space of paths: Preliminary results. Technical Report CMU-RI-TR-06-51, Robotics Institute, Pittsburgh, PA, November 2006. 2.4.1

[36] Arthur Gretton, Dino Sejdinovic, Heiko Strathmann, Sivaraman Balakrishnan, Massimiliano Pontil, Kenji Fukumizu, and Bharath K Sriperumbudur. Optimal kernel choice for large-scale two-sample tests. In *Advances in neural information processing systems*, 2012. 4.2.2

[37] Hugo Grimmett, Rimi Paul, Rudolph Triebel, and Ingmar Posner. Knowing when we don't

know: Introspective classification for mission-critical decision making. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4531–4538. IEEE, 2013. 3.1.2

[38] Hugo Grimmett, Rudolph Triebel, Rohan Paul, and Ingmar Posner. Introspective classification for robot perception. *The International Journal of Robotics Research*, page 0278364915587924, 2015. 3.1.2, 3.4.2

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*, 2015. 2.2.1

[40] Ruijie He, Sam Prentice, and Nicholas Roy. Planning in information space for a quadrotor helicopter in a gps-denied environment. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1814–1820. IEEE, 2008. 1.3

[41] Alex Holub, Pietro Perona, and Michael C Burl. Entropy-based active learning for object recognition. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*, pages 1–8. IEEE, 2008. 3.1.2

[42] Stefan Hrabar and Gaurav Sukhatme. Vision-based navigation through urban canyons. *Journal of Field Robotics*, 26(5):431–452, 2009. 1.3

[43] Fu Jie Huang and Yann LeCun. Large-scale learning with svm and convolutional for generic object categorization. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 284–291. IEEE, 2006. 3.2.2

[44] Ammar Husain, Heather Jones, Balajee Kannan, Uland Wong, Tiago Pimentel, Sarah Tang, Shreyansh Daftry, Steven Huber, and William L Whittaker. Mapping planetary caves with an autonomous, heterogeneous robot team. In *Aerospace Conference, 2013 IEEE*, pages 1–13. IEEE, 2013. 1.3

[45] Nataraj Jammalamadaka, Andrew Zisserman, Marcin Eichner, Vittorio Ferrari, and CV Jawahar. Has my algorithm succeeded? an evaluator for human pose estimators. In *Computer Vision–ECCV 2012*, pages 114–128. Springer, 2012. 3.2.1

[46] I Hong Jhuo, Guangnan Ye, Shenghua Gao, Dong Liu, Yu Gang Jiang, DT Lee, and Shih Fu Chang. Discovering joint audio–visual codewords for video event detection. *Machine vision and applications*, 25(1):33–47, 2014. 3.2.2

[47] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross B. Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *CoRR*, abs/1408.5093, 2014. URL http://arxiv.org/abs/1408.5093. 2.2.1, 2.2.1, 3.4.1

[48] Eagle S Jones and Stefano Soatto. Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *The International Journal of Robotics Research*, 30 (4):407–430, 2011. 2.2.3

[49] Ashish Kapoor, Kristen Grauman, Raquel Urtasun, and Trevor Darrell. Gaussian processes for object categorization. *International Journal of Computer Vision*, 88(2):169–188, 2010. 3.1.2

[50] Sertac Karaman and Emilio Frazzoli. High-speed flight in an ergodic forest. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2899–2906. IEEE, 2012. 1.2

[51] Alonzo Kelly, Anthony Stentz, Omead Amidi, Mike Bode, David Bradley, Antonio Diaz-Calderon, Mike Happold, Herman Herman, Robert Mandelbaum, Tom Pilarski, et al. Toward reliable off road autonomous vehicles operating in challenging environments. *The International Journal of Robotics Research*, 25(5-6):449–483, 2006. 2.4.1

[52] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007. 2.4.2

[53] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012. 2.2.1, 2.2.1, 2.3, 3.2, 3.2.2, 3.4.2, 4.2.2, 4.2.2

[54] Norbert Kruger, Peter Janssen, Sinan Kalkan, Markus Lappe, Ale Leonardis, Justus Piater, Antonio Jose Rodriguez-Sanchez, and Laurenz Wiskott. Deep hierarchies in the primate visual cortex: What can we learn for computer vision? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1847–1871, 2013. 3.2.2

[55] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2556–2563. IEEE, 2011. 3.4.1

[56] Matjaz Kukar. Estimating confidence values of individual predictions by their typicalness and reliability. In *ECAI*, volume 16, page 1045, 2004. 3.1.2

[57] Jack Langelaan and Steve Rock. Towards autonomous uav flight in forests. In *Proc. of AIAA Guidance, Navigation and Control Conference*, 2005. 1.2

[58] David Lentink and Andrew A Biewener. Nature-inspired flightbeyond the leap. *Bioinspiration & biomimetics*, 5(4):040201, 2010. 1.2

[59] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *arXiv preprint arXiv:1504.00702*, 2015. 4.1

[60] Lihong Li, Michael L Littman, Thomas J Walsh, and Alexander L Strehl. Knows what it knows: a framework for self-aware learning. *Machine learning*, 82(3):399–443, 2011. 3.1.2

[61] Christopher H Lin, Andrey Kolobov, Ece Kamar, and Eric Horvitz. Metareasoning for planning under uncertainty. *arXiv preprint arXiv:1505.00399*, 2015. 3.1.2

[62] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *arXiv preprint arXiv:1411.4038*, 2014. 2.2.1

[63] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015. 4.3, 4.2.2

[64] Todd Lupton and Salah Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *Robotics, IEEE Transactions on*, 28

(1):61–76, 2012. 2.2.3

[65] Masayoshi Matsuoka, Alan Chen, Surya PN Singh, Adam Coates, Andrew Y Ng, and Sebastian Thrun. Autonomous helicopter tracking and localization using a self-surveying camera array. *The International Journal of Robotics Research*, 26(2):205–215, 2007. 1.3

[66] Luis Mejias, Srikanth Saripalli, Pascual Campoy, and Gaurav S Sukhatme. Visual servoing of an autonomous helicopter in urban areas using feature tracking. *Journal of Field Robotics*, 23(3-4):185–199, 2006. 1.3

[67] Jeff Michels, Ashutosh Saxena, and Andrew Y Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, 2005. 4.1

[68] Aaron Christopher Morris. *Robotic introspection for exploration and mapping of subterranean environments*. ProQuest, 2007. 3.1.2

[69] Michael Muhlbaier, Apostolos Topalis, and Robi Polikar. Ensemble confidence estimates posterior probability. In *Multiple Classifier Systems*, pages 326–335. Springer, 2005. 3.1.2

[70] Urs Muller, Jan Ben, Eric Cosatto, Beat Flepp, and Yann L Cun. Off-road obstacle avoidance through end-to-end learning. In *Advances in neural information processing systems*, pages 739–746, 2005. 2.2.1

[71] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011. 2.4.2

[72] NVIDIA. http://www.nvidia.com/gtx-700-graphics-cards/gtx-titan-z, 2014. 2.2.1

[73] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *Neural Networks, IEEE Transactions on*, 22(2):199–210, 2011. 4.1

[74] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. Technical report, DTIC Document, 1989. 2.2.1, 4.1

[75] Ali S Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, pages 512–519. IEEE, 2014. 3.2.2

[76] Martin Riedmiller. Advanced supervised learning in multi-layer perceptronsfrom backpropagation to adaptive learning algorithms. *Computer Standards & Interfaces*, 16(3): 265–278, 1994. 2.2.1

[77] Stéphane Ross, Narek Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J Andrew Bagnell, and Martial Hebert. Learning monocular reactive uav control in cluttered natural environments. In *Robotics and Automation (ICRA), International Conference on*, 2013. 2.5.3, 3.6, 3.4.3, 4.1, 4.2, 4.2.1

[78] M. Rumpler, S. Daftry, A. Tscharf, R. Prettenthaler, C. Hoppe, G. Mayer, and H. Bischof. Automated end-to-end workflow for precise and geo-accurate reconstructions using fidu-

cial markers. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-3:135–142, 2014. 2.2.2

[79] Markus Rumpler, Alexander Tscharf, Christian Mostegel, Shreyansh Daftry, Christof Hoppe, Rudolf Prettenthaler, Friedrich Fraundorfer, Gerhard Mayer, and Horst Bischof. Evaluations on multi-scale camera networks for precise and geo-accurate reconstructions from aerial and terrestrial images with user guidance. *Computer Vision and Image Understanding*, 2016. 2.2.2

[80] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. Learning depth from single monocular images. In *Advances in Neural Information Processing Systems*, pages 1161–1168, 2005. 2.3

[81] 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. 1.3

[82] Majura F Selekwa, Damion D Dunlap, Dongqing Shi, and Emmanuel G Collins. Robot navigation in very cluttered environments by preference-based fuzzy behaviors. *Robotics and Autonomous Systems*, 56(3):231–246, 2008. 1.2

[83] Shaojie Shen, Yash Mulgaonkar, Nathan Michael, and Vijay Kumar. Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor. In *Robotics: Science and Systems*. Citeseer, 2013. 2.2.3

[84] David Shim, Hoam Chung, H Jin Kim, and Shankar Sastry. Autonomous exploration in unknown urban environments for unmanned aerial vehicles. In *Proc. AIAA GN&C Conference*, 2005. 1.2

[85] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 2.2.1

[86] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014. 3.2.2

[87] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 3.4.1

[88] Military Specification. Flying qualities of piloted airplanes. *United States Department of Defense*, 1980. 2.4.3

[89] Jan Stühmer, Stefan Gumhold, and Daniel Cremers. Real-time dense geometry from a handheld camera. In *Pattern Recognition*, pages 11–20. Springer, 2010. 2.2.2

[90] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014. 2.2.1

[91] Elham Tabassi and Patrick Grother. *Fingerprint image quality*. Springer, 2009. 3.2.1

[92] Yichuan Tang. Deep learning using linear support vector machines. *arXiv preprint arXiv:1306.0239*, 2013. 3.2.2

[93] Todd Templeton. Accurate real-time reconstruction of distant scenes using computer vision: The recursive multi-frame planar parallax algorithm. 2009. 2.2.2

[94] Todd Templeton, David Hyunchul Shim, Christopher Geyer, and S Shankar Sastry. Autonomous vision-based landing and terrain mapping using an mpc-controlled unmanned rotorcraft. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1349–1356. IEEE, 2007. 1.3

[95] Carlo Tomasi and Takeo Kanade. *Detection and tracking of point features*. School of Computer Science, Carnegie Mellon Univ., 1991. 2.4.2

[96] Rudolph Triebel, Hugo Grimmett, Rohan Paul, and Ingmar Posner. Driven learning for driving: How introspection improves semantic mapping. In *Proc of Intern. Symposium on Robotics Research (ISRR)*, 2013. 3.1.2

[97] SA UEE and ZAE Weiergewan. Confidence estimation in classification decision: A method for detecting unseen patterns. 2007. 3.2.1

[98] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3169–3176. IEEE, 2011. 3.2.2

[99] Stephan Weiss, Markus W Achtelik, Simon Lynen, Margarita Chli, and Roland Siegwart. Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 957–964. IEEE, 2012. 2.2.3

[100] Peter Welinder, Max Welling, and Pietro Perona. A lazy man's approach to benchmarking: Semisupervised classifier evaluation and recalibration. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3262–3269. IEEE, 2013. 3.2.1

[101] Andreas Wendel, Michael Maurer, Gottfried Graber, Thomas Pock, and Horst Bischof. Dense reconstruction on-the-fly. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1450–1457. IEEE, 2012. 2.2.2

[102] Manuel Werlberger, Thomas Pock, and Horst Bischof. Motion estimation with non-local total variation regularization. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2464–2471. IEEE, 2010. 3.4.1

[103] Kwangjin Yang and Salah Sukkarieh. 3d smooth path planning for a uav in cluttered natural environments. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 794–800. IEEE, 2008. 1.2

[104] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, 2014. 4.2.2

[105] Peng Zhang, Jiuling Wang, Alireza Farhadi, Martial Hebert, and Devi Parikh. Predicting failures of vision systems. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3566–3573. IEEE, 2014. 3.2.1, 3.4.2, 3.4.2

[106] Weiyu Zhang, Stella X Yu, and Shang-Hua Teng. Power svm: Generalization with exemplar classification uncertainty. In *Computer Vision and Pattern Recognition (CVPR), 2012*

IEEE Conference on, pages 2144–2151. IEEE, 2012. 3.1.2

[107] YM Zhang, A Chamseddine, CA Rabbath, BW Gordon, C-Y Su, S Rakheja, C Fulford, J Apkarian, and P Gosselin. Development of advanced fdd and ftc techniques with application to an unmanned quadrotor helicopter testbed. *Journal of the Franklin Institute*, 350 (9):2396–2422, 2013. 3.3

[108] Kemin Zhou, John Comstock Doyle, Keith Glover, et al. *Robust and optimal control*, volume 40. Prentice hall New Jersey, 1996. 2.4.3