Enabling Autonomous Mobile Robots in Dynamic Environments with Computer Vision

by

Jacob Michael Perron

B.Sc., York University, 2014

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Spring 2018

Approval

Name:	**Jacob Michael Perron**
Degree:	**Master of Science (Computing Science)**
Title:	**Enabling Autonomous Mobile Robots in Dynamic Environments with Computer Vision**
Examining Committee:	**Chair:** Anoop Sarkar Professor
	Richard Vaughan Senior Supervisor Professor
	Ping Tan Supervisor Associate Professor
Date Defended:	**April 19, 2018**

Abstract

Autonomous mobile robots are becoming more prevalent in our society as they become more useful. Often the environments we would like to see robots working in will be dynamic, with objects like people moving throughout. This thesis explores methods to make autonomous mobile robots more effective in the presence of dynamic objects. Specifically, two applications are developed on top of existing computer vision techniques where robots interact directly with moving objects.

The first application involves multiple robots collaboratively sensing and following an arbitrary moving object. This is the first demonstration of its kind where robots jointly localize themselves and an object of interest while planning motion that is sympathetic to the vision system. Live robot experiments are conducted demonstrating the efficacy of the proposed system.

The second application is a novel human-robot interaction system based on face engagement applied to an unmanned aerial vehicle (UAV). A unique use of facial recognition software enables an uninstrumented user to command a UAV with only their face. A series of experiments demonstrate the effectiveness of the interaction system for sending UAVs on a variety of flight trajectories.

Keywords: mobile robotics; visual SLAM; human-robot interaction; computer vision;

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Now more than ever, autonomous mobile robots are being incorporated into our societies helping improve our standard of living. We see warehouses full of robots moving consumer goods, drones acting as videographers, robots guiding tours at museums, robotic vacuums in our homes, and most recently, autonomous vehicles appearing on our roads. It appears that it will not be too long before autonomous mobile robots are commonplace.

As more mobile robots start integrating in different areas of society they will need to operate in a wide variety of environments. Often, these environments will be dynamic; objects move and structures physically change. Less dynamic environments can be characterized by physical changes that occur over the course of days, weeks, or months, whereas more dynamic environments involve continuously moving objects, such as, humans or vehicles. As an example, autonomous cars have to operate in the presence of other moving vehicles and pedestrians (more dynamic) as well as changes to road infrastructure (less dynamic). Mobile robots cannot assume the world is static if we expect them to work effectively.

In order for a mobile robot to work autonomously in a dynamic setting it must have a way to sense its surroundings. Camera sensors are ubiquitous among modern robots and can provide a great deal of information about the environment. Robotics applications can employ techniques from the computer vision community to accomplish tasks such as object recognition, 3D reconstruction, and mapping and localization. This thesis explores methods for mobile robots to autonomously operate in more dynamic environments with vision as the primary sensor. Specifically, two mobile robot applications are developed that build upon available computer vision systems.

Both applications involve novel demonstrations where robots interact directly with other moving objects. The first application involves multiple robots working indoors to scan an object with the aid of a visual localization and mapping technique. The second application occurs outdoor with an aerial robot interacting face-to-face with a human. The main contributions of this thesis are the novel robot systems presented and demonstrated for each application.

In Chapter 2, a collaborative method for following arbitrary moving objects is investigated with multiple ground robots. An indoor application is developed that is closely coupled with a vision system for localizing the robots and reconstructing the 3D position of an object of interest. The contributions of this work are: (i) the first demonstration of multiple robots solely using the visual localization and mapping system for object following and scanning; (ii) a description of a simple motion controller that tracks and orbits objects while respecting camera pose constraints imposed by the vision system. The results of this work are published at the *3rd Workshop on Multi View Geometry in Robotics* at *Robotics Science and Systems* [28].

In Chapter 3, a novel face-based interaction method is described involving a flying robot and a human interaction partner. An outdoor application is developed that uses common face recognition software to enable an uninstrumented user to send a robot on configurable trajectories. The contributions of this work are the novel interaction method and techniques for face-based UAV interaction. In particular, benefits of a runtime-determined facial expression as a control signal are demonstrated in several real robot experiments. The results of this work are published at the *14th Conference on Computer and Robot Vision* [3].

Related work for each of the two applications is presented within their respective chapters.

Chapter 2

Following with Collaborative Visual SLAM

The work presented in this chapter is motivated by the application of autonomous 3D modelling of moving objects in an arbitrary scene. Such a system has applications in industries such as entertainment, recreation, security, and health. As an example, having access to 3D models of humans over time can assist in producing automatic diagnostics of orthopedic patients or performance analysis of athletes [21]. It is inherent that robots must work in dynamic environments to solve the problem of gathering data for 3D modelling. Furthermore, in fulfilling the 3D modelling task, robot motion to acquire data will be affected directly, or indirectly, by the motion of the object of interest.

To approach the problem of autonomously modelling moving objects we can break the problem into two parts: *localization* of the object and *motion planning* of the robot. A robot cannot effectively plan what to do if it lacks the means to determine the location of the object with respect to itself. Once the robot can localize the object of interest it can plan its motion accordingly, subject to the constraints of the modelling task.

To solve the object localization problem, we turn to Simultaneous Localization and Mapping (SLAM). SLAM is a well studied problem in robotics. Durrant-Whyte and Bailey give an excellent overview of the history of the SLAM problem dating research back as early as 1986 [12]. Solutions to the SLAM problem are useful as it provides a means for a robot to situate itself in an environment with little or no prior information. This is especially important for robots to effectively navigate in GPS denied environments (e.g. indoors). In particular, *visual SLAM* focuses on solving the SLAM problem with vision sensors (e.g. cameras). In 2007, the landmark paper from Davison et al [8], shortly followed by another landmark paper from Klein and Murray [19], demonstrated the first successful SLAM applications with a single freely moving camera. Since then, significant progress has been made towards improving visual SLAM systems [14, 15, 26].

The Markov assumption (often referred to as the *static world assumption*) is prevalent throughout the majority of proposed SLAM systems. This means that many systems are

susceptible to dynamic environments resulting in poor performance. In an attempt relax the static world assumption, *collaborative SLAM* systems that involve more than one sensor have demonstrated not only a robustness to dynamic scenes, but the capability to incorporate pose estimates of dynamic objects into their world model [25, 37]. Specifically, we turn our attention to the collaborative visual SLAM system, *CoSLAM* [37], because of the rich data available from camera sensors that lends itself to 3D modelling.

Motion planning should be synergistic with the method of localization, especially in the case of modelling a moving object. As an example, it may be a bad decision to move a robot such that the object of interest leaves the robot sensing field-of-view. It may be better that the robot move another direction to not only keep the object in sensor range, but gather observations from another point of view.

The remainder of this chapter investigates the viability of performing 3D reconstruction of moving objects with a team of robots using a collaborative visual SLAM system. Work related to collaborative SLAM and multi-robot object following are discussed in 2.1.1 and 2.1.2 respectively. A multi-robot system design to work with CoSLAM is presented in 2.2 followed by experiments in 2.3. Finally, this chapter concludes with a discussion in 2.4 and possible future directions in 2.5.

2.1 Related Work

This section discusses existing work related to collaborative SLAM and object following with multiple robots.

2.1.1 Collaborative SLAM

Collaboratively localizing robots with multiple cameras has been previously investigated. Chang et al. [5] implement a vision based localization and tracking system for humanoid robots in a soccer field environment. The system is limited to 2D environments and shape detection is used to track the other robots and the ball. Dhiman et al. [9] use reciprocal observations of fiducial markers for camera pose estimation. Their approach is not designed for dynamic scenes and the fiducial markers need to be present in the image views. Dias et al. [10] propose an uncertainty based multi-robot cooperative triangulation method for object position estimation. However, they do not solve the camera poses and object detection problem. In [16], multiple flying robots perform monocular visual SLAM and estimate their motion individually. A centralized ground station receives preprocessed results and localizes the robots in a global coordinate frame. Due to the lack of cooperation between robot cameras, the system is not able to handle dynamic environments.

CoSLAM [37] is a recently-developed collaborative visual SLAM method that operates by receiving video frames from independent moving monocular cameras. By cooperatively tracking and creating a global 3D map, multiple independent cameras are able to detect and

track dynamic feature points in real time. It is evaluated on high-quality, video sequences that are manually synchronized and processed on a single machine after data acquisition.

2.1.2 Following a Moving Object with Multiple Robots

Robot controllers for object following have been widely studied under different settings in robotics. Dang et al. [7] track a moving object with multiple unmanned aerial vehicles (UAVs) in a formation anchored to a single leader. In [17], multiple robots localize and encircle an object in a distributed manner with range-finders. Aranda et at. [1] propose a general method for enclosing an object with any 3D geometric formation. Unlike the aforementioned approaches, the desired controller must consider the camera sensor constraints so that the target object can be maintained within the sensor's field-of-view. Yang et al.'s [36] work is similar to ours in that the robots move in order to achieve a desired sensor reading while communicating sensor information with each other. Their solution is distributed, with one-hop communication between neighbours, and evaluated in simulation with range sensors. Unlike the controller proposed in 2.2.2, they do not consider constraints imposed by non-holonomic robots or camera sensors.

2.2 System Design

The proposed system for following a generic moving object consists of multiple, non-holonomic, ground robots and two software components: CoSLAM and the motion planner. An example of what the system looks like in action can be seen in Figure 2.1. The robots each run a monocular camera mounted orthogonal to its heading (90 degrees counterclockwise) and an onboard controller. The cameras are mounted orthogonally due to the non-holonomic nature of the robots and its synergy with the motion planner. CoSLAM runs on a base station receiving video frames from each robot to produce pose estimations for all robots and the moving object of interest. Communications between robots and base station are done over WiFi.

2.2.1 Adapting CoSLAM for Robots

As CoSLAM was developed with offline processing of pre-recorded high-quality videos that are synchronized manually, there are difficulties in applying it to platforms with limited sensing payload in real time applications. First, the original CoSLAM system proposed by Zou and Tan [37] is discussed, followed by modifications required for real time operation on robots.

CoSLAM

CoSLAM follows the conventional sequential structure-from-motion pipeline, but using multiple cameras as input. Each camera is calibrated beforehand and is expected to move inde-

Figure 2.1: Two robots following and orbiting a human using CoSLAM as the only sensor.

pendently during operation. An initialization stage is required where all the cameras must look at the same scene to initialize a global 3D map. This initialization stage provides cameras with a common frame of reference as well as a common scale, as it is well known that monocular SLAM systems can only provide reconstructions up to an arbitrary scale. After initialization camera poses are tracked by registering 2D image features to the corresponding map points. The 2D features for an individual camera are detected and tracked with an efficient Kanade-Lucas-Tomasi (KLT) [35] tracker. When the number of visible map points in a camera image drops below a threshold, new map points can be generated with feature matches from a single camera or multiple cameras.

To handle dynamic scenes, at every frame map points are classified as *static*, *dynamic*, or *false*. All map points are labeled as *static* when they are first triangulated. For all visible static map points a reprojection error is computed by projecting them into both the current and previous frames. A large reprojection error can be explained by one of two reasons: the point is dynamic (ie. moving) or the point is generated from false feature matches. A key observation is that the dynamic points should be visible at the same position in space for different cameras at the same time. 3D points are triangulated again, this time with features matched between different cameras. If re-triangulated points have small reprojection errors they are labeled as *dynamic*, otherwise they are labeled as *false* points and should be discarded.

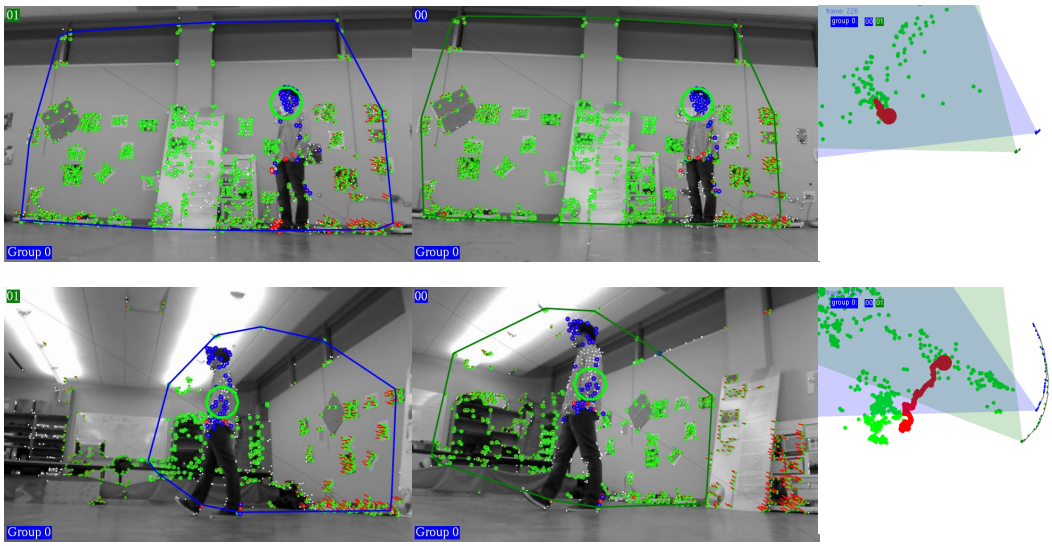

Figure 2.2: Tracking results of two robots following a walking person at frame 226 (first row) and at frame 1025 (second row). Dynamic points (blue), static points (green), camera overlap region (polygons) and 3D map (right) are displayed. The estimated 3D position of the object is projected to both image planes as a green circle. The 3D object trajectory is drawn in red.

Time Synchronization and System Initialization

In practice, robots send compressed images back with timestamps to the ground station over a wireless network. Time synchronization between frames is needed for dynamic points detection, as discussed in the previous section. *Network Time Protocol (NTP)* [20] helps synchronize the clocks of the robot computers and the ground station. Images are received at the ground station and synchronized according to their timestamps.

Two issues need to be resolved before camera poses obtained from CoSLAM can help guide autonomous control of the robots. First, the 3D map from CoSLAM has scale ambiguity. Second, the rigid transformation ${}^{\mathrm{C}}T_{\mathrm{W}}$ needs to be known between the camera frame F_{C} and the world frame F_{W} for all robots. Both problems can be solved simultaneously by introducing Augmented Reality (AR) tags to the initialization stage. Each AR tag is assigned 3D coordinates in F_{W}. Relative positions of the AR tags are detected in all cameras. The 3D coordinates of the AR tags in F_{C} are triangulated using the initial estimated camera poses. The scaling factor s and the rigid transformation ${}^{\mathrm{C}}T_{\mathrm{W}}$ are solved by registering the 3D coordinates of the AR tags in F_{C} and F_{W}. The tags are removed after the system is successfully initialized. The camera poses are transformed into F_{w} before being sent to the controller. Note, that since scale is not corrected again once the system is running it will drift over time.

Object Position Estimation

As feature points on the dynamic object are difficult to track, the distribution of dynamic points on the object changes over time. Moreover, point location inaccuracies are caused by motion blur and the rolling shutter effect during camera movement. These factors lead to noisy 3D reconstruction of dynamic points.

To obtain a stable position estimation of the object, a simple and effective approach is adopted. Since most of the dynamic points should be distributed in a constrained space of similar size to the object, we discard the points that are further than the expected distance travelled from the last observed object location. The geometric median of the remaining dynamic points is used to estimate the object's position. A low-pass filter helps smooth out the moving object's trajectory. The tracking results of a walking person are presented in Figure 2.2.

2.2.2 Orbiting Controller

We desire a robot control scheme sympathetic to the requirements of CoSLAM for following a moving object; specifically, keeping the object in view of each camera and maintaining overlap between neighbouring cameras. By fulfilling these two requirements the robots are able to detect and follow a moving object with CoSLAM. Furthermore, it would be beneficial to ensure that the object does not occupy too much, or too little, of the camera view. In other words, there is a desired distance to keep from the object based on its size.

An orbiting controller similar to [17] allows our non-holonomic robots adapt easily to a variety of motions. The robots are able to keep up with a moving object assuming they move a magnitude faster than the object. As the optical axis of the camera is perpendicular to the robot heading, the goal is to drive perpendicular to the vector between robot and object to ensure that the object is centered in the camera view. In addition, the smoother rotations produced by orbiting are preferable to CoSLAM as opposed to sharp or erratic motions.

Figure 2.3 provides an overview of the 2D geometry between a robot, its neighbour, and the object it is following. Given poses of the robot and object from the base station a desired global yaw angle $\hat{\theta}$ for the robot is computed using Equation 2.1. θ_v is the polar angle of the vector $\vec{v}$ from robot to object and θ_u is the polar angle of $\vec{u}$ that is perpendicular to $\vec{v}$. ρ is a weight between zero and one computed as a function of distance to the object in Equation 2.2. The distance between the object and image plane is denoted by d. As d increases, the robot needs to move closer to the target. The desired distance to the object $\hat{d}$ is determined a priori based on the desired object height as it appears in the image $\hat{h}$ (pixels). Given $\hat{h}$, focal length f of the camera, and the estimated physical height H of the object, $\hat{d}$ can be computed with Equation 2.3. In practice, the object's height may be fixed if it is available a priori or estimated from the 3D pose given by CoSLAM.

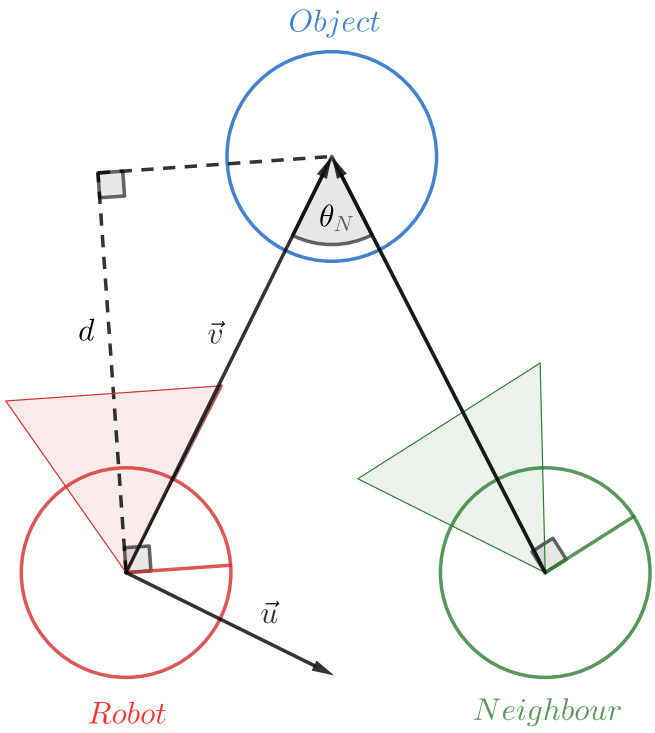

Figure 2.3: Geometry relating a robot (red), its neighbour (green), and the moving object they are orbiting (blue). Shaded triangles represent the cameras mounted orthogonal to the robot's heading.

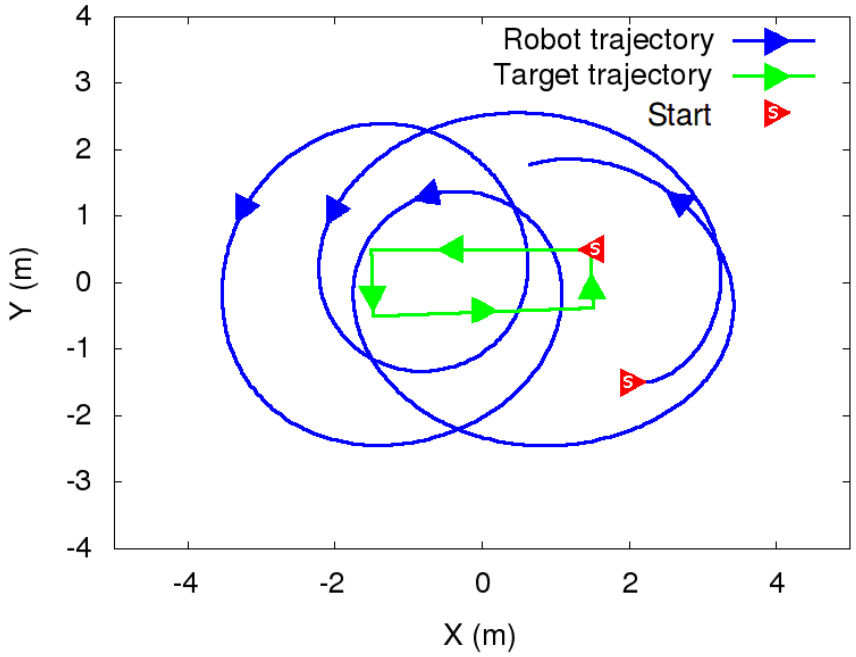

Figure 2.4: The trajectory of a single robot orbiting a moving object in simulation.

$$\hat{\theta} = \rho \cdot \theta_v + (1 - \rho) \cdot \theta_u \tag{2.1}$$

$$\rho = min(1, max(0, -(\hat{d}/d) + 1)) \tag{2.2}$$

$$\hat{d} = \frac{f \cdot H}{\hat{h}} \tag{2.3}$$

The combination of angles used to produce $\hat{\theta}$ lets the robot adjust its distance to the object dynamically. A PID controller is used to throttle the robot's rotational velocity in order to approach $\hat{\theta}$. This PID controller with a constant velocity produces trajectories similar to Figure 2.4.

Keeping the Object in View

It is possible that rotational motions produced by the orbiting controller cause the object to be outside field-of-view θ_{FOV} of the camera. To prevent losing sight of the object, upper and lower bounds for $\hat{\theta}$ are determined based on θ_{FOV} with Equation 2.4 and Equation 2.5. Hence, $\hat{\theta}$ is clamped according to the constraint 2.6. This constraint is particularly useful if the object is first detected too far away and the robots need to close the gap without losing sight. In the event that the object cannot be detected, the robots will use the object last known position in an attempt to recover.

$$\theta_{\text{LB}} = \theta_u - \frac{\theta_{\text{FOV}}}{2} \tag{2.4}$$

$$\theta_{\mathrm{UB}} = \theta_u + \frac{\theta_{\mathrm{FOV}}}{2} \tag{2.5}$$

$$\theta_{\mathrm{LB}} \leq \hat{\theta} \leq \theta_{\mathrm{UB}} \tag{2.6}$$

Maintaining Overlap

So far we have only considered a single robot orbiting an object of interest, but in order to detect a dynamic object CoSLAM requires camera overlap. Therefore, a final constraint is applied that throttles the robot's forward velocity given the position of its nearest neighbouring robot. The idea is to have each robot maintain a distance with another robot in the orbit. A neighbour robot can be chosen a priori or dynamically simply by choosing the closest robot.

A desired distance $\hat{d_{\mathrm{N}}}$ to a robot's neighbour is computed with Equation 2.7. $\hat{\theta_{\mathrm{N}}}$ is the desired angle between a robot and its neighbour in the orbit. As $\hat{\theta_{\mathrm{N}}}$ becomes smaller more overlap is created between cameras. If the actual distance to a robot's neighbour is less than $\hat{d_{\mathrm{N}}}$, then the robot is too close and risks collision. Likewise, if the distance is greater than $\hat{d_{\mathrm{N}}}$, then the robot is too far away, reducing overlap. Next, the bearing to a neighbour is used to determine if the robot is in front or behind. Each robot follows a simple set of rules listed in Table 2.1 to set its velocity. A PID-controller helps smooth changes in velocity.

$$\hat{d_{\mathrm{N}}} = 2\hat{d} \cdot sin(\frac{\hat{\theta_{\mathrm{N}}}}{2}) \tag{2.7}$$

Table 2.1: Rules for adjusting robot velocity.

Neighbour distance	Bearing to neighbour	Result
Too close	Behind	Decrease velocity
Too close	Front	Increase velocity
Too far	Behind	Increase velocity
Too far	Front	Decrease velocity

2.3 Demonstrative Experiments

The system is deployed onto slow-moving, low-cost non-holonomic ground robots with generic web-cam sensors. Due to the limited speed of the robots, slow-moving objects are chosen to follow. The objects are non-cooperative and do not act as if the robots are present.

The system was tested in two different environments: a visual-feature-rich lab (Figure 2.1) and a sparsely featured boardroom. The data described below is from a representative run in the lab environment. A video demonstration[1] shows a team of two robots scanning a moving object in both the lab and boardroom environments.

[1] `https://www.youtube.com/watch?v=8xGa_mETnPc`

Figure 2.5: iRobot Create 1 Platform with Odriod U3 and Gumstix computers

2.3.1 Apparatus

The robots used in the experiments are the iRobot Create 1 platform with an onboard Odroid U3 (cellphone-class) computer to compress and transmit camera frames and a Gumstix computer for robot control (See Figure 2.5). The base station was a laptop with an Intel Core i7 (2.5 GHz) processor and NVIDIA GeForce GTX 850M graphics card. CoSLAM ran on the laptop posting pose information to a Redis server. The moving object of interest was a stuffed teddy bear attached to a non-cooperating robot standing 0.86 metres tall.

2.3.2 Method

In each experiment the moving object travelled a pre-planned rectangular trajectory until it covered a set distance. Two or three robots were given the task of tracking the object. After CoSLAM initialization, the object started driving into view of the robot's cameras. Once the object was detected, the controller was engaged and the robots began their encirclement. The system ran until the moving object completed a full traversal of a three metre by one metre rectangle.

The object drove at a speed of 0.06m/s, while the robots drove at around 0.3m/s. Slow speeds were chosen to avoid motion blur and rolling shutter artifacts that would negatively affect the vision system. The object height of 0.86 metres was set a priori and desired height ratio in the view was 0.6. This means that the robots should encircle with a radius of around

1.92 metres. Neighbours were manually assigned to maintain 30 degrees between them in the circle.

2.3.3 Results

All relevant data was recorded during each experiment, which includes the 3D point-clouds generated by CoSLAM, video from both onboard and external cameras and all derived coordinates, calculations, and commands passed to and from the controllers. By examining a representative successful run we can assess how the system achieves the original goal of tracking a dynamic object.

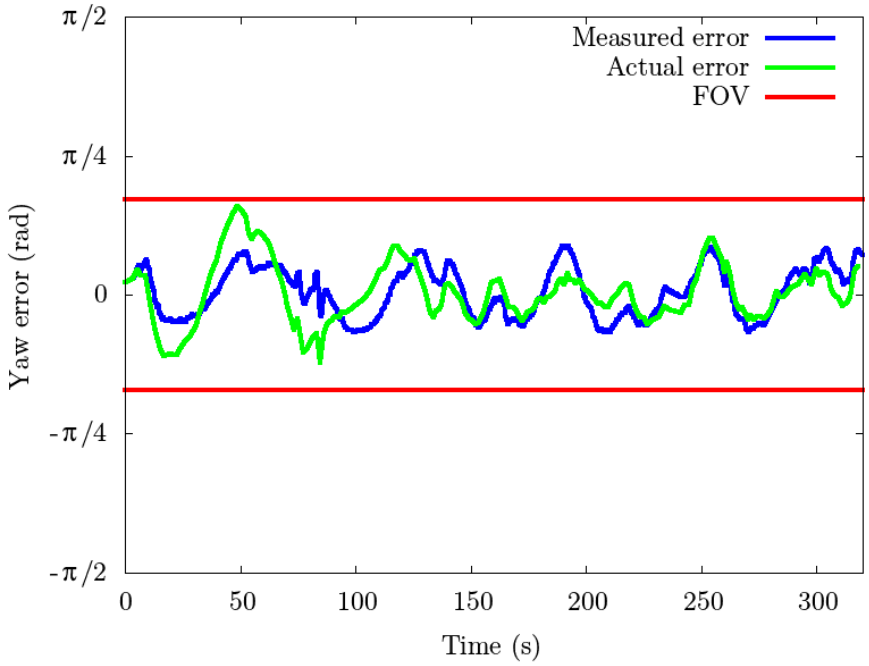

Figure 2.6: Yaw error over time (blue) compared to error obtained from ground truth (green). Although error fluctuates object never leaves field-of-view (red).

Three error metrics are considered: *yaw error*, *ratio error* and *neighbour error*. Yaw error is defined as the difference between θ_C and robots yaw. Ratio error is the difference between ratio of the object's height in the image and the desired ratio. Neighbour error is the difference between θ_N and $\hat{\theta_N}$. These three errors reflect the constraints outlined in 2.2.2.

We want to show that the errors measured by the system actually correspond to the errors in the real world. To support this assertion, we examined the video recorded by the cameras on the robots themselves. By manually annotating the videos we gained a ground truth measurement for the yaw error and ratio error over time. We were unable to obtain ground truth measurements for the neighbour error.

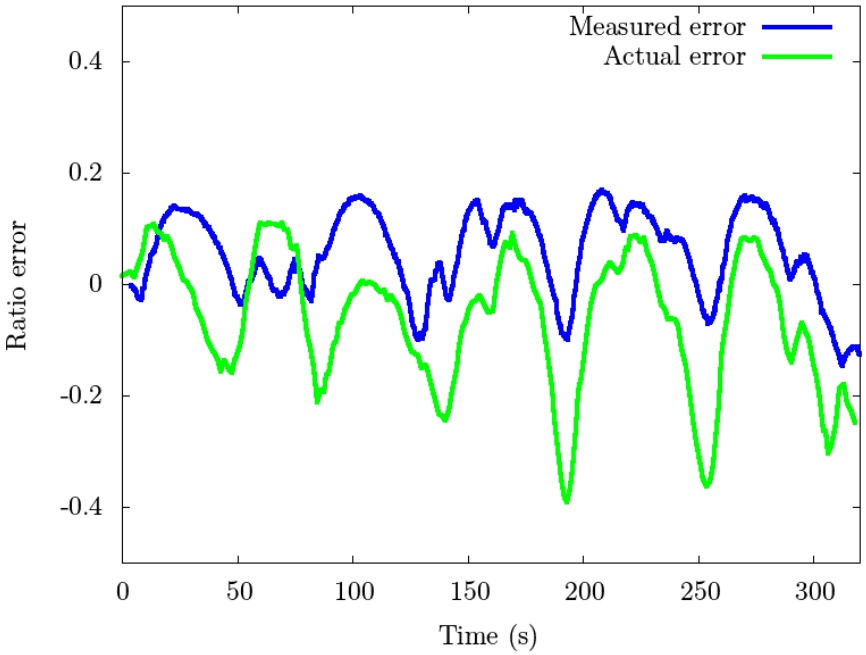

Figure 2.7: Ratio error (blue) compared with error obtained from ground truth (green). The ratio measured is the object's height in the camera view.

The representative results shown are from a run performed in the lab environment with two robots. Yaw error for one of the robots can be seen in Figure 2.6. The correlation between the measured and actual error shows that CoSLAM provides reasonable localization that is required to keep the object in view. Discrepancies are caused by changes in the distribution of features on the object due to changes in viewing angle. Different distributions move the median around the object's actual centre.

A similar correlation can be observed in Figure 2.7 for ratio error, again showing reasonable localization from CoSLAM. However, the difference between the measured and actual error increases over time caused by scale drift in CoSLAM's measurements. Robots will continue to increase their radius from the object increasing the risk of collision with the environment's boundary. Correcting scale drift in the future will allow for a more stable system.

It is difficult to infer how accurately the measured neighbour error reflects their actual error without a ground truth. However, the neighbour error in Figure 2.8 accurately reflects the fact that the robots never collided.

We find that the system is able to successfully track the moving object, keeping it in view of at least two cameras at all times. CoSLAM provides the necessary pose information to direct the controllers and the robots move to allow CoSLAM to continue object tracking. The robots orbit the object as it travels, so it is observed from all sides over time. This is

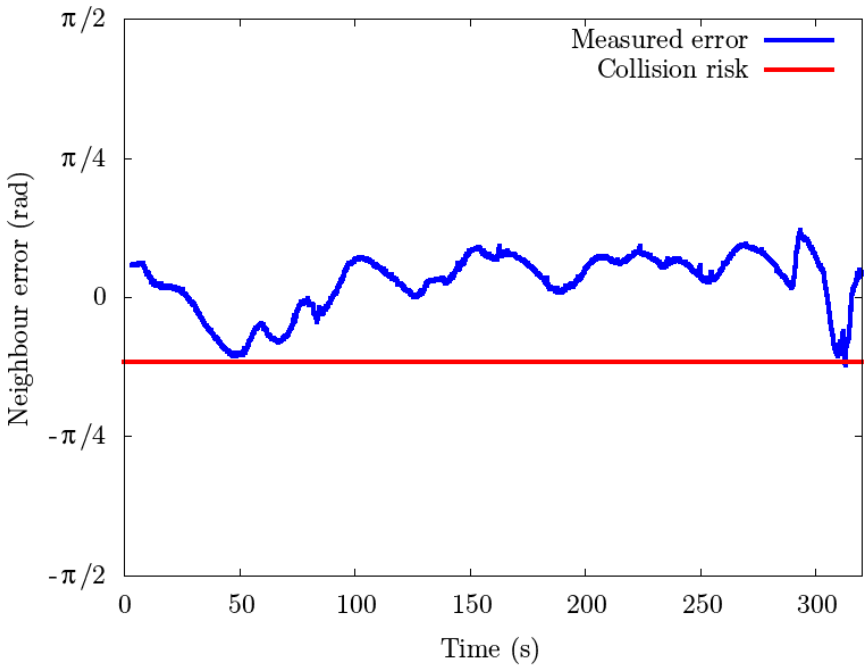

Figure 2.8: Neighbour error (blue) over time. Lower error increases overlap with neighbours camera view. The red line indicates that the robots are too close and risk colliding.

the behaviour we seek for our system to eventually capture 3D models of the object in real time.

2.4 Discussion

An initial demonstration of the feasibility of cooperative visual-SLAM-only control for robot teams to scan a moving object was conducted. Results are encouraging in the direction of multi-robot object-following systems based on vision. While some constraints remain to be overcome, the experiments nevertheless demonstrate a functional robot team exploiting CoSLAM to achieve subject tracking using only colour cameras and network communications. This capability, previously limited to more expensive and precise range-finding sensor suites, opens up new research possibilities using simple robots to achieve reliable visual coverage of uninstrumented, non-cooperating subjects.

2.5 Future Work

More work will be needed to improve and extend this system by making it robust to platforms with high dynamics (e.g. UAVs), increasing the number of participants and untethering the robots from a control station.

Future work may include attempts at integrating the system with faster robots with more degrees of freedom of motion, including UAVs. This would likely involve very different robot control schemes and possible more complex motion planning.

Also of interest is creating a fully distributed version of CoSLAM. This could be beneficial for larger robot teams, where wireless bandwidth limits is a practical problem, by increasing tolerance of intermittent communications or individual vehicle failures.

The system currently captures 3D point clouds of the object, but we have not yet investigated their quality or scope for building solid, time-varying object models - this is another possible future direction for this work.

Chapter 3

Face-based HRI

Recent innovations have brought us very capable, small-size, and low-cost unmanned aerial vehicles (UAVs). These have many applications and new industries forming around them. Current commercial operator control interfaces for UAVs have either a dedicated hardware controller or functionally similar software running on a tablet or smartphone. Researchers have begun to conduct user studies that investigate how untrained users choose to interact with robots using only 'natural' interfaces where the human participant is entirely uninstrumented, i.e. they carry no equipment, their appearance is unaltered, and little training is required [4, 27, 33]. The long term goal is to enable people to interact with robots as we now interact with people and trained animals, just as long imagined in science fiction. Consequently, this provides robots with more autonomy in dynamic scenarios that involve interaction with people. Value is added by offloading cognitive load on the user by increasing autonomy of the robot. Towards this goal, a new method is proposed for robots and humans to interact that is hands-free; no hardware is required to be handled by the human.

This chapter describes a novel human-robot interaction (HRI) system for controlling UAV flight trajectories primarily by face engagement with the UAV. The interaction system only requires the commodity sensors available onboard vehicles that cost a few hundred dollars today. Image processing is currently done off-board over a wireless network, but the necessary computation power will be available onboard even low-cost UAVs in the very near future. The developed application is tailored for flight trajectories of a UAV, but the concept presented can easily be extended to other robot types[1].

The proposed HRI system uses visual and orientation sensors onboard a quadrotor UAV to *1)* learn the identity and facial expressions of a user, *2)* accept user input through touch interaction, and *3)* aim the flight trajectory of the robot based on the relative pose of the user and the robot. This interaction is modelled on the act of drawing a bow or slingshot, in which the launch azimuth and elevation angles are lined up by eye, and the launch power

[1]Indeed, during early development the concept was extended to control a simulated and live ground robot on parabolic arcs constrained to the ground plane

is set by the magnitude of the draw. The 'slingshot' interaction allows the user to send the robot into predictable trajectories in 3D space without a hardware controller. Three different trajectory types are implemented: straight-line, parabola, and circling, that have different applications.

First, work related to uninstrumented HRI with UAVs is reviewed in 3.1. The proposed HRI system with UAV trajectory types and their applications are presented in 3.2. Three experiments evaluating different aspects of the system are described in 3.3. Finally, this chapter concludes with a discussion in 3.4 and possible future directions in 3.5.

3.1 Related Work

Existing work on uninstrumented HRI with UAVs has focused on gesture-based interfaces [22, 23, 27, 29], although voice [2, 11, 31] and touch-based [32] interactions have also been proposed. Monajjemi et al. [22] have the user wave their arms to establish mutual attention with a moving UAV. The interaction distance between user and robot is too great for current face detection solutions to be feasible. Pfeil et al. [29] explore several types of gestural interfaces with the body and arms to command a UAV. Their implementation requires an off-board RGB-D sensor positioned in front of the user, severely affecting mobility. In contrast, the proposed 'slingshot' interaction does not inhibit user motion as sensing is done onboard the robot.

Face detection has also been used in human-UAV interaction systems. In [32], selection between multiple robots is accomplished with face engagement as an attentional cue. In [23], the position of the user's face assists in localizing human waving gestures. Their work is similar to the proposed 'slingshot' interaction in that a small gesture vocabulary is used to signal the execution of high-level tasks to a UAV. One major difference the proposed method has with the aforementioned work is that the face is used as the primary means for controlling the robot.

Hulens and Goedemé [18] control a UAV with onboard vision sensing and processing for filming a moving person under cinematographic rules. A human detector and tracker are input to a yaw controller, while a face detector is simultaneously used to control the UAVs roll. Depth measurements from a stereo camera are used to control the UAVs pitch, or distance to the person. Their work is a great example of a UAV operating autonomously in close proximity to a human. The proposed 'slingshot' interaction is different in that it offers a more general user interface for commanding a UAV.

The method proposed in this chapter uses the line-of-sight vector from the user's face to the robot to indicate a direction (see Figure 3.1). This has been investigated in related work involving gaze direction [6, 34], and is similar to pointing gestures [30] in which a line is drawn between the user's eye and fingertip, communicating a pointing vector to a robot.

However, the method presented in this thesis is the first demonstration of a pointing-like gesture to control a UAV using only monocular camera sensing. [2]

3.2 System Design

The proposed system involves three main phases:

- *Ready*— the user's identity and facial expressions are learned and input is provided through touch-based interaction
- *Aim*— the robot starts flying and keeps its user centered in its camera view, while the user lines up the trajectory and chooses its power by "drawing back" analogous to firing a bow or slingshot
- *Fly!*— the user signals to the robot to begin a preset parameterized trajectory. The robot executes the trajectory with parameters observed at the end of the *Aim* phase. Three possible trajectory types are described in 3.2.3.

In this section each phase of the system is described in detail along with the components involved at each step.

3.2.1 Facial Expressions—Ready

Face recognition techniques require training to identify users. Training can be done in a separate phase and identity models can be assumed *a priori*, but there are advantages to including the training phase at runtime. In particular, the system will learn a model that is automatically calibrated to the current environment, lighting conditions, and transient appearance details of the user such as clothing, hairstyle, and worn accessories. Although these advantages were noted in practice, they were not thoroughly evaluated in the subsequent experiments. To facilitate training at runtime, the interaction begins with the robot in the user's hand and the user's identity is learned to sufficient confidence (100% precision over a three second window in this case) before the robot takes flight. This enables a degree of confidence that the robot will be able to find its user again once it has launched. Training time is typically less than one minute, and can be faster if it is the same user from a previous session.

The ROS *face_recognition* package [13] is used to learn the user's identity given images from a conventional camera sensor onboard the robot. As face recognition software is not yet meeting human performance, false positives can be expected that could cause the robot to take commands from other humans or human-like objects. In order to mitigate this

[2] It may be argued that [18] can also make the same claim. Note that both [18] and [3] were published at the same time.

Figure 3.1: Parabolic UAV trajectories being aimed by the user's face as described in this thesis. The vector between the face and the robot determines the launch angle, and the size of the face determines the distance.

Figure 3.2: Example learned faces taken from our experiments. The top row shows the *neutral* face, and the bottom row shows the *trigger* face that is used to send the start signal. The two columns on the left were captured with the robot in the user's hand, while the two on the right were captured in flight. Note the low image quality: the off-the-shelf face recognition software was able to handle the poor imagery up to a distance of several meters, generalizing successfully between scales.

weakness in the current state of face recognition technology, we borrow the concept of two-factor authentication where face identity alone is not sufficient to control the robot. Although a sensor modality independent from vision would be ideal for this purpose, once in flight, vision is the most feasible conventional sensor for detecting and interacting with humans. To address dual problems of signalling to a flying robot and providing robustness to false positives, the concept of a runtime-determined *trigger expression* is employed.

A convenient side effect of current off-the-shelf face recognition is that the same user showing dramatically different facial expressions (Figure 3.2) can be detected as distinct face identities. This side effect is exploited by training first on the user's *neutral expression* and, upon reaching 100% precision over a rolling time window, informing the user to choose a *trigger expression* (Figure 3.3). The *trigger expression* functions as a signal to start a particular flight behavior, and incidentally functions as a security measure to ensure that other humans in the area who do not know the chosen expression cannot command and recover the robot.[3]

[3]The interaction method described in this thesis generalizes straightforwardly to other types of gestures: the system can be modified and extended with a variety of other physical interactions as gesture recognition technology improves.

Figure 3.3: The two stages of the *Ready* phase: (left) learning the *neutral expression*; (right) learning the *trigger expression*.

The user rotates the robot in their hand to signal that their *trigger expression* is being displayed and the system trains to recognize this new expression to 100% precision. The direction of rotation is remembered during the *Fly!* phase, where it can determine parameters of the flight path (such as direction to circle). Once training is complete, the robot takes flight and the *Aim* phase begins.

3.2.2 Face Position—Aim

Once the robot has begun flying, it starts checking for the learned user's face to appear in view. When the user's face is recognized, the robot controls its yaw heading to keep the face centered in its field-of-view. Adjusting its yaw heading helps keep the user visible and most importantly, allows the user to aim the robot in a desired direction. The line-of-sight vector between the center of the user's face and the robot defines the direction of flight in the next phase. In addition to the location of the face for direction, an analogy to shooting a bow is applied for distance control: in shooting a bow, the power of the shot is determined by how much energy is put into the arrow, typically by how far the string is drawn back. Borrowing this concept, the magnitude of the flight path is determined by the size of the user's face in the image: a smaller face means the user is farther away and more power is put into the shot, so the robot travels farther in the *Fly!* phase.

The *trigger expression* learned in the first phase acts as the launch signal. Once the robot detects this expression, it initiates a trajectory defined by the line-of-sight vector and distance to the user as part of the *Fly!* phase.

3.2.3 Trajectory Execution—Fly!

Once the trigger signal has been received, the UAV executes a flight path. Three different trajectories are defined by analogy to different familiar projectile modalities: the *beam*, the *slingshot*, and the *boomerang*.

Beam

Beam is a straight path along the user's line-of-sight vector to the robot at launch time. The distance along this path is determined by modulating a predefined base distance with the size of the user's face on launch signal, a smaller face indicating more power and a greater distance. Once reaching the location the robot can perform an application specific behavior such as video capture or mapping. The utility of *Beam* is that it allows the robot to be sent to a location in 3D space at arbitrary altitude, for example for localized inspection of a tall building.

Slingshot

Slingshot is a parabolic trajectory launched at the angle specified by the user's line-of-sight vector to the robot. Similar to *Beam*, the ground distance covered by the trajectory is determined by modulating a predefined base distance with the size of the user's face. The robot's vertical velocity is decreased at a constant rate during the flight path, thereby following a parabolic arc analogous to a thrown projectile, until reaching the original launch altitude at which point the robot executes an application specific behaviour. The utility of *Slingshot* is that it can send a robot over an arbitrary-sized vertical obstacle and into an area out of line-of-sight, perhaps to perform video capture.

Boomerang

Boomerang is a circular path tangent to the user's line-of-sight vector to the robot, analogous to a boomerang that is thrown forward and curves around to return to the thrower. The direction of the circle (curving leftward or rightward) is determined by the direction of rotation during the touch interaction in the *Ready* phase. The radius of the circle is determined by the size of the user's face on launch signal. Smaller faces indicate a more powerful throw, and therefore a larger circle. The utility of *Boomerang* is to define a survey path, perhaps to obtain a 360-degree scan of an object such as a building or statue.

3.3 Evaluation Experiments

Three experiments are performed to investigate the efficacy of the interaction system, evaluating several aspects of its performance. The first experiment targets the reliability of the

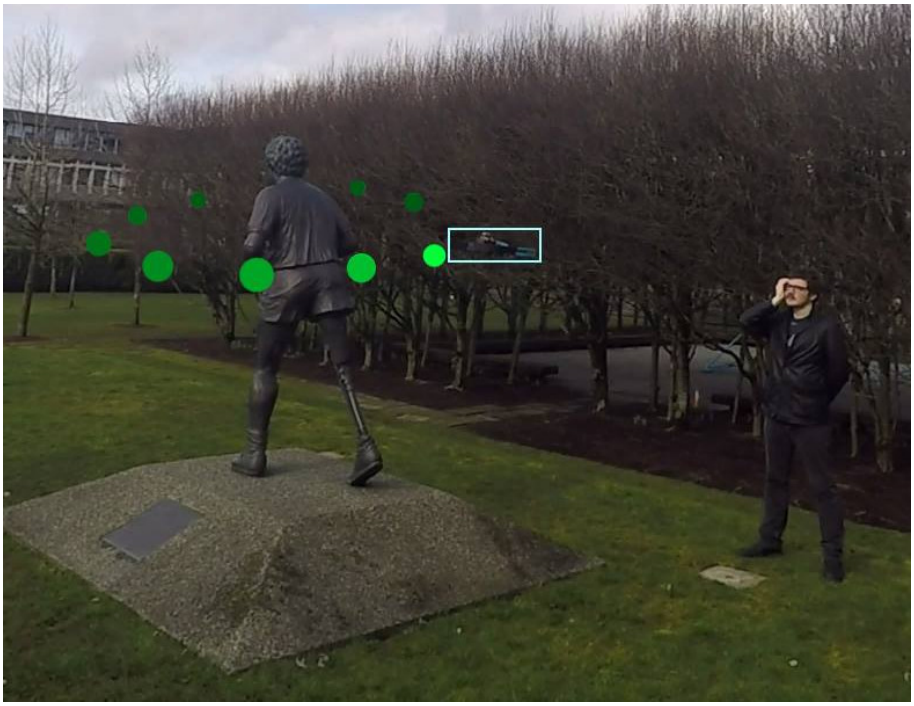

Figure 3.4: User signalling the robot to begin the *boomerang* trajectory orbiting a statue, with robot highlighted in blue.

face recognition system. The last two experiments focus on the performance of the *Beam* and *Slingshot* trajectories respectively.

For the *Boomerang* trajectory, simple outdoor demonstration trials were conducted where the user attempted to send the UAV on a circular path in a particular direction, ending the trajectory back near the user. Figure 3.4 illustrates a sample trajectory successfully orbiting a statue at close range. Although no evaluation metric is provided for these demonstrations, example footage is included in the accompanying video that can be seen at `https://youtu.be/sHkcVIJt2_Y`. The video also presents an end-to-end interaction involving the *Slingshot* trajectory to complete a surveillance mission.

3.3.1 Apparatus

The robot for all experiments and demonstrations is a Parrot Bebop quadrotor (Figure 3.5), which is commercially available $800 platform. The robot has a well-stabilized frontal camera with a resolution of 640x368 at 30 frames per second and high-quality onboard orientation and velocity estimation. The open-source `bebop_autonomy`[4] ROS package provides access to the onboard sensors and ability to control the robot. Due to the built-in image stabilization, the roll rotation during the *Ready* phase is almost imperceptible in the video

[4] `http://github.com/AutonomyLab/bebop_autonomy`

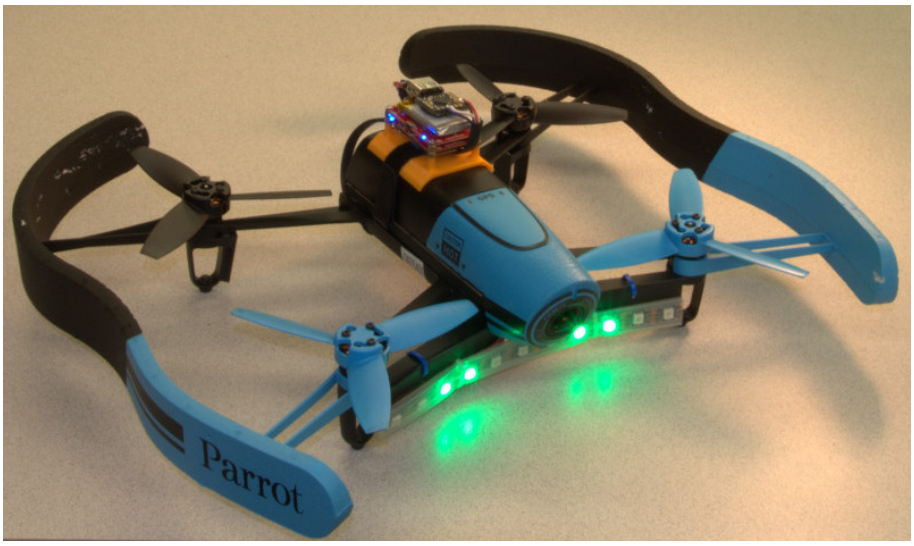

Figure 3.5: Parrot Bebop quadrotor used in our experiments, equipped with a programmable colored LED strip for visual feedback to the user.

feed, greatly simplifying the image processing and the use of orientation as a user signal. The only augmentation to the robot is the addition of a programmable colored light strip that is useful for user feedback and debugging. Specifically, the light strip indicates training progress and request for the *trigger expression* during the first phase. Video processing and control computation is performed off board over the built-in wireless network of the robot on a consumer-quality laptop with a quad-core Intel processor at 2 GHz and 8GB of RAM.

3.3.2 Experiment #1—Face Recognition

In this experiment, the security aspect of the system is evaluated by determining whether the robot would reject face command attempts from a human who was not the learned user. 20 trials were performed with each of three users. In each trial the robot would complete the training phase and then hover, and only land when presented with the correct user's face. Users stood facing the hovering robot, approximately 1.5 metres away. Half of the trials included only the learned user, and the other half had distractors attempting to command the robot to land for 30 seconds before the user entered the frame. A trial was considered to be successful if the robot accepted a command from only the authorized learned user, and a failure if it ever accepted a command from a non-trained (unauthorized) user. Success rates and time taken are reported in Table 3.1. Two failures were observed in 60 trials.

3.3.3 Experiment #2—Beam

To evaluate the straight-line trajectory, users attempted to direct the UAV through a 0.8 metre diameter hoop located eight metres from the user in our indoor motion-capture lab. Four target hoops were each located at a different bearing and azimuth. Eight trials were

Participant	Successes	Mean Time (s)	Std. dev. (s)
1	20/20	3.49	1.61
2	18/20	8.63	9.61
3	20/20	3.57	0.71
Overall	58/60	5.23	6.13

Table 3.1: Results of face recognition trials.

Participant	Successes	Mean Error (	Std. dev. (
1	6/8	12.78	9.20
*2	8/8	6.78	1.87
3	7/8	15.21	8.39
4	7/8	20.09	17.25
Overall	28/32	13.71	11.70

Table 3.2: Results of *beam* trials. The participant marked with the asterisk (*) was an expert user—the developer of the system.

performed per user, two for each target hoop. The resulting robot trajectory was measured with an external Vicon motion capture system and compared with the ideal, straight-line, trajectory. The results are reported in terms of the error in azimuth and elevation against the true angle to the target. The robot did not fly perfectly straight due to the inevitable errors of real-world robotics, so the angle error is computed based on a least-squares line through the trajectory as determined by the first principal component of a singular value decomposition on the array of positions for each trajectory.

Trials were conducted with four users. The majority of trajectories had a root-mean-square deviation of less than ten centimetres from the line of best fit. A trial was considered successful if the angle error to the target was the lowest among the four possible hoops. Trial results with errors are reported in Table 3.2 and two sample trajectories are shown relative to the four hoops in Figure 3.6.

3.3.4 Experiment #3—Slingshot

The ballistic trajectory was evaluated in an outdoor environment, by having users attempt to direct the robot to land in a 0.8 metre diameter hoop placed on the ground 18.5 metres from the robot. Users stood 1.5 metres in front of the robot. The base distance of the system was calibrated such that the largest face size would result in a flight distance of three metres, and the smallest resulting in a distance of 45 metres. On completing the ballistic trajectory, the robot lands marking the end location of the flight path. Due to the small size of the target relative to the traversal distance, a trial is considered successful if the robot landed with five metres of the center of the hoop. Five trials are conducted for each of three users. Accuracy is measured with respect to the robot's end location in metres from the center of the hoop. The robot began each trial with a yaw of 45° to the target to ensure that users

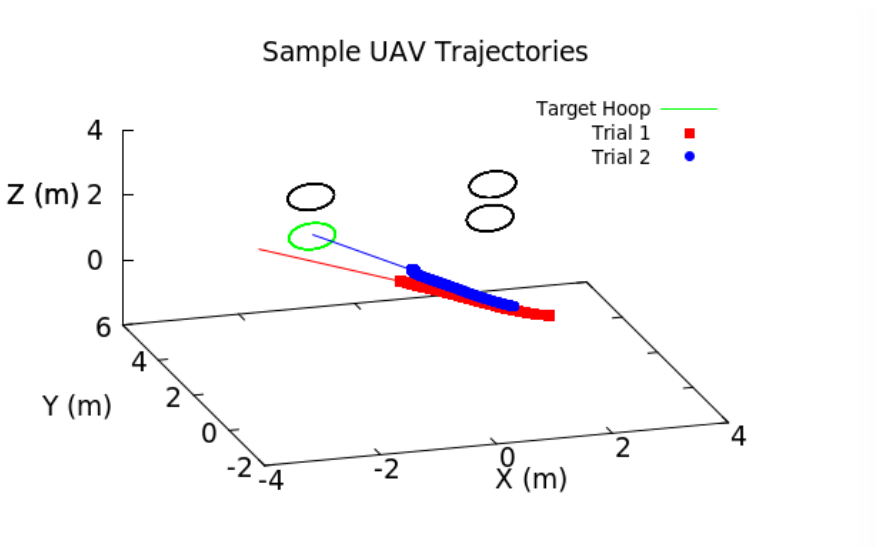

Figure 3.6: Two sample trajectories from the *beam* experiments, with the target hoop in green. Best fit lines show that the robot would have flown very close to the target, although trajectory lengths were limited in these trials for safety reasons. The trajectories shown are taken from the first two trials of Participant 1.

had to both aim the angle of the trajectory and adjust the distance. Results are shown in Table 3.3 and landing locations are shown in Figure 3.7.

Participant	Successes	Mean Error (m)	Std. dev. (m)
1	4/5	3.77	1.30
2	4/5	3.96	2.15
3	4/5	2.99	1.17
Overall	12/15	3.57	1.65

Table 3.3: Results of *slingshot* trials.

In addition to the experiment, *slingshot* trajectories were executed over distances of up to 60 metres and out of line-of-sight. Without accurate ground-truth (GPS data were not accurate enough), there is not sufficient data to provide quantitative evaluations. As an example, the system was tested in the short-range scenario depicted in Figure 3.8a with the goal of taking a photo of a target object that is occluded from the user's view. The user signals to the robot to fly a *slingshot* trajectory over a dirt pile (3.8b), the robot takes a picture (3.8c), and returns to the user.

3.4 Discussion

This chapter has presented a novel, face-based, HRI system that increases autonomy of a flying robot in the presence of humans by leveraging computer vision techniques for face recognition. The proposed system is the first that uses only face recognition to send

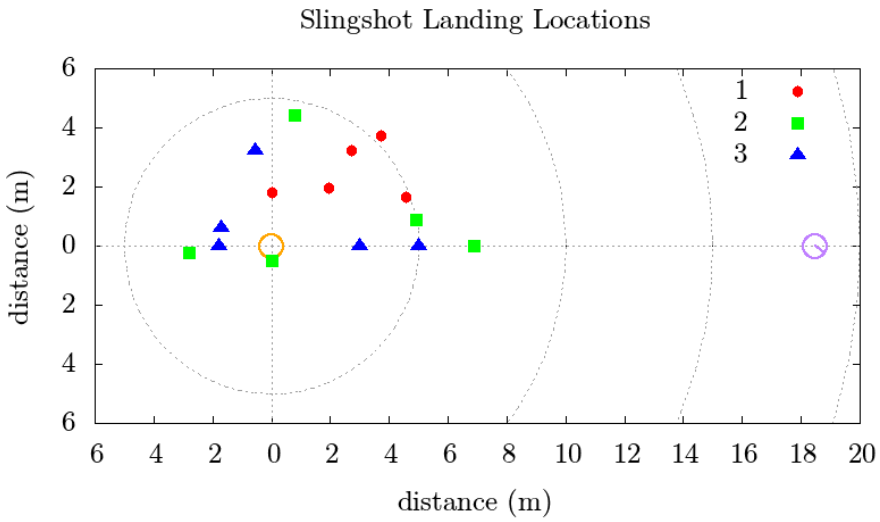

Figure 3.7: Landing locations of the robot in the *slingshot* trials. The initial pose of the robot is shown on the right in purple (note the orientation of 45° away from the target hoop), and the orange circle shows the location and diameter of the target hoop. The three different glyphs identify the different users.

commands to a flying robot. Although this work is preliminary and should be considered primarily a set of recipes for interaction with UAVs, the results are encouraging.

By training the face recognition system to 100% precision in the *Ready* phase, the false positive rate during flight was very low making it reliable for controlling and interacting with the UAV. Although false negatives can delay a successful interaction, false positives can be catastrophic due to the initiation of an arbitrary, undesired trajectory. It is also an advantage for a UAV to accept commands only from its known user since this provides a degree of security, and with only two failures out of 60, we can be relatively confident in this aspect of the system.

In the *beam* experiment, users were able to send the UAV toward the correct target hoop in 87.5% of the trials performed, with a mean angle error to the target of 13.7 (+/- 11.7) degrees. This is sufficient for positioning the robot at a coarse location in 3D space, and represents a new option for a hands-free natural interface to control a UAV for video capture or to initiate mapping. More precise control of the UAV may require better flight controllers and increased resolution in human face-pose estimation. It was also observed that expert users such as the system developer performed significantly better than untrained users, which indicates that users get better through experience with the system. This suggests

(a) Target is occluded from the user by a dirt pile.

(b) *Slingshot* trajectory commanded by user.

(c) An image acquired by the robot.

Figure 3.8: An example scenario where the user commands the robot to inspect an occluded area.

the possibility of future work evaluating the performance gains for new users during the training process.

The *slingshot* experiment shows agreement with the angular predictions of the *beam* experiment based on the spread of landing locations around the target. It demonstrates that users are able to use the slingshot metaphor to produce trajectories that land in the vicinity of the target. Given a range of possible distances from three to 45 metres, the starting distance of 18.5 metres from the target, and the lack of any precise robot-to-user feedback during the *Aim* phase, an average position error of 3.5 metres from the target is encouraging result. In only one trial, the robot came very close to landing inside of the target hoop. With the limited feedback from the onboard LEDs, hours of practice may be required before developing a sense of the distances produced (as in shooting a real slingshot), and the users in these trials had never used the system before.

For the demonstration of the *boomerang* trajectory, a statue was successfully orbited with the UAV using the interaction method described in Section 3.2. The user was able to specify an appropriate direction and size for the circular trajectory, albeit with a couple of practice attempts. During this trial and error period remote operator intervention was required to prevent UAV collisions with other objects in the environment (e.g. the statue), until the user calibrated themselves for the task. Although the original interest was in investigating the behavior of the robot, the camera data from such an orbiting behaviour would be useful creating 3D reconstructions.

While the demonstrations in this chapter have sent the robot on flights of ten metres indoors and 45 metres outdoors, these interactions scale to hundreds of metres without modification. If the UAV was able to visually servo to a target of interest after reaching the peak of its trajectory (for example another person, as described in [24]) we might be able to "throw" the UAV from one person to another over a kilometre or more.

3.5 Future Work

From the HRI perspective, it makes sense to conduct formal user study to determine whether users prefer the proposed slingshot-analogue method over other interaction methods and, as mentioned above, determine how well user performance increases with experience. It would be interesting to see how performance is affected by improving user feedback. In video games, trajectories are often indicated by overlaying a virtual arc on the screen while aiming (similar to Figure 3.1), and it seems reasonable to expect that a similar augmented-reality aspect in a system like this could improve performance by assisting the user in judging control parameters like distance and angle.

Another potential direction of future work is to further investigate the usefulness of the the proposed system in application specific domains, such as 3D mapping or surveying. For

example, the proposed system could be compared with other methods for acquiring image data from a robot for the purpose of constructing a 3D model of a moving person.

The proposed design has the UAV flying very close to the user (and potentially other obstacles), which has serious safety implications. For user safety in these experiments we rely on the inherent safety of the lightweight vehicle and a remote operator that can override the robot at a moment's notice. But, the general problem of having UAVs actively maintain safety when working closely around people is important for future work.

Other potential improvements include using a dedicated facial expression detector, as opposed to the dual-identity face recognition method, or extending the gesture vocabulary to include arms for more complex interactions. In the future, face recognition software is likely to be increasingly invariant to facial expression, but powerful software will no doubt continue to be developed for recognizing facial expressions and other complex gestures.

Chapter 4

Conclusion

This thesis has explored and demonstrated two methods for autonomous mobile robots to work effectively in dynamic scenarios with the aid of computer vision techniques. Specifically, it contains (i) the first demonstration of a multi-robot system using visual SLAM for the purpose of scanning a moving object and (ii) a novel, face-based HRI system for commanding a UAV. Both methods involve interacting directly with another moving object in the environment, which acts as a primary source of information for the robot system.

Motion planning in the multi-robot following system is primarily dictated by the location of the moving object. A visual collaborative SLAM system helps localize the moving object with respect to the robot team. Given the poses of all robots in a common frame of reference, a novel orbiting control scheme is applied in order to scan the moving object while keeping it in view under constraints of the SLAM system.

For the face-based HRI system, the main source of information for trajectory control comes from the moving human. Once trained on two facial expressions, the system uses one as a trigger, and the location of the face determines the angle and power of the flight. In addition to the robust signalling function of the facial expression system, it provides a degree of security to prevent false positives in face identity recognition from triggering behavior: the trigger expression is required before initiating motion, and this expression is known only by the primary user.

Both of the applications developed as part of this thesis are steps towards integrating autonomous mobile robots into our daily lives. They demonstrate effective techniques for robots operating in dynamic environments, though specific to the application scenario (collaborative following or human interaction with UAVs). It is not just enough to have a robot system that is robust to complex environments. Future work should aim towards building robot applications that leverage information regarding dynamic objects (e.g. being socially aware of humans) in order to operate more effectively.

Bibliography

[1] Miguel Aranda, Gonzalo López-Nicolás, Carlos Sagüés, and Michael M. Zavlanos. Three-dimensional multirobot formation control for target enclosing. In *Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ International Conference on*, pages 357–362. IEEE, 2014.

[2] Randal W Beard, Derek Kingston, Morgan Quigley, Deryl Snyder, Reed Christiansen, Walt Johnson, Timothy McLain, and Michael Goodrich. Autonomous vehicle technologies for small fixed-wing uavs. *Journal of Aerospace Computing, Information, and Communication*, 2(1):92–108, 2005.

[3] Jake Bruce, Jacob Perron, and Richard Vaughan. Ready-aim-fly! hands-free face-based hri for 3d trajectory control of UAVs. In *Computer and Robot Vision (CRV), 2017 14th Conference on*, Edmonton, AB, Canada, May 2017. IEEE.

[4] Jessica R Cauchard, LE Jane, Kevin Y Zhai, and James A Landay. Drone & Me: an exploration into natural human-drone interaction. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 361–365. ACM, 2015.

[5] Chun-Hua Chang, Shao-Chen Wang, and Chieh-Chih Wang. Vision-based cooperative simultaneous localization and tracking. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5191–5197. IEEE, 2011.

[6] Alex Couture-Beil, Richard T Vaughan, and Greg Mori. Selecting and commanding individual robots in a vision-based multi-robot system. In *Proceedings of the 5th ACM/IEEE international conference on Human-robot interaction*, pages 355–356. IEEE Press, 2010.

[7] AnhDuc Dang and Joachim Horn. Formation control of leader-following uavs to track a moving target in a dynamic environment. *Journal of Automation and Control Engineering Vol*, 3(1), 2015.

[8] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007.

[9] Vikas Dhiman, Julian Ryde, and Jason J Corso. Mutual localization: Two camera relative 6-dof pose estimation from reciprocal fiducial observation. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1347–1354. IEEE, 2013.

[10] A Dias, J Almeida, P Lima, and E Silva. Uncertainty based multi-robot cooperative triangulation. In *RoboCup Symposium Proceedings, Brasil. LNCS (LNAI). Springer*, 2014.

[11] Mark Draper, Gloria Calhoun, Heath Ruff, David Williamson, and Timothy Barry. Manual versus speech input for unmanned aerial vehicle control station operations. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 47, pages 109–113. SAGE Publications Sage CA: Los Angeles, CA, 2003.

[12] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[13] Shervin Emami and Valentin Petrut Suciu. Facial recognition using OpenCV. *Journal of Mobile, Embedded and Distributed Systems*, 4(1):38–43, 2012.

[14] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

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

[16] Christian Forster, Simon Lynen, Laurent Kneip, and Davide Scaramuzza. Collaborative monocular slam with multiple micro aerial vehicles. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3962–3970. IEEE, 2013.

[17] Antonio Franchi, Paolo Stegagno, Maurizio Di Rocco, and Giuseppe Oriolo. Distributed target localization and encirclement with a multi-robot system. In *7th IFAC Symposium on Intelligent Autonomous Vehicles*, pages 151–156, 2010.

[18] Dries Hulens and Toon Goedemé. Autonomous flying cameraman with embedded person detection and tracking while applying cinematographic rules. In *Computer and Robot Vision (CRV), 2017 14th Conference on*, Edmonton, AB, Canada, May 2017. IEEE.

[19] 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.

[20] David L Mills. Internet time synchronization: the network time protocol. *IEEE Transactions on communications*, 39(10):1482–1493, 1991.

[21] Thomas B Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2):90–126, 2006.

[22] Mani Monajjemi, Jake Bruce, Seyed Abbas Sadat, Jens Wawerla, and Richard Vaughan. UAV, Do You See Me? establishing mutual attention between an uninstrumented human and an outdoor UAV in flight. In *Intelligent Robots and Systems (under review), 2015 Int. Conf. on*. IEEE, 2015.

[23] Mani Monajjemi, Jens Wawerla, Richard Vaughan, and Greg Mori. HRI in the Sky: Creating and commanding teams of UAVs with a vision-mediated gestural interface. In *Intelligent Robots and Systems, 2013 Int. Conf. on*, pages 617–623, Nov 2013.

[24] Valiallah (Mani) Monajjemi, Sepehr MohaimenianPour, and Richard Vaughan. Uav, come to me: End-to-end, multi-scale situated hri with an uninstrumented human and a distant uav. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS'16)*, October 2016.

[25] Diluka Moratuwage, Ba-Ngu Vo, and Danwei Wang. Collaborative multi-vehicle slam with moving object tracking. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 5702–5708. IEEE, 2013.

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

[27] Wai Shan Ng and Ehud Sharlin. Collocated interaction with flying robots. In *RO-MAN, 2011 IEEE*, pages 143–149. IEEE, 2011.

[28] Jacob M. Perron, Rui Huang, Jack Thomas, Lingkang Zhang, Ping Tan, and Richard Vaughan. Orbiting a moving target with multi-robot collaborative visual slam. In *Proceedings of the 3rd Workshop on Multi VIew Geometry in RObotics (MVIGRO) at Robotics Science and Systems (RSS'15)*, Rome, Italy, July 2015.

[29] Kevin Pfeil, Seng Lee Koh, and Joseph LaViola. Exploring 3d gesture metaphors for interaction with unmanned aerial vehicles. In *Proceedings of the 2013 international conference on Intelligent user interfaces*, pages 257–266. ACM, 2013.

[30] Shokoofeh Pourmehr, Mani Monajjemi, Jens Wawerla, Richard T. Vaughan, and Greg Mori. A robust integrated system for selecting and commanding multiple mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2013.

[31] Shokoofeh Pourmehr, Valiallah (Mani) Monajjemi, Seyed Abbas Sadat, Fei Zhan, Jens Wawerla, Greg Mori, and Richard Vaughan. "You Are Green": A touch-to-name interaction in an integrated multi-modal multi-robot HRI system (late-breaking abstract). In *Proceedings of the 2014 ACM/IEEE International Conference on Human-robot Interaction*, HRI '14, pages 266–267, New York, NY, USA, 2014. ACM.

[32] Shokoofeh Pourmehr, Valiallah Mani Monajjemi, Richard T. Vaughan, and Greg Mori. "You two! Take off!": Creating, modifying and commanding groups of robots using face engagement and indirect speech in voice commands. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS'13)*, Tokyo, Japan, November 2013.

[33] Shokoofeh Pourmehr, Jack Thomas, and Richard Vaughan. What untrained people do when asked "make the robot come to you" (late-breaking abstract). In *Proceedings of the 2016 ACM/IEEE International Conference on Human-robot Interaction*, HRI '16, 2016.

[34] Charles Rich, Brett Ponsler, Aaron Holroyd, and Candace L Sidner. Recognizing engagement in human-robot interaction. In *Human-Robot Interaction (HRI), 2010 5th ACM/IEEE International Conference on*, pages 375–382. IEEE, 2010.

[35] Jianbo Shi and Carlo Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[36] Peng Yang, Randy A Freeman, and Kevin M Lynch. Distributed cooperative active sensing using consensus filters. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 405–410. IEEE, 2007.

[37] Danping Zou and Ping Tan. Coslam: Collaborative visual slam in dynamic environments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(2):354–366, 2013.