Faculty of Aerospace Engineering

Stability-based Scale Estimation of Monocular SLAM for Autonomous Quadrotor Navigation

Seong Hun Lee

August 21, 2017

TU Delft Delft University of Technology

Challenge the future

Stability-based Scale Estimation of Monocular SLAM for Autonomous Quadrotor Navigation

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering
at Delft University of Technology

Seong Hun Lee

August 21, 2017

Faculty of Aerospace Engineering · Delft University of Technology

Delft University of Technology

Delft University Of Technology
Department Of
Control and Simulation

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance a thesis entitled **"Stability-based Scale Estimation of Monocular SLAM for Autonomous Quadrotor Navigation"** by **Seong Hun Lee** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: August 21, 2017

Readers:

Dr. Guido C. H. E. de Croon

Prof. Dr. Ir. Jacco Hoekstra

Dr. Julian Kooij

Preface

This report consists of two parts: (1) the final techincal paper, and (2) preliminary report which had already been graded by my thesis supervisor.

When you compare these two works, you will find that I have made quite a few changes along the course of the project. However, the main research objective and approach did not change significantly. For background information on visual odometry/SLAM, scale estimation and stability-based methods, it may help to read the preliminary report before the technical paper.

Stability-based Scale Estimation of Monocular SLAM for Autonomous Quadrotor Navigation

Student: Seong Hun Lee
Supervisor: Guido de Croon

***Abstract*— We propose a novel method to deal with the scale ambiguity in monocular SLAM based on control stability. We analytically show that (1) using unscaled state feedback from monocular SLAM for control can lead to system instability, and (2) there is a unique linear relationship between the absolute scale of the SLAM system and the control gain at which instability arises. Using this property, our method estimates the scale by adapting the gain and detecting self-induced oscillations. Unlike conventional monocular approaches, no additional metric sensors are used for scale estimation. We demonstrate the ability of our system to estimate the scale for performing autonomous indoor navigation with a low-cost quadrotor MAV.**

I. INTRODUCTION

For a micro air vehicle (MAV) to perform autonomous flight tasks in GPS-denied environments, accurate self-localization is an important requirement. In the absence of external motion capture systems, the robot must rely solely on the measurements from its onboard sensors to localize and navigate itself with respect to the unknown surroundings. This motivates the relevance of the so-called Simultaneous Localization and Mapping (SLAM) problem.

To solve the SLAM problem on MAVs, monocular cameras are widely used as the main exteroceptive sensor [1]–[3]. This is primarily due to its small size, weight and power consumption compared to other sensors, such as laser scanners or RGB-D cameras. Also, as opposed to stereo systems, monocular approaches have the advantage that they can handle a large range of scene distances [4]. Moreover, the recent development of monocular visual odometry (VO) and SLAM methods such as [5]–[7] has provided the possibility to estimate the six degrees-of-freedom (DOF) camera pose in real time with high accuracy.

However, the main weakness of monocular SLAM is that the absolute scale of the reconstructed scene and camera motion is inherently unobservable. Without any supplementary information on the metric scale of the SLAM system, one cannot naively use the "unscaled" pose estimates to control and navigate the MAV. Traditional methods therefore employ additional metric sensors, such as an inertial measurement unit (IMU) [8] or altimeter [3] to estimate the absolute scale.

In this work, we propose a novel, stability-based, adaptive control strategy to achieve autonomous flight of quadrotor MAVs using only monocular SLAM. Inspired by the previous work [9], our method estimates the absolute scale of the SLAM system by inducing and detecting vertical oscillations in hover. We implement this in an adaptive control framework which allows us to effectively manipulate the MAV's vertical motion, while simultaneously stabilizing its horizontal position. Once the adaptive scale estimation is completed, we show that various autonomous flight tasks such as hovering and waypoint following can be performed.

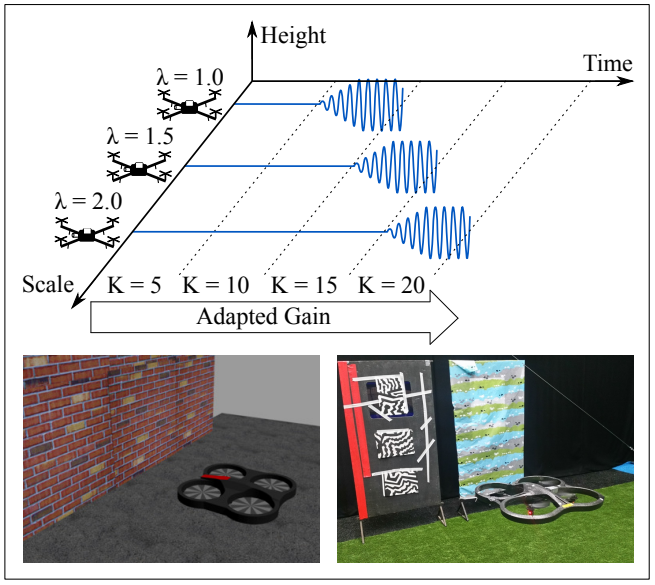


Fig. 1: We propose to solve the scale ambiguity of monocular SLAM using the control instability that arises when applying a certain gain to the unscaled state feedback from the SLAM system. **Top:** Example illustration of the linear relationship between the absolute scale λ and control gain K at which the system becomes unstable. **Bottom:** We verify the proposed method in simulation (left) and in real-world experiments (right).

The main contributions of this work are two-fold: First, we analytically show that when a quadrotor autopilot directly controls the unscaled velocity (or position) estimates from a monocular SLAM system, there is a unique linear relationship between the absolute scale of the SLAM system and the control gain at which instability arises. An example is illustrated at the top of Fig. 1. Second, we provide an adaptive technique to estimate the scale of a monocular SLAM system based on the hover stability of a quadrotor. Our method does not rely on additional sensor modalities, such as IMU, depth sensors, or altimeters. We demonstrate the proposed system both in simulation and on a real quadrotor platform using

The authors are with Micro Air Vehicle Laboratory of Faculty of Aerospace Engineering, Delft University of Technology, 2629HS Delft, The Netherlands (e-mail: sunghoon315@gmail.com; G.C.H.E.deCroon@tudelft.nl)

a Parrot AR.Drone[1] (see the bottom of Fig. 1). A video demonstration is available at:

https://youtu.be/SM17L8PD5Oc

II. RELATED WORK

When using monocular visual odometry (VO) or SLAM for MAV navigation, conventional approaches estimate the absolute scale by acquiring additional metric information from either range or IMU measurements.

1) Using Range Measurements: In this case, the distance between the robot and its surrounding environment is directly measured by a range sensor, such as a laser scanner [10] or RGB-D camera [11]. In [3], a closed-form solution is proposed to estimate the absolute scale using an ultrasonic altimeter of a Parrot AR.Drone. The two main drawbacks of this method are: (1) its applicability is limited to those MAVs equipped with onboard altimeters, and (2) it assumes a flat ground surface. These drawbacks do not apply to our method, as we do not rely on range sensor measurements for scale estimation.

2) Using IMU measurements: This approach is called *visual-inertial fusion*, as visual and inertial measurements are fused together to estimate the camera motion and scene structure in a metric scale. Accelerometer measurements provide the metric information about the robot's motion. State-of-the-art examples consist of filter-based [12], [13] and optimization-based [14]–[16] approaches. Although these methods have been shown to achieve very high accuracy and consistency, it is often necessary to provide appropriate initial states and accurate multi-sensor calibration to avoid the convergence towards suboptimal local minima [17]. Our method requires neither complex multi-sensor calibrations nor carefully tailored initialization.

In contrast to these passive, sensor-based approaches, a stability-based method retrieves metric information from the active control response to a scale-ambiguous visual input, such as optical flow divergence. Recently, one of the authors proposed a stability-based approach to estimate the height of a quadrotor MAV using a downward-looking onboard camera [9]. This method exploits a linear relationship between the height and control gain at which the adaptive divergence control manifests self-induced oscillations. In this work, we extend this idea to a scale estimation problem for monocular SLAM. The main difference from [9] is that our method does not use optical flow divergence for control, but instead uses unscaled pose estimates from a monocular SLAM system. We achieve this by (1) deriving similar stability proofs as in [9] for monocular SLAM-based control systems, (2) modifying the oscillation detection algorithm, and (3) devising a full adaptive control system for scale estimation.

The remainder of the paper is organized as follows: In Section III, we investigate the stability properties of quadrotor control using monocular SLAM. Subsequently, we detail the proposed adaptive control strategy for scale estimation in Section IV. Section V discusses the results of the scale estimation (Section V-A) and real-world flight experiments (Section V-B). Finally, conclusions are drawn in Section VI.

[1] https://www.parrot.com/fr/drones/parrot-ardrone-20-elite-edition

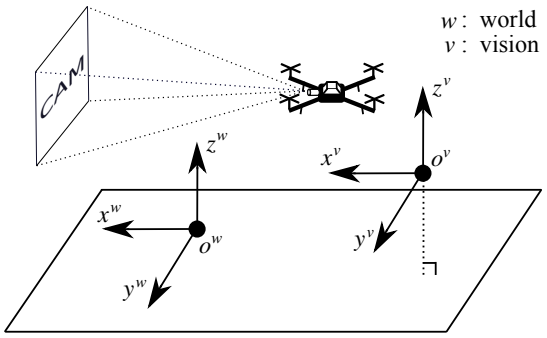


Fig. 2: World $(o^w x^w y^w z^w)$ and vision $(o^v x^v y^v z^v)$ reference frames.

III. INSTABILITY OF VERTICAL CONTROL SYSTEM

A. Discretized Model for Velocity Control

To study the instability phenomenon, it is sufficient to use a simple one-dimensional model describing the robot's vertical motion with double-integrator dynamics. Consider the following state space model and an output of unscaled vertical velocity from a monocular SLAM system:

$$\begin{aligned}\dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + B\mathbf{u}(t) \\ &= \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\mathbf{x}(t) + \begin{bmatrix}0\\ 1/m\end{bmatrix}u_z(t),\end{aligned} \tag{1}$$

$$\begin{aligned}\mathbf{y}(t) &= C\mathbf{x}(t) + D\mathbf{u}(t) \\ &= \begin{bmatrix}0 & 1/\lambda\end{bmatrix}\mathbf{x}(t),\end{aligned} \tag{2}$$

where $\mathbf{x}(t) = [Z^w(t),\ V_z{}^w(t)]^\top$, $\mathbf{y}(t) = V_z{}^v(t) = V_z{}^w(t)/\lambda$, m is the mass of the robot, $u_z(t)$ is the total thrust command, and λ is the unknown true scale of the monocular SLAM system. We use Z and V_z to denote the height and vertical velocity of the robot, and superscripts w and v to indicate the reference frames pertaining to a fixed world and visual SLAM system, respectively. This is illustrated in Fig. 2.

Discretizing (1) and (2) with zero-order hold (ZOH) yields:

$$\begin{aligned}A_d &= \begin{bmatrix}1 & T_s\\ 0 & 1\end{bmatrix}, \quad B_d = \begin{bmatrix}T_s^2/(2m)\\ T_s/m\end{bmatrix}, \\ C_d &= \begin{bmatrix}0 & 1/\lambda\end{bmatrix},\ D_d = [0],\end{aligned} \tag{3}$$

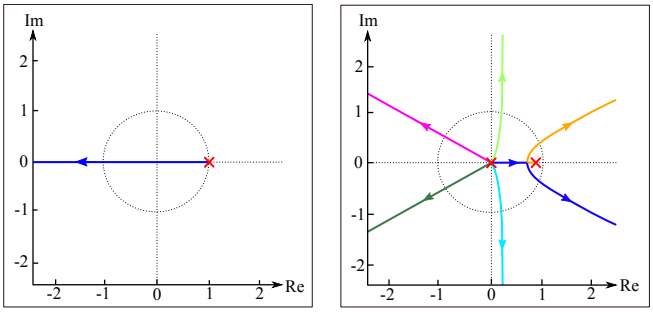
where T_s is the sample time used for discretization. This model is equivalent to the following transfer function:

$$\begin{aligned}G(\sigma) &= C_d(\sigma I - A_d)^{-1}B_d \\ &= \frac{T_s}{\lambda m(\sigma - 1)},\end{aligned} \tag{4}$$

where we use σ as the Z-transform variable instead of z to avoid confusion with the height Z.

Now, consider a simple proportional gain controller:

$$u_z = K\left(V_z{}^* - V_z{}^v\right), \tag{5}$$



(a) (b)

Fig. 3: (a) Root locus plot of the ZOH-model in a vacuum environment without noise and delay. (b) Root locus plot of the ZOH-model including drag and delay ($N = 5$).

where V_z^* is the desired vertical velocity in the vision frame. Then, the closed-loop transfer function for (4) is given as:

$$H(\sigma) = \frac{KT_s}{\lambda m(\sigma - 1) + KT_s}. \tag{6}$$

Equation (6) indicates that $H(\sigma)$ has a single pole located at $\sigma = 1$ when $K = 0$ and a negative infinite zero. This is shown in the root locus plot in Fig. 3*a*. As the gain increases, the pole moves towards the negative infinite zero along the real axis, and the system becomes unstable as soon as it crosses the unit circle at $\sigma = -1$. The control gain that induces such transition in the system's stability characteristics is called a *critical gain*. Setting $\sigma = -1$ and equating the denominator of (6) to 0, the critical gain can be found as:

$$K_{cr} = \frac{2m}{T_s}\lambda. \tag{7}$$

This implies that given a specific mass m and sample time T_s, there always exists a unique positive linear relationship between the critical gain and absolute scale of the monocular SLAM system.

B. Including Drag and Delay

The ZOH-model in the previous section can be extended to account for the aerodynamic drag and time delay (e.g., caused by visual processing and communication). Including drag, the acceleration of the robot can be modeled as:

$$\dot{V}_z{}^w = \frac{u_z + f_D}{m}, \tag{8}$$

with the drag force f_D:

$$f_D = \mathrm{sgn}(V_{\text{wind}} - V_z{}^w)\frac{1}{2}\rho C_D A\,(V_{\text{wind}} - V_z{}^w)^2, \tag{9}$$

where ρ is the air density, C_D is the drag coefficient, and A is the reference area of the robot. Linearization of (8) gives:

$$\Delta\dot{V}_z{}^w = \frac{\Delta u_z}{m} - \mathrm{sgn}(V_{\text{wind}} - V_z{}^w)\frac{\rho C_D A}{m}\,(V_{\text{wind}} - V_z{}^w)\,\Delta V_z{}^w, \tag{10}$$

where V_{wind} and $V_z{}^w$ are given the values at the linearization point. Hereafter, a constant p will be used to avoid cluttering the formulas, which is defined as:

$$p := \mathrm{sgn}(V_{\text{wind}} - V_z{}^w)\frac{\rho C_D A}{m}\,(V_{\text{wind}} - V_z{}^w). \tag{11}$$

As a result, the linearized equation of motion including the drag gives the following continuous state space model:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & -p \end{bmatrix},\ B = \begin{bmatrix} 0 \\ 1/m \end{bmatrix},\\ C = \begin{bmatrix} 0 & 1/\lambda \end{bmatrix},\ D = [0]. \tag{12}$$

Discretization of this model using ZOH with a sample time T_s leads to:

$$A_d = \begin{bmatrix} 1 & \frac{\left(1 - e^{-pT_s}\right)}{p} \\ 0 & e^{-pT_s} \end{bmatrix},\ B_d = \begin{bmatrix} \frac{T_s}{mp} - \frac{\left(1 - e^{-pT_s}\right)}{mp^2} \\ \frac{\left(1 - e^{-pT_s}\right)}{mp} \end{bmatrix},\\ C_d = \begin{bmatrix} 0 & 1/\lambda \end{bmatrix}, \qquad D_d = [0], \tag{13}$$

which is equivalent to the following transfer function:

$$G(\sigma) = \frac{1 - e^{-pT_s}}{\lambda mp(\sigma - e^{-pT_s})}. \tag{14}$$

Now, introducing a delay of N sampling periods in (14) gives:

$$G(\sigma) = \frac{1 - e^{-pT_s}}{\lambda mp\sigma^N(\sigma - e^{-pT_s})}. \tag{15}$$

The closed-loop transfer function with a proportional control is thus given as:

$$H(\sigma) = \frac{K\left(1 - e^{-pT_s}\right)}{\lambda mp\sigma^N(\sigma - e^{-pT_s}) + K\left(1 - e^{-pT_s}\right)}. \tag{16}$$

Equation (16) shows that, for $K = 0$, the system has N poles at $\sigma = 0$ and one additional pole at $\sigma = e^{-pT_s}$. This later pole is located on the positive part of the real axis inside the unit circle, as $p > 0$ and $T_s > 0$. Also, there are $N + 1$ zeros at equiangular infinities, creating asymptotes $2\pi/(N + 1)$ radians apart from each neighbor. An example of a root locus plot with $N = 5$ is shown in Fig. 3*b*.

Using the same procedure as in the previous section, the critical gain K_{cr} can be found by solving the following two equations:

$$\sigma = \cos\theta + j\sin\theta = e^{j\theta} \tag{17}$$

and

$$\lambda mp\sigma^N\left(\sigma - e^{-pT_s}\right) + K_{cr}\left(1 - e^{-pT_s}\right) = 0, \tag{18}$$

where θ is the phase of the pole that intersects the perimeter of the unit circle. Given the parameters m, p and T_s, we can solve these two equations for the two unknowns θ and K_{cr}. Equation (17) and (18) can be rearranged to give:

$$e^{-pT_s} = \frac{\sin\left((N + 1)\,\theta\right)}{\sin\left(N\theta\right)} \tag{19}$$

and

$$K_{cr} = \frac{\lambda mp\ \sin\theta}{\left(1 - e^{-pT_s}\right)\sin\left(N\theta\right)}. \tag{20}$$

The detailed derivations of (19) and (20) are provided in Appendix A. The computational analysis solving θ and K_{cr} from (19) and (20) gives rise to the following observations:

1) While there exist multiple possible solutions (θ, K_{cr}) that suffice (19) and (20) simultaneously, the poles that cross the unit circle for the first time are those with the smallest magnitude of θ. These poles uniquely determine the critical gain K_{cr}.
2) Although (11) and (20) suggest that K_{cr} depends on the parameter p which is a function of ${V_z}^w$ at the linearization point, the sensitivity of K_{cr} to different values of ${V_z}^w$ is almost negligible in hovering condition. For example, the maximum change in K_{cr} is less than 0.1% given $N < 50$ and $|V_{\text{wind}} - {V_z}^w| < 2m/s$ with $m = 1\text{kg}$, $T_s = 0.01\text{s}$, and $\frac{1}{2}C_D A = 0.5$. This effect weakens as N is decreased.
3) K_{cr} decreases with an increase in time delay N.

The first two observations allow us to represent K_{cr} in (20) as:

$$\begin{aligned} K_{cr} &= \lambda f(m, T_s, N, p({V_z}^w, \rho, C_D, A, m), \theta(p, T_s, N)) \\ &= \lambda g(m, T_s, N, {V_z}^w, \rho, C_D, A) \\ &\approx \lambda h(m, T_s, N, \rho, C_D, A) \end{aligned} \tag{21}$$

where $f(\cdot)$, $g(\cdot)$ and $h(\cdot)$ are functions that take different set of input to compute K_{cr}/λ. Since the parameters of $h(\cdot)$ are assumed to be fixed, (21) can be written as:

$$K_{cr} \approx \alpha\lambda, \tag{22}$$

where α is a positive constant. Similarly to the previous case, this suggests that a unique linear relationship exists between the critical gain and scale.

C. Height Control Using Velocity Commands

Some quadrotor systems allow users to manually control the flight using velocity commands. In this case, we may also design the vertical control system to incorporate velocity commands instead of thrust input. Such control systems exhibit similar type of self-induced oscillations when regulating the height. We provide a rudimentary explanation for this behavior using a simplified model in Appendix B.

IV. ADAPTIVE CONTROL STRATEGY FOR SCALE ESTIMATION

Motivated by the unique linear relationship between the critical gain and absolute scale of the SLAM system as suggested in the previous section, we propose an adaptive control framework that estimates the scale and enables autonomous quadrotor navigation. Our framework builds on two autopilot modes: adaptive control mode and Ready-to-Fly (RTF) mode.

Fig. 4 illustrates the pipeline of the adaptive control mode. Its main function is to estimate the scale of the SLAM system by adapting the gain K iteratively towards the critical gain and detecting the self-induced oscillations. To minimize the horizontal drift during the process, we apply PID control on the unscaled estimates of the horizontal position. The resultant horizontal commands are then scaled by the latest scale estimate λ' to compensate the unscaled input.

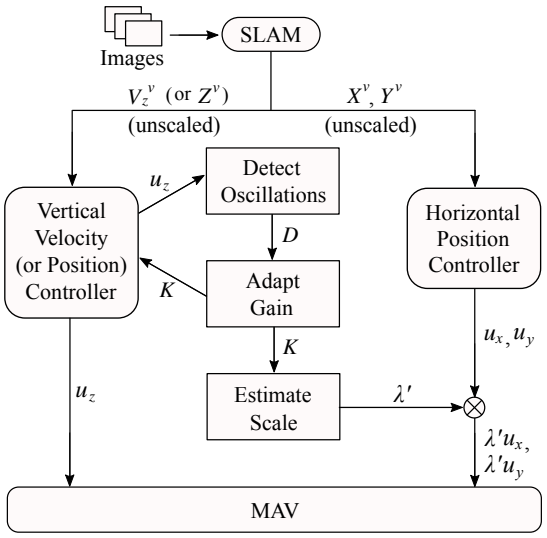


Fig. 4: Adaptive control pipeline for scale estimation.

Once the critical gain is found, the final scale is estimated using (22) and the autopilot switches to RTF mode. In this mode, we directly scale the position and velocity estimates from the SLAM system to perform autonomous flight tasks. Note that the heading control is independent of the scale, and it is thus identical in both adaptive and RTF modes.

In the following, the three main components of the adaptive control method are explained: detecting oscillations, adapting the gain, and controlling the horizontal position during the process.

A. Detection of Self-induced Oscillations

Traditional methods to detect self-induced oscillations typically use a fast Fourier transform (FFT) to identify the specific resonant frequency of the system [18] [19]. In this section, we propose a more computationally efficient method to detect self-induced oscillations. First, we define a heuristic variable D to quantify and detect oscillations. Specifically, we compute the windowed maximum of the average total variation in vertical velocity. For continuous systems, let t_0 be the beginning of the main time window, T_w the length of the sub-window for averaging the total variation, and n the maximum number of the sub-windows allowed before restarting the main window and updating t_0. At time t given $t_0 \le t \le t_0 + nT_w$, we define D as:

$$D(t, t_0, T_w) := \max_{t' \in [t_0, t]} \frac{\int_{t'-T_w}^{t'} \left|{a_z}^w(\gamma)\right| d\gamma}{T_w}, \tag{23}$$

where ${a_z}^w$ is the vertical acceleration in the world frame.

If the drag force is assumed to be relatively small, (23) can be approximated as:

$$D(t, t_0, T_w) \approx \frac{\max_{t' \in [t_0, t]} \int_{t'-T_w}^{t'} \left|u_z(\gamma)\right| d\gamma}{mT_w}. \tag{24}$$

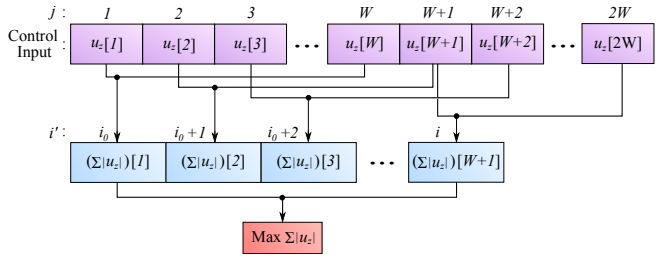


Fig. 5: Computing the discrete detection variable $D(i,i_0,W)$ with $n=2$, $i_0=W$ and $i=nW$. For each W past control input, we compute $\sum|u_z|$ and set its maximum value in the current window as the numerator of (25).

For discrete systems, this becomes:

$$D(i,i_0,W) = \frac{\max\limits_{i'\in\{i_0,\cdots,i\}} \sum\limits_{j=i'-W+1}^{i'} |u_z[j]|T_s}{mWT_s} = \frac{\max\limits_{i'\in\{i_0,\cdots,i\}} \sum\limits_{j=i'-W+1}^{i'} |u_z[j]|}{mW}, \quad (25)$$

given $i_0 \le i \le i_0+nW$ where W is the discrete window size, which is equal to T_w/T_s. This process is illustrated in Fig. 5. The onset of oscillations is detected as soon as D exceeds a preset threshold D_{thr}.

So far, we have assumed that the system controls vertical velocity using thrust commands as input. For height control systems using velocity commands (as in Section III-C), we need to modify (23) by reducing its order by one:

$$D(t,t_0,T_w) := \max_{t'\in[t_0,t]} \frac{\int_{t'-T_w}^{t'} |V_z{}^w(\gamma)|d\gamma}{T_w}, \quad (26)$$

$$\approx \max_{t'\in[t_0,t]} \frac{\int_{t'-T_w}^{t'} |u_z(\gamma)|d\gamma}{T_w}, \quad (27)$$

For discrete systems, this is equivalent to:

$$D(i,i_0,W) = \frac{\max\limits_{i'\in\{i_0,\cdots,i\}} \sum\limits_{j=i'-W+1}^{i'} |u_z[j]|}{W}. \quad (28)$$

Note that this is the same computation as (25) but without mass.

B. Adaptive Gain Control

In the following, we propose a two-step process to adaptively control the gain towards the critical gain with a good trade-off between accuracy and estimation speed. The idea is first to increase the gain adaptively until the onset of oscillations is detected (i.e., $D > D_{thr}$) and then to decrease the gain slowly until the system is considered stable again (i.e., $D < D_{thr}$). The critical gain is then estimated as the average of the two gains at the end of each process.

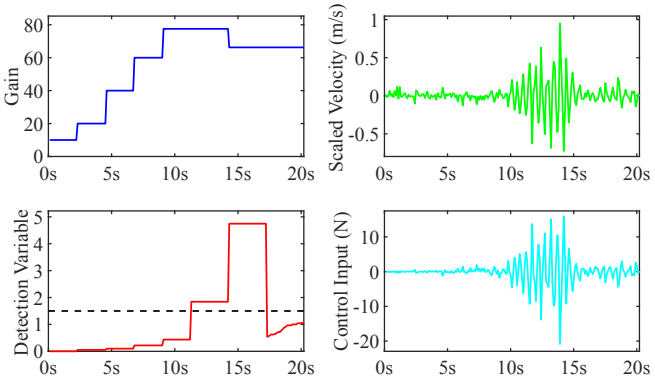


Fig. 6: Critical gain estimation with $D_{thr}=1.5$.

During the first step, the gain is increased using the following adaptive control:

$$K_{i+1} = K_i + PK_i\frac{(D_{thr}-D_i)}{D_i}, \quad (29)$$

where P is a proportional gain. This is slightly different from the method proposed in [9], as we divide the increment term by D_i. This modification was found to boost the convergence speed while also reducing the overshoot. Once the gain is adapted, the system waits for some time to induce sufficient response, and the next adaptation takes place if the oscillations are not detected until $t \ge t_0+nT_w$. Fig. 6 plots a time evolution of the relevant variables in simulation.

C. Pseudo-scale for Horizontal Position Control

As the system adapts the gain to find the critical gain, we can also recurrently compute the intermediate estimate of the scale at each iteration i using K_i from (29). This is done by introducing a new variable called a *pseudo-scale* λ':

$$\lambda'_i := \frac{K_i}{\alpha}, \quad (30)$$

where α is the constant in (22) which is to be calibrated empirically. Note that, ideally, the pseudo-scale λ' approaches to true scale λ as the adapted gain K approaches to K_{cr}.

As shown in Fig. 4, each updated λ' is used to scale the horizontal commands (u_x,u_y) computed by the horizontal position controller. This is because the controller takes unscaled translational states (X^v,Y^v) as input, and as a result, the commands (u_x,u_y) are also unscaled. Although at first the MAV may start drifting as the initial estimates of λ' are much smaller than λ, it quickly stabilizes in the horizontal position after a few iterations.

V. EXPERIMENTS AND RESULTS

We conducted a series of experiments in both simulation and the real world to validate the proposed approach. We discuss the results in two parts: In Section V-A, we verify the proposed scale estimation method, and investigate the influence of the scene distance. In Section V-B, we evaluate the performance of our system's autonomous flights.

We used *tum_simulator* [20] for simulations, and *ardrone_autonomy* [21] to operate a real AR.Drone. Note

that *ardrone_autonomy* is a ROS package based on official AR.Drone SDK, so it uses velocity commands as control input. On the other hand, *tum_simulator* can use thrust commands. Therefore, we implemented, respectively, velocity control for simulations (see Section III-A and III-B) and height control for the real drone (see Section III-C).

We used monocular ORB-SLAM [5] in both simulation and real-world experiments. Note that our method can be used with any other monocular SLAM system, as the visual pose estimation is treated as a black box. In the experiments, all computations were performed off-board on a mobile workstation laptop, and control commands were transmitted to the drone via WiFi connection. The ground-truth flight trajectories were tracked by an OptiTrack[2] motion capture system for comparison purposes only.

A. Scale Estimation

1) Critical Gain vs Scale: Using the proposed scale estimation method, we ran multiple tests to verify the linear relationship between the critical gain and absolute scale of the SLAM system as suggested in Section III. We used well-textured planar walls (see Fig. 1), and kept a fixed distance of 2 m between the wall and MAV to ensure that ORB-SLAM provides a consistent level of accuracy. We gathered 180 data points from simulation and 60 from real-world experiments. The results are shown in Fig. 7. Linear regression with zero intercept gives R^2 values of 0.96 (simulation) and 0.91 (experiments). This clearly indicates that there is a unique linear relationship between the critical gain and absolute scale of the SLAM system.

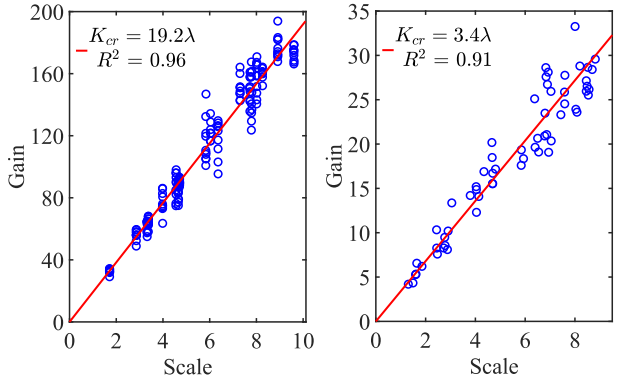


Fig. 7: Positive linear relationship between the critical gain K and scale λ verified with simulation (left) and a real quadrotor (right).

2) Critical Gain vs Scene Distance: Our model in Section III predicts that given a fixed scale, the critical gain should be constant across different scene distances. In reality, however, this is not necessarily the case. Fig. 8 plots the estimated critical gains when the MAV is made to hover at different distances from a planar scene. In the top plot, the simulation using velocity control clearly exhibits the tendency that the larger the scene distance, the lower the critical gain. This leads to an increased scale estimation error when subjected to a wide range of scene distances, as shown in the middle plot.

[2] https://optitrack.com/

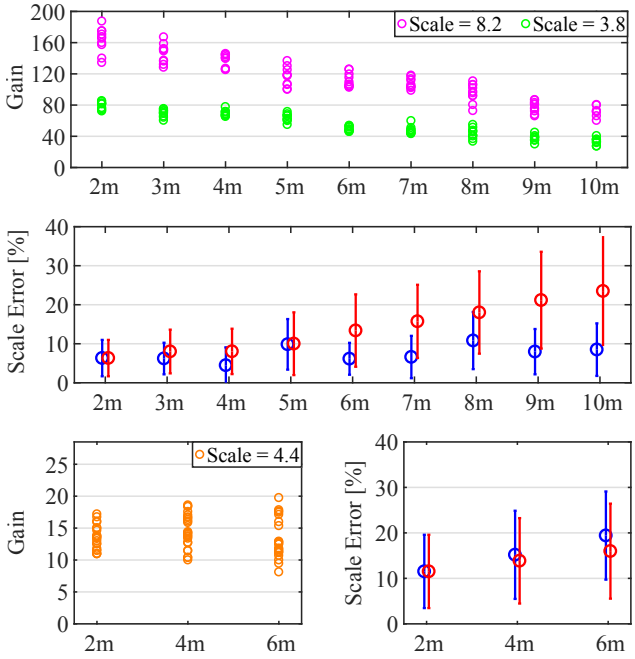


Fig. 8: **Top:** Simulation of critical gain estimation at different scene distances using velocity control. **Middle:** Scale error computed by evaluating the results at each individual scene distance (blue) or combining those less than or equal to each distance (red). We used both magenta and green data points to compute the error bars. **Bottom row:** Real-world experiment results using height control.

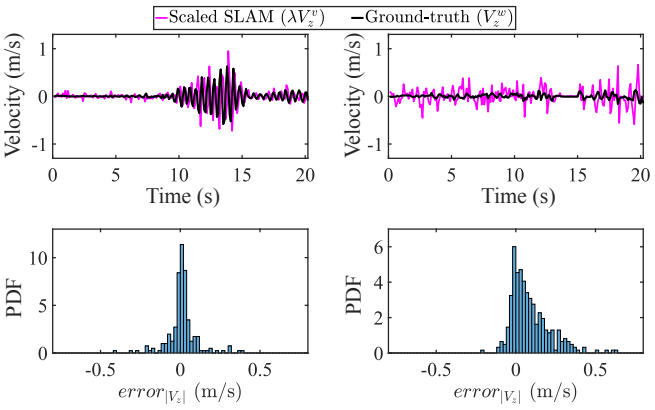


Fig. 9: **Top:** Estimated vertical velocity and ground-truth at the scene distance of 2m (left) and 10m (right). **Bottom:** Probability density function of the difference between the corresponding absolute velocities.

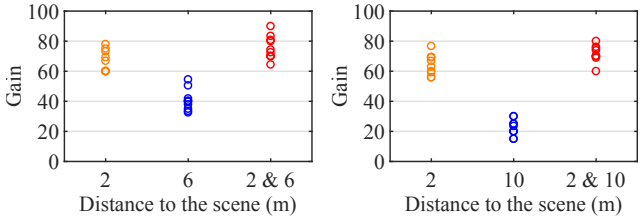
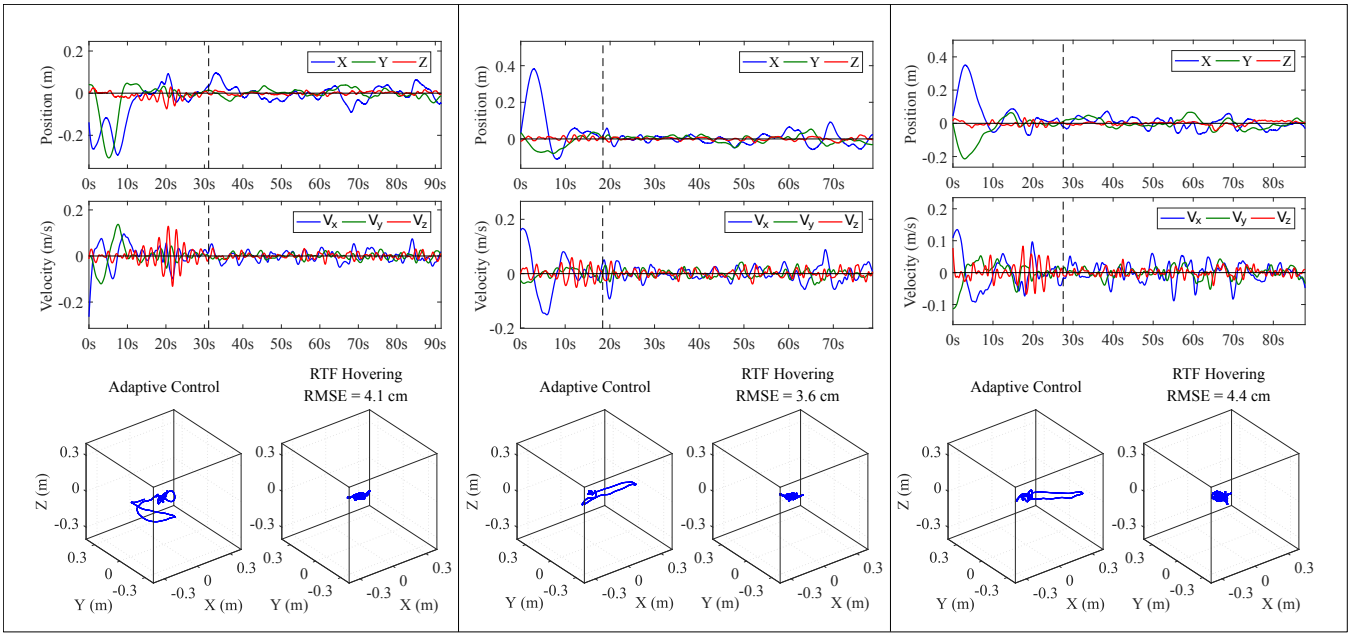


Fig. 10: Comparison of the estimated critical gains when subjected to planar (orange and blue) or nonplanar (red) scene structures.

Interestingly, the real-world experiments using height control displayed lower sensitivity to different scene distances at the cost of generally higher scale estimation error. This is shown in the bottom rows of Fig. 8. Within the scene distance



(a) Scene Distance = 2 m (b) Scene Distance = 4 m (c) Scene Distance = 6 m

Fig. 11: **Top two rows:** Time evolution of the MAV's position and velocity during the adaptive control followed by 60 seconds of position hold in RTF mode. The vertical dashed line marks the transition between the two modes. **Bottom row:** Ground-truth trajectories for the adaptive control (left) and RTF hovering (right).

between 2 m and 6 m, the average scale estimation error is estimated to be 13.4% and 16.0% in simulation and the real world, respectively.

The correlation between the critical gain and scene distance can be explained by the fact that ORB-SLAM overestimates velocity as its pose estimation accuracy decreases with larger scene distances. In Fig. 9, we compare the estimated velocities V_z^v from ORB-SLAM and the ground-truth V_z^w using two sample runs of simulation at 2 m and 10 m scene distance. We additionally scaled V_z^v by the true scale λ for comparison in metric scale. In contrast to the relatively accurate estimation at 2 m, considerable amount of noisy overestimation is observed at 10 m. This leads to a premature detection of oscillations, thus a lower critical gain. For quantitative comparison, the bottom plots of Fig. 9 present the probability density functions (PDF) of the observed difference between the two absolute velocities:

$$error_{|V_z|} = |\lambda V_z^v(t)| - |V_z^w(t - t_d)|, \tag{31}$$

where t_d is the time delay between the two signals. Note that the PDF at 10 m has a significantly heavier right tail, which indicates that the magnitude of velocity is mostly overestimated. Bootstrap two-sample test further confirmed that the two PDFs do not come from the same distribution.

Although such influence of the scene distance on the critical gain may be seen as an obstacle to accurate scale estimation, it can act as an advantage in terms of flight stability and control. As shown in Fig. 9, using the correct scale at a large scene distance would render the system unstable because of the amplified noise in the observed velocity. Since our adaptive control scheme takes this into account by definition, it provides adequately lower estimates of the critical gain and pseudo-scale at larger scene distances.

So far, we have considered only planar scenes. Yet typical scenes are nonplanar and contain objects at varying distances. To reveal the effect of nonplanar scenes, we conducted another set of simulations where we placed a small object between the camera and the wall, such that ORB-SLAM could track a small number of feature points (less than 20) at a closer distance (2 m) while the rest of the tracking points are located at either 6 m or 10 m distance. As shown in Fig. 10, the results suggest that the smallest scene distance has the dominant effect on the estimation of critical gain. This implies that having at least a few close tracking points during the adaptive process can mitigate the effect of large scene distances and reduce the scale estimation error.

B. Real-world Autonomous Flights

1) Hovering: Fig. 11 shows the ground-truth trajectories of the quadrotor MAV as it undergoes the adaptive control process followed by position hold. To explicitly show the pose stabilization effect, we initially made the drone drift before activating the adaptive control mode. The results show that our system can quickly correct the initial drift within the first 10–20 seconds and maintain a stable hover afterwards. Similar levels of convergence speed and flight stability were observed across the scene distances between 2 m and 6 m: scale estimation time around 20–30 seconds and root mean squared error (RMSE) between 3.6 cm and 4.4 cm during the RTF hovering. In terms of flight stability, this is comparable to the result reported in [3] (i.e., RMSE between 4.9 cm and 7.8 cm for indoor flight). Note that in both [3] and this work, Parrot AR.Drone is used as a platform. However, the

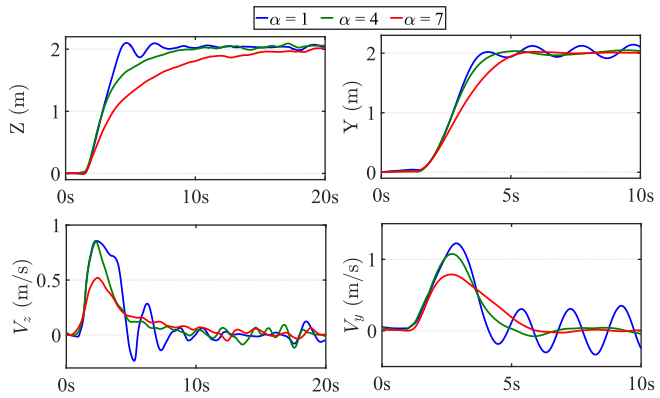


Fig. 12: Ground-truth states for single waypoint-following tasks in a vertical (left column) or horizontal (right column) direction.

TABLE I: Average convergence time and peak speed in position control

Setpoint position (m) (x, y, z)	$(0, 0, 2)$	$(0, 2, 0)$
Convergence time (s)	5.4 ± 0.9	2.7 ± 0.2
Peak speed (m/s)	0.86 ± 0.05	1.13 ± 0.05

main difference is that our method does not use altimeter measurements for scale estimation.

2) Waypoint Following: To evaluate the flight performance, we made the MAV fly to a waypoint located at 2 m distance from the initial hover position. This was done in either vertical or horizontal direction. Note that using inaccurate values of the calibration parameter α in (30) can either under- or overestimate the actual scale λ, degrading the control performance. Fig. 12 shows the ground-truth states for three example runs of each vertical and horizontal flights with α manually set to 1, 4 and 7. We can observe that setting $\alpha = 4$ gave the best performance in both vertical and horizontal flights. This is not surprising, since the correct value for α is given as 3.4 in Fig. 7. With $\alpha = 1$, the pseudo-scale λ' overestimated the actual scale λ, which resulted in overshoot and instability. On the other hand, setting $\alpha = 7$ underestimated the scale, which led to a slower convergence speed. We repeated the same task 15 times with $\alpha = 4$ and summarized the average convergence time and peak flight speed in Tab. I. The convergence was considered to be reached when the Euclidean distance between the MAV and waypoint fell below 10 cm.

3) Figure Flying: We demonstrate that our system can fly simple figures, similar to that demonstrated in [22]. Fig. 13 shows two example flights consisting of 11 successive waypoint following tasks. We estimated the flight accuracy by measuring the RMSE of the ground-truth trajectory with respect to the desired path set by the nearest waypoints. The results are given in the figure.

VI. CONCLUSIONS

In this work, we have presented a stability-based method to estimate the absolute scale of a monocular SLAM system on a hovering quadrotor MAV. Theoretical analysis of

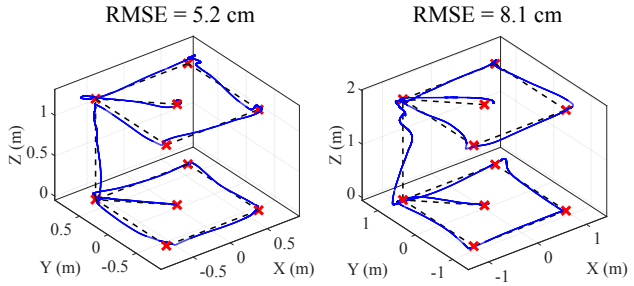


Fig. 13: Ground-truth trajectories for figure flying. **Left:** Small figure (1m × 1m × 1m). **Right:** Large figure (2m × 2m × 2m).

discretized quadrotor models has revealed a unique linear relationship between the scale of the SLAM system and control gain at which instability arises. Our method exploits this property and estimates the scale by adaptively inducing and detecting vertical oscillations in hover. We have demonstrated that our system is able to (1) estimate the scale without relying on additional metric sensors, and (2) perform various autonomous navigation tasks in unknown indoor environments.

Although our scale estimation method is not as accurate as the state-of-the-art approaches with additional metric sensors, we believe that it is definitely competitive (if not better) for achieving autonomous control using monocular SLAM. Future work may compare the control performance of our system to that of sensor-based approaches at varying scene distances.

APPENDIX

A. Derivation of (19) and (20

Let us start by substituting (17) into (18):

$$\begin{aligned}
&\lambda mp\left(e^{j\theta}\right)^N\left(e^{j\theta}-e^{-pT}\right)+K_w\left(1-e^{-pT}\right)=0,\\
&\lambda mp\left(e^{j(N+1)\theta}-e^{jN\theta}e^{-pT}\right)+K_w\left(1-e^{-pT}\right)=0,\\
&\lambda mp\left[\cos\left((N+1)\,\theta\right)+j\sin\left((N+1)\,\theta\right)\right.\\
&\left.-e^{-pT}\left(\cos\left(N\theta\right)+j\sin\left(N\theta\right)\right)\right]+K_w\left(1-e^{-pT}\right)=0.
\end{aligned} \tag{32}$$

The imaginary part of (32) gives:

$$\lambda mp\left[\sin\left((N+1)\,\theta\right)-e^{-pT}\sin\left(N\theta\right)\right]=0,$$

$$e^{-pT}=\frac{\sin\left((N+1)\,\theta\right)}{\sin\left(N\theta\right)},$$

which confirms (19).

Likewise, the real part of (32) gives:

$$\lambda mp\left[\cos\left((N+1)\,\theta\right)-e^{-pT}\cos\left(N\theta\right)\right]+K_w\left(1-e^{-pT}\right)=0,$$

which can be rearranged into:

$$K_w=-\frac{\lambda mp}{\left(1-e^{-pT}\right)}\left(\cos\left((N+1)\,\theta\right)-e^{-pT}\cos\left(N\theta\right)\right).$$

Using the result from (19), the right-hand side of this equation can be written as:

$$
\begin{aligned}
&-\frac{\lambda mp}{(1-e^{-pT})}\left(\cos\left((N+1)\,\theta\right)-\frac{\cos\left(N\theta\right)\sin\left((N+1)\,\theta\right)}{\sin\left(N\theta\right)}\right)\\
&=\\
&-\frac{\lambda mp\left(\sin\left(N\theta\right)\cos\left((N+1)\,\theta\right)-\cos\left(N\theta\right)\sin\left((N+1)\,\theta\right)\right)}{(1-e^{-pT})\sin\left(N\theta\right)}\\
&=-\frac{\lambda mp\ \sin\left(N\theta-(N+1)\theta\right)}{(1-e^{-pT})\sin\left(N\theta\right)}\\
&=-\frac{\lambda mp\ \sin\left(-\theta\right)}{(1-e^{-pT})\sin\left(N\theta\right)}\\
&=\frac{\lambda mp\ \sin\theta}{(1-e^{-pT})\sin\left(N\theta\right)}.
\end{aligned}
$$

which confirms (20).

B. Instability of Height Control Using Velocity Commands

First, assume a perfect system where the actual velocities instantaneously follow the velocity commands. Now, consider the following state space model with an output of unscaled vertical displacement from a monocular SLAM system:

$$
\begin{aligned}
\dot{\mathbf{x}}(t) &= A\mathbf{x}(t)+B\mathbf{u}(t)\\
&= \mathbf{u}(t),
\end{aligned}
\tag{33}
$$

$$
\begin{aligned}
\mathbf{y}(t) &= C\mathbf{x}(t)+D\mathbf{u}(t)\\
&= \frac{1}{\lambda}\mathbf{x}(t),
\end{aligned}
\tag{34}
$$

where $\mathbf{x}(t)=Z^w(t)$, $\mathbf{y}(t)=Z^v(t)=Z^w(t)/\lambda$, and $\mathbf{u}(t)=u_{vz}(t)$ is the vertical velocity command.

Discretizing the system with zero-order hold (ZOH) yields:

$$
A_d=\left[1\right],\ B_d=\left[T_s\right],\ C_d=\left[1/\lambda\right],\ D_d=[0], \tag{35}
$$

which is equivalent to the following transfer function:

$$
G(\sigma)=\frac{T_s}{\lambda(\sigma-1)}. \tag{36}
$$

Now, consider the following simple proportional feedback controller:

$$
u_z=K\left(Z^*-Z^v\right), \tag{37}
$$

where Z^* is the desired vertical displacement from the initial hovering height in the vision frame. Note the difference from (5), as we regulate the height instead of vertical velocity. From (36) and (37), the closed-loop transfer function becomes:

$$
H(\sigma)=\frac{KT_s}{\lambda(\sigma-1)+KT_s}. \tag{38}
$$

Following the same procedure as in Section III-A, the critical gain is computed as:

$$
K_{cr}=\frac{2}{T_s}\lambda, \tag{39}
$$

which indicates a unique, system-specific linear relationship between the critical gain and scale.

REFERENCES

[1] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based MAV navigation in unknown and unstructured environments," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2010, pp. 21–28.

[2] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart, "Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 3056–3063.

[3] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," in *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2012, pp. 2815–2821.

[4] D. Scaramuzza and F. Fraundorfer, "Visual odometry: Part I: The first 30 years and fundamentals," *IEEE Robot. & Automat. Mag.*, vol. 18, no. 4, pp. 80–92, 2011.

[5] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, 2015.

[6] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: fast semi-direct monocular visual odometry," in *Proc. IEEE Intl. Conf. on Robotics and Automation, (ICRA)*, 2014, pp. 15–22.

[7] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," in *arXiv:1607.02565*, 2016.

[8] G. Nützi, S. Weiss, D. Scaramuzza, and R. Siegwart, "Fusion of IMU and vision for absolute scale estimation in monocular SLAM," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1, pp. 287–299, 2011.

[9] G. C. H. E. de Croon, "Monocular distance estimation with optical flow maneuvers and efference copies: a stability-based strategy," *Bioinspiration & Biomimetics*, vol. 11, no. 1, 2016.

[10] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a navigation system for autonomous indoor flying," in *Proc. IEEE Intl. Conf. on Robotics and Automation, (ICRA)*, 2009, pp. 2878–2883.

[11] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Proc. IEEE Intl. Sym. of Robotics Research (ISRR)*, 2011, pp. 235–252.

[12] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to mav navigation," in *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 3923–3929.

[13] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proc. IEEE Intl. Conf. on Robotics and Automation, (ICRA)*, 2007, pp. 3565–3572.

[14] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[15] S. Shen, N. Michael, and V. Kumar, "Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft mavs," in *Proc. IEEE Intl. Conf. on Robotics and Automation, (ICRA)*, 2015, pp. 5303–5310.

[16] R. Mur-Artal and J. D. Tards, "Visual-inertial monocular SLAM with map reuse," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 796–803, 2017.

[17] J. Kaiser, A. Martinelli, F. Fontana, and D. Scaramuzza, "Simultaneous state initialization and gyroscope bias calibration in visual inertial aided navigation," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 18–25, 2017.

[18] G. V. Chowdhary, S. Srinivasan, and E. N. Johnson, "Frequency domain method for real-time detection of oscillations," *Journal of Aerospace Computing, Information, and Communication*, vol. 8, no. 2, pp. 42–52, 2011.

[19] P. Rzucidło, "The detection of pilotinduced oscillations," *Aviation*, vol. 11, no. 1, pp. 15–22, 2007.

[20] H. Huang, "tum_simulator," 2014. [Online]. Available: http://wiki.ros.org/tum_simulator

[21] M. Monajjemi, "ardrone_autonomy," 2012. [Online]. Available: http://wiki.ros.org/ardrone_autonomy

[22] J. Engel, J. Sturm, and D. Cremers, "Scale-aware navigation of a low-cost quadrocopter with a monocular camera," *Robotics and Autonomous Systems*, vol. 62, no. 11, pp. 1646–1656, 2014.

Faculty of Aerospace Engineering

Stability-based Scale Estimation for Monocular Visual Odometry of Quadrotor MAVs

Preliminary Report

Seong Hun Lee

March 22, 2017

TU Delft Delft University of Technology

Challenge the future

Stability-based Scale Estimation for Monocular Visual Odometry of Quadrotor MAVs

Preliminary Report

Preliminary Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering
at Delft University of Technology

Seong Hun Lee

March 22, 2017

Faculty of Aerospace Engineering · Delft University of Technology

Delft University of Technology

Delft University Of Technology
Department Of
Control and Simulation

Dated: March 22, 2017

Readers:

Dr. G. C. H. E. de Croon

Reader Two

Reader Three

Acronyms

BA	Bundle Adjustment
DFT	Discrete Fourier Transform
DOF	Degree of Freedom
DPPTAM	Dense Piecewise Planar Tracking and Mapping
DSO	Direct Sparse Odometry
EKF	Extended Kalman Filter
ESM	Efficient Second Order Minimization
FAST	Features from Accelerated Segment Test
FOV	Field of View
fps	frame per second
GNC	Guidance, Navigation and Control
ICIA	Inverse Compositional Image Alignment
IMU	Inertial Measurement Unit
IRLS	Iteratively Reweighted Least Squares
MAV	Micro Air Vehicle
ORB	Oriented FAST and rotated BRIEF
PTAM	Parallel Tracking and Mapping
RANSAC	Random Sample Consensus
SAD	Sum of Absolute Difference
SBC	Single Board Computer
SBI	Small Blurry Image
SFM	Structure From Motion
SLAM	Simultaneous Localization and Mapping
SVO	Semi-direct Visual Odometry
VO	Visual Odometry
ZOH	Zero-order Hold

Contents

Chapter 1

Introduction

When a Micro Air Vehicle (MAV) performs autonomous flight tasks in GPS-denied environments without any external motion capture systems, the Guidance, Navigation and Control (GNC) system relies solely on the measurements from its onboard sensors to compute the most appropriate control input for reaching the desired state (e.g., pose and/or velocity). Especially, for the maneuvers such as waypoint/trajectory-following and path-planning with collision avoidance, it is extremely useful to know the accurate estimation of 6 Degree of Freedom (DOF) pose[1] of the MAV, as well as the 3D structure of the surrounding environment. A technique which solves this problem simultaneously in real-time is called Simultaneous Localization and Mapping (SLAM)[2]. One of the most popular approaches today for achieving autonomous indoor flight of MAVs is to use vision to solve SLAM problems. By employing vision-based SLAM on the image stream from a single or multiple onboard camera(s), the system is capable of estimating both camera pose (thus MAV pose) and 3D reconstruction of the environment in real-time. These results can be then processed as input for high-level decision-making and outer-loop control, such as waypoint/trajectory generation and following. However, it should be noted that visual SLAM is merely one of many existing methods which can be adopted to achieve the autonomous navigation of MAVs in unknown GPS-denied environments. Admittedly, there exist multiple other alternative solutions (e.g., using different sensor modalities or different control strategies). In the next two paragraphs, it will be explained why SLAM or Visual Odometry (VO)[3] is chosen to be used for this work.

First, why use vision? The success and popularity of using vision in MAVs could be explained by the following factors: (1) There are already many cheap, small and lightweight cameras available with moderate quality. This is certainly a strong advantage over other heavier and more expensive alternatives such as laser, because weight, size and budget are often critical factors in the design and operation of MAVs. (2) Significant advancements of computer vision

[1] Six because there are three orthogonal components of translational movements in x, y, z direction and three corresponding Euler angles along each axis

[2] Localization is also known as (camera pose) tracking. Mapping means 3D reconstruction of the scene observed by the camera, and it usually takes the form of point cloud.

[3] Visual odometry and SLAM are similar, but not exactly identical. See Section 3-1-1 to understand the difference between VO and SLAM.

algorithms (e.g., VO and SLAM) can allow us to process the visual input efficiently and interpret it as meaningful data for various autonomous tasks. The technology is already mature and proven to work in many different platforms. (3) Using only vision is generally better and more strongly recommended than using only Inertial Measurement Unit (IMU) measurements. This is mainly because of the intrinsic sensor noise and drift of IMU measurements. Thus, for example, using visual landmarks for localization is much less prone to drift than relying on pure IMU measurements (Woodman, 2007; Hu & Chen, 2014). Besides, IMU-only system cannot map the environment, which makes it unusable for tasks where high-level autonomy is required, such as collision avoidance in indoor environments. Furthermore, note that it is also possible (often even more desirable) to combine both vision and IMU measurements using sensor-fusion techniques, which can lead to a gain in accuracy, robustness and efficiency - this is called **visual-inertial fusion**, and it will be briefly introduced in the literature study (Section 3-2-2).

Second, why use VO/SLAM? In recent years, there have been major advancements of VO and SLAM methods, pushing the limits in terms of accuracy and efficiency. As a result, it is now possible to implement many of these state-of-the-art VO systems for control and navigation of MAVs using only onboard camera and processor. Above all, the fact that both 6 DOF pose and 3D map of the environment can be estimated accurately and simultaneously in real-time is the most attractive advantage of SLAM-based autonomous systems. This is also the main difference from the other existing real-time vision-based approaches which do not make use of SLAM, such as velocity estimation using optical flow vectors.

So far, the relevance of VO/SLAM in autonomous flight of MAVs has been discussed. Like any other approaches, the state-of-the-art VO/SLAM algorithms still have large room for further improvements in terms of performance and functionality. Theoretically speaking, however, there is one critical weakness that cannot be overcome by simply improving the system performance as far as monocular approaches are concerned. ***That is, the intrinsic unobservability of the absolute scale***[4]. One intuitive way to understand this phenomenon is by looking at what is called 'miniature effect'. This is demonstrated in Figure 1-1. You can clearly see that it is simply not possible to retrieve the absolute scale from just a monocular sequence of images. Instead, only the relative scale[5] can be perceived, although this sense perception may still turn out to be very wrong in case of insufficient parallax motion. You may then ask: "Why not avoid the trouble by using a stereo or multiple camera system?". The rationale behind choosing the monocular system over stereo system will be elaborated later in the literature survey (Section 3-1-1).

In order to estimate the 3D translational pose of the camera and coordinates of the map points in the correct metric scale, the scale factor must be applied to these states. For monocular systems, the following methods are commonly used to estimate the absolute metric scale factor:

1. Using an additional range sensor (Section 3-2-1)
2. Incorporating the accelerometer measurements in sensor fusion (Section 3-2-2)
3. Initializing the initial map with a target of known size (Section 3-2-3)

[4]This means the knowledge of metric distance (or length) between the two distinct 3D points in the perceived environment. The 3D points may also include the 3D position of the camera itself. For more explanation, see Section 3-2.

[5]This means the knowledge of ratio of distance (or length) between the two distinct 3D points in the perceived environment.

(a) A miniature model of the RMS Titanic (Disario, 1997)

(b) A photograph taken during the filmmaking (Anonymous, 1997)

Figure 1-1: Demonstration of intentional miniature effect devised during the filmmaking of 1997 film "Titanic". The audience could not notice that they were watching a miniature model of the ship, because it is impossible to observe the absolute scale (i.e. dimension of an object in meters) from a monocular sequence of images. Only the relative scale (i.e. ratio of length or size) could be perceived.

Each of these methods has their own limitations - it either requires additional sensors or special target to initialize. Especially, the sensor fusion approach, either filter-based or optimization-based, is often subject to convergence towards an inaccurate scale factor given a poor initialization (e.g., inappropriate initial movements or inaccurate calibration parameters).

In this work, a novel stability-based scale estimation method is proposed which exploits the unique self-induced oscillation behavior of quadrotor MAVs when a constant divergence control strategy is implemented - that is, the control gain which starts inducing the instability is linearly proportional to the distance to the ground surface from which the flow divergence is estimated. The distance estimation strategy that takes advantage of this property was first proposed by (de Croon, 2016), which was verified with simulations and real-world experiments. Based on the original work, the main contribution of this work will be as follows:

1. The stability-based distance estimation is used to initialize the absolute (metric) scale of a monocular VO system. This can enable various autonomous flight modes, including hovering, landing and waypoint-following.
2. Instead of estimating the divergence from optical flow vectors, the unscaled 3D position and velocity output from the VO system is used to compute the divergence directly. This will bring about multiple advantages, such as higher accuracy and robustness (see Section 4-1-1 for more discussions).
3. (OPTIONAL) Improvements of the distance estimation strategy may be proposed (e.g., more accurate oscillation detection method, etc.)

This preliminary report is organized as follows: First, a research analysis is given in Chapter 2 where the relevant research objective, questions and tasks are defined. It is then followed by a literature survey in Chapter 3. The literature survey is structured in two folds, discussing the state-of-the-art VO/SLAM systems (Section 3-1) and metric scale estimation strategies

(Section 3-2). In Chapter 4, a novel stability-based scale estimation method is proposed and elaborated in the context of a monocular VO pipeline. Also, the set-up for simulations and real-world experiments is addressed. In Chapter 5, a preliminary analysis of the most relevant open-source systems and datasets is provided. Finally, the planning and conclusions are presented in Chapter 6 and Chapter 7, respectively.

Chapter 2

Research Analysis

2-1 Main research objective

First of all, the main research objective is formulated as follows:

The main research objective is to achieve an autonomous indoor navigation of a quadrotor MAV using the onboard processor and forward-looking camera by implementing a monocular VO pipeline with the novel stability-based scale estimation.

The primary reason that a forward-looking camera is chosen over a downward-looking camera is due to the fact that indoor floor texture is generally weak, which worsens the performance of any vision-based algorithms. In contrast, the indoor scene visible to the forward-looking camera usually contains stronger texture, such as edges and corners with strong contrast, so-called, *feature-points*. Also, note that the research objective strictly requires that only the onboard processor be used for the VO system. Despite the disadvantage that it imposes a strong constraint on the computational complexity of the overall system, onboard processing can reduce the risk of MAV-to-ground-station link failure or intermittent disconnections, as well as the time-delay introduced between the vision and control system. Such risks can be fatal for stability-based strategies where large or inconsistent time delay would most certainly degrade the accuracy of the system.

2-2 Sub-goals/Tasks

In order to achieve the main objective, the following tasks must be achieved in the following months:

1. Implement a complete simulation program which links the following modules:
    - Graphical interface

- Quadrotor dynamics
- VO/SLAM system
- Scale estimation using the stability-based distance estimation method.

2. Verify the proposed scale estimation strategy with the simulation.
   - If necessary, improve the scale estimation method and repeat.
3. Establish the link between the Single Board Computer (SBC), camera and MAV.
4. Validate the proposed scale estimation strategy with real-world experiments.
   - If necessary, adjust the control parameters and repeat.
5. Evaluate the autonomous waypoint-following flights by comparing against the ground truth
   (a) Implement the autonomous waypoint-following flight plan on PaparazziUAV[1]
   (b) Collect and compare the odometry result with the ground-truth trajectory captured by the OptiTrack[2] system.
6. Write a thesis report.

For more information on the set-up of simulation and real-world experiments, see Section 4-2. The planning and logistics of the task execution is elaborated in Chapter 6.

2-3 Research questions

The following research questions will be answered in the following months to accomplish the research objective successfully. The main research questions and their sub-questions are listed in a bullet-point style.

1. Which monocular VO system is the most suitable for the proposed pipeline and platform?
2. How accurate is the stability-based distance estimation compared to the ground-truth?
   (a) How accurate is it in simulation?
   (b) How accurate is it in real-world experiments?
3. What is the difference between the simulation and real-world experiments, and why so?
4. What modifications (if any) did you make in the algorithm to improve the accuracy of the stabilization-based scale estimation?
5. How accurate is the pose estimation given by the proposed system compared to the ground-truth (e.g. RMSE[3])?
6. What are the common failure modes of the proposed system?
7. What are the limitations of the proposed method, and what possible recommendations do you have to overcome these limitations?

[1] `https://wiki.paparazziuav.org/wiki/Main_Page`
[2] `https://optitrack.com/`
[3] Root Mean Squared Error

Chapter 3

Literature survey

In this chapter, the most relevant literature are reviewed. In Section 3-1, the state-of-the art VO/SLAM systems are categorized and discussed. In Section 3-2, existing methods that are frequently used for metric scale estimation are investigated.

3-1 Visual Odometry/SLAM Systems

This section consists of two parts. In Section 3-1-1, it is first shown that there exist multiple ways to categorize VO/SLAM systems based on different characteristics. Also, it is argued why certain approaches are more suitable for this work than the others. In Section 3-1-2, the relevant open-source state-of-the-art VO/SLAM systems are reviewed.

3-1-1 Categorization and Justification for choice

The vision system can be categorized in the following ways:

- Visual Odometry *vs* Visual SLAM
- Monocular *vs* Stereo
- Sparse *vs* Dense *vs* Semi-dense
- Filter-based *vs* Keyframe-based
- Feature-based *vs* Direct *vs* Semi-direct

In the following discussion, the taxonomy of categories is explained. Also, the reason for choosing **a keyframe-based sparse monocular visual odometry approach** will be argued.

Visual Odometry vs Visual SLAM

Visual odometry and visual SLAM are fundamentally the same in the sense that both methods attempt to estimate the camera pose and reconstruct the scene structure. In other words, these two approaches come from the common principle called Structure From Motion (SFM). This is why some researchers use the term VO and V-SLAM interchangeably. Technically speaking, however, these two methods have different philosophies. VO aims at recovering the camera pose incrementally, and potentially, it may build a local map and trajectory using windowed optimization. SLAM, on the other hand, tries to produce a globally consistent map and trajectory, such that the system can detect when the camera returns to a previously explored position and correct the drift accordingly. This is known as 'loop-closure'. In summary, one can distinguish between VO and SLAM by looking whether it has a loop-closure functionality or not. Therefore, a full SLAM system has generally higher computational complexity compared to a pure VO system. **It is for this reason why VO will be used for this work.** For more discussion on the difference between visual odometry and SLAM, see (Scaramuzza & Fraundorfer, 2011).

Monocular vs Stereo

Stereo methods have several advantages over monocular methods - for example, it is possible to directly compute the 3D map points in the the absolute scale. Besides, stereo schemes are generally more robust to drift in case of small motions. However, the depth estimates from stereo vision deteriorate when the scene is far away from the cameras. This is illustrated in Figure 3-1. Since $\triangle PO_lO_r$ is similar to $\triangle Pp_lp_r$, the following equations hold:

$$\frac{d}{b} = \frac{d-f}{b-x_l+x_r} \tag{3-1}$$

$$\text{Hence:} \quad \frac{d}{b} = \frac{f}{x_l-x_r} \tag{3-2}$$

Equation (3-2) means that if the depth-to-baseline becomes large, the disparity $(x_l - x_r)$ will be small, and vice versa. So when the matched feature is actually located far away from the cameras (e.g., at infinity), the computation of this feature's depth would rely on a very small disparity value, which implies that the estimation is most likely to be sensitive to small errors or even noise introduced in the stereo matching process. One way to mitigate this problem is by increasing the stereo baseline. However, this is often undesirable or impractical for lightweight MAVs. With this trade-off between monocular and stereo approaches in mind, the focus is placed on the fact that (1) the target platform is a quadrotor MAV with a strong constraint on the size, and (2) large-scale navigation should be possible using the same method used for the small-scale navigation. **Therefore, a monocular scheme is chosen**, and this is explicitly specified in the research objective (see Section 2-1).

Sparse vs Dense vs Semi-dense

Based on the sparsity of the tracking/mapping points, VO/SLAM systems can be classified as either sparse, dense or semi-dense method (see Figure 3-2). Sparse methods track and map

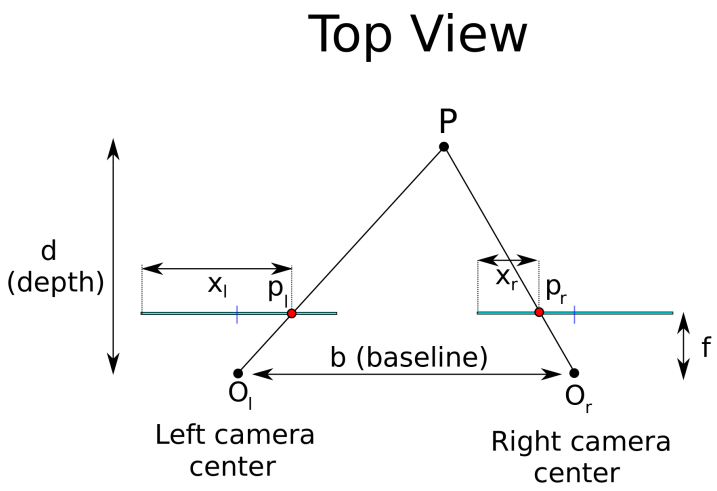


Figure 3-1: Stereo-vision disparity and depth relationship are explained. p_l and p_r are the 2D projections of the 3D feature point P onto the left and right camera's image planes respectively.

only a selected subset of sparse points (e.g. corners, feature-points), whereas dense methods use the entire pixels in the 2D image. The difference in sparsity is clearly manifested in the appearance of the 3D point cloud reconstruction result. The most representative dense SLAM methods are (Newcombe, Lovegrove, & Davison, 2011) and (Pizzoli, Forster, & Scaramuzza, 2014). On the other hand, semidense approach uses only the high-gradient pixels in the image, which usually correspond to edges and corners. In general, the number of 3D tracking (and mapping) points in the semidense methods is larger than the sparse methods, but smaller than the dense methods. Examples include (Engel, Sturm, & Cremers, 2013) and (Engel, Schöps, & Cremers, 2014). The primary tracking and mapping threads of DPPTAM (Concha & Civera, 2015) also make use of semidense pixel points. **For this work, only sparse methods will be considered due to the limited processing resources of onboard processors.**

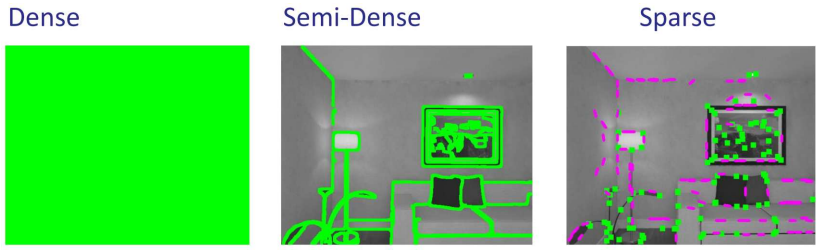


Figure 3-2: Illustration of different sparsity levels (Scaramuzza, 2016). Only the colored pixels are used for tracking and mapping, and the rest are simply discarded.

Filter-based vs Keyframe-based

Filter-based methods, such as (Davison, Reid, Molton, & Stasse, 2007), estimate the pose and 3D feature positions by including them as in the state vector and updating their probability distributions sequentially using Extended Kalman Filter (EKF)-framework. On the other hand, keyframe-based methods use optimization (e.g., *bundle adjustment*) on a selected subset of past frames known as *keyframes* to compute the most likely pose parameters and

feature positions. In (Strasdat, Montiel, & Davison, 2010), these two approaches are compared. Their results showed that keyframe-based methods give higher accuracy, unless the processing resources are significantly limited, which is not the case for most commercially available embedded processors today. **Thus, a keyframe-based approach is chosen for this work.**

Feature-based vs Direct vs Semi-direct

Visual motion estimation methods can be classified into three categories: Feature-based (or indirect), direct, and semi-direct (or hybrid). Feature-based methods estimate 3D geometry of a scene and camera pose by performing an optimization technique called "Bundle Adjustment (BA)" which tries to minimize the reprojection error from a set of keypoint-matches (see Figure 3-3). Motion-only BA is then expressed as Equation (3-3), where Cost(·) represents a robust cost function (e.g. Huber or Tukey (Huber, Wiley, & InterScience, 1981)). Examples of feature-based approach are PTAM (Klein & Murray, 2007) and ORB-SLAM (Mur-Artal, Montiel, & Tardos, 2015).

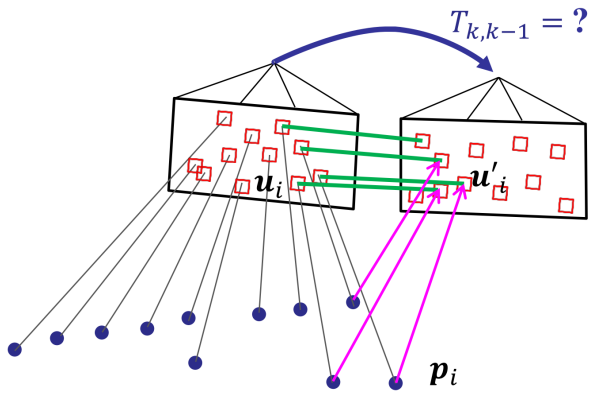


Figure 3-3: Feature-based methods minimize the cost function of reprojection error (Scaramuzza, 2016). See Equation (3-3).

$$T_{k,k-1} = \arg\min_{T} \sum_{i} \text{Cost}\left(\mathbf{u'_i} - \pi(\mathbf{p_i})\right) \tag{3-3}$$

In contrast, direct methods do not extract or match feature-points. Instead, they make use of the photometric error (i.e., difference in the pixel intensity) in order to estimate the 3D geometry of the scene and camera pose. This is shown in Figure 3-4 and Equation (3-4). Examples of direct approach are DTAM (Newcombe et al., 2011), LSD-SLAM (Engel et al., 2014) and DPPTAM (Concha & Civera, 2015). Compared to the sparse feature-based methods, dense direct methods have been shown to be more robust in scenes with little texture (Lovegrove, Davison, & Ibanez-Guzman, 2011), occlusion and camera blur caused by the rapid motion or defocus (Newcombe et al., 2011) compared to feature-based methods. Semi-direct methods, such as SVO (Forster, Pizzoli, & Scaramuzza, 2014), use the photometric error for the initial pose estimation after each keyframe addition, and perform BA to refine the camera pose and structure.

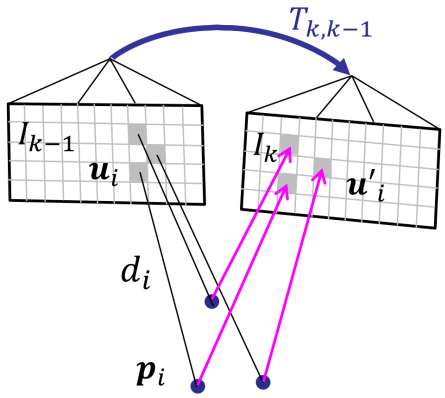


Figure 3-4: Direct methods minimize the cost function of photometric error (Scaramuzza, 2016). See Equation (3-4).

$$\begin{aligned} T_{k,k-1} &= \arg\min_T \sum_i \text{Cost}\left(I_k(\mathbf{u_i'}) - I_{k-1}(\mathbf{u_i})\right) \\ \text{where } \mathbf{u_i'} &= \pi\left(T \cdot \left(\pi^{-1}(\mathbf{u_i}) \cdot d_i\right)\right) \end{aligned} \tag{3-4}$$

For the moment, the question as to which method is the most appropriate for this work is left unanswered because there are large spectrum of systems of each type and it is difficult to generalize which approach is better than another. Instead, some of the most relevant state-of-the-art VO systems of each type (i.e., feature-based, direct and semi-direct) are tested and compared in Section 5-2.

3-1-2 Open-source VO/SLAM Systems

In this section, the following stat-of-the-art open-source VO/SLAM systems (except for the sparsified DPPTAM, which is closed-source) are reviewed. Note that all these methods take a keyframe-based monocular approach.

1. **PTAM** (Klein & Murray, 2007) and **Modified PTAM** (Weiss, Siegwart, & Kumar, 2012)
2. **SVO** (Forster et al., 2014)
3. **DPPTAM** (Concha & Civera, 2015) and **Sparsified DPPTAM**
4. **ORB-SLAM** (Mur-Artal et al., 2015)
5. **DSO** (Engel, Koltun, & Cremers, 2016a)

PTAM

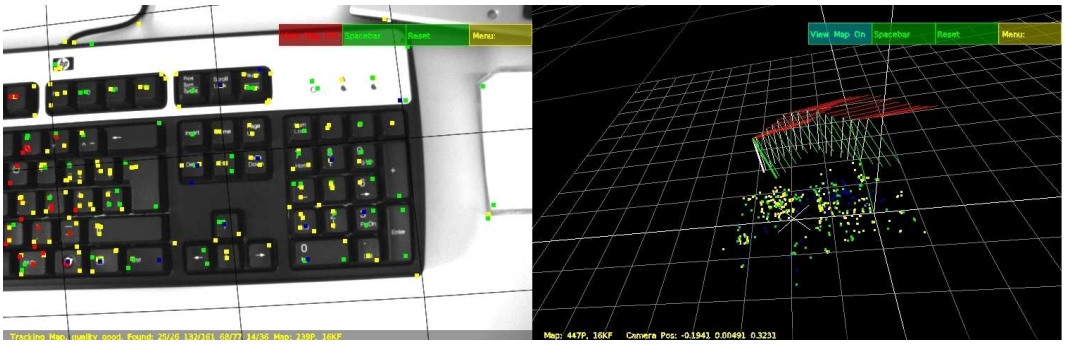


Figure 3-5: Visual output from PTAM demo. **Left:** Camera input and tracked feature points. The different colors of the feature points represent the different pyramid levels at which the tracking is taking place. **Right:** 3D visualization of the camera trajectory and mapped points.

Parallel Tracking and Mapping (PTAM) (Klein & Murray, 2007) is the first keyframe-based algorithm which separates the tracking and mapping process in two parallel threads, resulting in a significant increase of the computational efficiency and system performance compared to the past SLAM algorithms, such as filter-based MonoSLAM (Davison et al., 2007). PTAM is characterized by the following points:

- The original release of PTAM initialized a map using five-point stereo algorithm (Stewénius, Engels, & Nistér, 2006) which estimates a fundamental matrix assuming non-planar initial scene, but this was later changed to a homography-based algorithm (Faugeras & Lustman, 1988) which assumes the initial scene to be piecewise planar.
- The Features from Accelerated Segment Test (FAST) algorithm (Rosten & Drummond, 2006) is used to extract corners, and they are represented as 8×8 locally planar patches. Four-level image pyramid is built for each keyframe. Each map-point has a reference to the source keyframe and pyramid level where the feature was first observed, as well as its 2D pixel location.

- The tracking thread updates the pose by optimizing the Tukey biweight objective function (Huber et al., 1981) of the reprojection error (see Equation (3-3)). To improve the chance of convergence, a motion model is implemented to predict the pose and use it as the initial guess. Also, coarse-to-fine scheme is employed search the patches more robustly in the rapid camera motions.
- Once tracking is lost, relocalization is attempted by comparing the photometric difference of Small Blurry Image (SBI) of the current frame with respect to SBIs of all past keyframes. Once the most similar keyframe is found, the position of the current frame is assumed to be the same as the position of the matching keyframe. Afterwards, Efficient Second Order Minimization (ESM) tracking (Benhimane & Malis, 2007) is carried out to compute the rotational transformation that best aligns the two images.
- A new keyframe is added by the mapping thread if a certain minimum time has passed **and** the current pose is farther than a certain minimum distance from the closest past keyframe. Then, the reprojection error between the projection and triangulation of the map-points from the closest keyframe is used in the bundle adjustment.
- The bundle adjustment consists of local bundle adjustment (where the recent keyframe poses and associated map-points are optimized) and global bundle adjustment (where the all keyframe poses and map-points are fully optimized after the map-points are locally converged.).

The limitations of PTAM are: First, the stereo initialization algorithm can easily fail unless a slow and smooth translational motion is given parallel to the image plane. Second, it cannot handle large occlusion, as self-occlusion by the map is not understood by the system. This makes PTAM inappropriate for applications, such as large-scale operations or tracking 360° movement around a target object.

Modified PTAM

Modified PTAM (Weiss, Achtelik, Lynen, Chli, & Siegwart, 2012) has the following adaptations to the original algorithm which improve the tracking quality and system performance:

1. It retains an upper bound in the number of keyframes used in order to minimize the computational complexity. The keyframe farthest away from the current keyframe gets deleted first along with the associated features. Also, the number of maximum tracking points for each keyframe is reduced from 1,000 to 300.
2. Features from the finest-scale pyramid level are not included in map-handling. This improves the keyframe-generation speed and tracking robustness against high frequent "self-similar" scenes, such as grass or asphalt.
3. Inverted index structure is implemented to filter out the map-points that are unlikely to be in the field-of-view of the current keyframe.

SVO

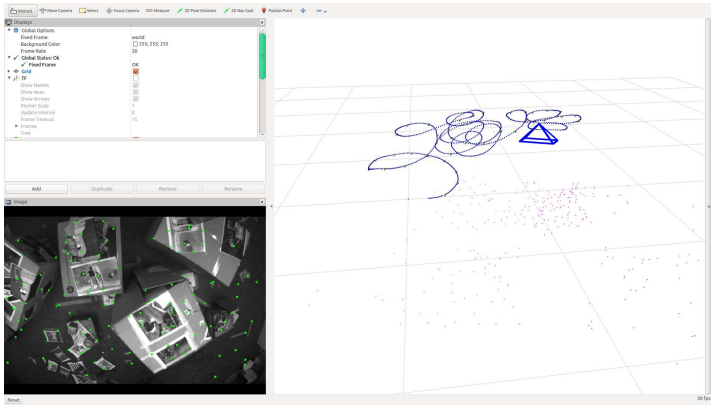


Figure 3-6: Visual output from SVO demo. ROS rviz package is used for online visualization.

Semi-direct Visual Odometry (SVO) (Forster et al., 2014) is a semi-direct monocular visual odometry method characterized by the following novelties:

- Tracking thread is divided into three steps of hybrid motion estimation approach:
  1. Sparse model-based image alignment: this direct step provides the initial estimate of the relative camera motion by minimizing the photometric error of 4×4 patches around FAST corners. Inverse Compositional Image Alignment (ICIA) technique is applied to accelerate the process (Baker & Matthews, 2004).
  2. Relaxation through feature alignment: this step refines the 2D feature correspondence by aligning the corresponding affine-warped feature-patches. ICIA approach is also used for this step.
  3. Pose and structure refinement: in this step, bundle adjustment is applied using the reprojection error from the previously obtained feature correspondence.
- Mapping thread uses a Bayesian depth filter from (Vogiatzis & Hernandez, 2011) to recursively estimate the probabilistic distribution of the inverse depths corresponding to the 2D features. Refer to (Civera, Davison, & Montiel, 2008) for more information on inverse depth parametrization.

The processing speed of SVO is remarkably fast: approximately 300 fps on the laptop[1] and 55 fps on the embedded platform[2]. Such high speed is the result of the semi-direct approach which saves computational cost by obviating the need for explicit feature extraction and matching at every frame, as well as the Bayesian depth filter which allows the system to use a small number of features (< 200) by assuring reliable depth estimates of the tracked features. However, the limitation of the current SVO implementation is that the tracking robustness deteriorates considerably if either (i) rolling shutter camera is used instead of global shutter camera, (ii) video frame-rate is low (≤ 30 Hz), or (iii) the dominant motion is the forward motion. If only such shortcomings can be overcome by means of an efficient visual-inertial fusion, SVO is an undoubtedly attractive choice due to its inherent speed advantage compared to other VO systems. An example of visual output is shown in Figure 3-6.

[1] Intel i7, 8 cores, 2.8 GHz
[2] Odroid-U2, ARM Cortex A-9, 4 ores, 1.6 GHz

DPPTAM

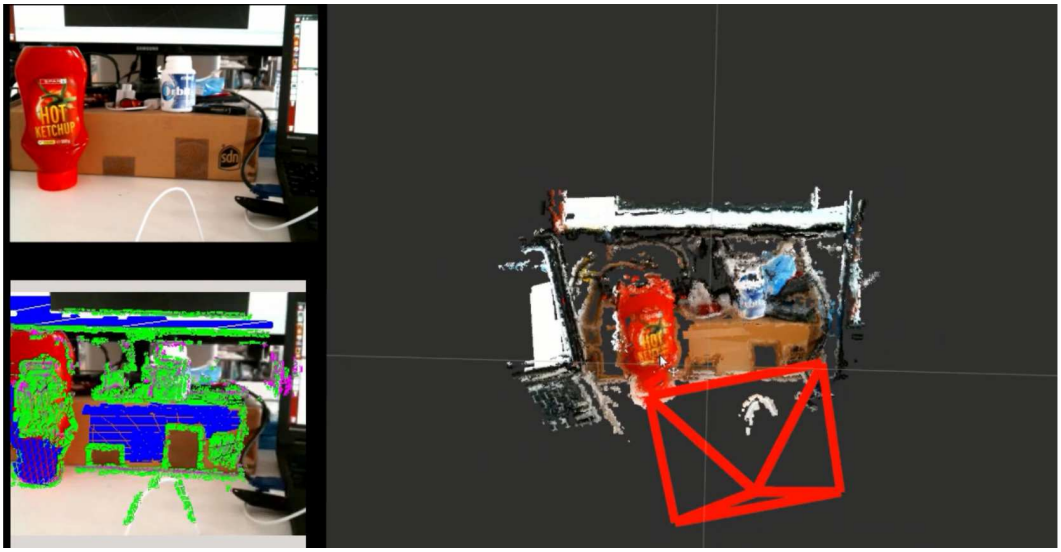


Figure 3-7: Visual output from DPPTAM demo in a small workspace environment. **Left top image:** input RGB frame. **Left bottom image:** Projections of the tracked points and 3D superpixels onto the image input . The green points have low photometric error, and the pink points have high photometric error. Blue planes are the successfully mapped 3D superpixels. **Right image:** 3D point cloud as a result of mapping. The 6-DOF camera pose is visualized by the red pyramid

Dense Piecewise Planar Tracking and Mapping (DPPTAM) (Concha & Civera, 2015) is a direct monocular VO algorithm consisting of three threads - semidense tracking, semidense mapping and dense piecewise plane (i.e. 3D superpixels) mapping. The semidense tracking thread estimates the camera pose by optimizing the weighted square sum of photometric errors corresponding to high-gradient pixels. The optimization is done by using the inverse compositional Iteratively Reweighted Least Squares (IRLS) algorithm (Baker & Matthews, 2004) with Gauss-Newton update and *Geman-McClure* weight function. In parallel with the tracking thread, the semidense mapping thread estimates the 3D location of the map points. The thread has two main functions: (a) inverse depth update through epipolar triangulation, and (b) inverse depth regularization. The epipolar search uses Sum of Absolute Difference (SAD) with 7×7 patch size, and afterward, the uncertainties of estimated inverse depths are updated by assuming a disparity deviation of one pixel, like in (Forster et al., 2014; Pizzoli et al., 2014). The inverse depth regularization performs "de-noising of the map" by means of outlier removal based on three criteria - temporal consistency, spatial consistency and accumulated uncertainty in geometric disparity error. Finally, the 3D superpixel mapping thread constructs piecewise planar surfaces by fitting planes (with RANSAC) to the semidense map points whose 2D projections belong to the superpixel contour obtained from the graph-based image segmentation (Felzenszwalb & Huttenlocher, 2004) of each mapping image.

In order to bootstrap the system, a random depth with extremely large uncertainty is uniformly assigned for the first frame parallel to the image plane. In most of the cases, it has been shown that the depth map converges to the correct value after the first few keyframes. Such bootstrapping method is similar to (Engel et al., 2014). Since this uniform depth at the

beginning is obviously not the ground truth value, DPPTAM does not visualize the initial local map results from the first few keyframes in the final 3D reconstruction. An example of visual output is given in Figure 3-7.

Sparsified DPPTAM

Sparsified DPPTAM is a closed-source sparse variation of DPPTAM which uses and reconstructs FAST corners in the images (see Figure 3-8). The main difference between the semi-dense DPPTAM (i.e. original DPPTAM implementation without 3D superpixel dense mapping) is that it selects corners for the inverse depth initialization, instead of high-gradient pixels. Additionally, the sparsified DPPTAM adapts FAST corner thresholds regionally to control the map sparsity and distribution of points throughout the image, similarly to (Engel et al., 2016a). The sparsified DPPTAM is not only faster than the original DPPTAM (i.e., increased tracking speed up to 60 fps for 320×240 resolution input), but also more accurate and robust to hostile changes in the appearance of the scene caused by large/rapid movements, occlusion, and illumination changes.

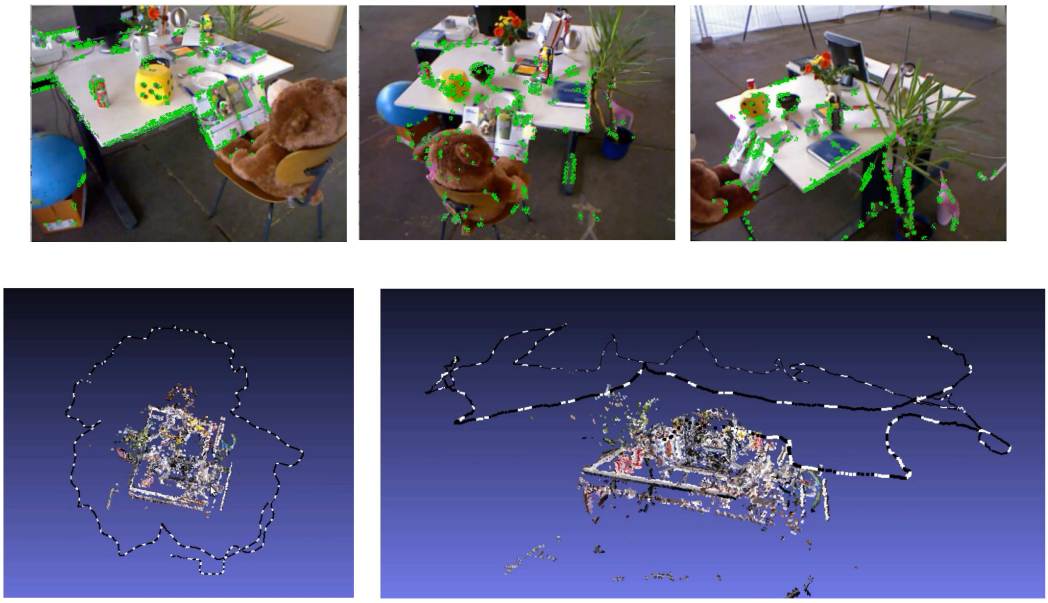


Figure 3-8: Sample results from the sparsified DPPTAM. **Top three images:** Sparse (sometimes semidense, depending on the corner threshold) projections of the tracked points onto the image input. **Bottom two images:** Final reconstruction of the scene and camera pose trajectory.

ORB-SLAM

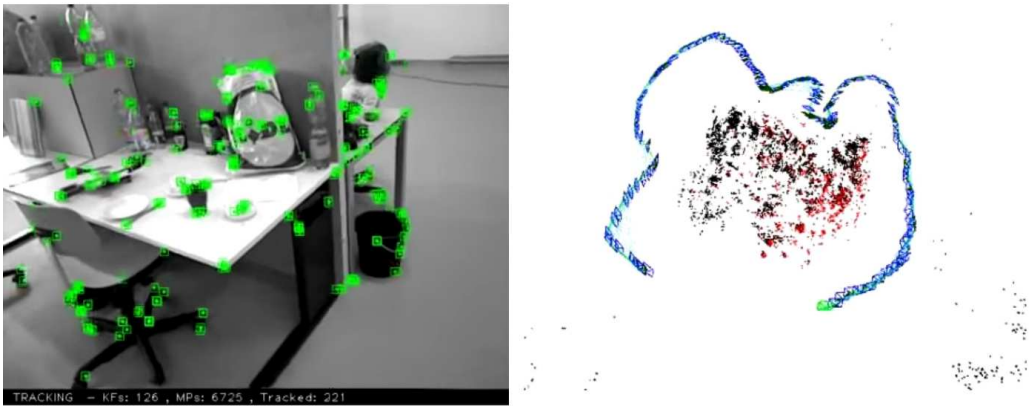


Figure 3-9: Visual output from ORB-SLAM demo. The red points represent the local map, and black points are the remaining feature points which are not visible in the local map. The past keyframe positions are shown as blue cameras.

ORB-SLAM (Mur-Artal et al., 2015) is a full SLAM algorithm (i.e., with loop closure) which takes a featured-based monocular sparse approach. The visual output is shown in Figure 3-9. The main contributions of ORB-SLAM are:

- Oriented FAST and rotated BRIEF (ORB) features (Rublee, Rabaud, Konolige, & Bradski, 2011) are used throughout all tasks, allowing real-time performance on CPUs and strong invariance to viewpoint and illumination changes.
- Real-time performance is ensured for even large-scale operations by using a covisibility graph (Strasdat, Davison, Montiel, & Konolige, 2011).
- Real-time loop-closure is achieved by optimizing a pose graph called the *Essential Graph*. An essential graph is made of (i) a spanning tree of keyframes, (ii) a subset of strong edges from the covisibility graph, and (iii) loop-closure links.
- Bags of words place recognition module based on DBoW2 (Galvez-Lopez & Tardos, 2012) are used for effective loop detection and relocalization.
- Automatic initialization procedure which can handle both planar or non-planar initial scenes.
- A *survival of the fittest* approach (i.e. recent map points culling and local keyframe culling) discards outlier map-points and redundant keyframes.

All optimizations are performed using the Levenberg-Marquardt algorithm implemented in the g2o framework (Kuemmerle, Grisetti, Strasdat, Konolige, & Burgard, 2011).

DSO

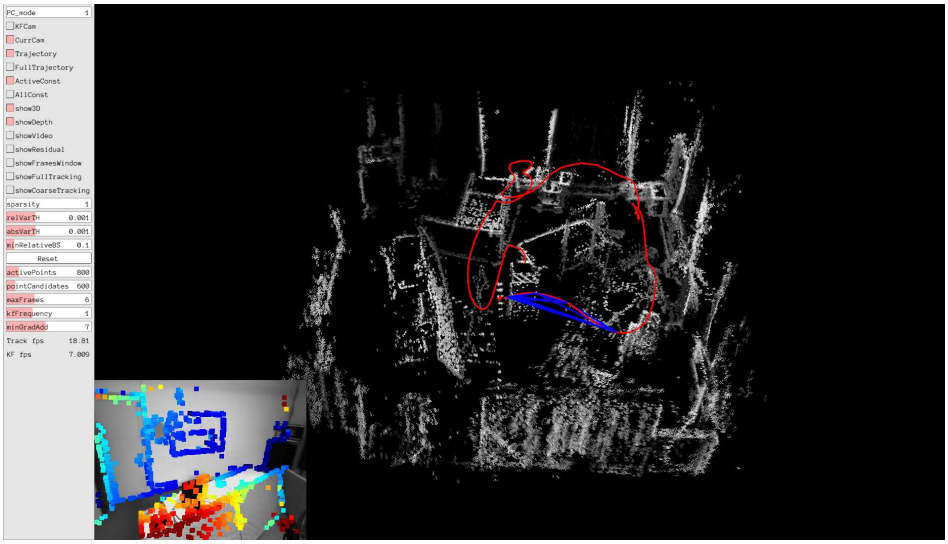


Figure 3-10: Visual output from DSO demo. The bottom left image shows the camera input with 2D projections of the tracked points with color-coded inverse depth. The red line represents the camera pose trajectory, and local constraints among the active keyframes are shown in blue color.

Direct Sparse Odometry (DSO) (Engel et al., 2016a) is a direct sparse monocular visual odometry method (as shown in Figure 3-10) characterized by the following novelties:

- It is a fully direct method that jointly optimizes the likelihood of all involved model parameters - camera poses, camera intrinsics, inverse depth values (and affine brightness transfer function parameters if the proposed photometric calibration (Engel, Koltun, & Cremers, 2016b) is absent).
- Visual odometry front-end performs data-selection and manages the frames and points with effective heuristics, providing accurate initialization for the back-end optimization.
- Windowed optimization, inspired by (Leutenegger, Lynen, Bosse, Siegwart, & Furgale, 2015), marginalizes the old variables (i.e. keyframes and all the points represented in them) using Schur complement in order to drop residual terms that affect the sparsity pattern of the Hessian in the Gauss-Newton optimization.

DSO is shown to outperform the odometry version of ORB-SLAM (Mur-Artal et al., 2015) in a hard-enforced real-time setting. Especially, it is worth noting that the proposed reduced setting (800 feature points, 6 active frames, 424×320 image resolution, ≤ 4 Gauss-Newton iterations) can be used to achieve 5 times real-time speed[3] on a high-end laptop[4]. However, it was shown that the performance of DSO quickly deteriorates in the presence of strong geometric noise originating from inaccurate camera calibration and/or rolling shutter camera effect. This suggests that indirect model (e.g. ORB-SLAM) may be generally superior for smartphones or off-the-shelf webcams.

[3]This is the speed when all images were loaded and rectified beforehand
[4]Intel i7-4910MQ CPU

3-2 Metric Scale Estimation Methods

The metric scale of a monocular VO can be estimated if any one of the following types of distance is known or measured in meters: (i) pose-to-pose distance, (ii) pose-to-point distance, and (iii) point-to-point distance, as shown in Figure 3-11. The VO system provides measures of these distances with its own arbitrary scale. Even though this scale is not metric, the distances are internally consistent with each other as long as the VO results are valid. Therefore, knowing at least one type of metric distance in the map and/or pose will allow us to directly compute the metric scale factor. In this section, four different scale estimation strategies are reviewed:

1. Using a range sensor
2. Visual-inertial fusion
3. Initialization with a known target
4. Stability-based distance estimation

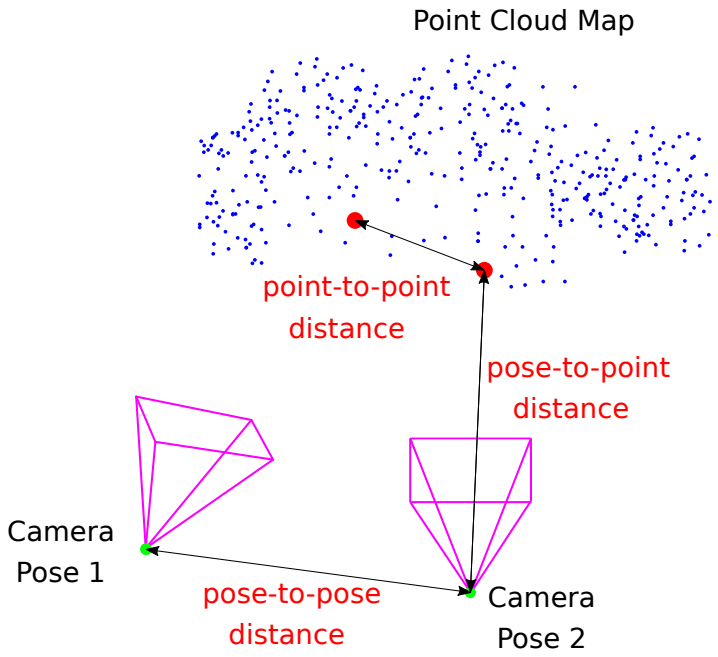


Figure 3-11: Basic principle of scale estimation illustrated: The metric scale of a monocular VO can be estimated if any one of the above distances is known or measured in meters.

3-2-1 Using a Range Sensor

Range sensors (e.g., laser, ultrasonic/pressure altimeter, RGB-D[5] cameras) or stereo cameras can directly provide the metric distance measurements. This corresponds to the pose-to-point distance in Figure 3-11. For each type of sensors, there exist a number of advanced odometry/SLAM methods which effectively incorporate the range measurements to estimate the metric states. For interested readers, refer to (Cole & Newman, 2006), (Engel, Sturm, & Cremers, 2012) and (Kerl, Stueckler, & Cremers, 2015) to learn more about the existing methods that use a laser, ultrasonic/pressure altimeter, and RGB-D camera, respectively. Although the direct availability of the distance measurements allows for higher accuracy and efficiency, there are few obvious weaknesses: First of all, lasers and RGB-D cameras are often too bulky or heavy to be mounted on MAVs. Second of all, ultrasonic/pressure altimeters are not trivial to acquire. Besides, their height measurements are also known to be quite noisy, and yield even lower reliability when the ground is more uneven.

3-2-2 Visual-Inertial fusion

Over the past decade, visual-inertial fusion methods have gained large popularity for variety of applications in robotics and perception field. This is due to the complementary nature of the two sensor modalities (i.e., a camera is an exteroceptive sensor, whereas an IMU is a proprioceptive sensor), as well as the availability of small, lightweight and cheap cameras and MEMS[6] IMUs. IMUs consist of tri-axis gyroscopes and accelerometers. The gyro measurements are especially useful when estimating rapid rotational motions which are generally known to be difficult to track in VO. However, since the gyro measurements are purely rotational, they do not provide any information on the metric distance. Therefore, in order to estimate the absolute scale of a monocular VO, accelerometer measurements must be used. For this reason, we only consider the visual-inertial fusion approaches that use both gyro and accelerometer measurements, thus excluding 'gyro-only' approaches such as (Forster, Zhang, Gassner, Werlberger, & Scaramuzza, 2016).

The simplest (but not the smartest) way to acquire metric distances would be by double-integrating the pure accelerometer measurements. If this is accurate enough, one can easily compute the pose-to-pose distance in Figure 3-11. However, such approach of pure IMU integration usually yields unreliable results due to the accumulation of sensor bias and noise (Woodman, 2007; Hu & Chen, 2014). Moreover, the accelerometer measurements are known be very noisy, and this is even worse for small robots, as their bodies are relatively more subject to vibration and most of them do not carry any dampening mechanisms due to their limit in size. This suggests the importance of more elaborate visual-inertial fusion methods.

The state-of-the-art visual-inertial fusion methods can be divided into two main types: **loosely-coupled** and **tightly-coupled** approaches. *Loosely-coupled* approaches (Weiss et al., 2013) consider the VO module as a 'black box' which yields the 6 DOF pose as output, and integrates this result with IMU measurements using EKF to estimate the states, including IMU biases and absolute scale. In contrast, *tightly-coupled* approaches (Mourikis & Roumeliotis, 2007; Leutenegger et al., 2015; Forster, Carlone, Dellaert, & Scaramuzza, 2015)

[5] A typical example is Kinect (`http://www.xbox.com/en-US/xbox-one/accessories/kinect`)
[6] an acronym for Microelectromechanical Systems

jointly estimate the states by including the raw measurements from the camera and IMU directly into a common filter or optimization problem. Studies have shown that tightly-coupled methods are usually more computationally expensive, but yield better estimation results than loosely-coupled methods (Leutenegger et al., 2015; Shen, Michael, & Kumar, 2015).

Tightly-coupled fusion systems can be subdivided into two categories: **filter-based** and **non-linear optimization-based** approaches. The most notable example of the filter-based approach is the *Multi-State Constraint Kalman Filter* (MSCKF) (Mourikis & Roumeliotis, 2007). This approach is *structureless* (i.e., positional tracking is the only important function, and it does not reconstruct the 3D structure of the environment) and is more efficient than EKF in that its computational complexity grows linearly in the number of features, unlike EKF where the complexity grows quadratically. On the other hand, nonlinear optimization-based approaches have proven to outperform the filter-based methods in terms of accuracy, but at the cost of increased computational complexity (Leutenegger et al., 2015; Forster et al., 2015). Recently, a simultaneous state initialization and gyro bias calibration method using a closed-form solution was proposed in (Kaiser, Martinelli, Fontana, & Scaramuzza, 2017).

Although the state-of-the-art visual-inertial fusion methods have demonstrated an impressive level of performance in terms of accuracy and efficiency, both filter-based methods and optimization-based methods have been shown to be subject to convergence towards an inaccurate state-estimation (or even divergence) unless carefully supervised initial movements and/or calibration parameters (e.g., IMU biases, camera-to-IMU calibration and time-synchronization between the two sensors) are provided.

3-2-3 Initialization with a Known Target

One straightforward way to estimate the metric scale is by using a special target object of known size (Davison et al., 2007; Eberli, Scaramuzza, Weiss, & Siegwart, 2011). If an object with known dimension such as a 2D feature pattern is placed in front of the camera at initialization, the object can be recognized using a simple detection algorithm. Subsequently, a point-to-point distance (see Figure 3-11) can be computed from the designated features on the target object, which is then used to initialize the scale. Furthermore, if the distance between the camera and target object is known, the features can be directly added to the map with absolute scale. This can greatly ease the bootstrapping of the VO system.

However, there are few limitations of this approach: First, it requires a special target object which must be easily recognizable and that the object be placed in a certain way during initialization of the system. Moreover, since the target object is used only at initialization, it is prone to scale drift over time.

3-2-4 Stability-based Monocular Distance Estimation

In order to understand the principle behind the stability-based method proposed by (de Croon, 2016), we first need to understand the meaning of the following two terms - **(flow) divergence D** and **visual observable θ_z**.

Divergence and Visual Observable

Using the camera coordinate system shown in Figure 3-12, the equations for divergence D at the optical center are given as follows[7].

$$D \equiv \frac{1}{2}\nabla\vec{q} \quad \left(\text{or } \frac{1}{2}\text{div}\,(\vec{q})\right) \tag{3-5}$$

$$= \frac{1}{2}\left(\frac{\partial u}{\partial x^{im}} + \frac{\partial v}{\partial y^{im}}\right) \tag{3-6}$$

$$= \frac{1}{2}\left(2\hat{V}_Z{}^c + \hat{V}_X{}^c Z_X{}^c + \hat{V}_Y{}^c Z_Y{}^c\right) \tag{3-7}$$

where $\vec{q}$ is an optical flow field at the optical center, $(x^{im}, y^{im}) = \left(f\frac{X^c}{Z^c}, f\frac{Y^c}{Z^c}\right)$ are 2D projection of 3D feature's position vector $(X^c, Y^c, Z^c)^T$ onto the camera's image plane, $(u, v)^T \equiv (\dot{x}^{im}, \dot{y}^{im})^T$ is an optical flow velocity, and $\hat{V}_X{}^c$, $\hat{V}_Y{}^c$, $\hat{V}_Z{}^c$ are ventral flow vectors (i.e. depth-scaled velocities):

$$\hat{V}_X{}^c = \frac{V_X{}^c}{Z^c},\ \hat{V}_Y{}^c = \frac{V_Y{}^c}{Z^c},\ \hat{V}_Z{}^c = \frac{V_Z{}^c}{Z^c} \tag{3-8}$$

For the full derivation of Equation (3-7), refer to (Longuet-Higgins & Prazdny, 1980) and (Subbarao, 1990). Assuming planar scene structure and zero translational velocity parallel to the image plane (i.e. $V_X{}^c = V_Y{}^c = 0$ and $Z^w = Z^c$), Equation (3-7) becomes:

$$D = \frac{V_Z{}^c}{Z^c} = \frac{V_Z{}^c}{Z^w} \tag{3-9}$$

Since the z-axes of the world reference frame and camera coordinate system are in the opposite direction, Equation (3-9) can be written as:

$$D = -\frac{V_Z{}^w}{Z^w} \tag{3-10}$$

Then, visual observable θ_z is defined as:

$$\theta_z \equiv -D = \frac{V_Z{}^w}{Z^w} \tag{3-11}$$

Thus, $D > 0$ and $\theta_z < 0$ when the MAV descends vertically.

With this background knowledge, there are two obvious ways to compute divergence:

1. Directly plug in the estimated value of $V_Z{}^w$ and Z^w into Equation (3-10).

[7] According to Equation (3-5), this is in fact a *half-divergence*. It is defined this way to follow the same convention as in the original work (de Croon, 2016)

2. Estimate the partial derivatives of the optical flow velocities at the optical center, and use Equation (3-6).

The problem with the first method is that the metric values of ${V_Z}^w$ and Z^w cannot be obtained with a monocular camera due to the intrinsic unobservability of the absolute scale. On the other hand, the problem with the second method is that the direct estimation of u_x and v_y is highly unpractical because it relies on the differentiation of a single measurement of optical flow velocities, which means that it would be extremely sensitive to the noise. Besides, there is no guarantee that a reliable feature will be detected precisely at the optical center and tracked accurately in the next frames. In (de Croon, 2016) and (Ho, Croon, Kampen, Chu, & Mulder, 2016), these problems are circumvented by using the 'size divergence' - they measure the size of lines between features in consecutive frames, and take the median or average of the estimates. Interested readers are referred to (Ho et al., 2016) for the mathematical explanation of this method.

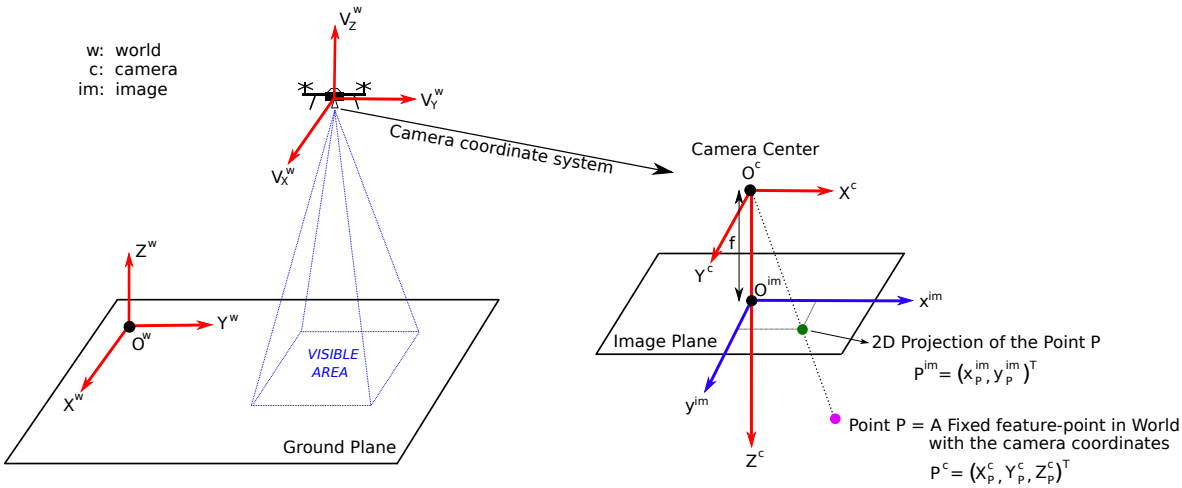


Figure 3-12: Reference frames employed in (de Croon, 2016). **Left**: World reference frame and MAV's velocity with respect to this frame. **Right**: 3D Camera and 2D image coordinate system. f is the focal length of the camera, which represents the distance between the camera center and image plane. Note that the positive Z^c direction is the same as the camera view direction, which is downward in this case.

Stability-based Strategy for Monocular Distance Estimation

In (de Croon, 2016), a stability-based method of estimating distance is proposed for quadrotor MAVs with only a downward-looking monocular camera. This strategy exploits the unique self-induced oscillation behavior of quadrotors that typically occurs during a constant divergence landing (i.e., thrust command is controlled such that D is kept constant). This strategy is motivated by the analytical evidence that the control gain K_z which starts causing the instability is linearly proportional to the distance to the ground surface z from which the flow divergence is estimated:

$$K_z = \frac{2}{T} z \tag{3-12}$$

where T is the time step used for discretization of the system with Zero-order Hold (ZOH). Such linear relationship was shown to hold even in the presence of wind, drag and time delay. Equation (3-12) implies that, at the onset of the oscillation, the knowledge of K_z can directly provide the information on z.

Three possible ways are proposed in the original work to estimate the oscillation-inducing gain K_z:

1. **Fixed-gain landing approach**: During a constant divergence landing, a fixed control gain K_z will start causing vertical oscillations when a certain height is reached. This oscillation can be detected by evaluating the covariance between the thrust and observed estimate of θ_z given at time t and window W:

$$\mathrm{cov}_t(u'_z, \hat{\theta}_z) = \sum_{i=t-W+1}^{t} \left(u'_z(i) - \overline{u'_z}\right)\left(\hat{\theta}_z(i) - \overline{\hat{\theta}_z}\right) \tag{3-13}$$

   where $\hat{\theta_z}$ is a delayed estimation of true θ_z, and u'_z is the upward thrust command in Newton which follows the control law $u'_z = K_z(\theta^*_z - \theta_z)$ with the set-point θ^*_z. If this (negative) covariance falls below a certain threshold, one can regard it as the onset of oscillations. In (Ho et al., 2016), a more robust oscillation detection method is proposed, which examines the windowed covariance of divergence and half-period-shifted divergence. However, this method requires the prerequisite knowledge of the resonance frequency which can be obtained from the Discrete Fourier Transform (DFT).

2. **Hovering approach**: While hovering, K_z is varied through adaptive gain control to induce oscillations at a given height. The detection of oscillations is identical to the previous approach.

3. **Landing-on-the-edge-of-oscillation approach**: This approach is similar to the hovering approach, except that the adaptive gain control keeps $\mathrm{cov}_t(u'_z, \hat{\theta}_z)$ and set-point θ^*_z at fixed negative values. This enables the MAV to descend "on the edge of oscillation" and estimate the height continuously during the landing.

The hovering approach is particularly interesting because it allows for the distance estimation without necessitating large movements like in the case of most monocular visual-inertial approaches. Instead, small vertical oscillations would suffice. Another advantage over visual-inertial methods is the simplicity of the algorithm, as you do not have to go through a tedious sensor calibration and synchronization process which is often required by visual-inertial fusion algorithms. On the other hand, the limitation of this stability-based strategy is that prior knowledge of the linear relationship $z = aK_z + b$ is required to estimate z based on K_z. In (de Croon, 2016), this knowledge was obtained by fitting a linear model to the data measured by an Optitrack motion tracking system.

Chapter 4

Methodology

In this chapter, a novel solution is proposed to estimate the absolute scale for monocular VO of quadrotor MAVs - that is, **stability-based scale estimation**. The proposed method is elaborated in the context of a complete VO pipeline in Section 4-1. In Section 4-2, the set-up for simulations and real-world experiments is also explained.

4-1 Proposed solution

4-1-1 Motivation and Contribution

As mentioned in Section 3-2-4, in the original work (de Croon, 2016; Ho et al., 2016), the divergence D is estimated from the 2D optical flow vectors in the image input provided by a downward-looking camera. This approach has several limitations, such as:

- The camera must be downward-facing.
- The ground visible to the camera must be planar and highly textured.
- A few inaccurately tracked features can have large influence on the divergence estimation, so some form of outlier rejection (e.g., Random Sample Consensus (RANSAC) (Fischler & Bolles, 1981)) should be implemented.
- Horizontal movements (e.g. caused by wind, etc.) must be kept minimal to ensure accurate estimation of divergence.

In this work, we propose a new method of computing the divergence by directly using the unscaled 3D position and velocity output from a monocular VO system. This exploits the following relations:

$$D = -\frac{{V_Z}^w}{Z^w} = -\frac{(1/\lambda)\cdot {V_Z}^w}{(1/\lambda)\cdot Z^w} = -\frac{{V_Z}^{vo}}{Z^{vo}} \tag{4-1}$$

where λ is an unknown metric scale factor which converts the arbitrarily-scaled translational velocity and position vectors (e.g., ${V_Z}^{vo}$ and Z^{vo}) from the monocular VO into metric values (e.g., ${V_Z}^{w}$ and Z^{w}). **The novelty of this idea lies in the fact that the knowledge of the scale λ is never required for the estimation of divergence, even though we are still using the unscaled results from VO** - it simply cancels out itself as shown in Equation (4-1).

At the cost of increased computational complexity, this method of divergence estimation overcomes the limitations of optical-flow-based approach mentioned earlier:

- The camera does not necessarily have to be downward-looking. It can be either downward-looking or forward-looking, depending on the environment and use-case. For example, you might prefer using a downward-looking camera for outdoor applications, and forward-looking camera for indoor navigation.
- Even if a downward-looking camera is used, the ground does not have to be planar. If a forward-looking camera is used, this approach is affected by neither ground texture nor its geometric structure.
- A few outliers in feature tracking will not significantly affect the accuracy of divergence estimation unless the entire VO system fails and loses tracking completely.
- Horizontal movements (e.g. caused by wind, etc.) will not affect the divergence estimation.

Using Equation (4-1) for the computation of divergence will hopefully improve the accuracy of the stability-based distance estimation strategy, since we can now directly use the accurate estimation of unscaled ${V_Z}^{vo}$ and Z^{vo} given by the state-of-the-art VO/SLAM system. Then, one can easily employ the stability-based distance estimation technique proposed to compute the absolute scale of a monocular VO by following a simple procedure:

1. Initialize a monocular VO (by manually moving/flying the MAV with gentle movements).
2. Place the MAV on the ground, and set $Z^{vo} = 0$.
3. Take off and hover at a random height A. This corresponds to ${H_A}^{w}$ in meters (unknown) and ${Z_A}^{vo}$ in arbitrary VO scale (known).
4. Estimate H_A^w in meters while hovering, using the technique proposed in (de Croon, 2016).
5. Compute the scale factor λ using:
$$\lambda = \frac{{H_A}^{w}}{{Z_A}^{vo}} \tag{4-2}$$
6. Scale the translational position and velocity results from the VO system using λ.
7. Use these scaled states for autonomous control and navigation of the MAV.

This method has several advantages over the other scale estimation methods mentioned in Section 3-2. First of all, it does not require any range sensors. Therefore, it can also be used for MAVs without any additional sensors other than a camera. Second of all, this method is more robust to inaccurate calibration parameters compared to visual-inertial fusion methods,

although the state-estimation results may not be as accurate when initialized with a very good calibration parameters. As discussed previously in Section 3-2-2, it is a known issue that visual-inertial fusion methods are often subject to convergence towards a wrong local minimum (or even divergence) given a poor initialization. Finally, the stability-based method does not require any special target with known dimension for initialization. This allows for remote operations where the ground-station is located far away from the MAV's flight area. Moreover, it enables reinitialization at any position during the flight - basically, all you need to do is hover and estimate the height again with respect to the initial ground level (i.e., where $Z^{vo} = 0$). This would not be possible if you initialized the scale with a special object with known dimension, unless you manually place the object in front of the camera again for reinitialization. The fact that the stability-based scale estimation can be performed at any time and location during the flight suggests that it is also possible to minimize the scale drift by correcting the scale multiple times at different positions.

4-1-2 Proposed System Overview

The pipeline of the proposed system is illustrated in Figure 4-1. In the figure, the monocular VO and stability-based height estimation modules are regarded as 'black boxes' for convenience. Although it is not explicitly shown in the figure, one should keep in mind that the stability-based height estimation module will be using the vertical controller implemented in PaparazziUAV autopilot. In fact, it is more appropriate to consider it as a part of the autopilot system. The figure is drawn like this in order to emphasize the use of the (scaled) estimation of states provided by the monocular VO system for replacing GPS-signals.

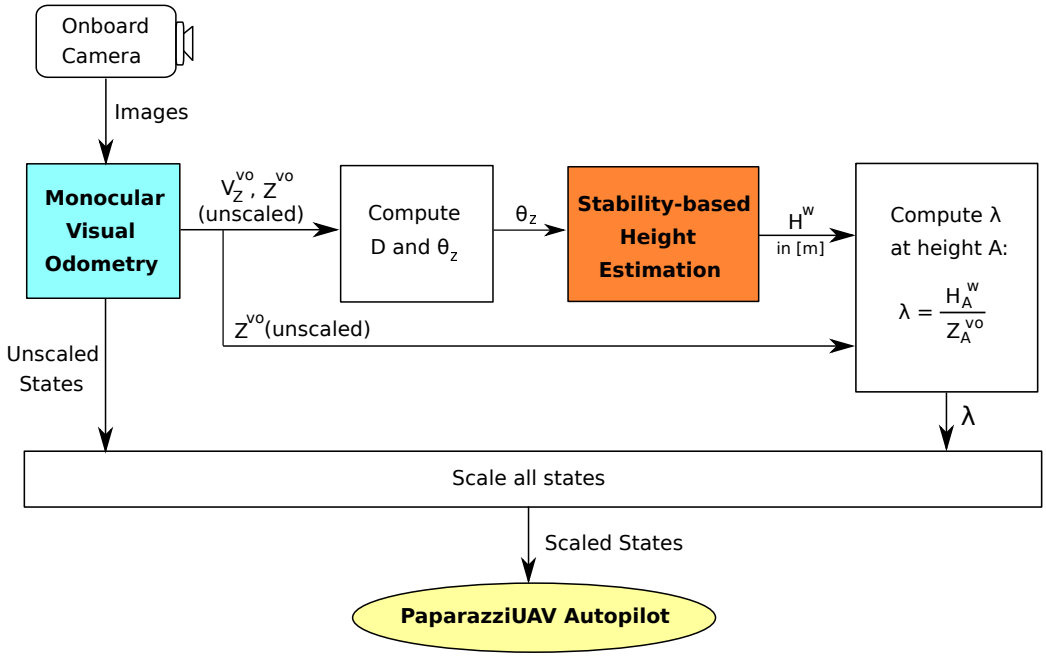


Figure 4-1: Proposed System Pipeline: Colored modules are treated as 'black box' for simplification. ${H_A}^w$ is the height in meters the MAV hovered around and estimated. ${V_Z}^{vo}$ and ${Z}^{vo}$ are unscaled vertical position and speed given by the VO system, respectively. ${Z_A}^{vo}$ denotes the corresponding ${Z}^{vo}$ at ${H_A}^w$.

4-2 Experimental set-up

In this section, a planning for the set-up of simulation, hardware and real-world experiments is presented.

4-2-1 Hardware

The following hardware will be used for simulation and real-world experiments:

- Laptop: Lenovo ThinkPad W541, i7-4810MQ at 2.80GHz, 15GB RAM
- SBC: Odroid XU4[1], Exynos5422 Octa core CPUs (A15 Quad 2.0GHz and A7 Quad 1.4GHz), 2GB RAM
- Camera: mvBlueFOX-MLC200w[2] (global shutter camera)
- Drone: Mavtec

4-2-2 Simulation set-up

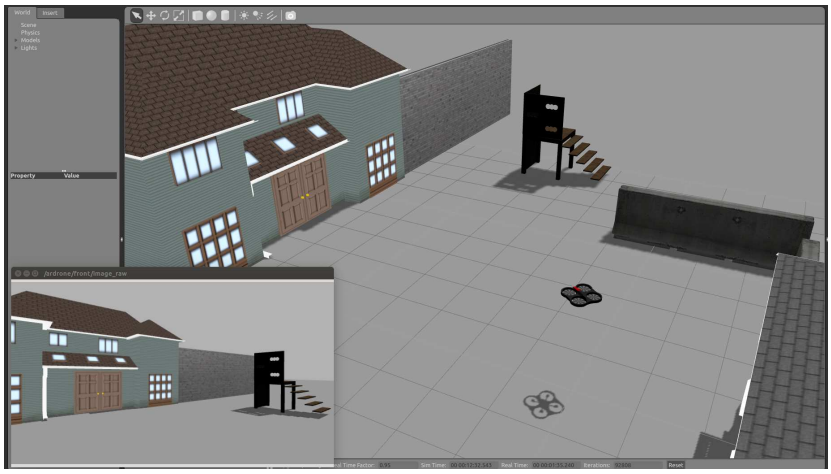


Figure 4-2: Visual output from tum_simulator. **Main window:** Simulated 3D world with AR-Drone model. **Bottom right window:** Image stream from the forward-looking camera on the drone.

The simulation of the complete system (which will be written in C++) will be carried out in ROS[3] environment. The simulation set-up consists of the following packages:

1. **tum_simulator** (Huang & Strum, 2013) is used as the main simulation platform (shown in Figure 4-2). This package is based on:

[1] `http://www.hardkernel.com`
[2] `https://www.matrix-vision.com/USB2.0-single-board-camera-mvbluefox-mlc.html`
[3] Robot Operating System, `http://www.ros.org/about-ros/`

(a) **tu-darmstadt-ros-pkg** (Meyer & Kohlbrecher, 2012) contains various simulation packages (e.g., Gazebo[4] plugins and models) necessary for robot simulation in ROS.

(b) **ardrone_autonomy** (Monajjemi, 2012): This is a ROS driver for Parrot AR-Drone 1.0 and 2.0, based on official AR-Drone SDK.

2. Monocular visual odometry will be run as an independent module that processes the image input provided by *tum_simulator* to generate the state output.

The stability-based control and estimation algorithm will take the state-estimation results from the monocular VO module as input and generate thrust commands. These commands will be used in the quadrotor controller of *tum_simulator*. A preliminary simulation of the stability-based scale estimation is implemented with a fixed-gain constant-divergence-landing using PTAM, and the playback can be accessed via `https://www.youtube.com/watch?v=jFx5mfMZ0aI`.

4-2-3 Real-world experiments set-up

The state-estimation results from the VO module will be recorded using the logger in PaparazziUAV. This will be compared with the ground-truth of the flight trajectory, which will be measured using the OptiTrack[5] motion capture system which is installed in CyberZoo flight arena [6] of TU Delft Aerospace Engineering faculty.

[4] `http://gazebosim.org/`
[5] `http://optitrack.com/`
[6] 10m × 10m × 7m dimension

Chapter 5

Preliminary Analysis

In this chapter, a preliminary analysis of the most relevant VO systems and datasets is presented. In Section 5-1, an overview of the five test datasets (i.e., **SW**, **TUM RGBD**, **TUM MonoVO**, **EuRoC MAV** and **KITTI**) is provided. In Section 5-2, the five selected VO/SLAM systems (i.e., **Modified PTAM**, **SVO**, **Sparsified DPPTAM**, **ORB-SLAM** and **DSO**) are evaluated qualitatively in terms of tracking ability. Note that MonoSLAM, LSD-SLAM and DPPTAM are not considered here because they are either filter-based or not sparse (see Section 3-1-1 for the justification of design choice).

5-1 Overview of Dataset

In total, five different datasets were used to test the VO/SLAM systems on a laptop and SBC (for hardware specifications, see Section 4-2-1):

1. **Small Workspace dataset**[1] (for convenience, an acronym **"SW dataset"** will be used hereafter) which I created by myself using a commercially available webcam[2] in a small workspace environment. Sample images from this dataset are provided in Figure 5-1.

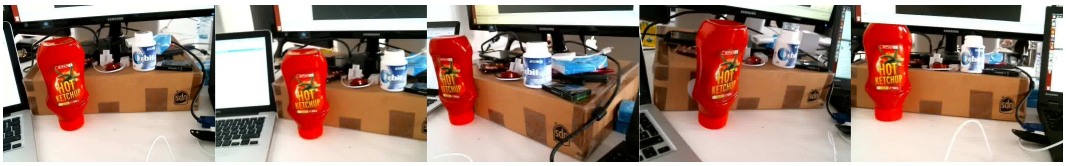


Figure 5-1: Sample images from one of the sequences of SW dataset. Note that there is no significant viewpoint change, as the camera's pointing direction is kept mostly unidirectional.

[1] Available in: `https://drive.google.com/drive/folders/0B7ka7aXEpgXKdi1qYWU2aTdzY2c?usp=sharing`. One of the image sequences from this dataset is used in Figure 3-7.

[2] Logitech C615

2. **TUM RGBD dataset** (Sturm, Engelhard, Endres, Burgard, & Cremers, 2012): This dataset contains large number of RGB-D benchmark sequences and ground-truth trajectory data. The following three sequences are used for testing:
   - *freiburg2_desk*[3]
   - *freiburg3_structure_texture_far*
   - *freiburg3_long_office_household*[4]

   Sample images from *freiburg2_desk* sequence are shown in Figure 5-2.

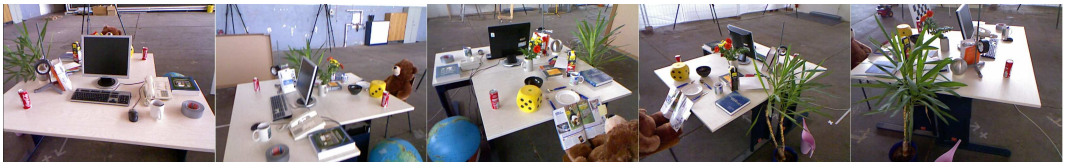


Figure 5-2: Sample images from *freiburg2_desk* sequence. The camera trajectory is room-scale.

3. **TUM MonoVO dataset** (Engel & Cremers, 2016): This dataset contains photometrically-calibrated sequences (i.e. taking account of the exposure times, and compensating the camera response function and lens attenuation factors) which were recorded in various indoor and outdoor environments. The following sequence is used for testing:
   - *sequence_42*

   Sample images from this sequence are shown in Figure 5-3.

Figure 5-3: Sample images from *sequence_42* of TUM MonoVO dataset. The camera trajectory is moderately large (i.e., approximately 3 minutes-long walk).

4. **EuRoC MAV dataset** (Burri et al., 2016): This dataset contains indoor image sequences collected on-board a MAV, as well as the accurate motion and structure ground-truth. The following sequences are used for testing:
   - *MH_01_easy*
   - *V1_01_easy*[5]
   - *V2_02_medium*

   Sample images from *MH_01_easy* sequence are shown in Figure 5-4.

[3]This sequence was used in Figure 3-8.
[4]This sequence was used in Figure 3-9.
[5]This sequence was used in Figure 3-10.

Figure 5-4: Sample images from *MH_01_easy* sequence. The camera trajectory is moderately large (i.e., 3 minutes-long indoor flying with a hex-rotor helicopter).

5. **KITTI odometry dataset** (Geiger, Lenz, & Urtasun, 2012): This dataset contains large-scale image sequences and ground-truth data collected on a driving car. The following sequences are used for testing:
    - *sequence_06*
    - *sequence_11*

Sample images from *sequence_11* are shown in Figure 5-5.

Figure 5-5: Sample images from *sequence_11*. The total camera trajectory and inter-frame motions are very large (i.e., approximately 2 minutes-long drive and 10 Hz frame rate).

Note that only the first two dataset contain images captured by rolling shutter cameras, while the other three dataset used global shutter cameras. The frame per second (fps) of the datasets is 30, 30, 22, 20 and 10, respectively. The easiest image sequence to track is the SW dataset, as the camera movement is generally slow, and the viewpoint is mostly unidirectional. It also contains sufficient translational movements to ensure accurate triangulation. TUM RGBD dataset is the second easiest dataset to track, containing mostly large parallax movements. TUM MonoVO dataset and EuRoC MAV dataset are more difficult to track compared to the first two dataset due to stronger rotations and faster movements. KITTI odometry dataset is the hardest to track, as its inter-frame movement is much larger than the others. Moreover, the most dominant motion is the forward motion, which makes triangulation more challenging due to the low parallax angle.

5-2 Qualitative Evaluation of VO systems

In this section, the qualitative evaluation of the five open-source VO/SLAM systems (PTAM, SVO, Sparsified DPPTAM, ORB-SLAM and DSO) is provided. Using the datasets mentioned previously, the tests were carried out on a workstation laptop and SBC respectively (see Section 4-2-1 for hardware specifications). The reason for the evaluation being qualitative is because the capabilities of the VO/SLAM systems to robustly track the camera pose are simply dichotomous for each dataset - it either tracks or not. In other words, the performance metric used in this case is the tracking robustness, rather than accuracy or efficiency. The results are summarized as follows:

PTAM

The tracking was only successful on SW dataset. This was true for both times when it was run on the laptop and SBC, as the system could still run fast enough to track 30 fps video without a problem on Odroid XU4. However, PTAM is not suitable for when large occlusion is introduced with large viewing angles (e.g., 360° movement around a target object). Also, the initialization is brittle, and it can easily fail when the scene is highly non-planar (due to homography-based initialization) or when inappropriate motion (i.e. fast, discontinuous, rotational or strong non-parallel motions) is provided.

SVO

SVO could not track any of the provided image sequences. The only occasion that SVO ran successfully was when it was given the rosbag file provided by the author. The rosbag file contains images that were taken by a downward-facing large Field of View (FOV) global shutter camera at 30 fps, moving in a slow circular motion parallel to the scene.

Sparsified DPPTAM

Sparsified DPPTAM could track and map the image sequences from SW dataset robustly and accurately, producing a good quality 3D reconstruction of the scene. As for TUM RGBD dataset, it could handle *freiburg2_ desk* and *freiburg3_ structure_ texture_ far* sequences fairly well, but not *freiburg3_ long_ office_ household* sequence due to the large occlusion and strong rotational motions. Also, the sparsified DPPTAM failed to track the other three datasets. As a sidenote, the original DPPTAM only worked with SW dataset.

ORB-SLAM

ORB-SLAM performed robustly on all datasets provided. However, it ran too slowly on the SBC, achieving the tracking rate around 3 fps on Odroid XU4. One way to increase the tracking speed would be by deactivating the loop-closure thread, turning it into a pure VO system. The question as to whether this would be enough to make ORB-SLAM run in real-time on the SBC still remains open.

DSO

Like ORB-SLAM, DSO also performed robustly on all datasets provided. The initialization was not perfectly smooth on *KITTI_06* sequence, as it attempted the initialization multiple times for the first few seconds. Once the initialization was complete, the system was bootstrapped successfully and performed well for the rest of the time. It was not possible to compile DSO on the SBC out-of-the-box, because some functions use SSE-optimized processing, which is not supported on ARM processors.

Summary and Discussion

The results on the workstation laptop are summarized in Table 5-1. SVO clearly showed the worst performance, and thus, will not be considered from here on. The sparsified DPPTAM cannot be used for the final thesis either because it is closed-source, and it can only serve as a reference for comparison with other VO systems. Therefore, the final decision on the VO system must be made between PTAM, ORB-SLAM and DSO. Since only PTAM worked out-of-the-box on Odroid, it seems only logical to choose PTAM as a basis at first, and consider the other two systems if and only if the performance of PTAM is shown to be unsatisfactory for our specific purpose - that is, stability-based scale initialization. If that turns out to be the case, then the final decision will be made between ORB-SLAM and DSO based on practicality, such as time-constraints.

Table 5-1: A summary of the qualitative evaluation of the five VO/SLAM systems on the workstation laptop.

		Datasets				
		SW	**TUM RGBD**	**TUM MonoVO**	**EuRoC MAV**	**KITTI**
VO/ SLAM	**Modified PTAM**	YES	NO	NO	NO	NO
	SVO	NO	NO	NO	NO	NO
	Sparsified DPPTAM	YES	Two YES and One NO	NO	NO	NO
	ORB-SLAM	YES	YES	YES	YES	YES
	DSO	YES	YES	YES	YES	YES

Chapter 6

Planning

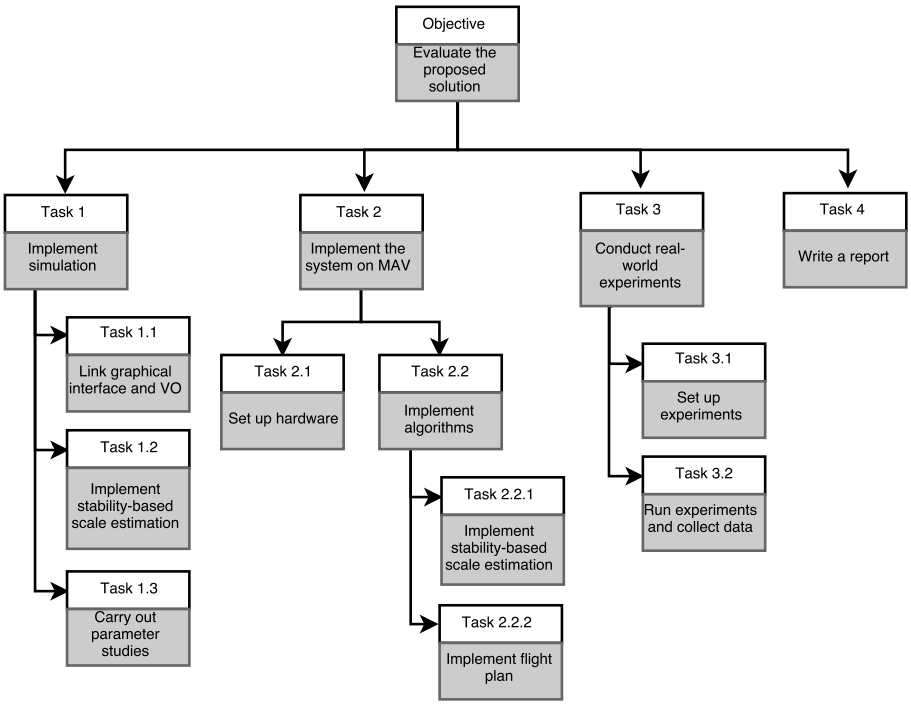


Figure 6-1: Top-down project task tree

In this chapter, the planning for the remaining months is presented. First, the top-down project task tree (also known as 'work breakdown structure') is given in Figure 6-1. Note that the task numbering is slightly different from the one in Section 2-2, as some of the tasks are broken down into smaller sub-tasks, or grouped together as one task unit. Also, due to the page limit, the verification and validation tasks are not shown in the figure. Instead, these steps are implied for all tasks expressed by the infinitive *"Implement"*. The order in which the tasks are listed in Figure 6-1 are not necessarily the sequential order for execution. The logistics of the task execution is elaborated and visualized in the Gantt Chart (Figure 6-2).

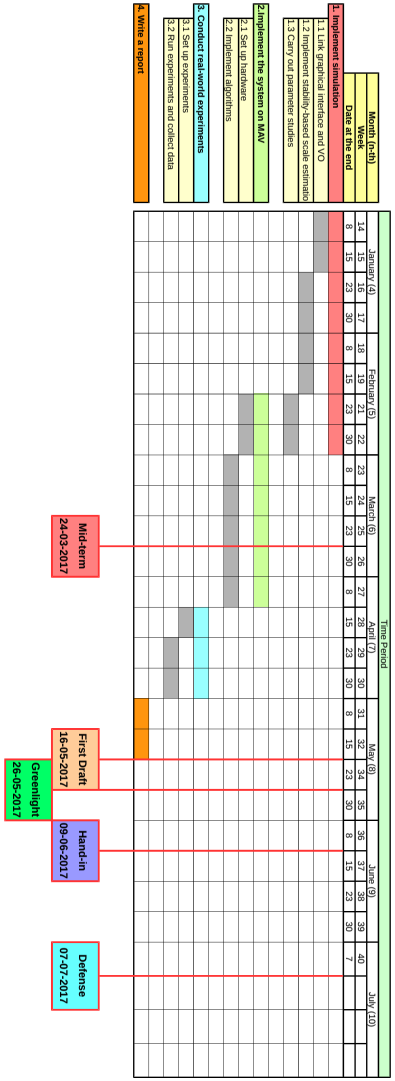


Figure 6-2: Gantt Chart with project milestones

Chapter 7

Conclusions

In this report, a preliminary literature study is given on the topic of state-of-the-art monocular visual odometry and metric scale estimation methods that are applicable for vision-based GNC of MAVs. In order for MAVs with a monocular camera to fly and navigate autonomously in unknown GPS-denied environments, it is important to estimate the absolute (metric) scale in one way or another. In that respect, a novel stability-based scale estimation method is proposed which can be readily implemented in conjunction with any existing monocular VO/SLAM system. The proposed solution neither requires any additional sensor modalities other than a monocular camera, nor a special visual target for initialization. Furthermore, the proposed method replaces the optical-flow-based estimation of divergence (as done in the original work (de Croon, 2016)) with the direct computation using unscaled output from a VO/SLAM system. This has several functional advantages, as well as potential performance gains.

Besides the literature survey on the state-of-the-art methods, a preliminary analysis of the open-source VO systems is provided. It suggests that the most suitable one at the moment is the modified PTAM (Weiss, Achtelik, et al., 2012). However, it should be noted that this is only a preliminary decision, and potential alternatives will be considered in case of unsatisfactory real-world performance: for example, loop-closure-deactivated ORB-SLAM (Mur-Artal et al., 2015) or DSO (Engel et al., 2016a).

In the following months, a complete simulation of the proposed scale estimation strategy will be designed and tested. A preliminary simulation of the stability-based scale estimation has already been implemented in a fixed-gain constant-divergence-landing scenario using the modified PTAM, and the result seems to be promising. Once the full system is thoroughly verified, the method will be implemented on a custom-built quadrotor MAV to demonstrate autonomous indoor navigation and flight.

Bibliography

Anonymous. (1997). *Titanic Fairlead Miniature.* http://www.thepropgallery.com/fairlead-miniature.

Baker, S., & Matthews, I. (2004). Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, *56*(3), 221-255.

Benhimane, S., & Malis, E. (2007, July). Homography-based 2D visual tracking and servoing. *International Journal of Robotics Research*, *26*(7), 661–676. Available from http://dx.doi.org/10.1177/0278364907080252

Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., et al. (2016). The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research*. Available from http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract

Civera, J., Davison, A., & Montiel, J. (2008). Inverse depth parametrization for monocular SLAM. *IEEE Trans. Robot. IEEE Transactions on Robotics*, *24*(5), 932-945.

Cole, D., & Newman, P. (2006, May). Using Laser Range Data for 3D SLAM in Outdoor Environments. In *IEEE International Conference on Robotics and Automation (ICRA).*

Concha, A., & Civera, J. (2015). DPPTAM: dense piecewise planar tracking and mapping from a monocular sequence. In *2015 IEEE/RSJ international conference on intelligent robots and systems, IROS 2015, hamburg, germany, september 28 - october 2, 2015* (pp. 5686–5693). Available from http://dx.doi.org/10.1109/IROS.2015.7354184

Davison, A. J., Reid, I. D., Molton, N. D., & Stasse, O. (2007). MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence IEEE Trans. Pattern Anal. Machine Intell.*, *29*(6), 1052-1067.

de Croon, G. C. H. E. (2016). Monocular distance estimation with optical flow maneuvers and efference copies: a stability-based strategy. *Bioinspiration & Biomimetics*, *11*(1).

Disario, J. (1997). *Titanic Model.* http://jeffdisario.com/project/titanic-beauty-model-crew/.

Eberli, D., Scaramuzza, D., Weiss, S., & Siegwart, R. (2011). Vision based position control for mavs using one single circular landmark. *Journal of Intelligent & Robotic Systems*, *61*(1), 495–512. Available from http://dx.doi.org/10.1007/s10846-010-9494-8

Engel, J., & Cremers, D. (2016). *Monocular Visual Odometry Dataset.* https://vision.cs.tum.edu/data/datasets/mono-dataset.

Engel, J., Koltun, V., & Cremers, D. (2016a, July). Direct sparse odometry. In *arxiv:1607.02565.*

Engel, J., Koltun, V., & Cremers, D. (2016b). A photometrically calibrated benchmark for monocular visual odometry. *CoRR*, *abs/1607.02555*. Available from http://arxiv.org/abs/1607.02555

Engel, J., Schöps, T., & Cremers, D. (2014, September). LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision (ECCV).*

Engel, J., Sturm, J., & Cremers, D. (2012). Camera-based navigation of a low-cost quadrocopter. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Engel, J., Sturm, J., & Cremers, D. (2013). Semi-dense visual odometry for a monocular camera. *2013 IEEE International Conference on Computer Vision*.

Faugeras, O. D., & Lustman, F. (1988). Motion and Structure From Motion in a Piecewise Planar Environment [Mosaic]. *Intern. J. of Pattern Recogn. and Artific. Intelige.*(3), 485-508.

Felzenszwalb, P. F., & Huttenlocher, D. P. (2004, September). Efficient graph-based image segmentation. *International Journal of Comput. Vision*, *59*(2), 167–181. Available from http://dx.doi.org/10.1023/B:VISI.0000022288.19776.77

Fischler, M. A., & Bolles, R. C. (1981, June). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, *24*(6), 381–395. Available from http://doi.acm.org/10.1145/358669.358692

Forster, C., Carlone, L., Dellaert, F., & Scaramuzza, D. (2015, July). IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In *Proceedings of robotics: Science and systems.* Rome, Italy.

Forster, C., Pizzoli, M., & Scaramuzza, D. (2014, May). SVO: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (p. 15-22).

Forster, C., Zhang, Z., Gassner, M., Werlberger, M., & Scaramuzza, D. (2016). Semi-direct visual odometry for monocular, wide-angle, and multi-camera systems. *IEEE Transactions on Robotics*.

Galvez-Lopez, D., & Tardos, J. D. (2012, October). Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, *28*(5), 1188–1197.

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR).*

Ho, H. W., Croon, G. C. H. E. de, Kampen, E. van, Chu, Q., & Mulder, M. (2016). Adaptive control strategy for constant optical flow divergence landing. *CoRR*, *abs/1609.06767*. Available from http://arxiv.org/abs/1609.06767

Hu, J. S., & Chen, M. Y. (2014, May). A sliding-window visual-imu odometer based on tri-focal tensor geometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (p. 3963-3968).

Huang, H., & Strum, J. (2013). *tum_simulator - ROS Wiki.* http://wiki.ros.org/tum_simulator.

Huber, P., Wiley, J., & InterScience, W. (1981). *Robust statistics*. Wiley New York.

Kaiser, J., Martinelli, A., Fontana, F., & Scaramuzza, D. (2017, Jan). Simultaneous state

initialization and gyroscope bias calibration in visual inertial aided navigation. *IEEE Robotics and Automation Letters*, *2*(1), 18-25.

Kerl, C., Stueckler, J., & Cremers, D. (2015). Dense continuous-time tracking and mapping with rolling shutter RGB-D cameras. In *IEEE International Conference on Computer Vision (ICCV)*.

Klein, G., & Murray, D. (2007, November). Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*. Nara, Japan.

Kuemmerle, R., Grisetti, G., Strasdat, H., Konolige, K., & Burgard, W. (2011, May). g2o: A general framework for graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (pp. 3607–3613). Shanghai, China.

Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., & Furgale, P. T. (2015). Keyframe-based visual-inertial odometry using nonlinear optimization. *International Journal of Robotics Research*, *34*(3), 314–334. Available from `http://dx.doi.org/10.1177/0278364914554813`

Longuet-Higgins, H. C., & Prazdny, K. (1980). The Interpretation of a Moving Retinal Image. *Royal Society of London Proceedings Series B*, *208*, 385–397.

Lovegrove, S., Davison, A. J., & Ibanez-Guzman, J. (2011). Accurate visual odometry from a rear parking camera. *2011 IEEE Intelligent Vehicles Symposium (IV)*.

Meyer, J., & Kohlbrecher, S. (2012). *tu-darmstadt-ros-pkg - ROS Wiki.* `http://wiki.ros.org/tu-darmstadt-ros-pkg/#Quadrotor_Simulation_and_Control`.

Monajjemi, M. (2012). *ardrone_autonomy - Official Documentation.* `http://ardrone-autonomy.readthedocs.io/en/latest/`.

Mourikis, A. I., & Roumeliotis, S. I. (2007). A multi-state constraint kalman filter for vision-aided inertial navigation. *Proceedings 2007 IEEE International Conference on Robotics and Automation*.

Mur-Artal, R., Montiel, J. M. M., & Tardos, J. D. (2015). ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Trans. Robot. IEEE Transactions on Robotics*, *31*(5), 1147-1163.

Newcombe, R. A., Lovegrove, S. J., & Davison, A. J. (2011). DTAM: Dense tracking and mapping in real-time. *2011 International Conference on Computer Vision*.

Pizzoli, M., Forster, C., & Scaramuzza, D. (2014). REMODE: Probabilistic, monocular dense reconstruction in real time. *2014 IEEE International Conference on Robotics and Automation (ICRA)*.

Rosten, E., & Drummond, T. (2006, May). Machine learning for high-speed corner detection. In *European Conference on Computer Vision* (Vol. 1, p. 430-443). Available from `"http://www.coxphysics.com/work/rosten_2006_machine.pdf"`

Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). ORB: An Efficient Alternative to SIFT or SURF. In *Proceedings of the 2011 International Conference on Computer Vision* (pp. 2564–2571). Washington, DC, USA: IEEE Computer Society. Available from `http://dx.doi.org/10.1109/ICCV.2011.6126544`

Scaramuzza, D. (2016). *A Tutorial on Visual Odometry.* `http://mrsl.grasp.upenn.edu/loiannog/tutorial_ICRA2016/VO_Tutorial.pdf`.

Scaramuzza, D., & Fraundorfer, F. (2011). Visual odometry part i: the first 30 years and fundamentals.

Shen, S., Michael, N., & Kumar, V. (2015). Tightly-coupled monocular visual-inertial fusion

for autonomous flight of rotorcraft MAVs. In *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015* (pp. 5303–5310). Available from http://dx.doi.org/10.1109/ICRA.2015.7139939

Stewénius, D. H., Engels, C., & Nistér, D. D. (2006). Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, *60*(4), 284–294.

Strasdat, H., Davison, A. J., Montiel, J. M. M., & Konolige, K. (2011). Double window optimisation for constant time visual SLAM. In *Proceedings of the 2011 International Conference on Computer Vision* (pp. 2352–2359). Washington, DC, USA: IEEE Computer Society. Available from http://dx.doi.org/10.1109/ICCV.2011.6126517

Strasdat, H., Montiel, J. M. M., & Davison, A. J. (2010, May). Real-time monocular SLAM: Why filter? In *2010 IEEE International Conference on Robotics and Automation* (p. 2657-2664).

Sturm, J., Engelhard, N., Endres, F., Burgard, W., & Cremers, D. (2012, Oct.). A Benchmark for the Evaluation of RGB-D SLAM Systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS).*

Subbarao, M. (1990). Bounds on time-to-collision and rotational component from first-order derivatives of image flow. *Computer Vision, Graphics, and Image Processing*, *50*(3), 329 - 341. Available from http://www.sciencedirect.com/science/article/pii/0734189X9090151K

Vogiatzis, G., & Hernandez, C. (2011). Video-based, real-time multi-view stereo. *Image and Vision Computing*, *29*(7), 434-441.

Weiss, S., Achtelik, M., Lynen, S., Achtelik, M., Kneip, L., Chli, M., et al. (2013). Monocular vision for long-term micro aerial vehicle state estimation: A compendium. *Journal of Field Robotics*, *30*(5), 803-831.

Weiss, S., Achtelik, M. W., Lynen, S., Chli, M., & Siegwart, R. (2012, May). Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments. In *2012 IEEE International Conference on Robotics and Automation* (p. 957-964).

Weiss, S., Siegwart, R., & Kumar, V. (2012). *Vision based navigation for micro helicopters.* ETH. Available from https://books.google.nl/books?id=FgBaMwEACAAJ

Woodman, O. J. (2007, August). *An introduction to inertial navigation* (Tech. Rep. No. UCAM-CL-TR-696). University of Cambridge, Computer Laboratory. Available from http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf