Stability-Based Scale Estimation for Monocular SLAM

Seong Hun Lee and Guido de Croon

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

***Index Terms*—Aerial systems, perception and autonomy, visual-based navigation, SLAM.**

I. Introduction

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

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. Moreover, the recent development of monocular visual odometry (VO) and SLAM methods such as [4]–[6] 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 metric scale of the reconstructed scene and camera motion is inherently unobservable. Without supplementary metric information, the pose estimates from the SLAM system are "unscaled". A naive use of unscaled states for control can easily lead to degraded performance or even instability. Traditional methods therefore employ additional metric sensors, such as an inertial measurement unit (IMU) [7] or a sonar altimeter [3] to estimate the metric scale.

In this work, we propose a novel, stability-based, adaptive method to solve the scale ambiguity in monocular SLAM. Inspired by the previous work [8], our method estimates the scale by exploiting the properties of self-induced oscillations in hover control. The main contributions of this work are two-fold: First, we analytically show that when certain types of control systems directly use the unscaled velocity (or position) feedback 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 based on the hover stability of a quadrotor MAV. Our method neither relies on additional sensor modalities, such as IMU, depth sensors or altimeters, nor imposes assumptions on the scene geometry. We demonstrate the proposed system both in simulation and on a real quadrotor platform using a Parrot AR.Drone 2 (see the bottom of Fig. 1). Video demonstrations are available at:

https://goo.gl/afq3tP

We also release our implementation as open-source.[1]

II. Related Work

Conventionally, monocular approaches estimate the absolute scale by acquiring additional metric information from either range or IMU measurements.

A. 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 [9] or RGB-D camera [10]. In [3], a closed-form solution is proposed to estimate the absolute scale using a sonar 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, so it leads to inaccuracies when flying over stairs or multiple large objects on the floor. These drawbacks do not apply to our method, as we do not rely on range sensor measurements for scale estimation.

Manuscript received September 9, 2017; accepted December 12, 2017. Date of publication January 4, 2018; date of current version January 25, 2018. This paper was recommended for publication by Associate Editor V. Lippiello and Editor J. Roberts upon evaluation of the reviewers' comments. *(Corresponding author: Seong Hun Lee.)*

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

Digital Object Identifier 10.1109/LRA.2018.2789841

[1]Source code and supplementary materials are available at https://github.com/sunghoon031/stability_scale

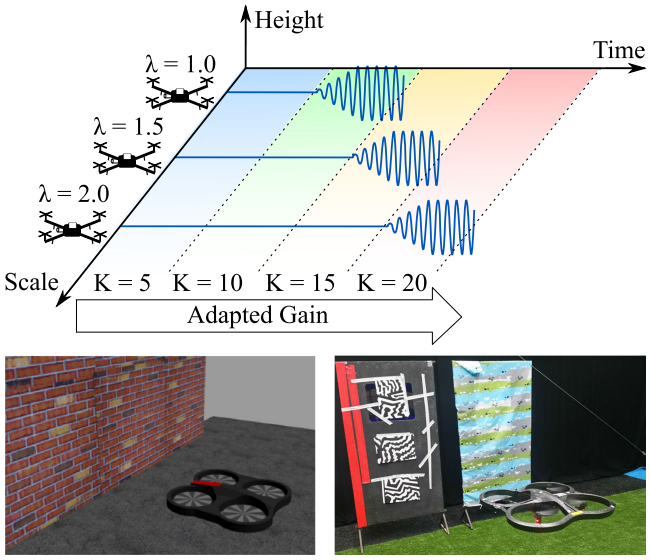

Fig. 1. We propose to solve the scale ambiguity of monocular SLAM using the control instability that arises when applying a certain gain to the unscaled state feedback from the SLAM system. **Top:** Example illustration of the linear relationship between the absolute scale λ and control gain K at which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

B. 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 metric scale. Accelerometer measurements provide the metric information about the robot's motion. State-of-the-art examples consist of filter-based [11], [12] and optimization-based [13]–[15] 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 [16]. Also, for pocket-sized (or smaller) MAVs, accelerometers readings are very noisy and often unreliable for scale estimation [17]. On the other hand, our method requires neither complex multi-sensor calibrations nor carefully tailored initialization, and it is applicable to both small MAVs and larger UAVs.

Our method is inspired by the previous work of one of the authors [8] where a *stability-based* method was first proposed to estimate the altitude of a quadrotor MAV using a single downward-looking onboard camera. This work showed that there is a unique linear relationship between the altitude and control gain at which optical flow divergence control manifests self-induced oscillations. Based on this relationship, it was shown that it is possible to deduce the altitude by detecting the onset of oscillations.

In this paper, we extend the idea of [8] to a scale estimation problem for monocular SLAM. Our method does not use optical flow divergence, but instead uses unscaled state estimates from the SLAM system. Therefore, our system has a different state space model, control law and factors that influence the system stability. We also incorporate different strategies for detecting the oscillations and adapting the controllers. Note that in comparison to [8], we apply the proposed method to achieve full 3D navigation instead of only vertical control. Furthermore, once the scale is estimated, the proposed SLAM-based system can use the same control gains independently of the distance to the scene, whereas an optical-flow-based method has to adapt its gains according to the distance.

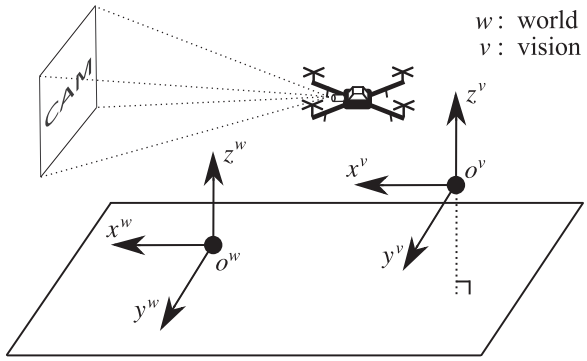

Fig. 2. World $(o^w\, x^w\, y^w\, z^w)$ and vision $(o^v\, x^v\, y^v\, z^v)$ reference frames.

The remainder of the paper is organized as follows: In Section III, we investigate the stability properties of certain control systems using unscaled state feedback. Subsequently, we detail the proposed adaptive method 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.

III. Instability of Unscaled Feedback Control

A. Discretized Model for Velocity Control

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

$$\begin{aligned}\dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + B\mathbf{u}(t)\\ &= \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\mathbf{x}(t) + \begin{bmatrix}0\\ 1/m\end{bmatrix}u_z(t), \qquad (1)\\ \mathbf{y}(t) &= C\mathbf{x}(t) + D\mathbf{u}(t)\\ &= \begin{bmatrix}0 & 1/\lambda\end{bmatrix}\mathbf{x}(t), \qquad (2)\end{aligned}$$

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 this model with zero-order hold (ZOH) yields:

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

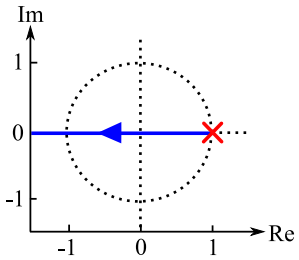

Fig. 3. Root locus plot of the ZOH-model in a vacuum environment without noise and delay.

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

$$G(\sigma) = C_d(\sigma I - A_d)^{-1}B_d = \frac{T_s}{\lambda m(\sigma - 1)}, \quad (4)$$

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:

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

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

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

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. 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*. By setting $\sigma = -1$ and equating the denominator of (6) to 0, the critical gain can be found as:

$$K_{cr} = \frac{2m}{T_s}\lambda. \quad (7)$$

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. Note that in [8], a similar relationship was found for optical flow divergence control, but with the distance to the scene instead of scale.

B. Including Drag and Delay

We can extend the ZOH-model in the previous section to account for the aerodynamic drag and time delay (e.g., caused by visual processing and communication). Here, we omit the derivations[1] and present only the resulting relationship between the critical gain K_{cr} and scale λ:

$$K_{cr} = \lambda f\left(m, T_s, {V_z}^w, \rho, C_D, A, N\right), \quad (8)$$

where $f(\cdot)$ is a positive function of the following variables: robot's mass m, discretization sampling period T_s, vertical velocity ${V_z}^w$ at the linearization point, air density ρ, drag coefficient C_D, reference area A, and time delay N as an integer number of sampling periods. Numerical analysis[1] further reveals two observations: First, $f(\cdot)$ has negligible sensitivity to ${V_z}^w$ at different linearization points near hover condition (i.e., within ± 0.5 m/s). Therefore, if we assume the remaining variables are constant, (8) becomes a linear equation between K_{cr} and λ with a constant coefficient.

Second, the value of $f(\cdot)$ decreases with an increase in either time delay N or sample time T_s. Together with (7), this suggests that the delay of the control system plays an essential role in the self-induced oscillations and hence in the relation between the critical gain and the scale of the SLAM system. A similar finding was reported in [8].

C. Height Control Using Velocity Commands

Some rotorcraft systems allow users to control the flight using velocity commands rather than thrust. In this case, regulating the height using velocity commands gives rise to a similar type of self-induced oscillations. In the following, we provide a rudimentary explanation for this behavior using a simplified model.

First, assume a perfect system where the actual velocities instantaneously follow the velocity commands. Then, we can use the following state space model:

$$\dot{\mathbf{x}}(t) = \mathbf{u}(t), \quad \mathbf{y}(t) = \frac{1}{\lambda}\mathbf{x}(t), \quad (9)$$

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.

Now, consider a simple proportional feedback controller:

$$u_{vz} = K\left(Z^* - Z^v\right), \quad (10)$$

where Z^* is the desired vertical displacement from the initial hovering height in the vision frame. Note that differently from (5), we regulate the height instead of velocity. Following the same procedure as in Section III-A, one can easily derive the discrete closed-loop transfer function and the corresponding critical gain, which is given as:

$$K_{cr} = \frac{2}{T_s}\lambda. \quad (11)$$

D. Generalized Relation Between Critical Gain and Scale

The analytical findings given as (7), (8) and (11) suggest that if we define

$$\alpha := \frac{K_{cr}}{\lambda}, \quad (12)$$

then α should be close to constant. We empirically find α by performing a linear fit between the ground-truth scale λ and critical gains K_{cr}^* obtained at different camera-to-scene distances within the operational range (e.g., see Fig. 7). We obtain K_{cr}^* values using the adaptive method presented in the following section.

Once α^* is estimated, we keep its value fixed and use it to estimate the scale from the given K_{cr}^*:

$$\lambda^* = \frac{K_{cr}^*}{\alpha^*}. \quad (13)$$

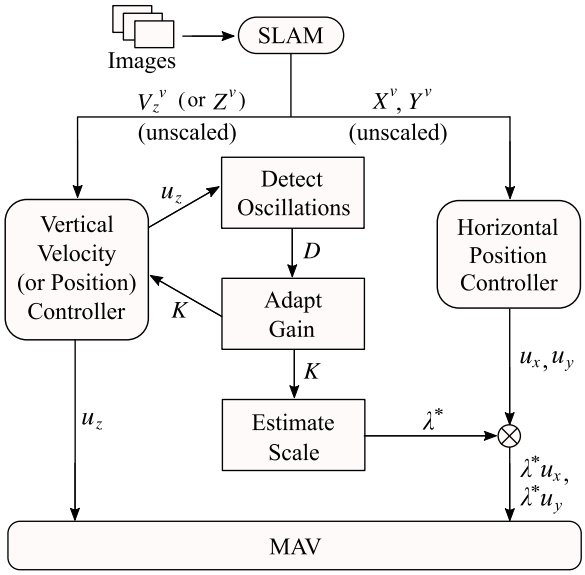

Fig. 4. Adaptive control pipeline for scale estimation.

We provide more details on how to iteratively estimate λ^* in Section IV-C. Note that the system does not have to know the mass in kilogram or thrust in Newton for estimating the metric scale as long as α^* is calibrated accurately using the ground-truth. Since the scale estimation accuracy depends on both accuracy of K_{cr}^* and α^*, a poor calibration of α^* can lead to a large scale error and degraded control performance (see Section V-B2).

IV. Adaptive Control Strategy for Scale Estimation

We propose an adaptive control framework that estimates the scale of a monocular SLAM system based on the control stability properties studied in the previous section. 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 for the unscaled input.

Once the critical gain is found, the final scale is computed 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 navigation. 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.

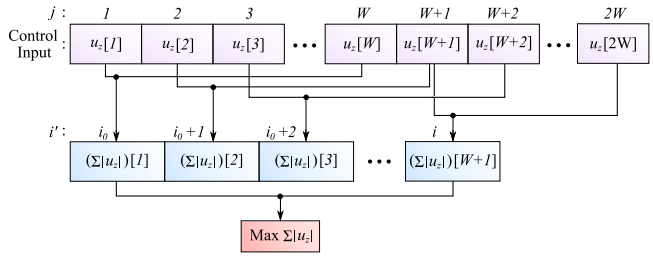

Fig. 5. Computing the discrete detection variable $D(i, i_0, W)$ at $i = i_0 + W$. For each W past control input, we compute $\sum |u_z|$ and update the maximum term in (16).

A. Detection of Self-Induced Oscillations

In this section, we propose a simple yet effective method to detect self-induced oscillations. To quantify and measure oscillations, we use a heuristic variable D defined as the maximum moving average of the total variation in vertical velocity. For continuous systems, let t_0 be the time at the beginning of the evaluation window and T_w the length of the subwindow for averaging the total variation. We define D at time t as:

$$D(t, t_0, T_w) := \max_{t' \in [t_0, t]} \left(\frac{1}{T_w} \int_{t'-T_w}^{t'} \left| a_z^w(\gamma) \right| d\gamma \right), \qquad (14)$$

where a_z^w is the vertical acceleration in the world frame. Since we cannot directly observe a_z^w from the SLAM system, we adopt an alternative formulation of D using the thrust command u_z. Assuming that the drag force is relatively small, (14) can be approximated as:

$$D(t, t_0, T_w) \approx \max_{t' \in [t_0, t]} \left(\frac{1}{mT_w} \int_{t'-T_w}^{t'} \left| u_z(\gamma) \right| d\gamma \right). \qquad (15)$$

For discrete systems, this becomes:

$$\begin{aligned} D(i, i_0, W) &= \max_{i' \in \{i_0, \cdots, i\}} \left(\frac{1}{mWT_s} \cdot \sum_{j=i'-W+1}^{i'} \left| u_z[j] \right| T_s \right) \\ &= \frac{1}{mW} \left(\max_{i' \in \{i_0, \cdots, i\}} \sum_{j=i'-W+1}^{i'} \left| u_z[j] \right| \right), \end{aligned} \qquad (16)$$

where W is the discrete window size, which corresponds to T_w / T_s. This process is illustrated in Fig. 5. The onset of oscillations is detected as soon as D exceeds a threshold D_{thr}. In our implementation, we empirically choose the appropriate values for W and D_{thr} in advance and keep them constant. Note that the maximum magnitude of oscillations during the instability is more or less constant, as the oscillations tend to occur in a unique "resonant" frequency and the control input u_z is bounded by the maximum throttle level. This allows us to apply a simple thresholding method with a fixed D_{thr}.

As for the height control system using velocity commands (described in Section III-C), we can effectively use a similar

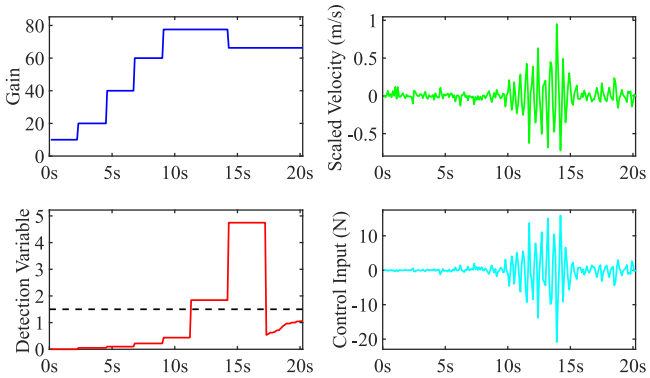

Fig. 6. **[Simulation]** Critical gain estimation with $D_{thr} = 1.5$.

approach by reducing the order of (14) by one:

$$D(t, t_0, T_w) := \max_{t' \in [t_0, t]} \left(\frac{1}{T_w} \int_{t'-T_w}^{t'} \left| V_z{}^w(\gamma) \right| d\gamma \right), \quad (17)$$

$$\approx \max_{t' \in [t_0, t]} \left(\frac{1}{T_w} \int_{t'-T_w}^{t'} \left| u_z(\gamma) \right| d\gamma \right), \quad (18)$$

For discrete systems, this is equivalent to:

$$D(i, i_0, W) = \frac{1}{W} \left(\max_{i' \in \{i_0, \cdots, i\}} \sum_{j=i'-W+1}^{i'} \left| u_z[j] \right| \right). \quad (19)$$

Note that this is the same computation as (16) but without mass.

B. Adaptive Gain Control

To find the critical gain, we first increase the gain adaptively until the onset of oscillations is detected (i.e., $D > D_{thr}$) and then decrease it back 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.

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

$$K_{i+1} = K_i + PK_i \frac{(D_{thr} - D_i)}{D_i}, \quad (20)$$

where P is a proportional gain. Once the gain is adapted, the system waits for some time to induce sufficient response. The next adaptation takes place if no oscillations are detected within a certain period of time after the last gain update. Fig. 6 plots a time evolution of the relevant variables in simulation.

C. Iterative Scale Update for Horizontal Position Control

As the system adapts the gain to find the critical gain, we can also update the intermediate estimate of the scale λ_i^* at each iteration i using (13) with K_i from (20):

$$\lambda_i^* := \frac{K_i}{\alpha^*}, \quad (21)$$

where α^* is to be empirically calibrated in advance (see Section III-D). Ideally, λ_i^* approaches the true scale λ as the adapted gain K approaches K_{cr}.

As illustrated in Fig. 4, we use λ_i^* after each update to scale the unscaled horizontal commands $(u_x,\ u_y)$ computed from the unscaled translational states $(X^v,\ Y^v)$. Although at first the MAV may drift as the initial magnitudes of λ^* are much smaller than the true λ, it quickly stabilizes in the horizontal position after a few iterations (see Section V-B1).

In principle, it is also possible to achieve "scalefree" control by directly updating the horizontal PID gains using a fixed set of appropriate ratios with respect to the vertical gain K. Although the metric map and motion will remain unknown, this approach can allow us to skip the calibration process for α^* and still enable autonomous control.

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 flight performance of our autonomous system.

We used *tum_simulator* [18] for simulations, and *ardrone_autonomy* [19] 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 Sections III-A and III-B) and height control for the real drone (see Section III-C).

We used monocular ORB-SLAM [4] 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, the video stream from the onboard forward-looking camera was transmitted to a ground-based laptop[2] via WiFi connection. All computations were performed off-board, and the resulting control commands were sent back to the AR.Drone at 50 Hz with the network latency around 60 ms [3]. On average, the onboard processing and transmission of the image took 170 ms. Additionally, ORB-SLAM's pose estimation took 30 ms for each frame, and our algorithm took 1 ms to detect oscillations and adapt the gain. In total, this gives time delays up to 280 ms. We used an OptiTrack[3] motion capture system to track the ground-truth flight trajectories for comparison purposes.

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

[2]Intel Core i7-4810MQ, 4 cores at 2.8 GHz with 15Gb RAM.
[3]https://optitrack.com/

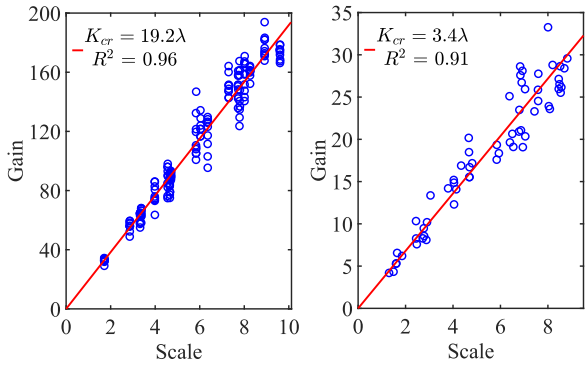

Fig. 7. Positive linear relationship between the critical gain and scale, i.e., $K = \alpha\lambda$. Left: **[Simulation]**, Right: **[Real-world]**.

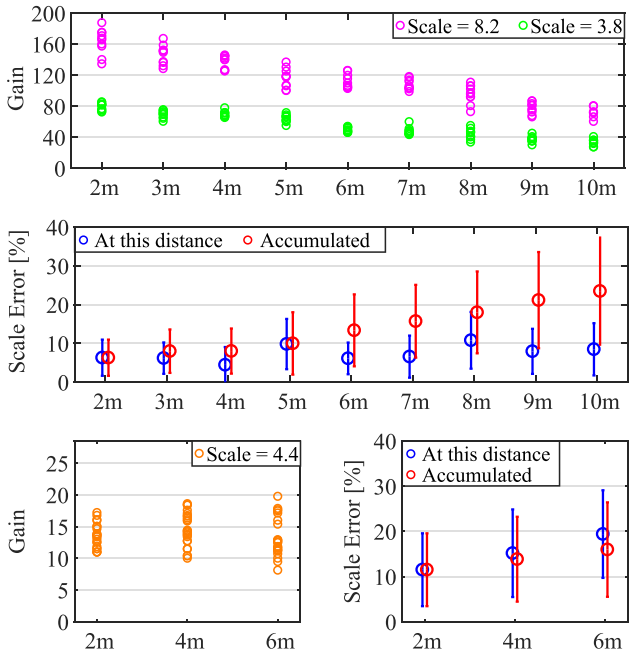

Fig. 8. **Top: [Simulation]** Critical gain estimation at different scene distances using velocity control. **Middle: [Simulation]** Scale error computed by evaluating the results at each individual scene distance or combining those less than or equal to each distance. We used both magenta and green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

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.

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. However, our results show that 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. 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

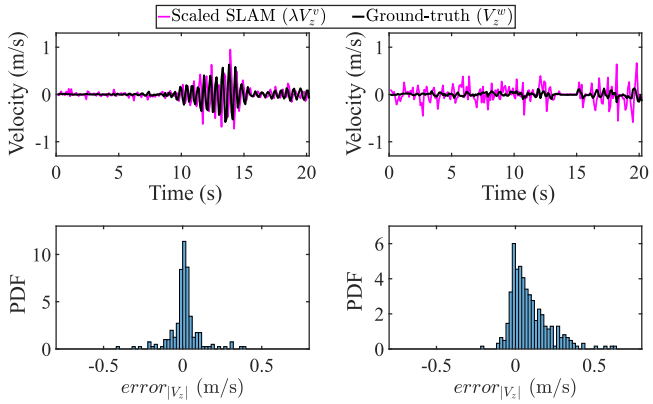

Fig. 9. **[Simulation] Top:** Estimated vertical velocity compared to the ground-truth at the scene distance of 2m (left) and 10m (right). **Bottom:** Probability density function of the difference between the corresponding absolute velocities.

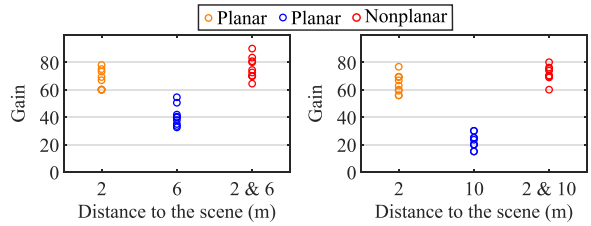

Fig. 10. **[Simulation]** Estimated critical gains when subjected to planar or nonplanar scene structures.

of Fig. 8. Within the scene distance between 2 m and 6 m, the average scale estimation error is estimated to be 13.4% and 16.0% in simulation and in the real world, respectively.

As for the velocity control system, 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. For quantitative comparison, the bottom plots of Fig. 9 present the probability density functions (PDF) of the observed difference between the two absolute velocities:

$$error_{|V_z|} = |\lambda V_z^v(t)| - |V_z^w(t - t_d)|, \tag{22}$$

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. A bootstrap two-sample test further confirmed that the two PDFs do not come from the same distribution.

Since the perceived magnitude of oscillation in ORB-SLAM is overestimated, so are the necessary control input $u_z = -KV_z{}^v$ and detection variable D in (16). As a result, the system detects the oscillations prematurely even if the actual magnitude of oscillation can be quite small (see top right of Fig. 9). This leads to a smaller magnitude of oscillations and lower critical gains at larger scene distances. We could not, however, observe

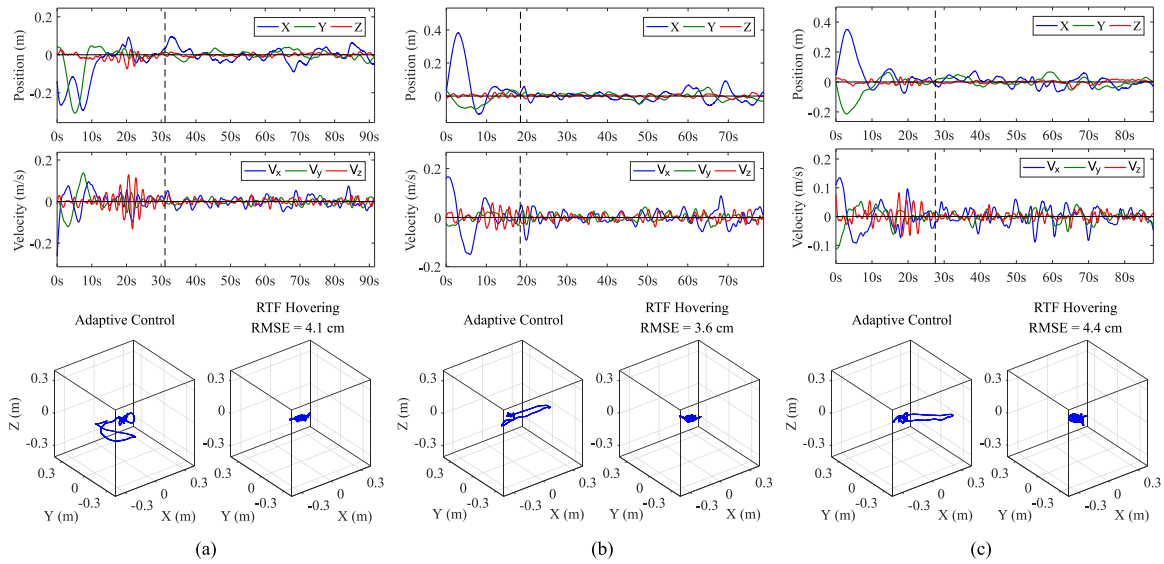

Fig. 11. **[Real-world] Top two rows:** Time evolution of the MAV's position and velocity during the adaptive control followed by 60 seconds of position hold in RTF mode. The vertical dashed line marks the transition between the two modes. **Bottom row:** Ground-truth trajectories for the adaptive control (left) and RTF hovering (right). (a)Scene Distance = 2 m. (b)Scene Distance = 4 m. (c)Scene Distance = 6 m.

such clear tendency in real-world experiments using height control (see the bottom row of Fig. 8 and Section V-B1).

Although at first glance such influence of the scene distance may seem like a critical weakness of the velocity control system, we point out two reasons why our method can still be used. First, we found that this "distance effect" weakens in typical large-scale indoor environments where a floor, ceiling or nearby walls (or objects) are visible. To show this, we conducted another set of simulations with nonplanar scenes by placing 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. The results are shown in Fig. 10. We see that the smallest scene distance has the dominant effect on the estimation of critical gain. This implies that (1) the variable we called "scene distance" in Fig. 8 was essentially the distance to the closest map point, and (2) having at least a few close tracking points (e.g., on the floor or ceiling) during the scale estimation process can already prevent the increase of scale estimation error.

The second reason to consider our method even at large scene distances is that the underestimated scale (due to an underestimation of the critical gain, if at all) may in fact be an advantage in terms of flight stability. For example, consider the hovering flight at 10 m scene distance as shown in Fig. 9. The RTF mode using the correctly scaled velocity $\lambda V_z{}^v$ (magenta) would render the MAV immediately unstable because of the large errors in the observed velocity. In this case, using an underestimated scale is preferred over the correct scale, since it reduces the velocity error caused by the noise and downscales the control input accordingly.

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

Also, note that the results in Fig. 11 do not display a clear correlation between the magnitude of oscillations and distance to the scene. For each scene distance of (2, 4, 6 m), the maximum oscillation amplitude and peak velocity was (0.07, 0.02, 0.04 m) and (0.1, 0.06, 0.08 m/s), respectively.

2) Waypoint Following: To evaluate the flight performance, we made the MAV fly to a waypoint placed at 2 m ground-truth distance from the initial hover position. This was done in either vertical or horizontal direction. Note that using a poorly calibrated α^* in (21) will either under- or overestimate the true scale λ, which can potentially degrade 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 $\alpha^* = 4$ gives the best performance in both vertical and horizontal flights. This is not surprising, since the appropriate value for α^* should be around 3.4 as given in Fig. 7. With $\alpha^* = 1$, λ^* overestimates the actual scale λ, causing overshoot and instability. On the other hand, setting $\alpha^* = 7$ underestimates the scale, leading 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 Table I. The convergence was considered to be reached when the Euclidean distance between the MAV and

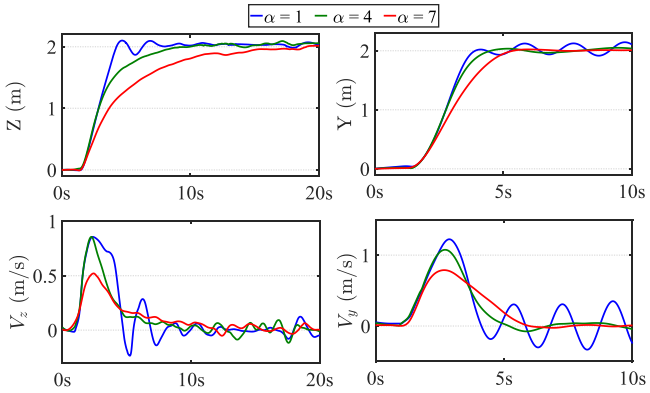

Fig. 12. **[Real-world]** Ground-truth states for single waypoint-following tasks in a vertical (left column) or horizontal (right column) direction.

TABLE I
[REAL-WORLD] AVERAGE CONVERGENCE TIME AND PEAK SPEED IN POSITION CONTROL

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

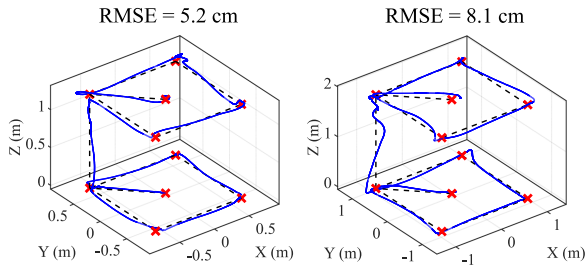

Fig. 13. **[Real-world]** Ground-truth trajectories for figure flying. Left: Small figure (1 m × 1 m × 1 m). Right: Large figure (2 m × 2 m × 2 m).

waypoint fell below 10 cm. In the videos, we additionally show how our system behaves when following approximately 10 m long straight line trajectories in a larger environment. Since the ground-truth data is not available for this experiment, we placed two markers on the ground at a 10 m distance and observed that the MAV hovered over each marker in close proximity at the beginning and end of the flight.

3) Figure Flying: We demonstrate that our system can fly simple figures, similar to that demonstrated in [20]. 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. CONCLUSION

In this work, we have presented a stability-based method to solve the scale ambiguity in monocular SLAM. Theoretical analysis has shown that, when using unscaled state feedback from monocular SLAM for control, a unique linear relationship exists between the absolute scale of the SLAM system and the control gain at which instability arises. Our method exploits this property to estimate 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 systems that use additional metric sensors, we believe that it is definitely competitive (if not better) for achieving autonomous control.

REFERENCES

[1] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based MAV navigation in unknown and unstructured environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 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 Int. Conf. Robot. Autom.*, 2011, pp. 3056–3063.
[3] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 2815–2821.
[4] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.
[5] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 15–22.
[6] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," vol. PP, no. 99, pp. 1–1, 2017.
[7] G. Nützi, S. Weiss, D. Scaramuzza, and R. Siegwart, "Fusion of IMU and vision for absolute scale estimation in monocular SLAM," *J. Intell. Robot. Syst.*, vol. 61, no. 1, pp. 287–299, 2011.
[8] 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, pp. 1–18, 2016.
[9] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a navigation system for autonomous indoor flying," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 2878–2883.
[10] 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 Int. Symp. Robot. Res.*, 2011, pp. 235–252.
[11] 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 Int. Conf. Intell. Robot Syst.*, 2013, pp. 3923–3929.
[12] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 3565–3572.
[13] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," *Int. J. Robot. Res.*, vol. 34, no. 3, pp. 314–334, 2015.
[14] S. Shen, N. Michael, and V. Kumar, "Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft MAVs," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 5303–5310.
[15] R. Mur-Artal and J. D. Tardós, "Visual-inertial monocular SLAM with map reuse," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 796–803, Apr. 2017.
[16] J. Kaiser, A. Martinelli, F. Fontana, and D. Scaramuzza, "Simultaneous state initialization and gyroscope bias calibration in visual inertial aided navigation," *IEEE Robot. Autom. Lett.*, vol. 2, no. 1, pp. 18–25, Jan. 2017.
[17] K. McGuire, G. C. H. E. de Croon, C. De Wagter, K. Tuyls, and H. Kappen, "Efficient optical flow and stereo vision for velocity estimation and obstacle avoidance on an autonomous pocket drone," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 1070–1076, Apr. 2017.
[18] H. Huang, "tum_simulator," 2014. [Online]. Available: http://wiki.ros.org/tum_simulator
[19] M. Monajjemi, "ardrone_autonomy," 2012. [Online]. Available: http://wiki.ros.org/ardrone_autonomy
[20] J. Engel, J. Sturm, and D. Cremers, "Scale-aware navigation of a low-cost quadrocopter with a monocular camera," *Robot. Auton. Syst.*, vol. 62, no. 11, pp. 1646–1656, 2014.