Omnidirectional vision odometry for low power hardware on flying robots

Simon Reich[1], Maurice Seer[1], Lars Berscheid[1], Florentin Wörgötter[1], and Jan-Matthias Braun[1]

***Abstract*— In this work we will show our method parts in more detail than possible in the original submission. However, some steps are omitted as they refer to standard text book examples. They are noted as such. Images are enlarged to ease understanding; Fig. 4 holds more camera sketches. The referenced sources are the same as in the original submission; however, the numbering does not align.**

I. METHOD

This section divides into three parts: In I-A the hardware setup is presented, and in I-B our algorithms to arrive at safe trajectory planning are shown. Lastly, we will discuss briefly the theoretical limit of the algorithms.

A. HARDWARE SETUP

Fig. 1 shows the hardware setup: A quadrocopter, controlled by a Raspberry Pi mini computer running a linux operating system. Connected to the Camera Serial Interface (CSI) port of the Raspberry Pi is a monocular RGB camera, which photographs with a resolution of 320x320 px at a frequency of 30 Hz. The camera is pointed upwards on a hyperbolically shaped mirror, which can also be replaced with a spherical shaped mirror. Later, we will discuss the advantages and disadvantages of these shapes. Additionally, we use an accelerometer, gyroscope, and ultrasonic sensor as input. Also, any contemporary bluetooth gaming controller can be attached. This allows easy control of high level features, e.g. issue the start or landing command. Of course, manual flight control is also possible—a feature not used in this work.

As software we use the Robot Operating System (ROS, see [1]). ROS offers efficient message transport capabilities, locally as well as via network connection. This function is heavily used to guarantee modular program design, as well as real time network connections. The latter is very important for debugging and visualization purposes on the flying robot via wlan. Fig. 3 lists all functional units as ROS nodes and thus provides a system overview. An Extended Kalman Filter (EKF) [2] performs sensor fusion and computes pose information. Afterwards, a PID controller, as demonstrated by Åström and Hägglund [3], adjusts the motor controllers to manipulate the quadrocopter into the goal pose. The goal position is defined by higher level algorithms (e.g. SLAM, see [4], corridor flight algorithms, see [5], etc.).

[1]Third Institute of Physics - Biophysics, Georg-August-Universität Göttingen, Friedrich-Hund-Platz 1, 37077 Göttingen, Germany
{sreich, mseer, lberscheid, worgott, jbraun}@phys.uni-goettingen.de
(The first two authors contributed equally to this paper.)

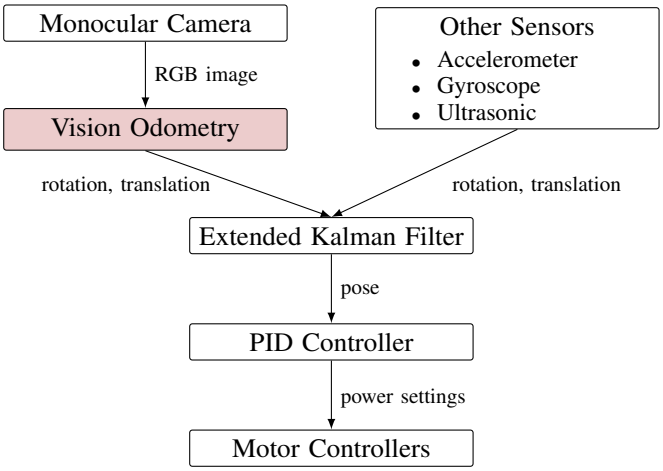


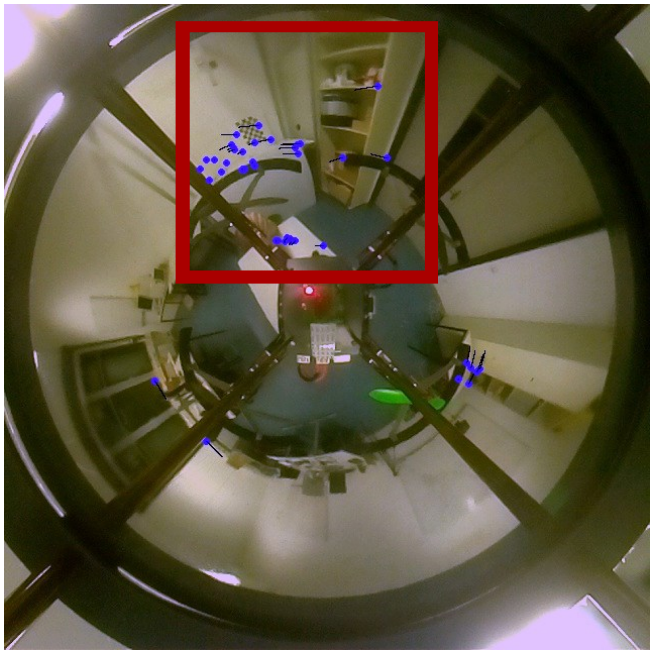
Fig. 3: Overview of the system nodes. Each block is a separate process; the Robot Operating System (ROS, [1]) exchanges messages between them. This way a modular design is guaranteed. In this work we focus on Vision Odometry (marked in red).

B. ALGORITHMS

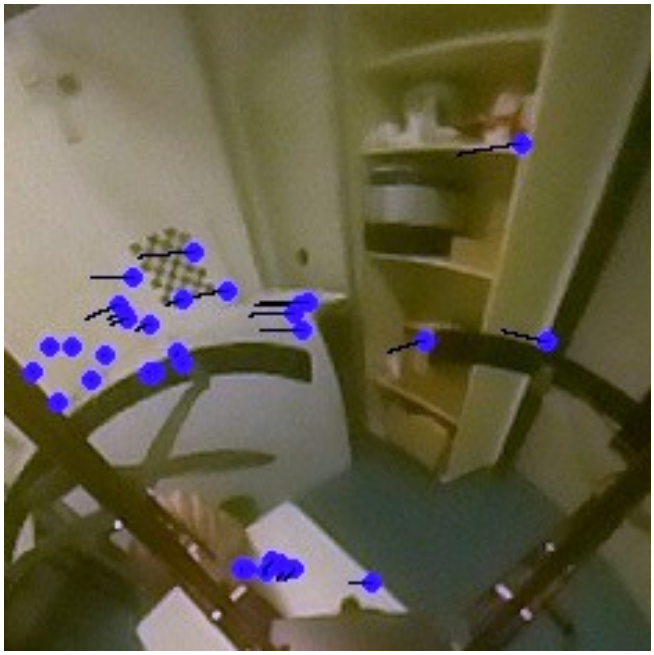
In this section we will describe how we arrive from an omnidirectional monocular RGB image at pose information. Please remember, that we are using a flying robot, which is very fast and agile. This means that a front facing camera will have problems recognizing features in adjacent camera frames; Features belonging to the same object will have a huge offset from one frame to the next. Because of this, we are using the omnidirectional camera setup as shown in Fig. 4a. Fig. 2 then takes you through the processing steps, which we will explain below in detail. First, features are computed on the omnidirectional image. Here we are using FAST features [6], but any desired feature set will work. On these features optical flow is computed. As we know the transfer function for image dewarping, we can now estimate the robots rotation and translation. Using this information and enriching it with data from the accelerometer, gyroscope, and ultrasonic sensor, we can compute new pose information. Additionally, we can track features over multiple frames—meaning, we have arrived at a list of pose information, each having a small offset. Combining both information we can perform 3D stereo vision and also estimate depth for each feature.

1) Vision Odometry Algorithm: In this section, we compute features on the raw camera image (Fig. 2a and Fig. 2b).

Fig. 1: The quadrocopter utilized in this work. In the center, a camera captures omnidirectional images via the mounted mirror (Compare Fig. 4a).



(a) Camera view of the hyperbola mirror with computed features and optical flow.



(b) Enlarged view of the red square in left image. The lower structure belongs to the robot.

(c) Dewarped Image showing the 360° view around the quadrocopter.

Fig. 2: Processed camera images with features and computed optical flow.

Features are areas in an image, which are easy to find, recognize, and track in consecutive frames—usually areas rich in texture. Afterwards, we compute the optical flow on these features. Both are very much solved problems and we will rather focus and describe methods dedicated to run on limited hardware.

There are numerous publications comparing different feature algorithms—the most prominent algorithms include FAST [6], GFTT [7], ORB [8], SIFT [9], and SURF [10]. In this work we tried FAST, SIFT, SURF, and GFTT. We ran quantitative tests, which are shown on our web page [11]. These tests showed that FAST offers the best trade-off between computational complexity and quality of found features. This result is not surprising, as FAST is known to be faster but also finds less features [12, 13]—as we work on very limited hardware, our focus lays on computational complexity.

For FAST, the image is first converted to greyscale. Then, areas with a high intensity variation are searched. The algorithm makes use of linearization and therefore offers low computational complexity. A detailed overview is given by Rosten and Drummond [6].

Next, we compute the optical flow. The optical flow is a vector field describing the apparent motion, usually the motion of tracked features. Let $I(x, y, t)$ be the intensity of the greyscale pixel position (x, y) at time $t \in \mathbb{R}$. We assume that the overall intensity does not change and therefore

$$\frac{\mathrm{d}I}{\mathrm{d}t} = \frac{\partial I}{\partial t} + \frac{\partial I}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial t} = I_t + I_x\vec{V_x} + I_y\vec{V_y} \stackrel{!}{=} 0 \quad ,$$

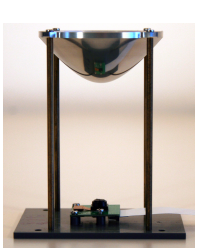
where $\vec{V}$ is the optical flow. This under-determined problem can be solved using the Lucas-Kanade method [14], which only assumes a constant flow in a local pixel neighborhood. A detailed introduction can be found in Dunkel [15], a rather pratical approach is given in Bradski and Kaehler [16]. This established method gives us the robots displacement relative to the features.

Next, we will discuss very shortly the camera pinhole model, which we will then generalize to the spherical mirror model. The pinhole model is a standard physics model and can be found in various undergraduate text books, e.g. [17, 18, 16].

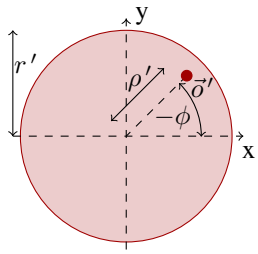
2) Pinhole Camera: A pinhole camera at position $\vec{c}$, orientation in x-y-plane $\vec{q}$, and focal length f points at an object at position $\vec{o}$ which lies in a plane with distance b; $\vec{c}, \vec{q}, \vec{o} \in \mathbb{R}^3$. We choose the origin of the camera's 2D-image space to be in the center of the sensor and denote image coordinates with a prime as in $\vec{o}'$. To map an image position $\vec{o}'$ to world coordinates $\vec{o}$, we define the transformation

$$T_P : \vec{o}' \longmapsto \vec{o} = \vec{c} + \frac{b}{f} \cdot R(\vec{q}) \cdot \vec{\Pi}(\vec{o}') \quad , \qquad (1)$$

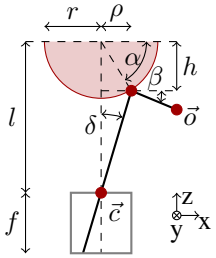
with the rotation matrix $R(\vec{q})$ and $\Pi(\vec{o}') = \left(o'_x, f, o'_y\right)^\intercal$. Analogously, we compute the inverse transformation using



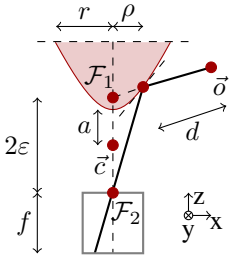
(a) The hyperbola mirror.



(b) Camera view of spherical mirror.



(c) Side view of spherical mirror.



(d) Side view of hyperbola mirror.

Fig. 4: Sketch of a camera observing an object $\vec{o}$, which appears at position $\vec{o}'$ in the image plane (bottom left). The top left figure depicts a simple pinhole model; on the top right the camera is pointed at a spherical mirror and at the bottom right at a hyperbolic mirror.

the depth $b = \left(R(\vec{q})^{-1}(\vec{o} - \vec{c})\right)_y$ as

$$\begin{aligned} \vec{\Pi} &= \frac{f}{b} R(\vec{q})^{-1}(\vec{o} - \vec{c}) \\ &= \frac{f R(\vec{q})}{\left(R(\vec{q})(\vec{o} - \vec{c})\right)_y}(\vec{o} - \vec{c}) \quad . \end{aligned} \qquad (2)$$

3) Spherical Mirror Model: Next, we will add a fixture holding a spherical mirror with its center in distance b (Fig. 4c). For simplicity, we will use a coordinate system that has its origin at the camera $\vec{c}$ with the z-Axis oriented towards the mirror. We will denote reflections on the mirror with the caret $\hat{}$ as in $\hat{\vec{o}}$. Using the real sphere radius r and radius r' in image space (Fig. 4b), the reflection point $\hat{\vec{o}}$ can be computed independently of camera parameters using the scaling factor $s = {}^{r}/_{r'}$:

$$\hat{\vec{o}} = \left(s\, o'_x, s\, o'_y, b - h\right)^\intercal \quad , \; h = \sqrt{r^2 - \rho^2} \quad . \qquad (3)$$

Given the distance d, we now compute the object position $\vec{o}$. Let $\vec{e}$ be a vector of unit length, satisfying the condition $\vec{o} - \hat{\vec{o}} = d\vec{e}$. This vector can be expressed with the angles β and φ (Fig. 4c, Fig. 4b):

$$\vec{e} = \begin{pmatrix} \cos\beta\cos\phi \\ \cos\beta\sin\phi \\ -\sin\beta \end{pmatrix} \quad ,$$

with $\rho = s\sqrt{{o'_x}^2 + {o'_y}^2}$, $\varphi = -\mathrm{atan2}(o'_y, o'_x)$, $\alpha =$

$\arccos\left({}^{\rho}/_{r}\right)$ and thus

$$\begin{aligned}\delta =& \arctan\left(\frac{\rho}{b-h}\right)\\ \beta =& 2\alpha+\delta-\frac{\pi}{2} \quad .\end{aligned}$$

Using the camera position and pose, we transform to world coordinates

$$T_S : \vec{o} = \vec{r} + R(\vec{q})\left(\hat{\vec{o}} + d\vec{e}\right) \quad .$$

The inverse transformation T_S^{-1} calculates $\vec{o}'$ from $\vec{o}$ and the camera pose. As the construction of T_S^{-1} is lengthy but does not add much to understanding of this work, we provide the solution at [11] and continue with the result

$$\begin{aligned}\cos\alpha =& (\theta-\sin\alpha)\\ &\cdot\tan\left(\arctan\left(\frac{-\Delta\sin\beta}{\rho-\cos\alpha}\right)-2\alpha+\frac{\pi}{2}\right)\end{aligned}$$

where $\Delta = {}^{d}/_{r}$. This function can only be solved iteratively. Using a generous approximation of small focal lengths ($\theta = {}^{b}/_{r}$) and large depths ($\Delta \ll 1$) we approximate

$$\begin{aligned}\cos\alpha \approx& \left(\frac{b}{r}-\sin\alpha\right)\\ &\cdot\tan\left(\arctan\left(\frac{-\sin\alpha-\Delta\sin\beta}{\rho}\right)-2\alpha+\frac{\pi}{2}\right).\end{aligned}$$

Still, the solution is nontrivial and computational very expensive. Considering that we are using autonomous robots, which perform all computations online on limited hardware this poses a problem.

4) Hyperbolic Mirror Model: Using a hyperbola, the inverse function can be computed easier and thus faster. The surface of a hyperbolic mirror is defined by

$$\frac{y^2}{a^2}-\frac{x^2}{b^2}=1 \quad , \; a,b\in\mathbb{R} \tag{4}$$

with the semi-major axis a. The focal points $\mathcal{F}_{1,2} = \left(0, \pm\sqrt{a^2+b^2}\right) =: (0, \pm\epsilon)$ are defined as in Fig. 4d. For the following derivation, we place the origin of the coordinate system $\vec{c}$ in the middle of these two focal points and place the focal point of the camera at $\mathcal{F}_2$. Given the image position $\vec{o}'$, the reflection point is $\hat{\vec{o}} = \left(\vec{s}_x, \vec{s}_y, a\sqrt{\rho^2/b^2+1}\right)^{\intercal}$. The unit vector $\vec{e}$ to the object $\vec{o}$ is

$$\begin{aligned}\vec{e} =& \frac{\hat{\vec{o}}-\mathcal{F}_1}{\left|\hat{\vec{o}}-\mathcal{F}_1\right|}\\ =& \frac{1}{\left|\hat{\vec{o}}-\mathcal{F}_1\right|}\left(s\,\vec{o}'_x, s\,\vec{o}'_y, a\sqrt{\rho^2/b^2+1}-\varepsilon\right)^{\intercal}\end{aligned} \tag{5}$$

and therefore the object position at a distance d is defined as $\vec{o} = \vec{r} + R\left(\hat{\vec{o}} + \mathrm{d}\vec{e}\right)$. It can be shown that the inverse transformation is

$$\rho = \frac{(\vec{o}-\vec{c})_{\rho}}{(\vec{o}-\vec{c})_{\rho}^2\cdot\varepsilon^2/b^2-1}\left((\vec{o}-\vec{c})_z\,\varepsilon+a\right) \quad . \tag{6}$$

All steps are shown in detail in [11]. The corresponding image position is given by $\vec{o}' = (\rho\cos\phi, \rho\sin\phi)^{\intercal}$. Different camera orientations $\vec{q}$ are accounted for by rotating the vector $(\vec{o}-\vec{c})$ before calculations.

5) Motion and Depth Estimation: Now, we have arrived at a point where we can first detect features, track them, and, furthermore, compute the robots displacement (translation and rotation) between consecutive frames from the dewarped image. An example of a transformed camera frame is given in Fig. 2c. As we store all tracked features, we can additionally perform simple 3D stereo vision on them. While in theory we would get a very good estimate, real world experiments show that quite a lot of noise gets introduced.

Estimating the depth for N features adds significant complexity to the problem. Currently, we try to estimate the 6D motion M—consisting of translation $\Delta\vec{r}$ and orientation $\Delta\vec{q}$. Our problem has now increased to $N+6$ dimensions. Changes in the feature set from frame $\vec{i}_{i,t-1}$ to frame $\vec{i}_{i,t}$ provide N equations, meaning features need to be tracked for at least 3 consecutive frames.

Matching features with the inverse estimation at time-step t:

1) Depth $d_{i,t-1}$ and motion M_t are initialized using previous data $d_{i,t-2}$ and motion M_{t-1}. The camera pose P_{t-1}, consisting of position $\vec{c}_{t-1}$ and rotation $\vec{q}_{t-1}$, is known.
2) For every feature i, calculate the global position $\vec{o}_{i,t-1}$ using the depth $d_{i,t-1}$, the image coordinates $\vec{o}'_{i,t1}$ and the camera pose P_{t-1}. The transformation T_X, $X \in \{P, S, H\}$ is chosen according to the camera setup, as detailed above.
3) Apply the inverse motion to all global positions $\vec{o}_{i,t-1}$. This results in the predicted global positions $\vec{o}^{p}_{i,t}$.
4) Use the inverse transformation T_X^{-1}, to compute the predicted image position $\vec{o}'^{,p}_{i,t} = T_X^{-1}\left(\vec{o}^{p}_{i,t}\right)$.
5) Lastly, we consider the environment as well as all global features to be static. Therefore, $\vec{o}_i$ and $\vec{o}'_i$ should be equal for corresponding features i: we minimize the sum of the squared distances for the last L time steps: $\mathrm{SD}\left(\mathrm{d_{i,t}}, \mathrm{M}\right) = \sum_{\mathrm{i=0}}^{\mathrm{N}}\sum_{\tau=-\mathrm{L}}^{0}\left\|\tilde{\mathrm{o}}'_{\mathrm{i,t}} - \tilde{\mathrm{o}}'^{,\mathrm{p}}_{\mathrm{i,t}}\right\|^2$.

Estimating the depth with the forward estimation:

1) Perform step 1. and 2. from the inverse estimation.
2) Our goal is to find the new depth $d_{i,t}$ based on the previous estimate $d_{i,t-1}$. In the pinhole model, the depth is defined as the y-component of the difference between the object position $\vec{o}_i$ and the camera position $\vec{c}$: $d_t = \left(R\left(\Delta\vec{q}_t\right)\left(\vec{o}-\vec{c}_{t-1}-\Delta\vec{c}_t\right)\right)_y$. In omnidirectional mirror models, the depth is $d_t = \left\|R\left(\Delta\vec{q}_t\right)\left(\vec{o}-\vec{c}-\Delta\vec{c}_t\right)-\hat{\vec{o}}^{p}\right\|$. The new reflection point $\hat{\vec{o}}^{p}$ is calculated with the inverse transformation T_X^1. For the spherical mirror, an approximation considering only rotations is easily possible. Let $\vec{k} = (0,0,b)^{\intercal}$ be the center of the spherical mirror. Then $\hat{\vec{o}}^{p} \approx R\left(\Delta\vec{q}\right)\left(\hat{\vec{o}}-\vec{k}\right)+\vec{k}$ is leading to the new depth

$$d_0 = \left\| R\left(\Delta\vec{q}_t\right)\left(\vec{o} - \vec{c} - \Delta\vec{r}_t - \hat{\vec{o}} + \vec{k}\right) - \vec{k}\right\|. \quad (7)$$

Due to $\Delta m \ll d$, the approximation can be considered to be vanishing.

3) Compute the new predicted pose $P_t = P_{t-1} + M_t$.
4) Compute predicted global positions $\vec{o}^{\,p}_{i,t=0}$ for every feature i based on the camera model.
5) The positions $\vec{o}_{i,t}$ and $\vec{o}^{\,p}_{i,t}$ should be equal for corresponding features i. We use this to minimize the sum of the squared distances

$$\mathrm{SD}\left(\mathrm{d_{i,t}}, \mathrm{M}\right) = \sum_{\mathrm{i}=0}^{\mathrm{N}} \sum_{\tau=-\mathrm{L}}^{0} \left\| \frac{\tilde{\mathrm{o}}_{\mathrm{i,t}-\tau} - \tilde{\mathrm{o}}^{\mathrm{p}}_{\mathrm{i,t}-\tau}}{\mathrm{d_{i,t}}-\tau} \right\|^2 .$$

The factor $d_{i,t}$ weights all summands consistently as the position-error scales linearly with d.

C. ACHIEVABLE ANGULAR RESOLUTION

Given a fixed camera resolution of $320\mathrm{x}320\,\mathrm{px}$ we can now compute the projection of the hyperbola mirror onto the camera. We assume that the object is at a distance of $2\,\mathrm{m}$ and we require five pixels width to separate it from adjacent objects. After straight forward application of above formulas, we arrive at a limit of approximately $1.9°$.

REFERENCES

[1] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[2] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[3] K. J. Åström and T. Hägglund, *Advanced PID control*. ISA-The Instrumentation, Systems and Automation Society, 2006.

[4] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós, "A comparison of loop closing techniques in monocular slam," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1188–1197, 2009, inside Data Association.

[5] S. Lange, N. Sünderhauf, P. Neubert, S. Drews, and P. Protzel, *Autonomous Corridor Flight of a UAV Using a Low-Cost and Light-Weight RGB-D Camera*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 183–192.

[6] E. Rosten and T. Drummond, *Machine Learning for High-Speed Corner Detection*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443.

[7] J. Shi and C. Tomasi, "Good features to track," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 1994, pp. 593–600.

[8] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *International conference on computer vision*. IEEE, 2011, pp. 2564–2571.

[9] D. G. Lowe, "Object recognition from local scale-invariant features," in *The proceedings of the seventh IEEE international conference on Computer vision*, vol. 2. IEEE, 1999, pp. 1150–1157.

[10] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[11] S. Reich, M. Seer, L. Berscheid, and J.-M. Braun, "Computational Neuroscience - Quadrocopter Vision Odometry," 2016, accessed: 2016-09-09. [Online]. Available: http://www.dpi.physik.uni-goettingen.de/cns/redir.php?s=quadrocopter

[12] M. El-gayar, H. Soliman, and N. Meky, "A comparative study of image low level feature extraction algorithms," *Egyptian Informatics Journal*, vol. 14, no. 2, pp. 175–181, 2013.

[13] J. Heinly, E. Dunn, and J.-M. Frahm, *Comparative Evaluation of Binary Features*. Springer Berlin Heidelberg, Oct 2012, pp. 759–773.

[14] B. D. Lucas, T. Kanade *et al.*, "An iterative image registration technique with an application to stereo vision." in *IJCAI*, vol. 81, no. 1, 1981, pp. 674–679.

[15] C. T. Dunkel, *Person Detection and Tracking Using Binocular Lucas-Kanade Feature Tracking and K-means Clustering*. ProQuest, 2008.

[16] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. "O'Reilly Media, Inc.", 2008.

[17] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[18] Z. Gan and Q. Tang, *Visual sensing and its applications: integration of laser sensors to industrial robots*. Springer Science & Business Media, 2011.