2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

A Monocular Pose Estimation System based on Infrared LEDs

Matthias Faessler, Elias Mueggler, Karl Schwabe and Davide Scaramuzza

***Abstract*— We present an accurate, efficient, and robust pose estimation system based on infrared LEDs. They are mounted on a target object and are observed by a camera that is equipped with an infrared-pass filter. The correspondences between LEDs and image detections are first determined using a combinatorial approach and then tracked using a constant-velocity model. The pose of the target object is estimated with a P3P algorithm and optimized by minimizing the reprojection error. Since the system works in the infrared spectrum, it is robust to cluttered environments and illumination changes. In a variety of experiments, we show that our system outperforms state-of-the-art approaches. Furthermore, we successfully apply our system to stabilize a quadrotor both indoors and outdoors under challenging conditions. We release our implementation as open-source software.**

SUPPLEMENTARY MATERIAL

An open-source implementation and video of this work are available at: `http://rpg.ifi.uzh.ch/software`

I. INTRODUCTION

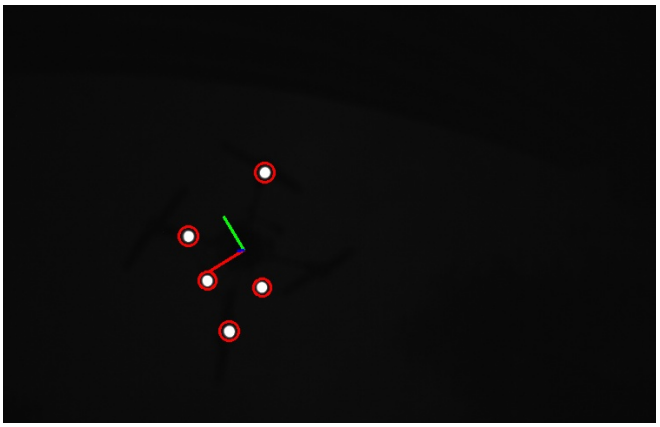
A. Motivation

Rescue missions in disaster sites are extremely challenging for robots since they have to deal with unforeseeable and unstructured environments. Furthermore, these robots need to have many attributes, such as being able to overcome obstacles, being reasonably fast, and being able to manipulate their environment. We plan on using a team of heterogeneous robots (aerial and ground) to collaborate and make use of their individual strengths to address these challenges. Since mutual localization is one of the most fundamental parts in controlling a team of mobile robots, a system that can accurately and reliably estimate the mutual pose of the robots is necessary. For both indoor and outdoor operations, it needs to be robust to cluttered environments, dynamic scenes, and illumination changes. Part of our robot team are quadrotors of approximately 0.5 m in size (see Fig. 1a). Small quadrotors have fast dynamics and, thus, need a frequent and precise estimate of their 6 DOF pose to be stabilized. Furthermore, small quadrotors have limited payload and battery power, as well as limited onboard-processing power. Hence, the pose estimation system must be lightweight, energy efficient, and computationally inexpensive. Existing systems lack the combination of all these requirements.

We propose a pose estimation system that consists of multiple infrared LEDs and a camera with an infrared-pass filter. The LEDs are attached to the robot that we want to track, while the observing robot is equipped with the camera. Since this system operates in the infrared spectrum, the LEDs are easy to detect in the camera image. This also applies to situations with cluttered backgrounds and illumination changes. The infrared LEDs can be detected by motion-capture systems and their position on the target object can, thus, be precisely determined. Furthermore, the camera only requires a short exposure time, which allows for high frame rates (up to 90 fps in our system). To track an object, only a few LEDs need to be mounted. However, this is not an issue in terms of power consumption or payload, even for small quadrotors.

In our experiments, we compare the performance of our

(a) Stabilizing a quadrotor above a ground robot.



(b) View from the camera on the ground robot.

Fig. 1: A camera with an infrared-pass filter is mounted on a ground robot and used to stabilize a quadrotor above it. The red circles in (b) indicate LED detections. The pose estimate is illustrated by the projection of the body-fixed coordinate frame of the quadrotor.

The authors are with the Robotics and Perception Group, University of Zurich, Switzerland—`http://rpg.ifi.uzh.ch`. This research was supported by the Swiss National Science Foundation through project number 200021-143607 ("Swarm of Flying Cameras") and the National Centre of Competence in Research Robotics.

system to a previous approach [1] and pose estimation from AprilTags [2] as well as a motion capture system (we use OptiTrack[1]). Furthermore, we show that our system meets all the stated requirements and can be used to stabilize a quadrotor.

B. Related Work

Our work is in line with other monocular vision-based pose estimation systems [1], [3], while improving on accuracy, versatility, and performance. Since [1] uses passive markers or LEDs in the visible spectrum, their performance decreases in cluttered environments and in low-light conditions. While their system is restricted to four markers, our algorithm can handle any number of LEDs to increase robustness. Additionally, our system is robust to false detections. The setup of [3] uses a special event-based camera [4] with LEDs that blink at different frequencies. The great advantage of this camera is that it can track frequencies up to several kilohertz. Therefore, pose estimation can be performed with very low latencies. Its precision, however, is limited due to the low sensor resolution (128x128 pixels).

Nowadays, artificial patterns, such as ARTags [5] and AprilTags [2], are often used for mutual localization. In addition to a pose estimate, they also provide a unique ID of the tag. Those patterns require a large, flat area on the target object. This makes them unsuitable for micro-aerial vehicles, since it would interfere with their aerodynamics.

Another popular method for pose estimation are motion-capture systems, such as OptiTrack and Vicon.[2] While these systems yield high precision at high frame rates (up to $350\,\text{Hz}$), they are proprietary, expensive, and typically require a fixed installation with many cameras. In single-camera mode, OptiTrack uses the marker size, d, for estimating its 3D position and can, thus, compute a $6\,\text{DOF}$ pose using only three markers. However, the marker size in the image, d^*, degrades quickly for larger distances to the camera z. Consequently, also the accuracy of the pose estimate degrades: $d^* \propto d/z$. Large markers are also not suitable for micro-aerial vehicles. Nonetheless, all findings of this paper could also be applied for single-camera motion capture systems.

Estimating the camera pose from a set of 3D-to-2D point correspondences is known as Perspective from n Points (PnP) (or resection) [6]. As shown in [6], the minimal case involves three 3D-to-2D correspondences. This is called Perspective from 3 Points (P3P) and returns four solutions that can be disambiguated using one or more additional point correspondences. To solve the P3P problem, we use the algorithm in [7], which proved to be accurate while being much faster than any previous implementation. A comprehensive overview of PnP algorithms can also be found in [7] and references therein. Furthermore, we use P3P to initialize an optimization step that refines the pose by minimizing the reprojection error based on all detected LEDs.

[1] http://www.naturalpoint.com/optitrack/

[2] http://www.vicon.com/

A heuristic approach that provides near-optimal marker configurations on the target object is presented in [8]. Since the geometry of micro-aerial vehicles restricts the configuration space drastically, we do not apply such algorithms and rely on some heuristics mentioned in Section II-A.

The remainder of the paper is organized as follows. In Section II, we describe the prerequisites of our system. Our algorithm is described in Section III and evaluated in Section IV.

II. System Prerequisites

A. Hardware

Our system consists of infrared LEDs at known positions on the target object and an external camera with an infrared-pass filter. With at least four LEDs on the target object and the corresponding detections in the camera image, we can compute the $6\,\text{DOF}$ pose of the target object with respect to the camera. However, to increase robustness, our system can also handle more than four LEDs on the target object. Furthermore, in case of self-occlusions or false positive detections, e.g. caused by reflections, we are still able to recover the full pose if at least four LEDs are detected.

The placement of the LEDs on the target object is arbitrary, but must be non-symmetric. In addition, the LEDs should not lie in a plane to reduce ambiguities of the pose estimation. To increase precision, they should span a large volume. Robustness can be increased if the LEDs are visible from as many view points as possible.

B. Calibration

As mentioned above, our system requires knowledge of the LED configuration, i.e. the positions of the LEDs in the reference frame of the target object. Since infrared LEDs are detectable by a motion capture system, we can use it to determine the positions of the LEDs with sub-millimeter accuracy. To do so, we first assign the desired coordinate frame to the target object in the motion capture system (we used OptiTrack) using a calibration stand (see Fig. 2). This can be achieved by knowing the exact marker positions on the calibration stand and mounting the target object on it. Then, we can track the target object in the motion capture system and read out the positions of the single LEDs, which can be transformed into the target-object coordinate frame. Furthermore, we need to know the intrinsic camera parameters, which we obtain using the camera calibration tools of ROS.[3]

III. Algorithm

A. Overview

The flowchart of our algorithm is presented in Fig. 3. The current camera image, the LED configuration, and previously estimated poses serve as inputs to our algorithm. In a first step, we detect the LEDs in the image. Then, we determine the correspondences using prediction or, if that fails, using a

[3] http://wiki.ros.org/camera_calibration/

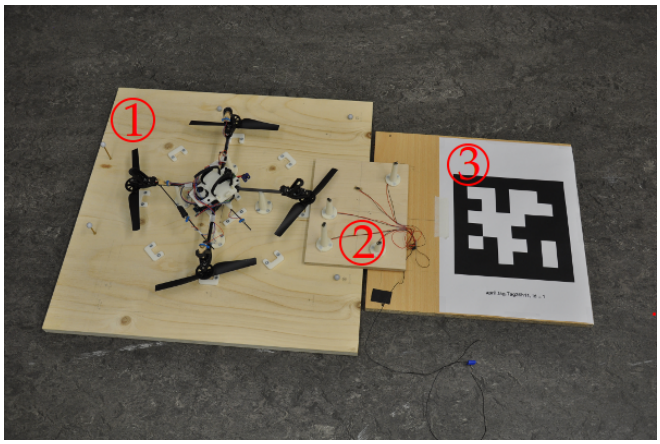


Fig. 2: Calibration stand (1) to determine the exact location of the LEDs in the reference frame of a target object using a motion-capture system. The quadrotor, the target object (2) and the AprilTag (3) were used to perform the experiments in Section IV.

combinatorial brute-force approach. Finally, the pose is optimized such that the reprojection error of all detected LEDs is minimized. This optimization also returns the covariance of the pose estimate, which is crucial information in further processing, e.g., in filtering or SLAM applications. All steps are described in more detail below.

B. Notation

We denote the LED positions on the target object as $\mathbf{l}_i \in \mathbb{R}^3$, the number of LEDs as $n_\mathcal{L}$, and the LED configuration as $\mathcal{L} = \{\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_{n_\mathcal{L}}\}$. The detections of the LEDs in the image are denoted as $\mathbf{d}_j \in \mathbb{R}^2$, measured in pixels. The number of detections is $n_\mathcal{D}$ and the set of detections is $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_{n_\mathcal{D}}\}$. Note, while $\mathcal{L}$ results from the calibration, $\mathcal{D}$ depends on the current image. A correspondence of an LED $\mathbf{l}_i$ and a detection $\mathbf{d}_j$ is denoted as $\mathbf{c}_k = \langle \mathbf{l}_i, \mathbf{d}_j \rangle \in \mathcal{C} \subset \mathcal{L} \times \mathcal{D}$. Poses are denoted as $P \in SE(3)$. We use grayscale images $\mathbf{I}(u, v) : \mathbb{N}^{w \times h} \to \{0, 1, \ldots, 255\}$, where w and h denote the image width and height, respectively.

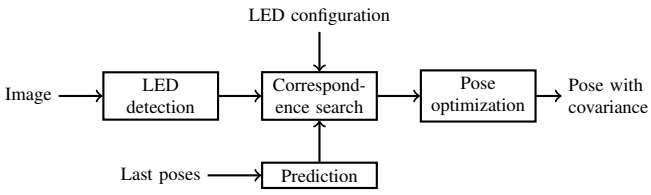


Fig. 3: Flowchart showing the main steps of our algorithm.

C. LED Detection

Since we are using infrared LEDs whose wavelength matches the infrared-pass filter in the camera, they appear very bright in the image compared to their environment. Thus, a thresholding function is sufficient to detect the LEDs $\mathcal{D}$,

$$\mathbf{I}'(u, v) = \begin{cases} \mathbf{I}(u, v), & \text{if } \mathbf{I}(u, v) > \text{threshold}, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

This threshold parameter depends on the shutter speed of the camera settings. However, we found that a large range of parameters works well (80–180). We then apply Gaussian smoothing and group neighboring pixels to blobs. To estimate the center of these blobs with sub-pixel accuracy, we weigh the pixels with their intensity. The center is then calculated using first image moments that are defined as

$$M_{pq} = \sum_u \sum_v u^p v^q \mathbf{I}'(u, v). \tag{2}$$

The weighted center, i.e. the (distorted) LED detection in the image, is then

$$\hat{u} = M_{10}/M_{00}, \tag{3}$$

$$\hat{v} = M_{01}/M_{00}. \tag{4}$$

In all calculations to come, we assume the standard pinhole camera model. Thus, we have to correct the detections $\mathbf{d}_j$ for radial and tangential distortion. We do this using the OpenCV library [9].

D. Correspondence Search

Since the different LEDs cannot be distinguished from each other in the image, we need to find the correspondences between the LED detections, $\mathcal{D}$, in the image and the LEDs, $\mathcal{L}$, on the target object. To do so, we make use of the P3P algorithm in [7] to compute four pose candidates for every combination of three detections in the image, $\mathcal{D}_3$, and every permutation of three LEDs on the target object, $\mathcal{L}_3$. For every pose candidate, we then project the LEDs that were *not* used to compute the pose candidate, $\mathcal{L} \setminus \mathcal{L}_3$, into the camera image. If such a reprojection has a nearest neighbor of the detections $\mathcal{D}$ closer than a threshold λ_r, we consider the LED to correspond to this detection. For the reprojection-distance threshold, we typically use $\lambda_r = 5\,\text{pixels}$. To be robust to outliers, we form a histogram with bins for every detection-LED pair. A histogram bin is increased whenever a pair is considered to be a correspondence. This procedure returns the set of correspondences $\mathcal{C}$ and is summarized in Algorithm 1. The procedure for finding the final correspondences from the histogram is illustrated in Fig. 5.

For $n_\mathcal{D}$ detections and $n_\mathcal{L}$ LEDs on the object, we will obtain N pose candidates,

$$N = 4 \cdot \binom{n_\mathcal{D}}{3} \cdot \frac{n_\mathcal{L}!}{(n_\mathcal{L} - 3)!}. \tag{5}$$

This number grows quickly for a large $n_\mathcal{D}$ or $n_\mathcal{L}$. However, since we use only a few LEDs (typically four or five) and false-positive detections are rare, this is not an issue. Numbers of pose candidates computed according to (5) are shown in Fig. 4.

$n_\mathcal{D}$ / $n_\mathcal{L}$	4	5	6	7	8
4	384	960	1,920	3,360	5,376
5	960	2,400	4,800	8,400	13,440
6	1,920	4,800	9,600	16,800	26,880

Fig. 4: Number of pose candidates N based on the number of detections $n_\mathcal{D}$ and the number of LEDs on the target object $n_\mathcal{L}$.

Algorithm 1 Correspondence search

```
for all D_3 ∈ Combinations(D, 3) do
    for all L_3 ∈ Permutations(L, 3) do
        L_r ← L \ L_3
        P ← P3P(D_3, L_3)
        for all P ∈ P do
            found ← False
            for all l ∈ L_r do
                p ← project(l, P)
                for all d ∈ D do
                    if ||d − p||^2 < threshold then
                        inc(histogram(l, d))
                        found ← True
                    end if
                end for
            end for
            if found then
                inc(histogram(L_3, D_3))
            end if
        end for
    end for
end for
```

$\mathcal{D}$ / $\mathcal{L}$	$\mathbf{l}_1$	$\mathbf{l}_2$	$\mathbf{l}_3$	$\mathbf{l}_4$	$\mathbf{l}_5$
$\mathbf{d}_1$	1	**12**	0	1	0
$\mathbf{d}_2$	0	3	2	1	**8**
$\mathbf{d}_3$	1	0	1	**13**	1
$\mathbf{d}_4$	1	0	1	4	1
$\mathbf{d}_5$	2	1	0	1	1
$\mathbf{d}_6$	**11**	3	0	2	2

Fig. 5: Correspondence histogram. The numbers indicate how often a small reprojection error of LED $\mathbf{l}_i$ to the detection $\mathbf{d}_j$ was found. Under ideal conditions, this value is $\binom{n_\mathcal{L}}{3}$ for a correspondence and zero otherwise. In practice, we iteratively search for the highest number in the histogram, take the respective LED and image point as the correspondence, and then ignore that column in all subsequent iterations. Note that this allows a detection to correspond to multiple LEDs, but not vice versa (cf. Section IV-B). In this example, in the first iteration, we match $\mathbf{l}_4$ and $\mathbf{d}_3$, i.e. $\mathbf{c}_1 = \langle \mathbf{l}_4, \mathbf{d}_3 \rangle$. All further correspondences are also marked in bold. Note that $\mathbf{l}_3$ was not matched since all remaining entries in its column are lower than a threshold (we chose $0.5\binom{n_\mathcal{L}}{3}$). This LED might be occluded (cf. Section IV-B) or its detection failed.

E. Prediction

Since the brute-force matching in the previous section can become computationally expensive, we predict the next pose using the current and the previous pose estimates. A constant-velocity model is used for prediction. The pose P is parametrized by twist coordinates $\boldsymbol{\xi}$. We predict the next pose linearly [10, p. 511],

$$\hat{\boldsymbol{\xi}}_{k+1} = \boldsymbol{\xi}_k + \Delta T \left(\boldsymbol{\xi}_k - \boldsymbol{\xi}_{k-1} \right), \tag{6}$$

$$\Delta T = \begin{cases} 0, & \text{if } n_P = 1, \\ (T_{k+1} - T_k)/(T_k - T_{k-1}), & \text{if } n_P \geq 2, \end{cases} \tag{7}$$

where T_k is the time at step k and n_P the number of previously estimated poses.

Using the predicted pose, we project the LEDs into the camera image. We then match each prediction with its closest detection, if they are closer than a threshold. (Note that this threshold is different from λ_r). This condition prevents false correspondences, e.g. if an LED is not detected. We typically use $5\,$pixels for that threshold. We then check if the predicted correspondences are correct. To do so, we compute the four pose candidates with the P3P algorithm for every combination of three correspondences. We then compute the projection of the remaining LEDs and check if at least $75\,\%$ of them are below the reprojection threshold λ_r. If this is true for one of the four pose candidates of more than $70\,\%$ of the combinations of correspondences, we consider them as correct. In case we could not find the correct correspondences, we reinitialize the tracking using the brute-force method from the previous section.

F. Pose Optimization

To estimate the target-object pose, P^*, we use all correspondences in $\mathcal{C}$ and iteratively refine the reprojection error [11, p. 286f.] starting with a solution from the P3P algorithm as an initial estimate, that is

$$P^* = \arg\min_P \sum_{\langle \mathbf{l}, \mathbf{d} \rangle \in \mathcal{C}} \| \pi \left(\mathbf{l}, P \right) - \mathbf{d} \|^2, \tag{8}$$

where $\pi : \mathbb{R}^3 \times SE(3) \to \mathbb{R}^2$ projects an LED into the camera image. For the optimization, we parametrize the pose using the exponential map and apply a Gauss-Newton minimization scheme.

The covariance of the final pose estimate, $\boldsymbol{\Sigma}_P \in \mathbb{R}^{6\times6}$, is a byproduct of the Gauss-Newton scheme, since it requires the computation of the Jacobian matrix, $\mathbf{J} \in \mathbb{R}^{2\times6}$. Using the derivation of [12, p. 182ff.], we can compute $\mathbf{J}$ in closed-form. The covariance of the pose, $\boldsymbol{\Sigma}_P$, is then obtained by [13]

$$\boldsymbol{\Sigma}_P = \left(\mathbf{J}^\top \boldsymbol{\Sigma}_\mathcal{D}^{-1} \mathbf{J} \right)^{-1}, \tag{9}$$

where $\boldsymbol{\Sigma}_\mathcal{D} \in \mathbb{R}^{2\times2}$ is the covariance of the LED detections, which we conservatively set to $\boldsymbol{\Sigma}_\mathcal{D} = \mathbf{I}_{2\times2} \cdot 1\,\text{pixel}^2$.

TABLE I: Comparison of pose estimation performance.

	April-Tags [2]	Breitenmoser et al. [1]	Our system	
Mean Position Error	1.41	1.5	**0.74**	cm
Standard Deviation	1.02	0.7	**0.46**	cm
Max Position Error	11.2	12.1	**3.28**	cm
Mean Orientation Error	1.53	1.2	**0.79**	°
Standard Deviation	1.61	**0.4**	0.41	°
Max Orientation Error	19.5	4.5	**3.37**	°

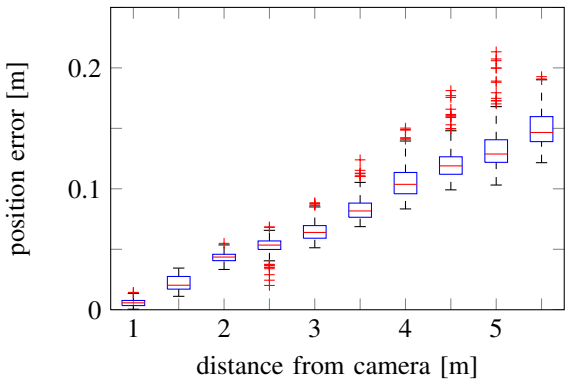
IV. Evaluation

A. Benchmarks

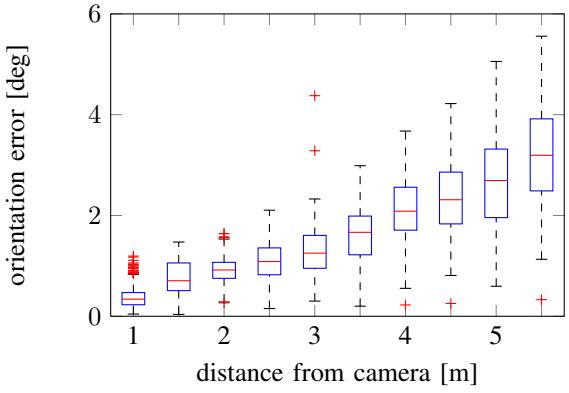
To evaluate our system, we compare it to a previous system [1] and to AprilTags [2]. A MatrixVision mvBlueFOX-MLC200w monochrome camera[4] with an infrared-pass filter, a resolution of 752x480 pixels, and a field of view of 90° was used for the experiments. Furthermore, we added reflective markers to the camera to obtain ground truth in an OptiTrack motion-capture system. On the target object, we mounted SMD LEDs (of type Harvatek HT-260IRPJ or similar) since they proved to have a wide radiation pattern. We used either a configuration of four or five infrared LEDs on the target object (see Fig. 2). Both configurations have a circumsphere radius of 10.9 cm. Since the infrared LEDs are directly visible in the motion capture system, no additional markers were needed to obtain the ground truth data of the target object. To have a direct comparison, we attached an AprilTag with edge length of 23.8 cm to the target object. For pose estimation from the AprilTags, we used a C++ implementation.[5]

In a first run, the target object is positioned at a fixed location while the camera is moving. We used $n_{\mathcal{L}} = 4$ LEDs on the target object and performed excitations of the camera in all six degrees of freedom. Fig. 6 shows position and orientation as well as the respective errors. Since our setup is virtually identical to [1] and the trajectory follows the similar excitations in all six degrees of freedom, we claim that the results are comparable. In Table I, we compare our performance to the system in [1] and to AprilTags [2]. As an orientation error metric, we used the angle of the angle-axis representation. Since we cannot measure the precise location of the center of projection of the camera, we use the first 10 % of the data for hand-eye calibration. We also estimate the pose of the AprilTag with respect to the target object in the same way. The dataset consists of 7,273 images. In 2 images (0.03 %), not all 4 LEDs could be detected. In another 2 images, no solution was found. Thus, in 99.94 % of all images, a good estimate was found.

In a second experiment, we evaluated the error with respect to the distance between the camera and the target object. We used $n_{\mathcal{L}} = 5$ LEDs on the target object to increase robustness. The camera was moved from 0.8 m to 5.6 m, while recording a total of 2,651 images. Fig. 7 shows the boxplots for both position and orientation. For the orientation error, we used again the axis-angle representation. In 3 images (0.04 %) at more than 5 m distance, incorrect correspondences lead to pose estimates that were off by more than 90° in orientation. We consider them as outliers and, thus, they are not shown in Fig. 7b.



(a) Position error.



(b) Orientation error.

Fig. 7: Boxplot of the pose estimation errors with respect to the distance between the target object and the camera. The target object was equipped with $n_{\mathcal{L}} = 5$ LEDs.

B. Occlusions and Alignment of LEDs

Here we take a deeper look at two special cases. First, we evaluate the estimation error in situations where an occlusion occurs. In Fig. 8b, we show such a situation. Since at least four LEDs are always visible for the entire duration of the occlusion, the estimation error does not change significantly (cf. Fig. 10).

Secondly, we look at the situations where two LEDs appear as one in the image (e.g. in Fig. 9b). In such situations, they cannot be detected separately. Thus, as soon as the two LEDs are detected as one, there is an immediate increase in the estimation error. As the LEDs appear closer to each other, the error decreases until the two LEDs are almost perfectly aligned. It then increases until the two LEDs can again be detected separately, whereafter it drops to the initial values. This behavior can be seen in Fig. 11.

[4] http://www.matrix-vision.com/
[5] http://people.csail.mit.edu/kaess/apriltags/

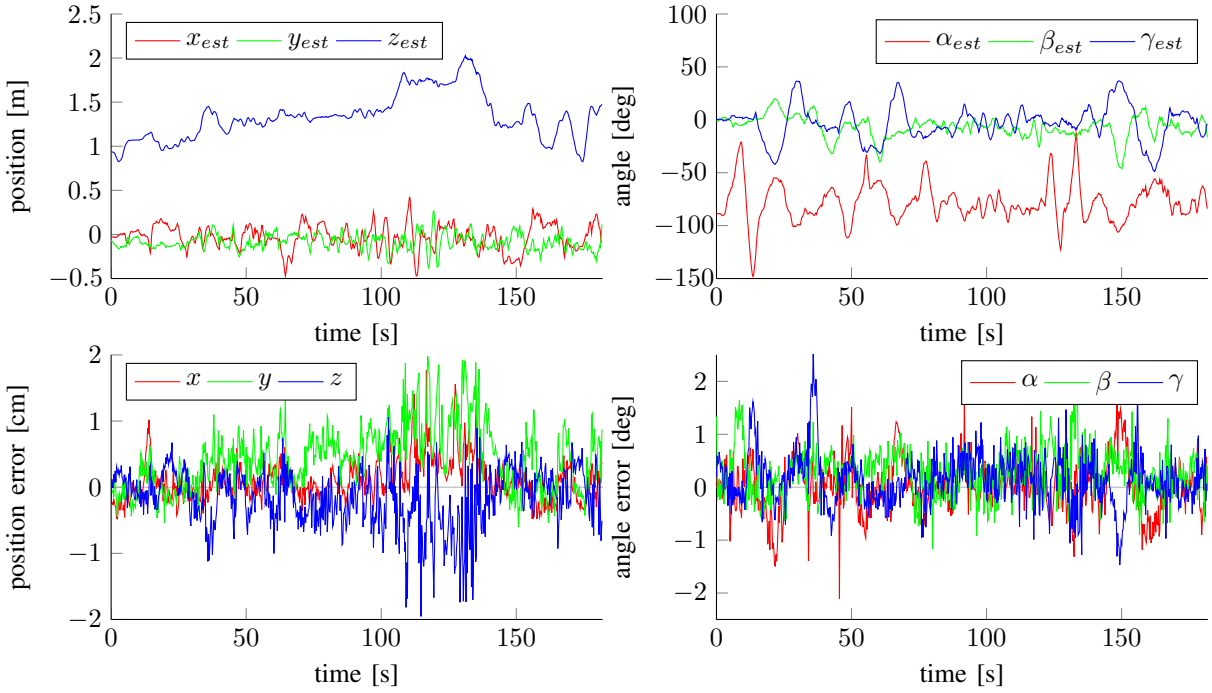
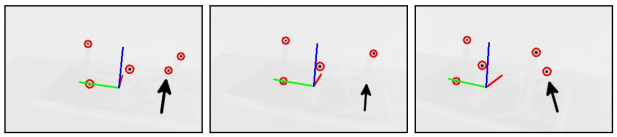
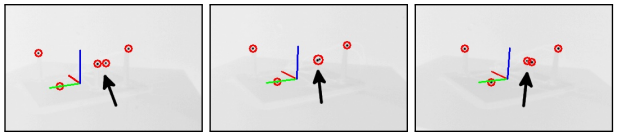


Fig. 6: Estimation of position and orientation, as well as the respective errors. Ground truth is not shown because there is no visible difference to the estimated values at this scale. The orientation is parametrized with Euler angles, i.e. yaw (α), pitch (β), and roll (γ). The target object was equipped with $n_{\mathcal{L}} = 4$ LEDs. For four out of a total of 7,273 images, no estimate could be resolved.



(a) T_1 (b) T_2 (c) T_3

Fig. 8: Camera images (a) before, (b) during, and (c) after an LED occlusion. The camera images were inverted for better contrast. The estimation errors for this experiment are shown in Fig. 10. The arrow indicates the LED that is occluded.



(a) T_1 (b) T_2 (c) T_3

Fig. 9: Camera images (a) before, (b) during, and (c) after an alignment of two LEDs. The camera images were inverted for better contrast. The estimation errors for this experiment are shown in Fig. 11. The arrow indicates the two LEDs that were aligned.

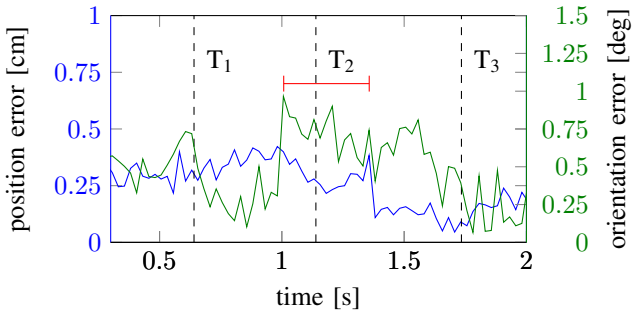


Fig. 10: Error plots of position and orientation during an LED occlusion. As an orientation error metric, we used the angle of the angle-axis representation. The red interval indicates the duration of the occlusion. The times T_i correspond to the images in Fig. 8.

C. Quadrotor Stabilization

To show the applicability of our system in a real-world scenario, we demonstrate closed-loop control of a quadrotor[6] using pose estimates of our system at 40 Hz. We attached $n_{\mathcal{L}} = 5$ LEDs to a quadrotor, which is based on the PIXHAWK platform [14] (see Fig. 2), and mounted the camera on a KUKA youBot [15] (see Fig. 1). We used a lens with field of view of 120°. Our system is robust enough to handle illumination changes from daylight to complete darkness, false detections, occluded LEDs, and dynamic backgrounds. It is also fast and precise enough to stabilize the quadrotor when it gets pushed or flies outdoors with unpredictable winds.

[6]A video is included as an attachment to this paper.

D. Execution Time

The mean and maximum execution times for each step of our algorithm can be found in Table II. They were measured while running our system on a dataset with 2,400 images and LED configurations consisting of 4 and 5 LEDs. For the timing, we enforced a brute-force correspondence search in each step. However, if we use prediction, this search is required less than 0.2 % of the time. We used a laptop with

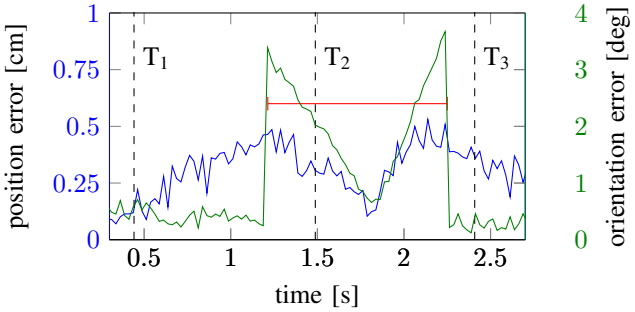


Fig. 11: Error plots of position and orientation when two LEDs appear as one in the camera image. As an orientation error metric, we used the angle of the angle-axis representation. The red interval marks the duration of the alignment. The times T_i correspond to the images in Fig. 9.

TABLE II: Execution times of the individual steps of our algorithm (corresponding to subsections III-C to III-F)

Number of LEDs		LED detection [ms]	Correspondence search [ms]	Prediction [ms]	Pose optimization [µs]	Total (w/o prediction) [ms]	**Total (w/ prediction)** [ms]
$n_\mathcal{L} = 4$	Mean	2.7	1.1	0.2	31	5.0	**3.8**
	σ	0.9	0.6	0.1	10	1.4	**1.1**
	Maximum	5.1	5.7	0.6	94	10.1	**6.5**
$n_\mathcal{L} = 5$	Mean	2.7	4.9	0.3	36	9.0	**3.8**
	σ	0.8	2.0	0.1	11	2.5	**1.1**
	Maximum	5.1	14.0	0.7	93	17.6	**9.3**

an Intel i7-3720 (2.60 GHz) processor. Note that on average the LED detection makes up 71 % of the execution time. This could be drastically reduced by defining a region of interest around the predicted detections, as is done in [1].

V. Conclusions

We presented an accurate, versatile, and robust monocular pose tracking system based on infrared LEDs. Comprehensive experiments showed its superiority over previous approaches for pose estimation and its applicability to robots with fast dynamics such as quadrotors. Our system is available as an open-source ROS [16] package, so that it can easily be integrated into other robotic platforms.

Future work will include the extension to track multiple objects and will integrate the dynamical model of the target object for prediction and filtering. Furthermore, we plan to use the system for mutual localization in a team of quadrotors.

Acknowledgement

We gratefully acknowledge the contribution of Flavio Fontana for helping with the quadrotor experiments.

References

[1] A. Breitenmoser, L. Kneip, and R. Siegwart, "A Monocular Vision-based System for 6D Relative Robot Localization," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[2] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[3] A. Censi, J. Strubel, C. Brandli, T. Delbruck, and D. Scaramuzza, "Low-latency localization by Active LED Markers tracking using a Dynamic Vision Sensor," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[4] P. Lichtsteiner, C. Posch, and T. Delbruck, "An 128x128 120dB 15us-latency temporal contrast vision sensor," *IEEE J. Solid State Circuits*, vol. 43, no. 2, pp. 566–576, 2007.

[5] M. Fiala, "ARTag, a fiducial marker system using digital techniques," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR)*, 2005.

[6] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[7] L. Kneip, D. Scaramuzza, and R. Siegwart, "A Novel Parametrization of the Perspective-Three-Point Problem for a Direct Compuation of Absolute Camera Position and Orientation," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR)*, 2011.

[8] T. Pintaric and H. Kaufmann, "A Rigid-Body Target Design Methodology for Optical Pose-Tracking Systems," in *Proc. ACM Symp. on Virtual Reality Software and Technology (VRST)*, 2008.

[9] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[10] J. Gallier, *Geometric Methods and Applications For Computer Science and Engineering*, ser. Texts in applied mathematics. New York, NY, USA: Springer, 2001.

[11] R. Szeliski, *Computer Vision: Algorithms and Applications*, 1st ed. New York, NY, USA: Springer, 2010.

[12] E. Eade, "Monocular Simultaneous Localisation and Mapping," Ph.D. dissertation, Cambridge University, 2008.

[13] B. Bell and F. Cathey, "The iterated Kalman filter update as a Gauss-Newton method," *IEEE Trans. on Automatic Control*, vol. 38, pp. 294–297, 1993.

[14] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys, "PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision," *Autonomous Robots*, vol. 33, pp. 21–39, 2012.

[15] R. Bischoff, U. Huggenberger, and E. Prassler, "KUKA youBot - a mobile manipulator for research and education," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[16] 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 *IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Software*, 2009.