Proceedings of the 33rd Chinese Control Conference
July 28-30, 2014, Nanjing, China

A Robust Vision System for a UAV Transporting Cargoes between Moving Platforms

Shiyu Zhao[1], Zhangyuan Hu[2], Mingfeng Yin[3], Kevin Z. Y. Ang[1], Peidong Liu[1], Fei Wang[4], Xiangxu Dong[4], Feng Lin[4], Ben M. Chen[1], Tong H. Lee[1]

1. Department of Electrical & Computer Engineering, National University of Singapore
E-mail: shiyuzhao@nus.edu.sg, kevinang@nus.edu.sg, elelp@nus.edu.sg, bmchen@nus.edu.sg, eleleeth@nus.edu.sg

2. Department of Information Engineering, The Chinese University of Hong Kong
E-mail: mua08p@gmail.com

3. School of Automation, Nanjing University of Science and Technology
E-mail: yinmingfengnjust@gmail.com

4. Temasek Laboratories, National University of Singapore
E-mail: wangfei@nus.edu.sg, tsldngx@nus.edu.sg, linfeng@nus.edu.sg

Abstract: Motivated by the 2013 International UAV Innovation Grand Prix (UAVGP), we design and implement a real-time vision system for an unmanned helicopter to transfer cargoes between two moving platforms fully autonomously. The key algorithms in the vision system include ellipse detection, ellipse tracking and single-circle-based pose estimation. Comprehensive experiments have verified the efficiency, accuracy and robustness of the algorithms. The developed vision system has helped us achieving a great success in UAVGP. Our team was ranked first in the final round competition.

Key Words: Vision-based navigation, Unmanned aerial vehicle, Ellipse detection, Circle-based pose estimation, OpenCV.

1 Introduction

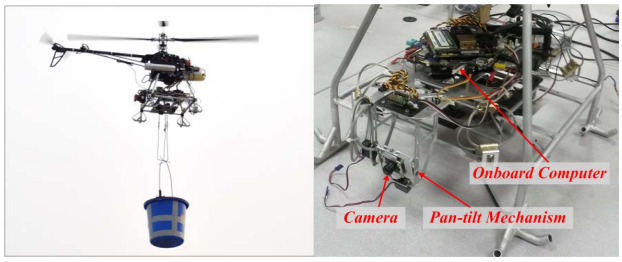
The work presented in this paper is motivated by an international competition, the 2013 International UAV Innovation Grand Prix (UAVGP), which was held in September 2013, Beijing, China. The competition requires an unmanned aerial vehicle (UAV) to autonomously transfer cargoes from one moving platform to the other (see Fig. 1). The cargoes are four buckets, each of which weighs about 1.5 kg. In addition to transferring cargoes, the UAV should also be able to autonomously take off and land. The entire competition task must be completed by the UAV fully autonomously without any human intervention. The techniques developed for the competition are potentially applicable to practical autonomous cargo transfer tasks such as vertical replenishment.

The team from the Unmanned Aircraft Systems (UAS) group at National University of Singapore has developed a sophisticated unmanned helicopter system (see Fig. 2). The unmanned helicopter successfully completed the required competition tasks. Our team was ranked first in the final round competition. The vision system plays a key role for autonomous cargo transfer. This paper primarily introduces the vision system of the unmanned helicopter. Details of the hardware, software, control and navigation of the helicopter can be found in [1, 2].

This paper mainly presents efficient and robust algorithms for ellipse detection, ellipse tracking and single-circle-based position estimation. These algorithms are not restricted to cargo transfer. They are also potentially applicable to a variety of vision-based tasks such as takeoff and landing [3–5], target tracking and following [6, 7] and visual servoing [8]when they use circles as the visual features. When designing the vision system, we put more emphasis on the efficiency of the algorithms and the convenience of implementation. The proposed algorithms can be conveniently implemented by the popular library OpenCV.

(a) The platform (b) The helicopter grabbing a cargo

Fig. 1: An illustration of the competition task.



(a) Unmanned helicopter (b) Onboard vision system

Fig. 2: The unmanned helicopter and the onboard vision system.

2 Overview of the Vision System

In terms of hardware, the vision system consists of a wide-angle camera, a pan-tilt mechanism, and an onboard computer (see Fig. 2). The camera is downward-looking and installed on the nose of the helicopter. The horizontal angle of view of the camera is 87°, and the vertical one is 71°. The camera is mounted on a pan-tilt mechanism, which can be controlled at 50 Hz to ensure that the image plane is perfectly level. The pan-tilt mechanism is essential for robust image tracking because it can compensate the fast dynamic rotation of the helicopter and hence the location of the tar-

get changes smoothly in the image. The onboard computer for vision processing is an AscTec Mastermind PC with the CPU as Intel Core2Duo SL9400 (2x1.86 GHz). The vision system can be executed at 10 Hz in the onboard computer.

In the competition, four cargoes (buckets) are initially placed inside four circles on one platform, respectively. The four cargoes need to be transferred one by one into the four circles on the other platform. Apparently, circles are the key features for the vision system to process. The vision system should be able to detect all the circle areas and cargoes in the image, and then intelligently decide which cargo the helicopter should load and where the cargo should be transferred. When loading or unloading a cargo, the target area must be continuously tracked and localized by the vision system. In some emergency cases (for example, all the cargoes are out of the field of view of the camera), the vision system should inform the control system such that the unmanned helicopter can take appropriate actions.

To complete the above tasks, we need to solve three key problems: ellipse detection, ellipse tracking and single-circle-based position estimation. In fact, the three problems are also crucial for any other vision-based tasks that use circles as visual features. Although a series of other algorithms have also been developed for the competition, we will mainly introduce our proposed algorithms for the three problems in this paper due to space limitations.

3 Ellipse Detection

Ellipse detection has been investigated extensively up to now. We choose ellipse fitting [9, 10] as the core of our ellipse detection algorithm. That is mainly because ellipse fitting is very efficient compared to, for example, Hough transform based ellipse detection algorithms [11]. Since ellipse fitting itself is not able to determine whether a contour corresponds to an ellipse or not, we present a three-step procedure to robustly detect ellipses. The procedure consists of 1) pre-processing, 2) ellipse fitting and 3) post-processing. The pre-processing is based on affine moment invariants (AMIs) [12]; the post-processing is based on the algebraic error between the contour and the fitted ellipse.

3.1 A Three-step Ellipse Detection Procedure

In order to detect ellipses, we need first obtain the contours that corresponds to the ellipses by using, for example, color thresholding or edge detection. Once the contours have been obtained, each of them will be processed by the following three steps to see if it corresponds to an ellipse.

3.1.1 Pre-processing based on AMIs

Moment invariants are very useful tools for pattern recognition. The most popular moment invariants were proposed by Hu [13]. Hu's moment invariants are invariant only under translation, rotation and scaling of the object. The work in [12] proposed AMIs that are invariant under general affine transformations. Since an arbitrary ellipse can be obtained by applying an affine transformation to a circle, all ellipses and all circles have exactly the same AMIs. Hence AMIs are powerful tools to detect ellipses [14, 15].

Four AMIs were proposed in [12, Section 2.1]. In our work, we only use the first three as the fourth one is much more complicated and less reliable than the first three. The first three AMIs are expressed as

$$\begin{aligned} I_1 &= (\mu_{20}\mu_{02} - \mu_{11}^2)/\mu_{00}^4, \\ I_2 &= (\mu_{30}^2\mu_{03}^2 - 6\mu_{30}\mu_{21}\mu_{12}\mu_{03} + 4\mu_{30}\mu_{12}^3 + 4\mu_{21}^3\mu_{03} \\ &\quad - 3\mu_{21}^2\mu_{12}^2)/\mu_{00}^{10}, \\ I_3 &= \big(\mu_{20}(\mu_{21}\mu_{03} - \mu_{12}^2) - \mu_{11}(\mu_{30}\mu_{03} - \mu_{21}\mu_{12}) \\ &\quad + \mu_{02}(\mu_{30}\mu_{12} - \mu_{21}^2)\big)/\mu_{00}^7, \end{aligned} \tag{1}$$

where the central moment μ_{ij} is given by

$$\mu_{ij} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}(x-\bar{x})^i(y-\bar{y})^j\rho(x,y)\mathrm{d}x\mathrm{d}y. \tag{2}$$

In the above equation, $(\bar{x}, \bar{y})$ is the coordinate of the centroid and $\rho(x, y)$ is the density distribution function.

We now calculate the AMIs of ellipses and circles. Since all ellipses and circles have the same AMIs, for the sake of simplicity, we can calculate the AMIs of a special one: a circle centered at the origin with the radius as one unit. Suppose $\rho(x, y) = 1$ when (x, y) is on the circle, and $\rho(x, y) = 0$ otherwise. Hence (2) can be rewritten as $\mu_{ij} = \int_0^{2\pi}\cos^i\theta\sin^j\theta\mathrm{d}\theta$. Then it is easy to calculate that $\mu_{00} = 2\pi$, $\mu_{11} = 0$, $\mu_{02} = \mu_{20} = \pi$ and $\mu_{12} = \mu_{21} = \mu_{03} = \mu_{30} = 0$. Substituting these values into (1) yields

$$I_1 = \frac{1}{16\pi^2} \approx 0.006332,\ I_2 = I_3 = 0. \tag{3}$$

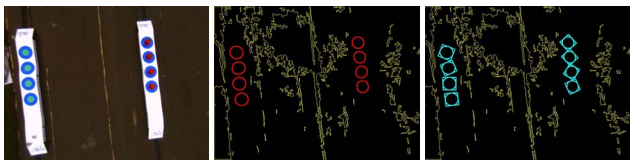
In summary, the pre-processing step is to compute the AMIs of each contour and compare them with the theoretical values in (3). If the AMIs of a contour is sufficiently close to the theoretical values, then the contour can be classified as an ellipse. According to our experience, we recommend the thresholds for I_1, I_2 and I_3 to be ± 0.0003, ± 0.0000001 and ± 0.000001, respectively. In addition, all the central moments required to calculate AMIs as shown in (1) can be conveniently computed by the OpenCV function *moments*. An example is shown in Fig. 3 to verify the effectiveness and robustness of the pre-processing. As can be seen, the contours that correspond to ellipses are all successfully detected in the presence of a large number of non-elliptical contours.

3.1.2 Ellipse Fitting

There are a variety of ellipse fitting algorithms in the literature (see, for example, [9, 10]). In our work, we choose the ellipse fitting function, *fitEllipse*, implemented in OpenCV. Experiments show that this function is efficient and accurate enough for our work. Fig. 3 gives an example to demonstrate the ellipse fitting. As shown in Fig. 3(c), all ellipses in the image are successfully obtained based on the elliptical contours detected by the pre-processing.

3.1.3 Post-processing based on Algebraic Error

In order to further improve the robustness of the ellipse detection algorithm, we adopt a post-processing step to calculate the algebraic error between a contour and its fitted



(a) (b) (c)

Fig. 3: An example to demonstrate the pre-processing and ellipse fitting. (a) Color image; (b) Elliptical contours detected based on AMIs; (c) Fitted ellipses with rotated bounding boxes.

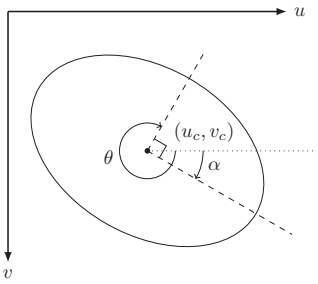


Fig. 4: An illustration of the ellipse parameters.

ellipse. If the algebraic error is larger than a threshold, the fitted ellipse will be classified as a false detection.

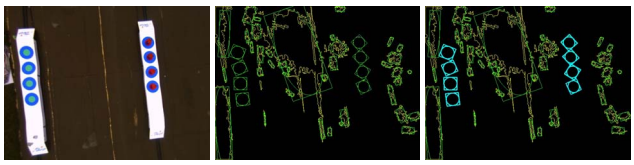
The algebraic error between a contour and the fitted ellipse is defined as follows. Let (u, v) be the coordinate of a pixel point in the image coordinate frame. As shown in Fig. 4, the u- and v-axis of the image frame defined in OpenCV are pointing right and downward, respectively. The origin of the image frame is located at the upper left corner of the image. Let (u_c, v_c) be the coordinate of the ellipse center, and a and b be the lengths of the semi-major and semi-minor axes, respectively. Denote α as the angle between the major axis and the u-axis (see Fig. 4). The angle α is positive clockwise based on the right-hand rule. The algebraic error between a contour and the fitted ellipse is defined as

$$e_{\text{alg}} \triangleq \frac{1}{n}\sum_{i=1}^{n}\left|\frac{[(u_i-u_c)\cos\alpha+(v_i-v_c)\sin\alpha]^2}{a^2} + \frac{[(u_i-u_c)\sin\alpha-(v_i-v_c)\cos\alpha]^2}{b^2}-1\right|, \quad (4)$$

where (u_i, v_i) with $i = 1, \ldots, n$ are the coordinates of the points on the contour. It is clear that the error will be zero if all the points of the contour are on the fitted ellipse.

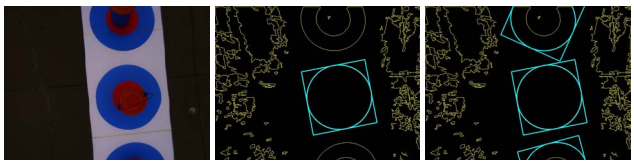
In summary, the post-processing will eliminate a fitted ellipse if its algebraic error is larger than a threshold. According to our experience, we recommend the threshold to be 0.1. An example is given in Fig. 5 to demonstrate the effectiveness and robustness of the post-processing. Note the ellipse detection in Fig. 5 is based merely on ellipse fitting and post-processing, while pre-processing is not used. As shown in Fig. 5(b), ellipses are fitted for all of the contours. Fig. 5(c) shows that all falsely fitted ellipses can be eliminated by the post-processing.

In practical implementation, an ellipse is described by a structure named *RotatedRect* in OpenCV. We can easily (but may not directly) obtain all the parameters required to compute the algebraic error from *RotatedRect*. The angle returned by *RotatedRect* is not α; instead, it is θ as shown in Fig. 4. According to a series of experiments we have



(a) (b) (c)

Fig. 5: An example to demonstrate the post-processing. (a) Color image; (b) Fitted ellipses for all contours (contours with too few points are excluded); (c) Good ellipses detected based on the algebraic error.



(a) Color image (b) Whole ellipses (c) All ellipses

Fig. 6: An example to demonstrate the detection of partially occluded ellipses.

conducted, θ is the angle between the minor-axis of the ellipse and the u-axis of the image frame. Moreover, the angle θ is positive clockwise, and it is always in the interval $[135°, 315°)$. Therefore, the angle α can be obtained as $\alpha = \pi/2 + \theta$. It is worth noting that adding $k\pi$ to α with k as an integer does not affect the algebraic error in (4).

3.2 Special Case: Partially Occluded Circle

In practice, it is common that only part of a circle can be seen by the camera due to occlusion or limited field of view. Then the contour corresponding to the circle will not be elliptical (see, for example, the top and the bottom contours in Fig. 6). We next show a procedure to fit ellipses for these contours. Firstly, compute the convex hull of each of the contours that are not detected as whole ellipses. The OpenCV function *convexHull* can be conveniently used to do this. Secondly, fit ellipse for the convex hull, and then compute the algebraic error between the fitted ellipse and the convex hull (not the contour). If the algebraic error is too large, the fitted ellipse will be eliminated, otherwise it can be classified as an occluded ellipse. Fig. 6 shows an example to demonstrate the effectiveness of the above procedure.

3.3 Summary of the Ellipse Detection Algorithm

The proposed ellipse detection algorithm is summarized in Algorithm 1. Several important remarks are given here. First, Algorithm 1 can detect both whole and partial ellipses. Second, as demonstrated in Fig. 3 and Fig. 5, either pre- or post-processing can independently detect ellipses. It is not necessarily required that the pre- and post-processing procedures must be both adopted in the ellipse detection algorithm. But in order to improve the robustness of the algorithm, it is recommended to use both of them unless the computational resource is extremely limited.

4 Ellipse Tracking

In practical applications, multiple ellipses may be detected in an image, but we may be only interested in one of them. After certain initialization procedure, the ellipse of interest needs to be tracked over the image sequence such that the pose of the corresponding circle can be estimated con-

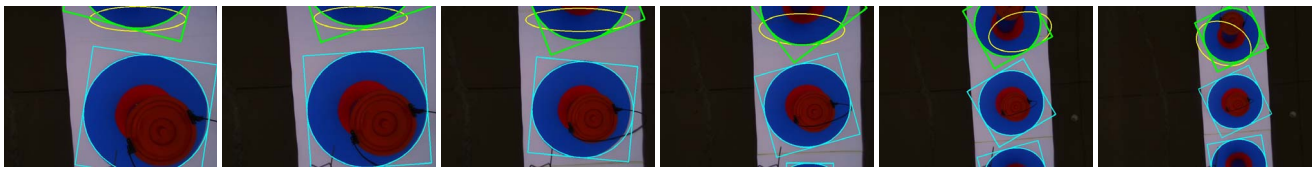


Fig. 7: An example to demonstrate ellipse tracking over consecutive images. In each image, all ellipses have been detected and drawn in cyan. The tracked ellipse is highlighted in green. The yellow ellipse is the target area returned by CAMShift.

Algorithm 1 Robust Real-time Ellipse Detection

1: **Preparation:** Detect contours in the image using, for example, edge detection or color thresholding.

2: **Pre-processing:** For each contour, compute its central moments with the OpenCV function *moments*, and then calculate I_1, I_2 and I_3 as shown in (1). Compare the calculated I_1, I_2 and I_3 with the values in (3). If

$$|I_1 - 1/(16\pi^2)| < 0.0003,$$
$$|I_2| < 0.0000001, \quad |I_3| < 0.000001, \tag{5}$$

then the contour is a good candidate for a whole ellipse.

3: **Ellipse fitting:**

a) For each contour that satisfies (5), use the OpenCV function *fitEllipse* to fit an ellipse.

b) For each contour that does not satisfy (5), use the OpenCV function *convexHull* to get the convex hull of the contour, and then fit an ellipse for the convex hull.

4: **Post-processing:**

a) For each contour that satisfies (5), compute its algebraic error e_{alg} as defined in (4).

b) For each contour that does not satisfy (5), compute its algebraic error e_{alg} between its convex hull and the fitted ellipse.

If $|e_{\text{alg}}| < 0.1$, then the fitted ellipse can be confirmed as a correct one.

Algorithm 2 Ellipse Tracking based on CAMShift

For frame $k+1$:

1: Update the tracking window: choose the initial tracking window for CAMShift as the tangent rectangle of the target ellipse in frame k.

2: Update the histogram: denote h_k and h_{k+1} as the histogram used for tracking in frame k and $k+1$, respectively. Let $\tilde{h}_k$ be the histogram of the target ellipse area in frame k. Then update h_{k+1} by

$$h_{k+1} = wh_k + (1-w)\tilde{h}_k, \tag{6}$$

where $w \in [0, 1]$ is a weight factor.

3: Compute the back projection image based on the updated histogram. This can be done by using the OpenCV function *calcBackProject*.

4: Obtain the final tracking window computed by CAMShift. This can be done by substituting the back projection image and the initial tracking window to the OpenCV function *CamShift*.

5: Find the ellipse that is located closest to the final tracking window given by CAMShift.

tinuously. We choose the efficient image tracking method CAMShift [16] as the core of our tracking algorithm. The proposed ellipse tracking algorithm is summarized in Algorithm 2. It is notable that the scale and shape of the target ellipse may dynamically vary when the camera and the target circle are relatively moving. As a result, the histogram of the target ellipse area needs to be updated continuously. In the histogram-update law as shown in (6), the new histogram is a convex combination of the old one and the one of the ellipse area in frame k. If $w = 1$, there will be no histogram update; if $w = 0$, the histogram would be exactly the one of the area enclosed by the target ellipse in frame k. According to a series of experiments, the value $w = 0.95$ can give a satisfactory tracking performance.

Fig. 7 shows an example to demonstrate the ellipse tracking performance. As can be seen, the target ellipse is tracked robustly though its position and scale keep changing.

5 Single-Circle-based Pose Estimation

The application of circles in camera calibration and pose estimation has been investigated extensively [3, 4, 17–20]. However, the existing work mainly focused on the cases of concentric circles [3, 4, 18, 19], while the aim of our work is to do pose estimation based only on *one single circle*. The topic addressed in [20] is similar to ours, but it is concluded in [20] that other information such as parallel lines are required to estimate the pose of a single circle. From a practical point of view, we can successfully solve the single-circle-based pose estimation problem by adopting a reasonable assumption. The main contribution of our work is that we analyze the specific properties of circle-based pose estimation and identify *four 3D-2D point correspondences*. The four correspondences can be used to estimate the pose of the single circle by various mature pose estimation algorithms (see, for example, [21–23] and the references therein).

5.1 Pose Estimation from Four Point Correspondences

We first introduce three coordinate frames involved in the pose estimation problem: world frame, camera frame and image frame. Without loss of generality, we set the world frame in the following way: the origin of the world frame coincides with the circle center, and the Z-axis of the world frame is orthogonal to the plane Π_1 that contains the circle. Thus the Z component of any points on the circle is zero, and the coordinate of the circle center is $(0, 0, 0)$ in the world frame. The camera frame has its origin located at the camera center and its z-axis orthogonal to the image plane Π_2. The image frame defined in OpenCV has been shown in Fig. 4. The pose estimation problem we are going to solve is given as below.

Problem 1. *Consider one single circle in the world frame. Its perspective projection, an ellipse, has already been detected in the image. Given the radius of the circle and the intrinsic parameters of the camera, estimate the coordinate of the circle center in the camera frame.*

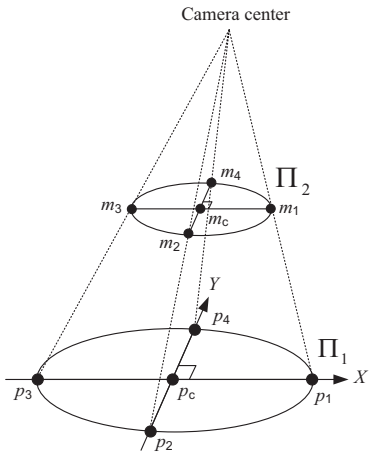


Fig. 8: Perspective projection of a circle and the four point correspondences.

Denote $p = [X, Y, Z]^T \in \mathbb{R}^3$ and $q = [x, y, z]^T \in \mathbb{R}^3$ as the coordinates of any point in the world and the camera frame, respectively. Let $R \in \mathbb{R}^{3\times3}$ and $T \in \mathbb{R}^3$ be the rotational and translational transformation from the world frame to the camera frame, respectively. Then we have $q = Rp+T$. Use the subscript c to denote the coordinate of the circle center. Then we have $p_c = [0, 0, 0]^T$ and hence $q_c = T$. Therefore, the translation T is the one we need to estimate.

We next identify four sets of corresponding points on the ellipse and the circle, respectively. Let m_1, m_2, m_3 and m_4 be the four vertexes of the ellipse in the image (see Fig. 8). Given the parameters of the ellipse such as (u_c, v_c), a, b, α as shown in Fig. 4, the coordinates of the four points can be easily obtained as $m_1 = (u_c + a\cos\alpha, v_c + a\sin\alpha)$, $m_2 = (u_c - b\sin\alpha, v_c + b\cos\alpha)$, $m_3 = (u_c - a\cos\alpha, v_c - a\sin\alpha)$ and $m_4 = (u_c + b\sin\alpha, v_c - b\cos\alpha)$. Let p_1, p_2, p_3 and p_4 be the four corresponding points on the circle. Point p_i corresponds to m_i for $i = 1, \ldots, 4$. The following assumption on $p_1, \ldots, p_4$ is the key to our algorithm.

Assumption 1. *The four points $p_1, \ldots, p_4$ are evenly distributed on the circle, which means p_1, p_c and p_3 are collinear, p_2, p_c and p_4 are collinear and the line p_1p_3 is perpendicular to p_2p_4.*

Remark 1. *It can be proved that Assumption 1 is valid if and only if the image plane is parallel to the plane that contains the circle. Due to space limitations, the proof will be given somewhere else. This assumption is valid in our work because the circles are placed on the horizontal ground and the camera is controlled such that its image plane is also always horizontal.*

Under Assumption 1, we are able to set the world frame such that the four points $p_1, \ldots, p_4$ are located at the X- and Y-axes, respectively (see Fig. 8). As a result, if the diameter of the circle is given as r, we have $p_1 = (r, 0, 0)$, $p_2 = (0, -r, 0)$, $p_3 = (-r, 0, 0)$ and $p_4 = (0, r, 0)$. It is worth noting that the orientation of the world frame does not affect $q_c = T$, and hence its orientation can be set freely. Thus, we successfully obtain four sets of point correspondences $\{m_1, p_1\}$, $\{m_2, p_2\}$, $\{m_3, p_3\}$ and $\{m_4, p_4\}$, which can be consequently used for pose estimation. Our pose estimation algorithm is summarized in Algorithm 3.

Algorithm 3 Single-Circle-based Pose Estimation from Four Point Correspondences

1: Obtain the four image points, which are the vertexes of the ellipse:

$$m_1 = (u_c + a\cos\alpha, v_c + a\sin\alpha),$$
$$m_2 = (u_c - b\sin\alpha, v_c + b\cos\alpha),$$
$$m_3 = (u_c - a\cos\alpha, v_c - a\sin\alpha),$$
$$m_4 = (u_c + b\sin\alpha, v_c - b\cos\alpha).$$

2: Obtain the four corresponding points on the circle:

$$p_1 = (r, 0, 0), \quad p_2 = (0, -r, 0),$$
$$p_3 = (-r, 0, 0), \; p_4 = (0, r, 0).$$

3: Substitute the four correspondences $\{m_1, p_1\}$, $\{m_2, p_2\}$, $\{m_3, p_3\}$ and $\{m_4, p_4\}$ to the OpenCV function *solvePnP*. Suppose T is the translational vector returned by *solvePnP*. Then T is the translation from the world frame to the camera frame. Thus $q_c = T$.

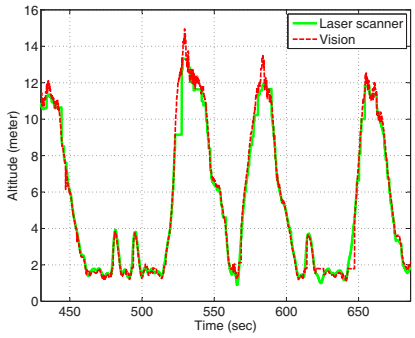


Fig. 9: The altitude estimated by vision VS the altitude measured by the laser scanner.

6 Experimental and Competition Results

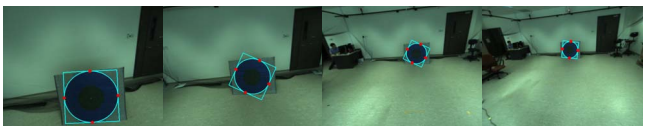
6.1 Competition Results

Successful completion of all competition tasks is the strongest evidence for the accuracy and robustness of the vision system. A video of the onboard vision processing for the final round competition can be found at http://youtu.be/GSeafBsASTs.

We now show the flight data of the final competition to quantitatively verify the vision system. Although the helicopter is equipped with a GPS, the accuracy of the GPS is much worse than that of the vision system. Fortunately, a laser scanner is mounted at the back of the helicopter to measure the altitude. The laser measurement is very accurate in general, and hence can be viewed as the ground truth for the altitude. Note the vision system can also estimate the altitude though it is not required. Thus, we can compare the altitude estimated by vision with the laser measurement. The flight data as shown in Fig. 9 indicates that the vision estimates are accurate as they coincide with the laser measurements well. As can be seen, when the altitude is less than 2 m, the error of the vision measurement is less than 0.2 m.

6.2 Experiments in a Vicon System

Experiments in a Vicon system have been conducted to verify the accuracy of Algorithm 3. The Vicon system is



(a) (b) (c) (d)

Fig. 10: Images captured in the experiment. The size of each image is 640×480 pixels.

Table 1: Pose estimation results using the images in Fig. 10.

Image	Circle center q_c by vision (m)	$\|q_c\|$ by vision (m)	$\|q_c\|$ by Vicon (m)	Error of $\|q_c\|$ (m)
(a)	$(-0.566, 0.074, 1.566)$	1.667	1.650	0.017
(b)	$(-0.278, 0.160, 2.361)$	2.383	2.363	0.020
(c)	$(0.644, 0.507, 4.120)$	4.201	4.218	0.017
(d)	$(1.044, 0.253, 5.194)$	5.304	5.253	0.051

used to obtain the ground truth of the range from the circle center to the camera center. The diameter of the target circle in our experiment is one meter. The images of the target circle are given in Fig. 10. Note the image plane is approximately parallel to the circle plane. The estimation results by Algorithm 3 are given in Table 1. As can be seen, the proposed algorithm can give very accurate estimates of the range from the circle center to the camera center.

7 Conclusions

This paper presented a robust and real-time vision system for ellipse detection, ellipse tracking, and single-circle-based pose estimation. Experimental results and the competition results have verified the efficiency, accuracy, and robustness of the proposed algorithms. The proposed algorithms are not restricted to the specific competition task. They are also applicable to a wide range of vision-based navigation tasks where circles are utilized as the visual features.

References

[1] F. Wang, P. Liu, S. Zhao, B. M. Chen, S. K. Phang, S. Lai, and T. H. Lee, "Guidance, navigation and control of an unmanned helicopter for automatic cargo transportation," to be presented at the 33rd Chinese Control Conference, Nanjing, China, July 2014.

[2] P. Liu, X. Dong, F. Wang, and B. M. Chen, "Development of a comprehensive software architecture for vertical replenishment by an unmanned helicopter," submitted to Journal of Aerospace Information Systems.

[3] S. Lange, N. Sunderhauf, and P. Protzel, "A vision based onboard approach for landing and position control of an autonomous multirotor UAV in GPS-denied environments," in *Proceedings of the 2009 International Conference on Advanced Robotics*, Munich, Germany, June 2009, pp. 1–6.

[4] D. Eberli, D. Scaramuzza, S. Weiss, and R. Siegwart, "Vision based position control for MAVs using one single circular landmark," *Journal of Intelligent and Robotic Systems*, vol. 61, pp. 495–512, 2011.

[5] S. Yang, S. A. Scherer, and A. Zell, "An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle," *Journal of Intelligent and Robotic Systems*, vol. 69, pp. 499–515, 2013.

[6] F. Lin, X. Dong, B. M. Chen, K. Y. Lum, and T. H. Lee, "A robust real-time embedded vision system on an unmanned rotorcraft for ground target following," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 2, pp. 1038–1049, February 2012.

[7] Y. Hu, W. Zhao, and L. Wang, "Vision-based target tracking and collision avoidance for two autonomous robotic fish," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 5, pp. 1401–1410, May 2009.

[8] H. Lang, M. T. Khan, K.-K. Tan, and C. W. de Silva, "Developments in visual servoing for mobile manipulation," *Unmanned Systems*, vol. 1, no. 1, pp. 143–162, July 2013.

[9] A. Fitzgibbon, M. Pilu, and R. B. Fisher, "Direct least square fitting of ellipses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 476–480, May 1999.

[10] S. J. Ahn, W. Rauh, and H.-J. Warnecke, "Least-squares orthogonal distances fitting of circle, sphere, ellipse, hyperbola, and parabola," *Pattern Recognition*, vol. 34, pp. 2283–2303, 2001.

[11] R. A. McLaughlin, "Randomized hough transform: Improved ellipse detection with comparison," *Pattern Recognition Letters*, vol. 19, pp. 299–305, March 1998.

[12] J. Flusser and T. Suk, "Pattern recognition by affine moment invariants," *Pattern Recognition*, vol. 26, no. 1, pp. 167–174, January 1993.

[13] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE Transactions on Information Theory*, vol. 8, no. 2, pp. 179–187, February 1962.

[14] P. L. Rosin, "Measuring shape: ellipticity, rectangularity, and triangularity," in *Proceedings of the 15th International Conference on Pattern Recognition*, Barcelona, Spain, 2000, pp. 952–955.

[15] F. Mokhtarian and S. Abbasi, "Shape similarity retrieval under affine transforms," *Pattern Recognition*, vol. 35, pp. 31–41, 2002.

[16] J. G. Allen, R. Y. D. Xu, and J. S. Jin, "Object tracking using CamShift algorithm and multiple quantized feature spaces," in *Proceedings of the Pan-Sydney area workshop on Visual information processing*, Darlinghurst, Australia, 2004, pp. 3–7.

[17] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *Proceedings of the 1997 IEEE Conference on Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, 1997, pp. 1106–1112.

[18] J.-S. Kim, P. Gurdjos, and I.-S. Kweon, "Geometric and algebraic constraints of projected concentric circles and their applications to camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 4, pp. 637–642, 2005.

[19] G. Jiang and L. Quan, "Detection of concentric circles for camera calibration," in *Proceedings of the 10th IEEE International Conference on Computer Vision*, Beijing, China, 2005, pp. 333–340.

[20] G. Wang, J. Wu, and Z. Ji, "Single view based pose estimation from circle or parallel lines," *Pattern Recognition Letters*, vol. 29, pp. 977–985, 2008.

[21] R. M. Haralick, H. Joo, D. Lee, S. Zhuang, V. G. Vaidya, and M. B. Kim, "Pose estimation from corresponding point data," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, no. 6, pp. 1426–1446, 1989.

[22] L. Quan and Z. Lan, "Linear n-point camera pose determination," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 8, pp. 774–780, August 1999.

[23] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An accurate $O(n)$ solution to the PnP problem," *International Journal of Computer Vision*, vol. 81, no. 2, pp. 155–166, February 2009.