Evaluation of tracking algorithms to autonomously following dynamic targets at low altitude

Hendrikus R. Oosterhuis[1], and Arnoud Visser[1]

Intelligent Systems Laboratory Amsterdam
Universiteit van Amsterdam

Abstract

One of the main requirements in enabling autonomous behaviour of Micro Aerial Vehicle is the tracking of visual targets. The MAV encounters several problems including a limited point of view, a motion blur due to constant aerial movement and the localization problems that come with aerial flight. A common visual target found in our human environment is the QR-code. In order to successfully track such a code the detection algorithms need to cope with its easily distorted features. While recognition is possible at close distance it is not reliable enough for real-time tracking. A solution to this problem is a hybrid of a recognition algorithm as provided by the ZXing[6] library and a robust tracking algorithm as provided by openTLD[5][4][7]. This paper is a survey of the experimental circumstances that influence the performance of such a hybrid system in a QR-code tracking task, which is one of the challenges of the Indoor Micro Aerial Vehicle 2013 competition.

1 Introduction

In robotics one of the main goals is to develop mobile robots that can operate autonomously in the real world environment. These autonomous robots have various purposes and are used for a wide range of applications such as inspection, exploration and rescue. Even though reasonable developments have been made in the robotics field, robots cannot operate autonomously in all circumstances the real world can provide.

One of the initiatives to promote the developments in autonomous robots is the International Micro Aerial Vehicle (IMAV) conference and competition, which is the basis of the Iran Open Flying robot competition[1]. The indoor challenge of this competition consist of an arena with several mission elements (see Fig 1). One of the mission elements are the 'Drop zones' (marked with ⑥ in Fig 1). For this mission element two drop zones are placed in the safe zone, both marked with a QR-code. One of the two zones moves over a given trajectory while the other remains static. Scoring is done by dropping a ball from the vehicle onto the drop zone. The static zone can be scored once whereas the moving zone can be scored as many times as possible. There is also a QR-code placed in the obstacle zone, points can be scored by finding and decoding it. (marked with ④)

The experiments described in this paper are performed prior to and after the Iran Open Flying robot competition. The drop zone mission element of the Iran Open proved to be considerately different from the IMAV and could not provide an indication of how well our algorithm performs.

AR.Drone SLAM [2] is a development framework for the Parrot AR.Drone developed and proposed by N. Dijkshoorn. This framework runs off-board and contains a real-time Simultaneous Localization and Mapping (SLAM) implementation based on a down-pointing camera.

[1] `http://www.imav2013.org/` and `http://2013.iranopen.ir/`

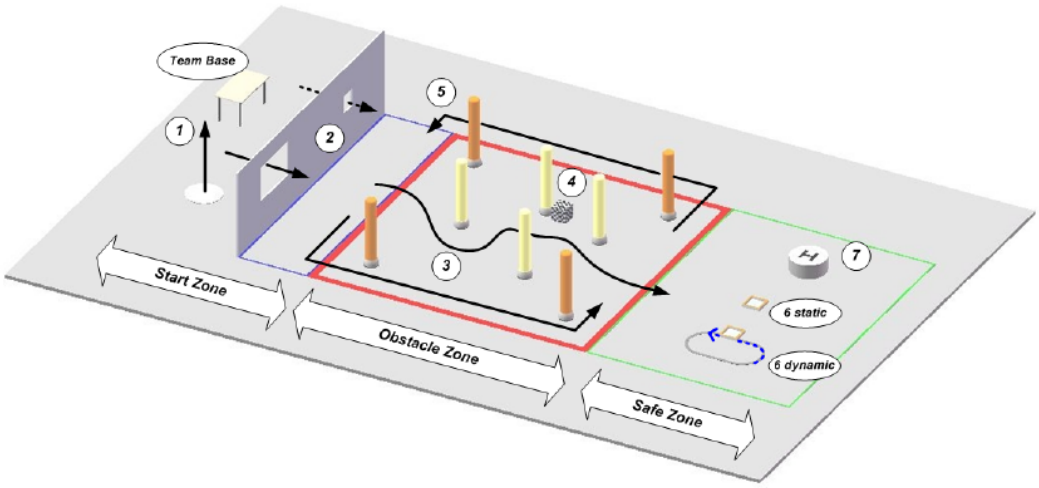

Figure 1: The indoor arena of the International Micro Aerial Vehicle (IMAV) competitions (Courtesy IMAV committee).

Therefore, it allows a MAV to know its position and movement in the environment by generating a feature map of the environment so the MAV can localize itself on this map. Furthermore, the framework facilitates control for a 3D mouse or a keyboard, and enables the generation of visual and elevation maps. The algorithms described in this study are an extension of this framework.

Indoor navigation is possible based on Visual SLAM, but when an object or location is marked by a symbol an algorithm developed for recognizing such a symbol is more beneficial. The human world is filled with symbols indicating important locations from helicopter platforms to street signs. When positioning systems based on satellites are unavailable systems have to rely on visual information. The variety of human symbols is immense, this paper limits a symbol to a QR-code. This allows us to use the IMAV to determine the performance of our implementation. [12] QR-codes are widely used and can be found in various public spaces, mainly they are intended for smart phones.

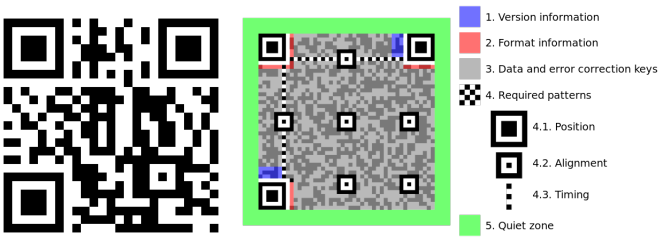

Figure 2: A QR-code and an overview of its general structure.

Source http://en.wikipedia.org/wiki/QR_code

QR-code is short for Quick Response code and was developed to be quickly decodable. It is mainly recognizable by its three corner squares, additionally the alignment cues ensure that the three corners are part of the same code.2 Arguably the most complete open source library of multi-format 1D/2D barcode image processing tools is ZXing[6]. It contains a QR-code decoder available in several languages including Java and C++. The performance of the decoder in relation to the angle at which a code is observed, has been researched.[10] This paper is a survey for the performance of ZXing when using a camera attached to a flying micro aerial vehicle. The influence of camera resolution and altitude are examined.

Therefore, the main research question is to determine the best altitude for a MAV with respect to the resolution of its camera in order to detect QR-codes. It is divided up in the following sub-questions:

- What is the optimal altitude for QR-code recognition?
- How does the resolution of the on board camera affect the optimal altitude?

Section 2 gives an overview is given over the related research regarding visual target recognition and UAV navigation. The approach of this paper is discussed in section 3. In section 4 the experiments are illustrated and in section 5 and 6 the results will be presented and discussed. In section 7 and 8 the conclusion of this paper will be presented and directions for future research will be proposed.

2 Related Work

This section gives an overview of the related research that has been done regarding autonomous navigation for unmanned aerial vehicles and QR-code detection. Since small quadcopters have become affordable, research on this platform is moving toward autonomous and intelligent applications. The mission elements from the IMAV show a great resemblance to autonomous landing, for both require tracking of a zone placed below the flying vehicle and navigation to keep the vehicle properly aligned. Beside research on the performance of ZXing this section briefly describes applications of autonomous landing for aerial vehicles.

Landing zone recognition needs to a real-time implementation to allow for a autonomous landing. A design and implementation has been proved to work[11], it relies on a simplification of the landing target design that significantly simplifies the computer vision tasks. Limiting them to corner detection and correspondence matching, customized algorithms allow for real time computation at a frame rate of 30 Hz.

Visually-Guided Landing is a task requiring the recognition of a landing zone and autonomous navigating of the Unmanned Aerial Vehicle. A successful design and implementation on a UAV has been presented.[9] In contrast with the MAV of this paper the design is made for a much larger vehicle. The landing algorithm is an integration of several algorithms for visual acquisition of the helipad and a navigation algorithm towards the helipad. The algorithm works with an arbitrary initial position and orientation. Navigation makes use of both GPS and vision whereas precise target detection and recognition is merely based on vision. The vehicle updates its landing target parameters based on vision and uses an onboard behaviour-based controller to follow a path to the landing site.

ZXing QR-code decoding among with other barcode formats have been evaluated in past research.[10] By creating a framework that can generate evaluation images by applying all distortions one should expect for a real-world application, a large scale evaluation of ZXing decoding tools was enabled. The QR-code delivers the best decoding results yet still suffers

from decoding problems within certain angles. It has the best decoding probability at 90° whereas it most problematic angles are 45°, 135°, 225° and 315°. Results with the framework have been compared to results of a real world scenario and shown to be reliable.

3 Approach

In this section the main approach that defines the autonomous behaviour for this paper is discussed.

3.1 Main Approach

The main approach of this paper for vision based tracking consists of two phases. These are the following phases:

In the **recognition phase**, the images received by the on board camera are scanned for QR-codes. If recognized an image of the found QR-code is stored.

The **tracking phase** using the image of a found QR-code openTLD[5][4][7] is used to track the QR-code. OpenTLD does not search for the defining features of a QRcode enabling it to track the QRcode when it is blurred or partly visible.

In the **navigation phase**, the drone is positioned above the observed QR-code, if no code was observed the system will descend to the optimal altitude for QR-code recognition.

These phases make up a closed loop system, first the recognition phase tries to find a QR-code if this fails the tracking phase tries to look for something that resembles the last observed QR-code. After one of first two phases succeeds or both fail the navigation phase is initiated. The approach is real-time and integrated in a ROS port of the AR.Drone SLAM [3] development framework.

3.1.1 Recognition Phase

In this paper the image received from the MAV is reformatted to a binary image. ZXing searches for the defining features of a QR-code, if found it determines the location of the three corner squares. Using the sonar at the bottom of the MAV it determines its altitude, given the size of the QR-code (for the IMAV it's 15cm by 15cm) it can calculate the dimensions of the QR-code as observed by its camera. Combining the size and the location of its corner squares, a bounding box is drawn around the QR-code. This bounding box is passed on to the openTLD tracker initiating its tracking and learning functions. After doing so ZXing attempts to decode the QR-code, if successful its contents will be displayed in the interface. This does not affect its navigation phase but may result in scoring points for other mission elements.

3.1.2 Tracking Phase

The tracking phase is only consulted if the recognition phase was unsuccessful. This means the QR-code is partially or not visible or has been distorted in such a manner that is it no longer recognizable as a QR-code. During flight this can be caused by blurring due to acceleration, brightness problems, a too low resolution caused by the MAV flying at a high altitude or a combination of the three. In these situations openTLD can still perform well, for it does not rely on features specific to QR-codes.

Instead openTLD is based on a Median-Flow tracker[5][1], it divides the bounding box in a grid of 10 x 10 points and estimates the motion of each of these points. By doing so it can find the QR-code even when it blurs or shrinks.

It also has a learning component that consists of two experts. The P-expert that predicts new occurrences of the object and thus increases generalization of the object detection and the N-expert generates negative training examples to prevent over-generalization. The two experts are applied simultaneously allowing the object detector to learn new occurrences while preventing it from giving false positives.

3.1.3 Navigation Phase

The navigation phase differs for the drop zones, navigating above a static drop zone is considerately easier than attaining a stable position above the moving drop zone. Therefore each zone has its own approach.

The **static zone** is after being recognized directly navigated towards. The dropping radius of the MAV has been determined and the vehicle navigates toward to the drop zone in such a manner that its dropping radius and the drop zone overlap. This trajectory is a straight line to its goal. If aligned the vehicle instantly drops a ball, since this zone can only be scored once the phase is not repeated.

The **dynamic zone** to avoid the complications of remaining aligned with a moving target, the MAV does not follow its target. Instead it makes use of the fact that the target repeats its trajectory. After its target has been recognized the MAV remains stationary, it waits for the target to move out of its view while tracking its movements. After the target has disappeared it moves towards the targets observed trajectory. Here it waits for the target to repeat its movement, when the target appears again the MAV will wait to align with the target and drop a ball when it has done so.

By avoiding fast movements that are required when a target is leaving the field of vision, the MAV has less quick accelerations and remains at a stable angle. This allows for better positioning and makes it less likely to overshoot.

With **no found zone** the MAV remains stationary, having no good estimate of where the QR-code is. This is preferred over flying towards the last direction the target was spotted, both zones will more likely be lost forever using this method. Whereas keeping in mind the property that the dynamic zone returns to all its previous locations, waiting for the QR-code is a safer approach. The MAV will slowly attain the altitude that is optimal for recognizing QR-codes (determined in the next section).

4 Experiments

This section describes the experiments performed to determine the optimal altitude for MAV onboard QR-code recognition using the ZXing library.

4.1 Platform

For this paper, the Parrot AR.Drone 2.0[2] was chosen as the MAV. The system is widely available and has both indoor on-board stabilization and is by default controlled through a wireless connection. The implementation was made in a ROS port of the AR.Drone SLAM [3] development framework. ROS[3][8] is short for Robot Operating System, furthermore a ROS port of openTLD was used.

[2] `http://www.parrot.com`
[3] `http://www.ros.org`

4.2 Camera Specifications and Modifications

The standard AR.Drone has a front camera and a bottom camera, the front camera has a resolution of 1280x720 pixels and a frame rate of 30 fps whereas the bottom camera only has 640x480 pixels and 60 fps. To examine the effect of camera resolution the front camera was turned 90° so its direction matches that of the bottom camera.

It should be noted that while the front camera records images at 1280x720, they are compressed to a 640x480 image before being send over the wireless connection.

4.3 Experiments

In order to investigate the optimal altitude for each camera, two experiments were conducted. The drone was flown manually above several QR-codes while footage from the cameras was recorded together with its corresponding altitude readings. For each camera a set of 6000 frames were randomly picked.Manually all the frames that fully displayed a QR-code were selected. A QR-code is considered fully displayed when all four corner squares are completely visible.
The collected frames are placed in bins corresponding to the altitude at which they were recorded. Each bin has frames that fall in a span of 83 mm, the amount of frames sampled per bin is shown together with the amount of frames where the QR-code was found and read. This gives us the probability of a visible QR-code being recognized or read.

The footage was recorded under several lighting conditions and varying backgrounds.

After recording the footage for the bottom camera it became clear the AR.Drone is unable to remain stable with an altitude of less then 40 cm. At these heights no frames from the front camera where taken from these altitudes for the AR.Drone is unable to keep a QR-code within its field of view. Instead for the front camera the experiment focused on altitudes greater than 40 cm since information about these heights are the most relevant.

4.4 Evaluation Criteria

To evaluate the detection algorithm we calculate recall, precision is abandoned since early testing showed no false positives at all. This is because ZXings approach is based on a real-time image feed. Instead of doing an exhaustive search on the image, ZXing gives up fairly quickly. It favors a search on the next image over the risk of false negatives. A reasonable decision considering the practical use of real-time recognition.

5 Results

In this section the results of the experiments will be presented.

Bottom Camera

Altitude (mm)	Samples	Found	Read	Chance of Finding	Chance of Reading
1 - 83	1	0	0	0	0
84 - 167	0	0	0	-	-
168 - 250	2	1	1	0.5	0.5
251 - 333	3	3	3	1	1
334 - 417	19	11	11	0.58	0.58
418 - 500	34	25	25	0.74	0.74
501 - 583	78	39	31	0.5	0.4
584 - 667	108	2	2	0.02	0.02
668 - 750	84	3	2	0.04	0.02
751 - 833	148	2	2	0.01	0.01
834 - 917	634	0	0	0	0
918 - 1000	169	1	0	0.01	0
1001 - 1083	38	0	0	0	0
1084 - 1167	47	0	0	0	0
1168 - 1250	31	0	0	0	0
1251 - 1333	68	0	0	0	0
1334 - 1417	116	0	0	0	0
1418 - 1500	52	0	0	0	0
1501 - 1583	28	0	0	0	0
1584 - 1667	35	0	0	0	0
1668 - 1750	53	0	0	0	0
1751 - 1833	49	0	0	0	0
1834 - 1917	40	0	0	0	0
1918 - 2000	22	0	0	0	0
2001 - 2083	17	0	0	0	0
2084 - 2167	35	0	0	0	0
2168 - 2250	11	0	0	0	0

Front Camera

Altitude (mm)	Samples	Found	Read	Chance of Finding	Chance of Reading
1 - 83	0	0	0	-	-
84 - 167	0	0	0	-	-
168 - 250	0	0	0	-	-
251 - 333	0	0	0	-	-
334 - 417	0	0	0	-	-
418 - 500	53	51	36	0.96	0.68
501 - 583	140	106	94	0.76	0.67
584 - 667	676	285	238	0.42	0.35
668 - 750	82	29	18	0.35	0.22
751 - 833	666	24	13	0.04	0.02
834 - 917	82	5	4	0.06	0.05
918 - 1000	58	0	0	0	0
1001 - 1083	68	0	0	0	0
1084 - 1167	289	0	0	0	0
1168 - 1250	655	0	0	0	0
1251 - 1333	121	0	0	0	0
1334 - 1417	69	0	0	0	0
1418 - 1500	58	0	0	0	0
1501 - 1583	104	0	0	0	0
1584 - 1667	77	0	0	0	0
1668 - 1750	118	0	0	0	0
1751 - 1833	202	0	0	0	0

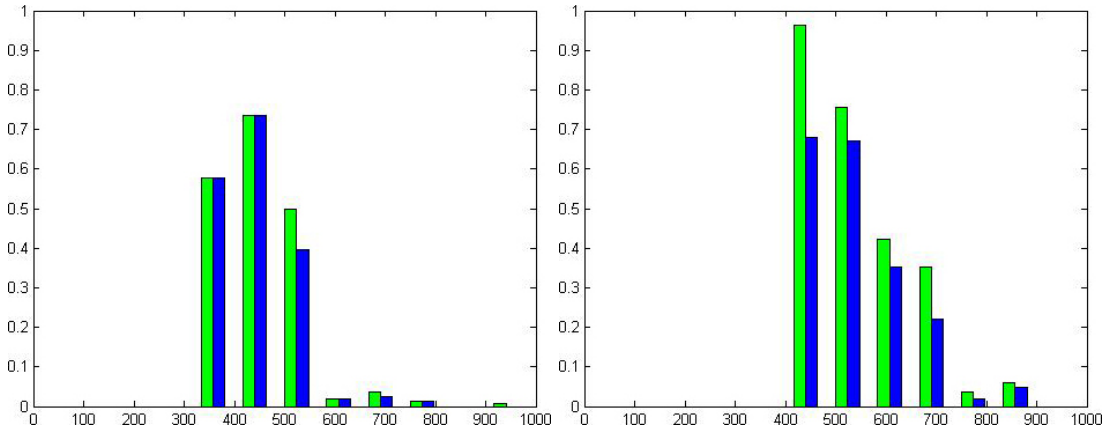

Figure 3: The probability of recognizing (green) and reading (blue) a visible QR-code per altitude in mm. Left: the bottom camera, Right: the front camera

6 Discussion

In this section the results of the experiments will be discussed.

As expected the results show front camera has a higher chance of detection at every altitude. Moreover its recognition remains probable with an altitude up to 75 cm whereas the bottom

camera is unreliable above 58 cm. Considering the frame rate of both cameras probabilities around 0.1 are not as bad as them seem at first glance. With a frame rate of the front camera at 30 fps a QR-code is expected to be found within a second.

Interestingly the probability of reading and recognizing is quite similar for the bottom camera, the front camera however is clearly more likely to find the QR-code than it is to read it. This difference can be attributed to the compression that takes place before the image is send to the controlling unit. The compression can reduce the effect of the motion blur gained by flight. It seems to make the corner features more distinct while distorting the data presentation, explaining the increased chance of locating and the absence of such an increase for reading.

From images recorded at an altitude above 1 meter no QR-codes where found. This suggests that the 640x480 resolution is to low to display the required features at this height.

6.1 Effect on Approach

This places the ideal altitude for locating QR-codes between 41 and 75 cm. For the main approach the choice has been made to remain at an altitude of 70 cm while trying to find a QR-code. While an altitude around 45 cm has a higher probability of recognizing a visible QR-code, 70 cm benefits of a larger field of view and more stable flight (and thus less motion blur) while still holding a great chance of recognition.

The recognition and tracking phase performed excellent in a testing environment. While the velocity of the the dynamic drop zone is not public the AR.drone was able to hit a smaller target with a slightly smaller trajectory on a wide range of velocities.

7 Conclusion

This paper proposed a three phased approach that can compete in the indoor Drop Zone Mission of IMAV 2013. It consists of a phase to recognize the QR-code indicating the location of a drop zone, a phase to keep track of the drop zone after it has been recognized and a phase that aligns the MAV with the drop zone.

Conclusively an experiment was conducted to determine the ideal altitude for recognizing the QR-code comparing the effectiveness of the two cameras on an AR.drone 2.0. The results show the higher resolution camera has a higher probability of finding a visible QR-code and the ideal altitude for recognition lies between 41 and 75 cm.

8 Future Research

The high difference in the probability of recognizing and reading for the front camera is suggested to come from compression. Programmers experienced with the ZXing library recommend the use of image compression, for the reduction in size allows for faster recognition and it can suppress the effects of blurring. However no research has been done on the effect of compression or other filters to boost the performance of ZXing. Perhaps a filter or compression method can boost the recognition on higher altitudes.

Acknowledgements

We would like to thank Gerald Poppinga for his support and guidance. Furthermore, we are grateful to Nick Dijkshoorn, Camiel Verschoor and Auke Wiggers for providing of this development framework, the ros-package ardrone_tools.

References

[1] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *Computer Vision–ECCV 2010*, pages 778–792. Springer, 2010.

[2] Nick Dijkshoorn. Simultaneous localization and mapping with the ar.drone. Master's thesis, Universiteit van Amsterdam, July 2012.

[3] Nick Dijkshoorn. Simultaneous localization and mapping with the ar.drone. Master's thesis, Universiteit van Amsterdam, July 2012.

[4] Zdenek Kalal, Jiri Matas, and Krystian Mikolajczyk. Online learning of robust object detectors during unstable tracking. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1417–1424. Ieee, 2009.

[5] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (99):1–1, 2011.

[6] Sean Owen. Zxing, October 2007.

[7] Giorgio Panin. *Model-based visual tracking: the OpenTL framework*. John Wiley & Sons, 2011.

[8] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, 2009.

[9] Srikanth Saripalli, James F Montgomery, and Gaurav S Sukhatme. Visually guided landing of an unmanned aerial vehicle. *Robotics and Automation, IEEE Transactions on*, 19(3):371–380, 2003.

[10] Constantin Scheuermann, Martin Werner, Moritz Kessel, Claudia Linnhoff-Popien, and Stephan AW Verclas. Evaluation of barcode decoding performance using zxing library. In *Second Workshop on Smart Mobile Applications*, 2012.

[11] Courtney S Sharp, Omid Shakernia, and S Shankar Sastry. A vision system for landing an unmanned aerial vehicle. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1720–1727. IEEE, 2001.

[12] Arnoud Visser. Vision-based road-following using a small autonomous aircraft. March 2013.

[13] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, University of North Carolina, Chapel Hill, NC, USA, 1995.