Landing - Extracting Angles From an Image

Submitted by: Li-tal Kupperman, Ziv Nachum

Advisors: Yair Atzmon, Aram Movsisian

Center for Intelligent System Laboratory, CS, Technion

July 2015

Abstract

This project is a part of a bigger project. The main objective of the project is to land a quadcopter on a moving target, using the image stream form the camera, mounted on the quadcopter.

The objective of this part of the project is to extract the angles of the taken image, which represents the position of the quadcopter in the world, with respect to the target in real world. This way, the quadcopter will be able to determine its next step, in order to approach the target.

Background

As mentioned above, we were trying to estimate the position and orientation of the camera, mounted on the quadcopter, with respect to the desirable target.

Calculating the estimated position, is possible using a set of N object points (3D) and their corresponding image projections (2D).

The image data from which the pose (position and orientation) of an object (=camera) is determined can be either a single image, a stereo image pair, or an image sequence where, typically, the camera is moving with a known speed.

The pose can be described by means of a rotation and translation transformation which brings the object from a reference pose to the observed pose. This rotation transformation can be represented in different ways, such as a rotation matrix, which is the information we are trying to extract.

One way of doing that is by using Analytic or geometric methods: Given that the image sensor (camera) is calibrated, the mapping from 3D points in the scene and 2D points in the image is known. If also the geometry of the object is known, it means that the projected image of the object on the camera image is a well-known function of the object's pose. Once a set of control points on the object, typically corners, edges or other feature points, has been identified, it is then possible to solve the pose transformation from a set of equations which relate the 3D coordinates of the points, with their 2D image coordinates.

Algorithms that determine the pose of a point cloud with respect to another point cloud are known as point set registration algorithms, if the correspondences between points are not already known.

Recognizing the target in the frame

In order to find the target in the image, we tried to use known algorithms and functions, one of which is the Hough transform, used to detect specific features in the images.

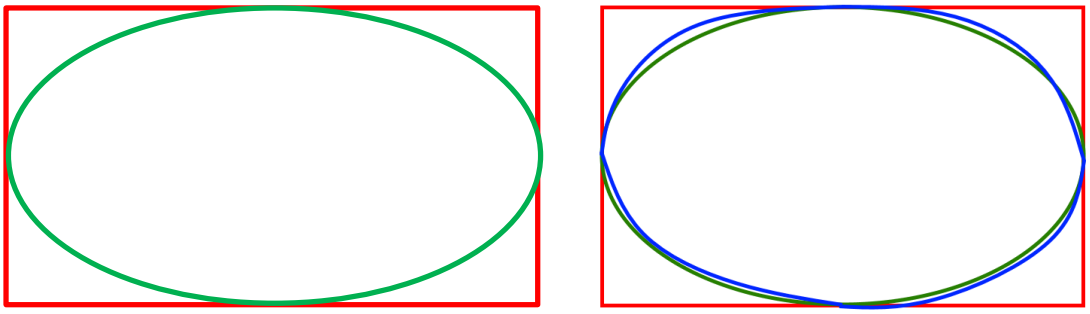
One of the implementations of the Hough transform allows finding circles in the image.

Trying to use the Hough transform, in order to detect ellipses, led to unwanted results.

This method was unable to detect the ellipses in the frame. We later learned that this method works well with fully rounded circles, but not with ellipses.

We decided to neglect this algorithm, and use our own heuristics, in order to detect the ellipses in the frame.

Notice that a rectangle corresponds to an ellipse and vice versa, by having the same width and height.



The Algorithm used:

1. Detect all the contours in the frame, using Canny edge detector.
2. For each contour:
    a. Find a minimal rotated rectangle which contains the contour.
    b. See if the area of the ellipse corresponding to the rectangle is similar enough to the area circumscribed by the contour.
        i. If not: continue to the next contour.
    c. Check if this ellipse was already found, or is very similar to another ellipse.
        i. If true: continue to the next contour.
        ii. Else: add this ellipse to the set of all the ellipses that were found.

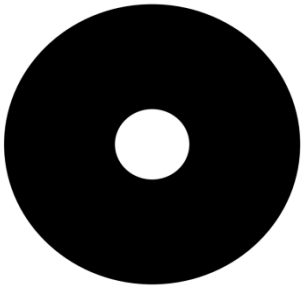
At the end of the algorithm we have a set of all the ellipses in the image.
According to the experiments we conducted, we saw that this heuristics allows us in fact to find most of the ellipses in the frame.
Now all we have to do is to identify all the relevant ellipses that form the target.

Choosing the Target

One of our secondary goals of the project was to find a suitable target, which will be clear to see and identify from the air, and will be noticeable (stick out) on the surface.

Our first trial was the original target that was planned for the Landing project: A circle within a circle.

The main problem we faced was to exclude the target from the surface. As it turns out, a lot of objects were identified as a circle with in a circle, thus causing the program to ignore the actual target, and choose another object as the target.



First Target

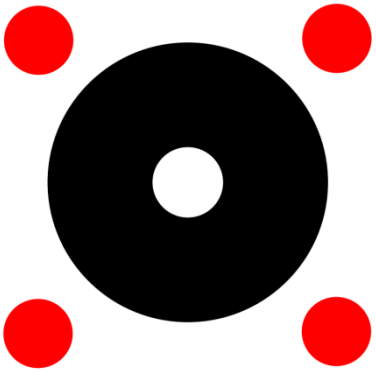
Our second trial included 4 other circles, surrounding the main target.

We tried to add more characteristics to the target, in order to exclude it from the rest of the surface.

This time, we were looking for 6 circles, forming a specific known composition of circles.

The problem we face now was related to the symmetry of the target. Our purpose was to extract the angles according to the image of the target, but due to the symmetry of the target, we could not decide which direction we were facing.

The angles we were extracting were unstable, and constantly changed, though the camera was standing still.



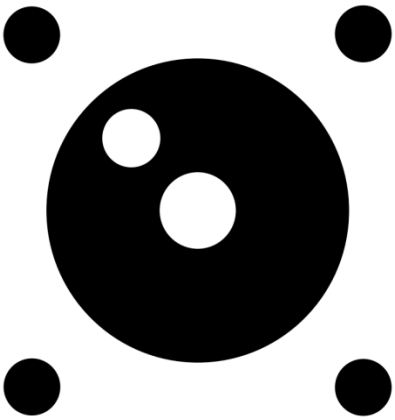
Second Target

In order to "break" the symmetry of the target, we decided to add a small circle, on the black ring.

Our purpose was to determine the direction of the target, therefore we wanted to decide which one of the peripheral circles, will be the first one to be recognized as part of the target (after revealing the two inner circles).

By adding another circle we were able to locate the same specific circle at all times, and find the right direction of the target.

This change led to stable results that matched the camera's direction in the real world.



Final Target

Extracting the Angles of the Camera

After detecting the target within the frame, we moved on to solve the main mission of the project – calculating the angles of the quadcopter\camera, with respect to the target, in the real world.

In order to do that, we used two sets of points:

1. Points in the 2D image plane, representing all the ellipses, found in the frame.
2. Points in the real 3D world, representing the real target, and its proportions.

We did that using Perspective-n-Points, a method which estimates a camera pose from given n 3D points and their projections in an image. Using the solvePnP function (openCV) , which gets these two sets of points as an input, we were able to retrieve the rotation vector and the translation vector. These two vectors represent the position of the camera, with regards to the real world.

It is important to mention, that until we reached our final target formation, the vectors receives from the solvePnP were inaccurate, and did not coincide with the real accepted results.

The last step was to extract the 3 rotation angles, in the X, Y and Z axes, from the rotation vector.

First, we used the Rodrigues function (openCV) , to convert the rotation vector into a rotation matrix, in order to simplify the calculation.

Then, we used a given calculation to calculate the 3 Euler angles needed:

$$M = \begin{pmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{pmatrix}$$

$$\theta_1 = atan2(m_{12}, m_{22})$$

$$c_2 = \sqrt{m_{00}^2 + m_{01}^2}$$

$$\theta_2 = atan2(-m_{02}, c_2)$$

$$s_1 = \sin(\theta_1) \qquad c_1 = \cos(\theta_1)$$

$$\theta_3 = atan2(s_1 m_{20} - c_1 m_{10}\ , c_1 m_{11} - s_1 m_{21})$$

After converting the Angels above, $\theta_1, \theta_2, \theta_3$, from radians to degrees, we finally received our desired angles.

As it was said before, these angles represent the relative position of the quadcopter\camera, with respect to the target.

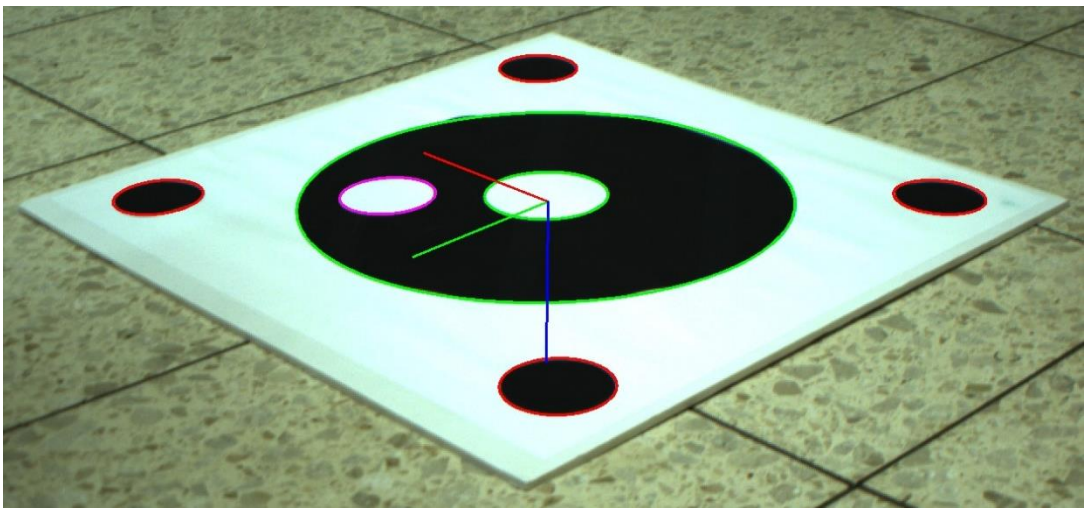
Using these angles, it is possible to calculate the next move of the quadcopter, toward the target.

Results

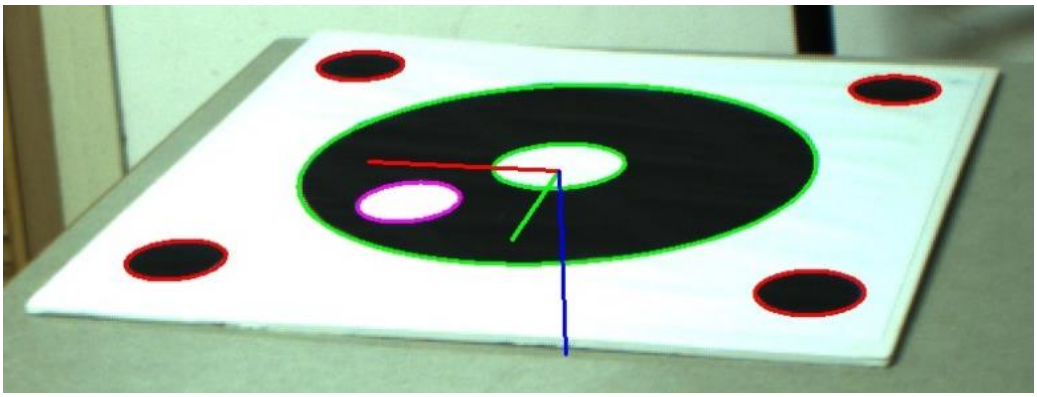
The following images display the results of our algorithm. It shows both the target recognition and the extracted angles, represented by the colorful coordinate system.

The colorful circles contour all the elements that form our chosen target. The following images indicate that the target is in fact being recognized by the algorithm, very accurately.

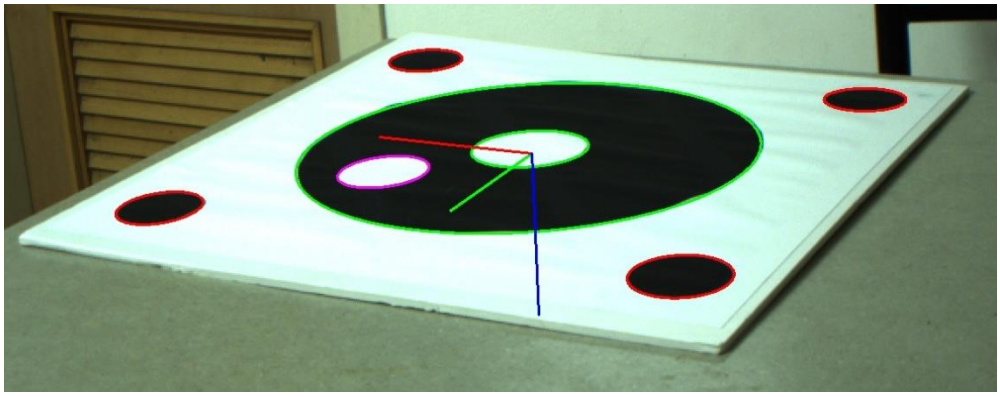
Giving this analysis for each frame the system is able to monitor the position of the camera, with respect to the target.



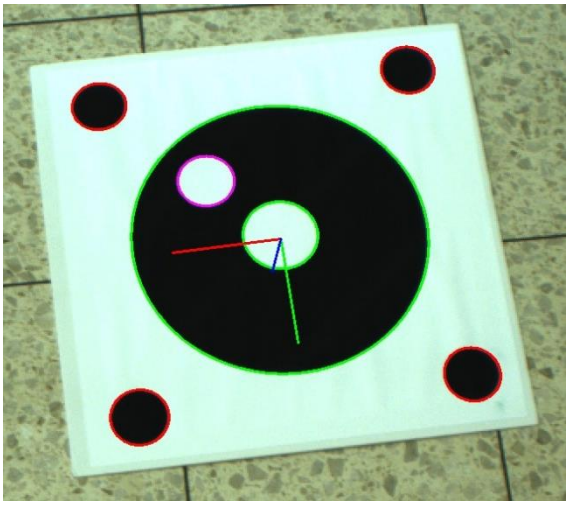
X:67.7977, Y:0.348954, Z:-134.144



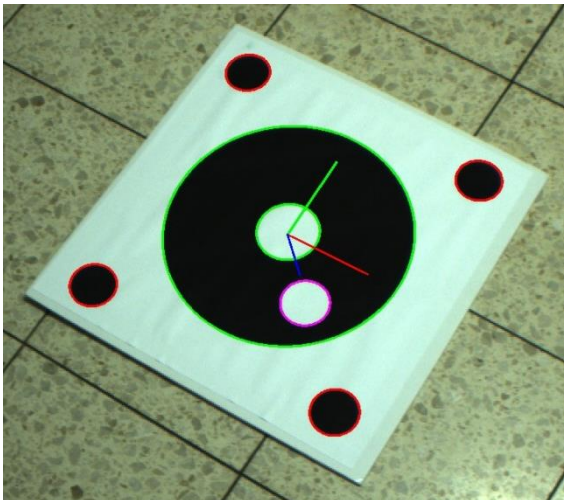
X:69.1828, Y:-1.93516, Z:-103.558



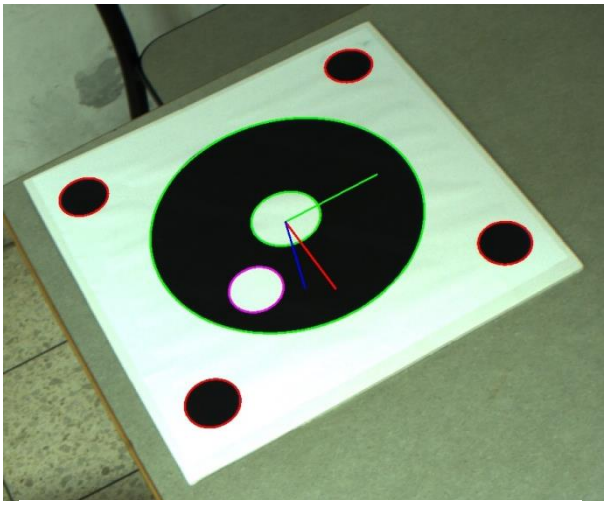
X:72.1469, Y:-2.32951, Z:-115.046



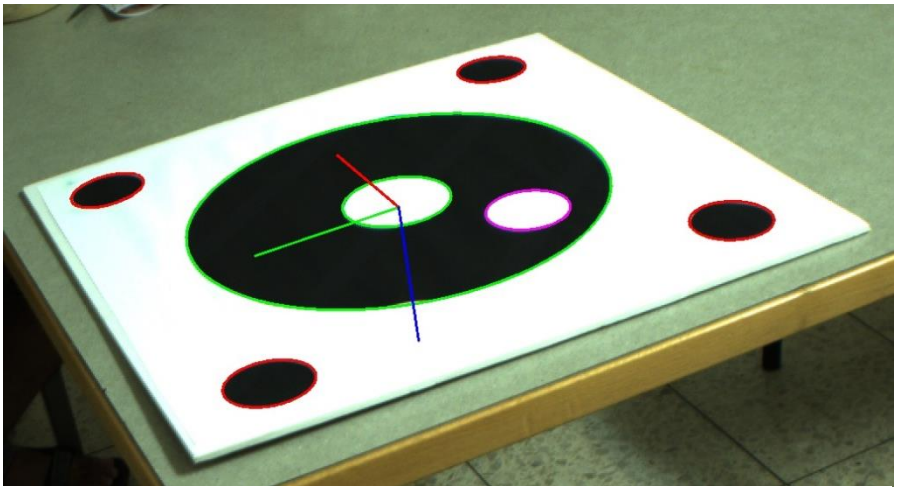
X: 17.4068, Y:6.23963, Z:-80.3897



X:25.249, Y:-8.4599, Z:57.6348



X: 40.0163, Y:-11.8851, Z:26.5399



X: 58.8856, Y:-6.60594, Z:-153.072

Suggestions for further word

1. Integrate between the algorithm results and the Quadcopter movement:
   Run the algorithm on an Odroid and use the results to form the relevant commands, needed to control the Quadcopter movement.

2. Improve the target:
   Making the target better, so it's recognition will be even more accurate, and more stable.

3. Improve the algorithm:
   Making the algorithm more efficient. Since the algorithm eventually produces results that need to be used in real-time, and control a flying Quadcopter, efficiency and agility are necessary.

Bibliography

1. https://en.wikipedia.org/wiki/Pose_(computer_vision).
2. https://en.wikipedia.org/wiki/3D_pose_estimation.
3. http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#bool solvePnP(InputArray objectPoints, InputArray imagePoints, InputArray cameraMatrix, InputArray distCoeffs, OutputArray rvec, OutputArray tvec, bool useExtrinsicGuess, int flags).
4. "Computing Euler angles from a rotation matrix", by Gregory G. Slabaugh.
5. "Extracting Euler Angles from a Rotation Matrix", by Mike Day, Insomniac Games. mday@insomniacgames.com