A comparative study of preprocessing techniques for marker based localization in UAVs

Alex Noel Joseph Raj, Akshay Chawla, Gautam Sridhar, Dheeraj Akshay

VIT University, Vellore

Abstract. Localization techniques involve estimating the robot's pose and uncertainty with respect to a world map by making use of sensors available on the robotic platform being used. Using the camera as a sensor involves preprocessing images to get range and bearing measurements of landmarks visible in the image with respect to the local frame of the camera. In this paper we explore template matching, phase correlation and contour based marker detection methods as preprocessors for landmark extraction for the localization problem. We compare performance of the three methods by embedding them in an Kalman filter framework running on a simulated quadrotor in V-REP.

Keywords: Kalman Filters, Localization, Camera, Quadcopter, Template Matching, Contour Extraction, Phase Correlation, V-REP, Simulation, Python

1 Introduction

Localization is an important task performed by mobile robots in order to autonomously navigate a given environment. A naive method for localization is dead reckoning in which an autonomous agent estimates its location solely on the basis of its internal state sensors. [1]. This is achieved by repeatedly integrating the estimates from the proprioceptive sensors such as inertial measurement units, or optical wheel encoders etc. Priori localization involves knowledge of a map of the environment beforehand and thus the robot is able to navigate it successfully by fusing the internal and external state sensors [8]. The map may be consisting of high level features such as visual markers placed in the environment that can be detected by a ground facing camera [4]. In addition to this, there also exists the concept of Simultaneous Localization and Mapping (SLAM), which requires the robot to build a map of the environment and localise itself in it concurrently. [1]. In our paper, the primary focus is priori localization.

Cameras as sensors on robots can perform multiple important tasks for the robot and are gaining much more popularity these days because of the cheap availability and high quality. Such cameras are designed to fit well with lightweight robots like quadrotors, and act like a multi-purpose sensor that can be used to detect position, velocity, and in our case to localize the agent with respect to the world environment by detection and recognition of fiducial markers placed in its path [4].

The application of the camera that we are focusing on in this paper is that of detection of markers. We use the commercially available A.R drone as our test bench as it is fitted with frontal and ground facing cameras [6]. We can model the Quadrotor and the environment in robotic simulation softwares. The simulation software used in this paper is called the Virtual Robotics experimentation platform (V-REP). [5]. Markers are placed in the environment to enable the quadrotor to localize itself when it executes a trajectory and attempts to navigate the environment. In this paper, we explore three different well known techniques used for detection of these markers which are Template matching, Phase Correlation and Contour based detection techniques.

2 Problem Formulation

The primary purpose of this project is to analyse and compare the working of three different marker detection algorithms for the localization problem. The comparison is made using the following performance metrics - 1. Height 2. Min error 3. Computational time requirements This section describes the localization algorithm used for the calibration of the camera on the drone model, and the three techniques that are compared: template matching, phase correlation and contour based matching.

2.1 Localization

The quadrotor performs localization using the Kalman filter. The Kalman filter is essentially an optimal linear estimator of any state given inaccurate or noisy observation. In the case of this technique, the observations are assumed to be affected by Gaussian noise, and the Kalman filter then minimises the mean square error of the estimated parameter.The Kalman filter is thoroughly described by Bishop et al. [12].

The terminology for using the Kalman filter in a general discrete time case is listed below-

1. x represents the state of the process that has to be estimated.
2. u is the control vector
3. z is the measurement of that state
4. w and v represent the process and measurement noise respectively with normal probability distributions. The covariance matrices of these two might change with each time step
5. A is a matrix that relates the state in the previous time step to the one in the current time step
6. B B relates the current control input to the next state.
7. H H is a transformation that transforms the current state to expected sensor value. It is derived from the sensor model.
8. Q,R,P are the covariance matrices of the process noise, measurement noise and the error respectively.

Given the above, we can write the following equations to describe the state and the measurement at a given state-

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \tag{1}$$

$$z_k = Hx_k + v_k \tag{2}$$

The Kalman filter now performs two steps: Time update and Measurement update. The time update is responsible for predicting the next state using proprioceptive sensors and the motion model of the autonomous agent. The measurement update uses the global measurement from extero-ceptive sensors to correct the estimate from the previous step. The equations are presented below.

For the time update step-

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \tag{3}$$

$$\hat{P}_k^- = AP_{k-1}A^T + Q \tag{4}$$

And for the measurement update-

$$K_k = \hat{P}_k^- H^T \left(H\hat{P}_k^- H^T + R\right)^{-1} \tag{5}$$

$$\hat{x}_k = \hat{x}_k^- + K_k\left(z_k - H\hat{x}_k^-\right) \tag{6}$$

$$P_k = \left(I - K_k H\right)\hat{P}_k^- \tag{7}$$

Here k represents the current time step, K_k is the Kalman gain. The first step in the measurement update is to calculate this gain value from which we calculate *aposteriori* values for x and P using the measures z. Finally we use these values in the time update step to find *apriori* values of the state and the error covariance in the next time step.

2.2 Calibration

The aim of the calibration step is the get an accurate mapping between the quadrotor height and the marker side length in pixels. The reason for this is so that the training images, i.e the dataset of all marker images can be resized before being compared to the extracted marker. This is important because depending on the height of the quadrotor from the floor, the size of the marker in the camera becomes different and thus comparing them with the template would require the template to be the same size as the marker on the extracted image, because the techniques used for matching are not scale invariant.

2.3 Template Matching

Template matching is the first technique that we apply to detect markers in the environment. The essence behind template matching is to find areas in a test image which are similar to a particular template that is known beforehand. Thus in template matching, we have one source image, and one template image which is moved one pixel at a time over the source image and at each point, a patch the same size as the template is compared to it. With this comparison we can accurately judge at which position in the source image, the template has been found. The metric that we use in this paper to judge a match is the normalized cross correlation(NCC) as described below-

$$R(x,y) = \frac{\sum_{x',y'} T(x',y') I(x+x',y+y')}{\sqrt{\sum_{x',y'} T(x',y')^2 \sum_{x',y'} I(x+x',y+y')^2}} \tag{8}$$

Here x',y' represent the pixels in the image $I(x,y)$ is the source image and $T(x,y)$ is the template image. The template image is smaller than the source image.

$R(x,y)$ is actually a matrix of values corresponding to the cross correlation, and its highest value will be achieved only when the template is properly aligned with the source image. Naturally, a template which is not present in the source will also give a peak at a location, hence, we choose an appropriate threshold on the NCC value in order to filter out false positives.

2.4 Phase Correlation

Phase correlation is the other technique we use in the comparison procedure. Phase correlation is a Fourier domain technique that is traditionally used for image registration applications. It involves calculating the Fast Fourier transform(FFT) between the two images that are being matched.

Phase correlation is based on the Fourier shift property which is encapsulated in the following mathematical formulation proposed by Kuglin and Hines in [7]-

Consider $f_1(x,y)$ to be the original image and $f_2(x,y)$ to be the image shifted by (x_0, y_0))

$$f_2(x,y) = f_1(x - x_0, y - y_0) \tag{9}$$

Then the corresponding Fourier relationship is given as

$$F_2(u,v) = \exp(-j2\pi(ux_0 + vy_0)) F_1(u,v) \tag{10}$$

Where F_1 and F_2 are the Fourier transforms of f_1 and f_2 respectively and the exponential term denotes the phase shift between the two. Given the above, we can define the Normalised Cross power spectrum as

$$\frac{(F_2(u,v) * F_1^*(u,v))}{|\ F_2(u,v) * F_1(u,v)\ |} = \exp(-j2\pi(ux_0 + vy_0)) \tag{11}$$

Where $F^*$is the complex conjugate of F.

The idea exploited here is that the phase difference between the two images represents the normalised cross correlation between images, if they purely differ through translation and this is the idea used in this paper during the detection of markers using this technique.

2.5 Contour Matching

Detection of markers can also be based on finding out the contours that exist in the camera image taken by the quadrotor and then matching that to an existing training set of markers. In this section the contour matching method that we have used is described in detail. It is somewhat similar to the technique used for detection of ArUco markers [3].

The following steps are followed in executing the contour matching approach-

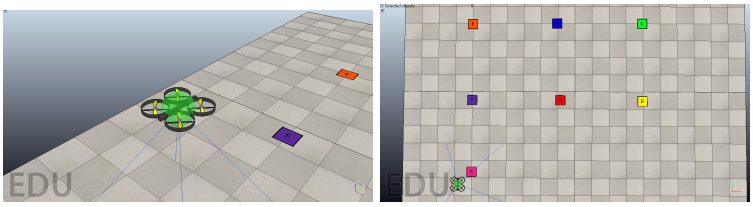
1. Convert the 24 bit RGB image to 8 bit grayscale.
2. Apply the bilateral filter. The bilateral filter is used to preserve major edges in an image while smoothing the rest of the image [11]. This is an important step because we need to remove the edges yielded by the floor, which otherwise lead to false positives and unnecessary computations.
3. Next we apply the canny edge detection technique to get the edges [2].
4. Find Contours from the binary image from the above step using Suzuki's method [10].
5. Then we sort the contours by area, and pick the largest contour, because our base assumption is that the largest contour is a candidate for being a marker.
6. We now have to approximate the contour using the Ramer-Douglas-Puecker Algorithm [9]. This algorithm essentially reduces the number of points on the contour.
7. If the approximated contour has 4 points (square or rectangular candidate, extract the square marker from the image.
8. Next we use the nearest neighbour technique based on the mean squared error criterion to find the best match of the extracted image patch to a resized training image If the mean square error is lesser than a given a threshold then we say that the marker has been detected.

3 Simulation Setup

The simulation for the above given methodology was done on V-REP. V-REP is a simulation software developed by Coppelia Robotics written in the programming language Lua. It supports many families of robots for simulation including aerial robots like quadrotors and various sensors like the RGB cameras. Another important feature of the software is the remote API for python which helped us control the quadrotor. This allows us to transfer and receive information from the

quadrotor model in V-REP. The information from the quadrotor was extracted in the form of odometry readings from the quadrotor. Gaussian noise was added to the odometry readings to ensure that they resembled real life quadrotor odometry readings.

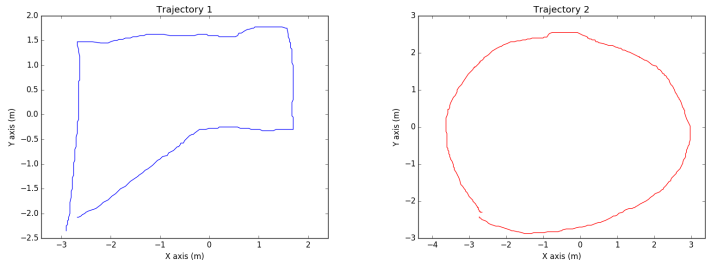
Figure 1 shows the simulation environment which consists of the quadrotor with the bottom facing camera attached to it, and the the positions of the markers. Coloured markers were chosen to ensure better matches for the template matching and phase correlation techniques as these were performed with RGB images.



(a) Quadrotor in simulation (b) Simulation environment

Fig. 1: Simulation Setup

The Quadrotor was flown multiple times on two different trajectories [Fig 2]. A single run of the trajectory might not always represent the performance of the algorithm, hence multiple runs were performed for averaged and consistent results.



(a) Trajectory 1 (b) Trajectory 2

Fig. 2: Trajectories for testing

4 Results

In this section we detail the experimental set-up and the procedure. This is followed by a comprehensive list of results. Separate experiments are performed

on each of the three different marker detection methods. The markers were stored in a database from which they could be easily accessed.

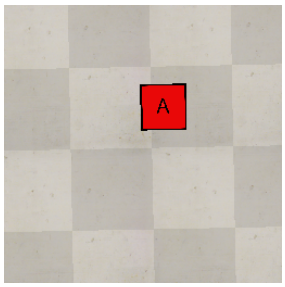
4.1 Methodology

We run the quadrotor over the given trajectories described in Section 3. During the run over a trajectory, the quadrotor is constantly running the program for localisation and marker detection. The markers essentially help in the measurement update described in section 2.1 as the quadrotor has the knowledge of the marker location with respect to the world reference frame and hence can adjust its estimate from an observation. We monitor and record the mean squared error(MSE) between the actual trajectory and the predicted trajectory. Additionally, we record the processing time required for the quadrotor to localize and predict its position accurately, given that, the marker detection program is running.

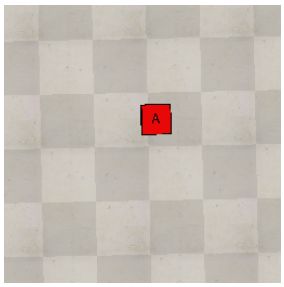
4.2 Simulation Results

Before the markers in the database are compared to the captured image, we need to resize the marker image as described in section 2 to ensure that both template and marker in captured image are of the same size.For this we perform the following steps:

1. Collect training examples of images from down facing camera and corresponding height. Some training examples are displayed in Figure 3.



(a) camera image at height=1.49m



(b) camera image at height=2.24m

Fig. 3: Training images

2. Fit a curve to Height of drone (m) v/s Marker side length (pixels) data points using regression. The polynomials is of the 4th degree. The curve that we fit for our experiment is displayed in Figure 4-

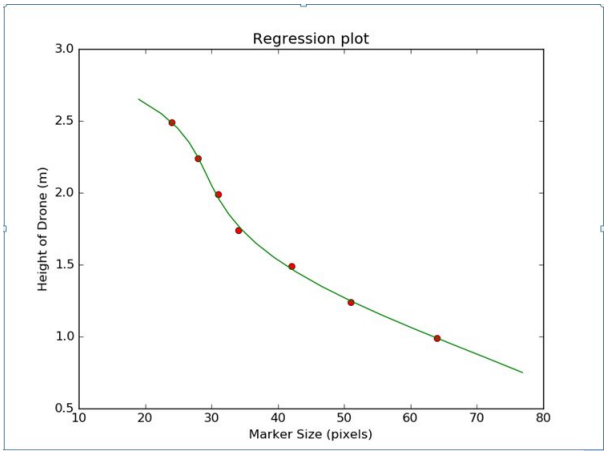


Fig. 4: Regression Plot for Height of drone(m) v/s Markers Size(pixels)

3. Use the polynomial to query the marker size (in pixels) given the height of the drone.

The first parameter we collect is computation time, which was collected as follows-

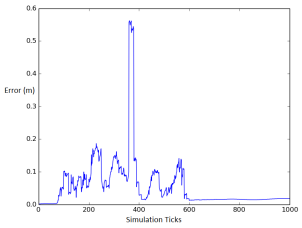
1. We took 20 images from down facing camera having markers in them.
2. For each algorithm for marker detection, we note the time required to calculate the marker position in the image.
3. Finally, we average this result over 20 images and the results are displayed in Fig.5.

Finally we receive the following table-

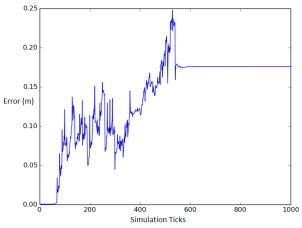
Contour Matching	Template Matching	Phase Correlation
0.02030	0.06194	0.05650

Fig. 5: Table for computation times

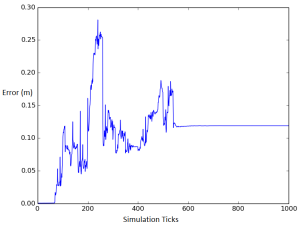
The second parameter we collect is the mean squared error between the actual trajectory from the simulator and the predicted trajectory from the localisation algorithm. We perform the localisation task over both trajectories for a total of 10 times and then averaged the predicted value. This average error was plotted simulation tick counts, where each tick count represents a one time data dump from the simulator. The graphs for the same are displayed in Figure 6,7-



(a) Contour matching MSE

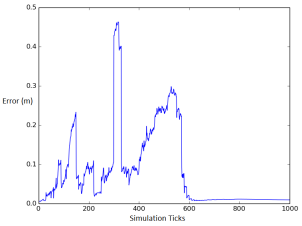


(b) Phase Correlation MSE

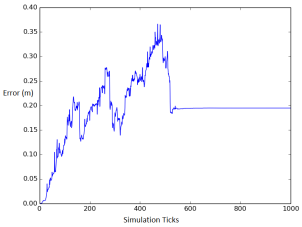


(c) Template Matching MSE

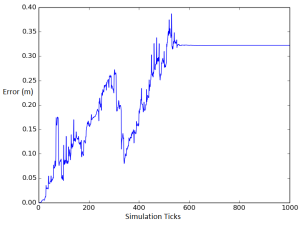
Fig. 6: Trajectory 1 MSE



(a) Contour matching MSE



(b) Phase Correlation MSE



(c) Template Matching MSE

Fig. 7: Trajectory 2 MSE

From the graphs we see that the contour matching technique yields sudden spikes in errors, which means that it detects false positives. Phase and template matching perform admirably and have consistently low errors. The initial error fluctuation is due to initial misalignments and errors due to odometry. The linear nature of Trajectory 1 means the quadcopter is less likely to encounter partial markers in the image, reducing the false positives in comparison to Trajectory 2, whjich has a circular nature. This means less error spikes in Trajectory 1.

5 Conclusions

This paper compares three methods used in quadrotor localisation and the error produced on different trajectories. The template matching based on normalized cross correlation in general out-performed the other two techniques and can be successfully used for localisation applications. The future scope of the this paper can involve adding and comparing more methods for marker detection and developing a Fuzzy Control system for which technique to use at which height.

References

1. Bailey, T.: Mobile robot localisation and mapping in extensive outdoor environments. Ph.D. thesis, Citeseer (2002)
2. Canny, J.: A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence (6), 679–698 (1986)
3. Garrido-Jurado, S., noz Salinas, R.M., Madrid-Cuevas, F., Marín-Jiménez, M.: Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognition 47(6), 2280 – 2292 (2014), http://www.sciencedirect.com/science/article/pii/S0031320314000235
4. Huang, A.S., Bachrach, A., Henry, P., Krainin, M., Maturana, D., Fox, D., Roy, N.: Visual odometry and mapping for autonomous flight using an rgb-d camera. In: International Symposium on Robotics Research (ISRR). vol. 2 (2011)
5. Ivaldi, S., Padois, V., Nori, F.: Tools for dynamics simulation of robots: a survey based on user feedback. arXiv preprint arXiv:1402.7050 (2014)
6. Krajník, T., Vonásek, V., Fišer, D., Faigl, J.: Ar-drone as a platform for robotic research and education. In: International Conference on Research and Education in Robotics. pp. 172–186. Springer (2011)
7. Kuglin, C.: The phase correlation image alignment method. In: Proc. Int. Conf. on Cybernetics and Society, 1975. pp. 163–165 (1975)
8. Leonard, J.J., Durrant-Whyte, H.F.: Mobile robot localization by tracking geometric beacons. IEEE Transactions on robotics and Automation 7(3), 376–382 (1991)
9. Ramer, U.: An iterative procedure for the polygonal approximation of plane curves. Computer graphics and image processing 1(3), 244–256 (1972)
10. Suzuki, S., et al.: Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing 30(1), 32–46 (1985)
11. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: Computer Vision, 1998. Sixth International Conference on. pp. 839–846. IEEE (1998)
12. Welch, G., Bishop, G.: An introduction to the kalman filter. Proceedings of the Siggraph Course, Los Angeles (2001)