Automated Landing of UAV

Target Localization and Motion Analysis for Landing

Department of Computer Science and Engineering
Indian Institute of Technology, Jodhpur

Presented by:
Qazi Sajid Azam (B16CS026)
Rahul Jindal (B16CS027)

Mentor:
Dr. Chiranjoy Chattopadhyay

Introduction

Abstract

We have designed a system that recognizes the given target pattern in frames received from the camera. Using successive frames from the camera, we are able to calculate the position, velocity and acceleration of the target with respect to the camera. This information can then be used to land the UAV.

Work Done

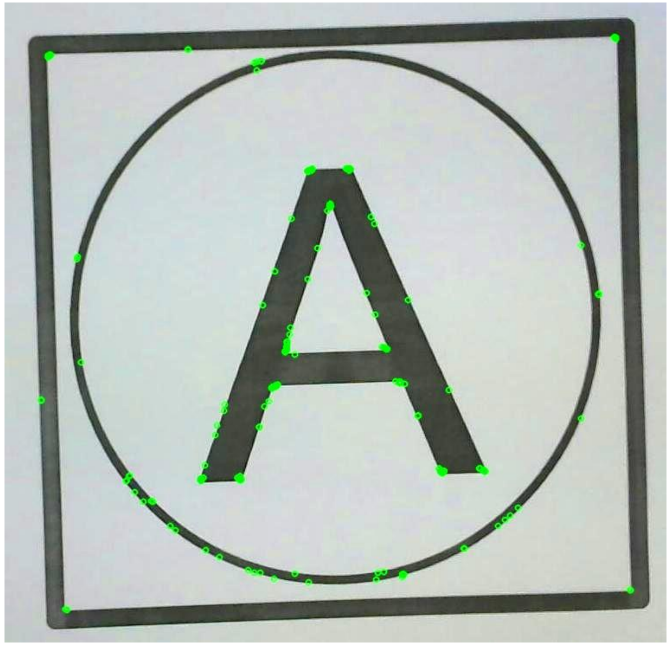
Naive Approach: Template Matching

- Given a photo of the target, we need to detect it in the camera footage.
- A window slides over the image and calculates the area with maximum correlation.
- This technique is not scale-rotation-skew invariant.
- A better method is to find the features in the image and use those for matching.

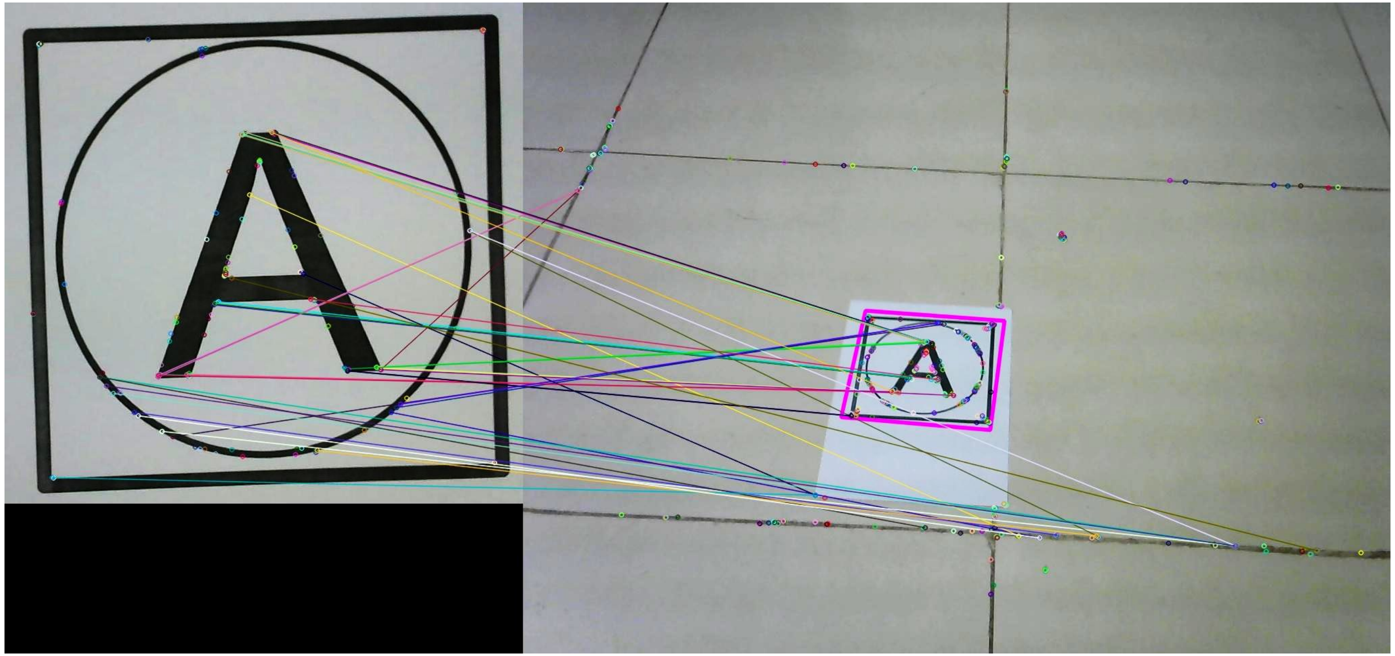
Better Approach: Feature Matching

- OpenCV has included libraries to find the features.
- We have used the ORB feature detector.
- We will find the features and compute their descriptors.
- Matching is done using OpenCV bruteforce hamming descriptor matcher.
- We take best 25% of the matches and consider them as good matches.
- Find the homography of the matched points to draw the bounding box.

Detected Keypoints



Results of Feature Matching



Finding Relative and Required Motion

- As all calculations are with relative motion, it will also work with moving platform.
- Using the data from consecutive frames, velocity and acceleration can be found.
- Using formula derived from equations of motion, we will align the target with camera.
- When the required relative acceleration is zero, we can descend.

Improvement of Accuracy

We use two techniques to improve the accuracy of the system:

1. Validation of bounding box
   1.1. Convexity and Complexity check
   1.2. Normalized standard deviation of side lengths
2. Buffering of successful matches

Showing the Live Stream

- Significant challenge to run this system in real time.
- Using several optimizations in the code, we have been able to achieve an average output frame rate of around 8 FPS.
- This will improve when we have a dedicated processor, and when we do not have to show the output images simultaneously.
- We will also print the values on the terminal window, for more clarity.

Calculation of Accuracy

Following events are successes:

- Valid match
- Invalid match but buffer available
- Not detected but buffer available

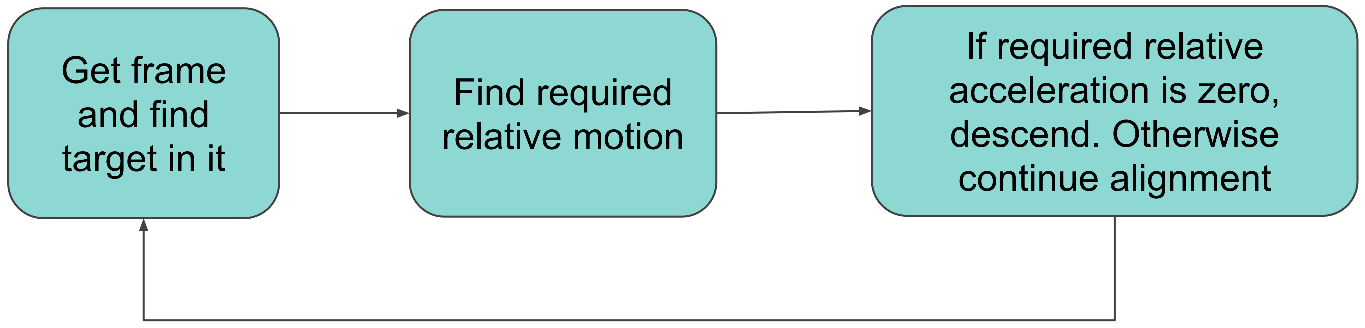
Following events are failures:

- Invalid match and empty buffer

Following events are not accounted for:

- Not detected and buffer empty

Methodology


Get frame and find target in it
Find required relative motion
If required relative acceleration is zero, descend. Otherwise continue alignment

Assumptions

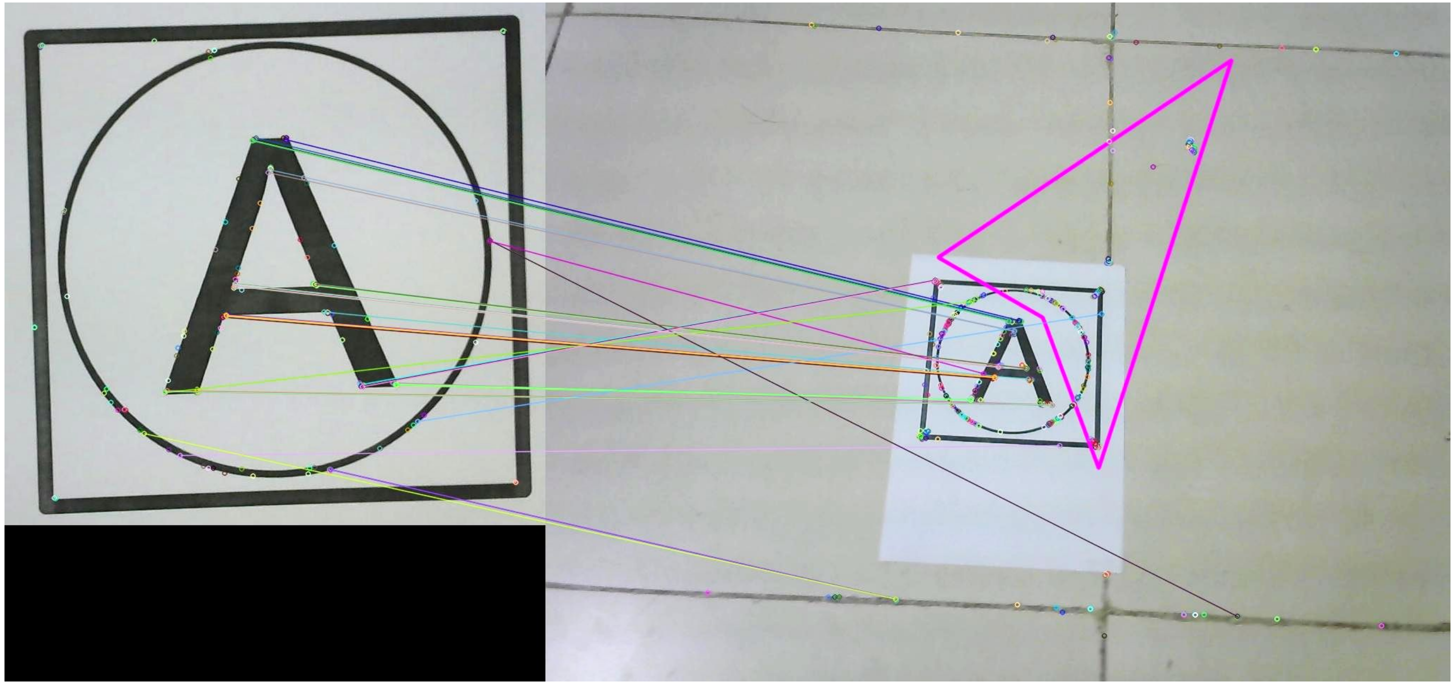
- Image from the camera is clear and not blurry.
- The target is visible to the camera.
- The target is occupying at least a fifth of the image.
- The background of the image (regions except the target) is uncluttered.

Challenges Faced

Challenges Faced

- Limitations of template matching with OpenCV
- Lack of dataset for Deep Learning
- Lack of actual equipment for testing of the system
- Blurry images from camera
- Unstable matching caused by shaking of camera
- Flickering of matched area
- Intermediate incorrect results

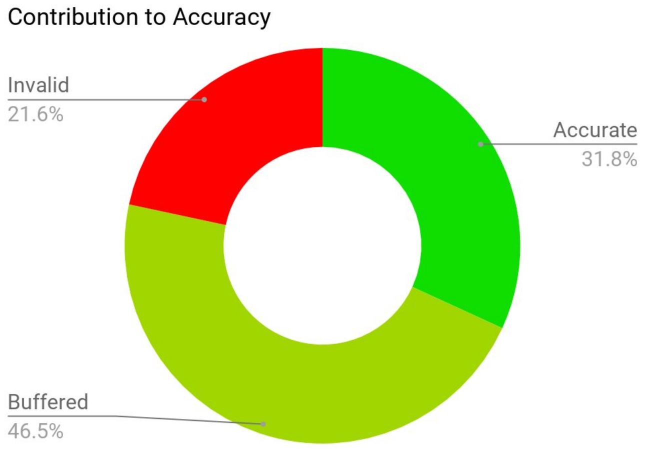
Incorrect Results of Feature Matching



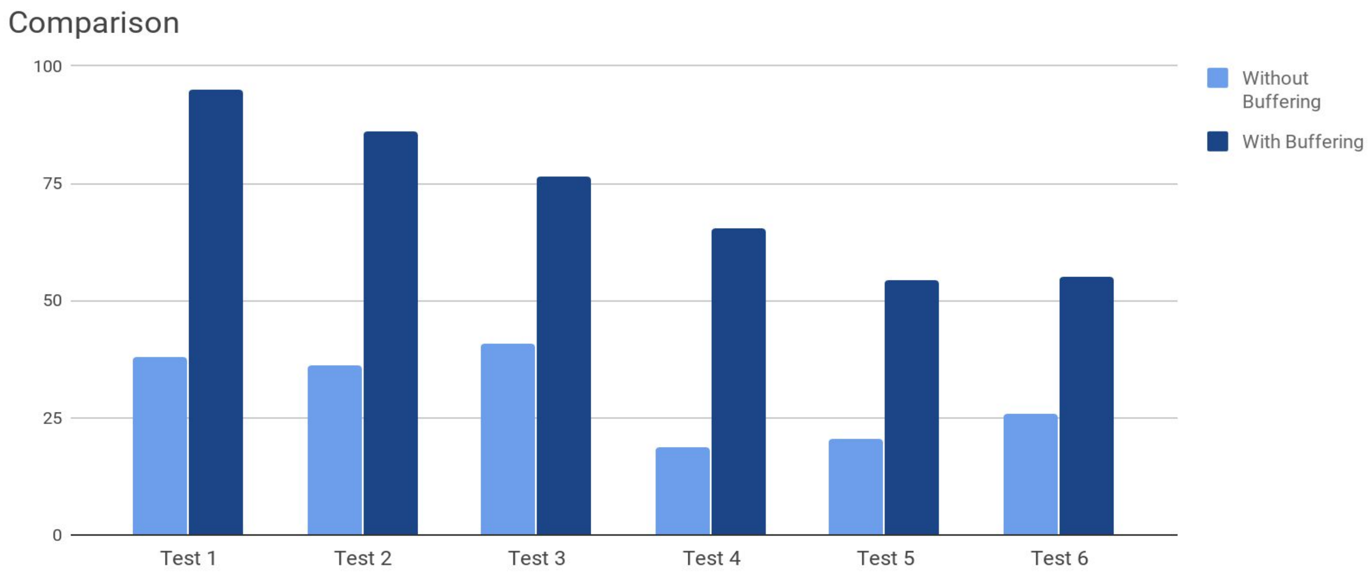
Results

Sample Output of Some Tests

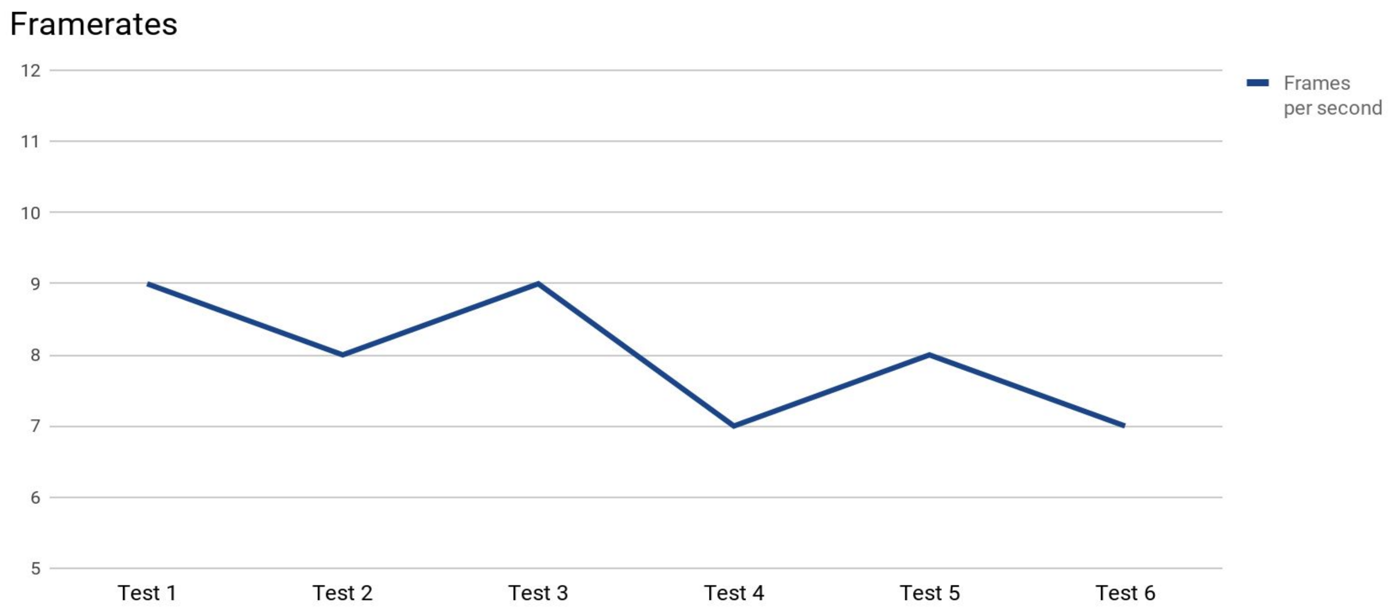
S.No	Number of Frames	Percentage Accuracy
1	869	96
2	529	88
3	1147	78
4	403	67
5	538	55
6	428	58



Benefits of Buffering



Sample Output Framerates



Live Demonstration

- Our system works fairly well under some known assumptions.
- It is nearly the best result we can achieve using these methods.
- Implementation was not done due to lack of equipment.
- We will now show the live demonstration of target localization using a handheld camera and a printed target.

77.78%

From the testing we have done, this is our success rate. It is the weighted mean of all accuracies in different tests, weighted by number of frames. We have done over 15 tests with total number of frames over 8000 to get these results. More tests were done but were not used in calculation of accuracy.

Future Work

Further Improvements

- Deep learning can be used to significantly improve the accuracy and performance.
- A more efficient method for landing would be to descend and align simultaneously.
- More accurate distance calculation with better camera.
- Real-time trajectory planning while avoiding obstacles.
- Implementation using actual equipment

Potential Applications

- Using automated systems to land UAVs will make them more convenient and more usable for the general public.
- The system can also be applied to any homing scenario, where the vehicle is chasing a target.

Thank You