Aerial Robotics Kharagpur Software Tasks Documentation

Archit Rungta *

***Abstract*— This documentation specifies the approach and results for ARL Software Team tasks 1 and 2.**

For task 1, I use RRT[1] whereas for the second task, a combination of SIFT[2] and homography projection[3] have been used to rectify images before creating a disparity map.

I. INTRODUCTION

A. TASK 1

The first task involves guiding a particle through a map with dynamic moving obstacles. The point particle starts from a specific region in the map and has to end up in a specific region while making sure that it never crosses obstacles. Obstacles are both static and moving. The maximum speed of this particle is restricted to 6 pixels per frame.

The first task involves guiding a particle through a map with dynamic moving obstacles. I use RRT to find a path from the start point to the end point initially. Once this path is found, the particle moves along this path. After each frame, the path is checked to make sure no obstacles come in this path. If the path is found to be obstructed, RRT is used again to find a new path. The initial difficulty was around figuring out a way to check for collisions quickly. After this was resolved, the problem was rather simple.

B. TASK 2

In the second task we have to create a disparity map using two pictures taken from a stereo camera. The pictures suffer from barrel distortion because of fish-eye lens in addition to not being rectified. This task is to be done without using any measurement of objects present in the picture and without calibration matrices. At first I tried to do this without any kind of modifications to the stereo images using pattern matching of OpenCV. This turned out to give a practically useless output and being extremely slow. The final version involves matching of SIFT features, homography based perspective correction and FFT 2D-convolutions[5] to generate the disparity map.

II. PROBLEM STATEMENT

This should cover one full column of page.

A. TASK 1

The first task involves guiding a particle through a map with dynamic moving obstacles. The point particle starts from a specific region in the map and has to end up in a specific region while making sure that it never crosses obstacles. Obstacles are both static and moving. The maximum speed of this particle is restricted to 6 pixels per frame.

*Ashwary Anand, Shubhika Garg

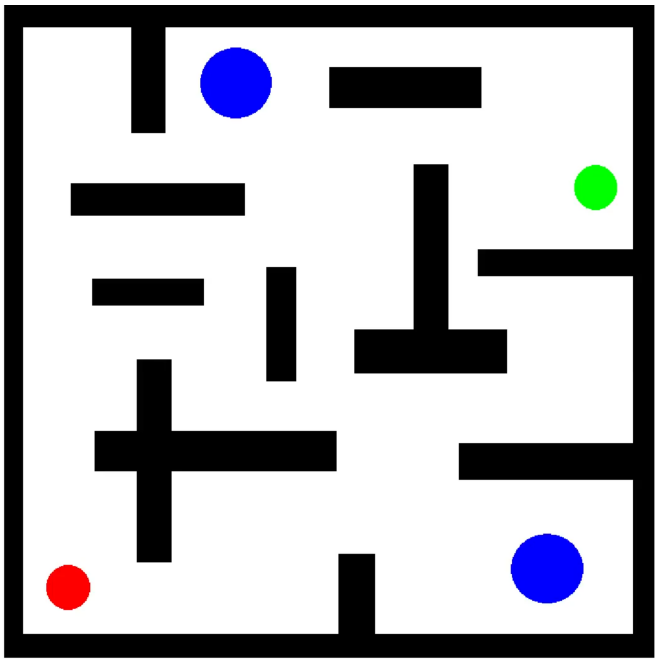

Fig. 1. Initial map

The black regions of the map are static obstacles whereas blue regions are the dynamic obstacles. The starting region is color coded by red and the stop region is color coded by green. The following image is of the map at a particular frame.

B. TASK 2

In the second task we have to create a disparity map using two pictures taken from a stereo camera. Disparity map is a picture which relates the amount of apparent pixel shift of the same point in 3D space in the two stereo images. The disparity map is indicative of depth from the cameras but it is not exactly proportional. The problem statement only involved creation of disparity map. The following two images are the images we had to use to calculate the disparity map.

III. RELATED WORK

A. TASK 1

RRT* is often used to solve the static version of this problem instead of RRT. RRT* is however slower than RRT and much more complicated to code. As such, I decided to implement RRT instead of RRT*.

Further algorithms like RRT* dynamic[4] are more robust and faster. However, the small size of the map and the

Fig. 2. Left image

Fig. 3. Right image

relative ease of obstacles meant that these algorithms were not required.

B. TASK 2

The first part of the problem - rectifying image without access to calibration - does not seem to have many other approaches. Surprisingly, the amount of literature on this topic is very limited and whatever I could find was beyond my understanding of the topic.

The second part has two distinct solution categories[6], global minimisation and local. First up, a cost function is defined which quantifies the likelihood that a certain pixel in one image has been mapped to a particular image in the second pixel. Local optimisation methods assume the pixel with the lowest associated cost to be the mapped pixel without considering anything else. Global optimisation methods on the other hand take into account facts such as objects with uniform texture will have approximately the same disparity. Also that stray pixels that have been mapped to very far pixels are probably errors. As such global optimisation methods tend to perform better with less stray pixels and uniform distances for continuous objects.

However, global optimisation methods are far more difficult to program and I decided to use a local optimisation method.

IV. INITIAL ATTEMPTS

A. TASK 1

The initial attempt and the final result are similar for the major part. I used RRT to find the path from start to end and follow this path till it gets obstructed. If it gets obstructed before reaching the goal, we find a new path from current position to end point and repeat. Initially collision detection was done by replacing the four colors in each frame using k-means with k=4 and then checking the color. The repeated call to k-means algorithm after every frame, however, made the code very slow.

B. TASK 2

Initially for the second task, I applied template matching on every 5x5 block of left picture and find the corresponding match in right picture. Then the displacement of center pixels was the value of the disparity map at this location. However, this approach gave a very rough output and was extremely slow because of template matching high time complexity and as the template matching was searching in the entire right picture. The output was rough because the pictures suffered from distortion, were not rectified to meet the constraint that shifts were only horizontal and had a lot of uniform texture in many of its regions.

V. FINAL APPROACH

A. TASK 1

At the first frame of the map, a path is generated using RRT algorithm from the start position to the end position. The particle through subsequent frames follows this path at the defined constant speed. After each frame, the path from current position to the end region is checked to make sure there are no collisions in this path. The particle keeps moving till either the particle reaches the end zone or the path is detected as being obstructed. Once the path is obstructed, a new tree is created discarding the previous work and a new path is found from the current location to the target region. After this path is found, the process continues as before.

The final attempt was almost identical to the initial attempt. Instead of using k-means to reduce the number of colors, I split the colors into constituent RGB components and then checked for ranges of values. This made collision checks much faster.

There are three major user definable constants in the algorithm. The first one controls the speed of the particle in pixels per frame, the second parameter refers to the probability that the random point chosen during RRT tree construction will be the center of the end region and the third point is the maximum straight line distance we can travel during the addition of a new node to the tree.

My solution assumes that there always exists a path from the current point to end point and that whenever a frame changes the particle does not land up in an obstacle.

As long as these two conditions are applicable and given time to reach the goal is infinite the algorithm will be able to guide the particle to the final objective region. Refer to

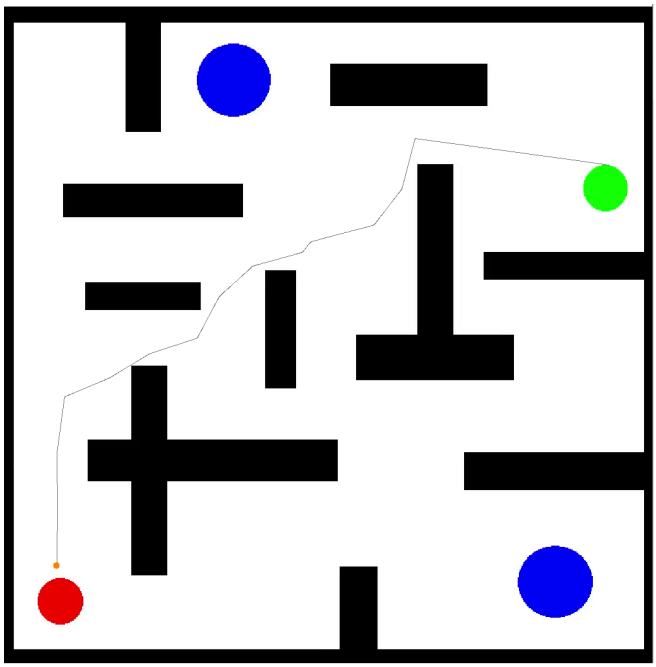

Fig. 4. Path taken by particle

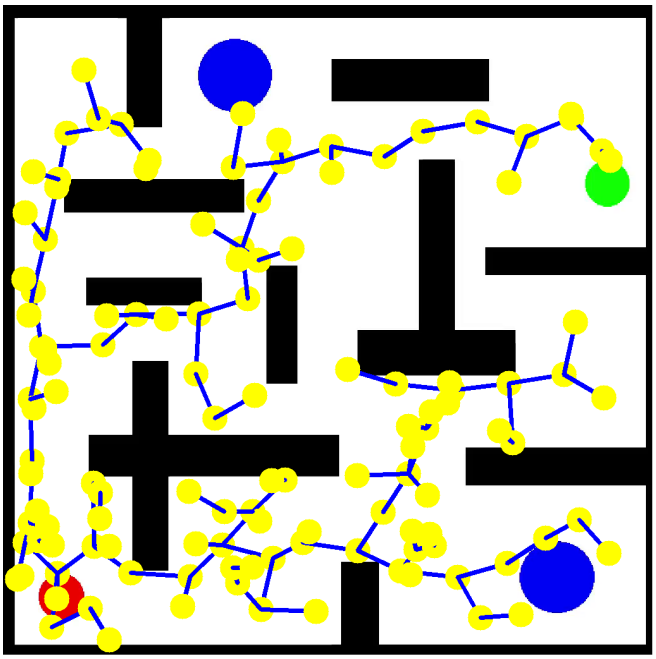

Fig. 5. Tree generated by RRT algorithm

figures 4 and 5 that show the generated tree and the path followed by the particle.

Algorithm 1 Collision Check algorithm
Input: R, G, B, x, y
Output: boolean
Code :
1: **if** $(R[x,y] > 240 and G[x,y] > 240 and B[x,y] > 240)$ **then**
2: return True
3: **end if**
4: return False

B. TASK 2

For task 2, the final solution has two distinct parts. The first part is rectification of the given images and the second part is generation of the disparity map from these rectified images.

For rectification of the images, in the first step we wish to remove the fisheye distortion. This is done in multiple steps. First, I use the SIFT algorithm to find features in both the left camera image and the right camera image. Then a brute-force matcher is used to match the same features from both of these images. Figure 6 shows 20 matched features. After this, we take the top 100 matched features and then use OpenCV's uncallibrated stereo features to find the fundamental and rectification matrices. After this, the images are rectified using their particular matrices and we get mostly fisheye distortion free images. However, certain lines that we know to be parallel to the axes are now very inclined and we have to fix that as-well.

In the second step, we use the fact that edges of the center wall should be perfectly horizontal and vertical to create a homography matrix on the 4 point method. This done individually for both of the images and then the images are remapped based on this homography matrix. Figure 7 and figure 8 show the final resultant pictures on which we create the stereo disparity map.

To create the disparity map I took the local minimisation approach. Initially I decided to use template matching to find the corresponding shift of every nxn block and use this shift as the value of the disparity. However, this approach turned out to be very slow. In the final version, I obtain a template for every pixel by taking the neighboring 4 pixels on each side. Next, this template is convoluted over a pre-defined neighboring search region and the resultant values for all center is obtained. Finally, we subtract the result of the template being convoluted with itself from the result of search region of convolution. Then, the pixel with value closest to zero is taken as our match. This is done repeatedly over all pixels in the image to obtain the final disparity map. Equations below show this mapping.

Let
x and y be the location of a certain pixel
S_x and S_y be search window dimension in pixels
T_x and T_y be template window dimension in pixels
T be template and S be the image

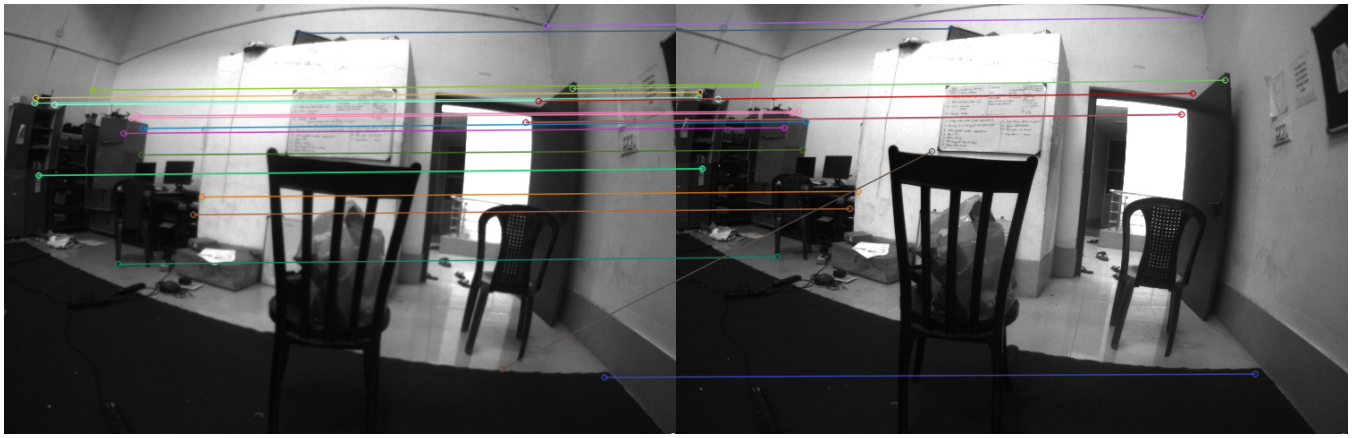

Fig. 6. SIFT features matched across images

Fig. 7. Transformed left camera image

Fig. 8. Transformed right camera image

Algorithm 2 Template matching algorithm

Output: shift

Initialisation :

min = $+inf$

target = $T \cdot T$

shift = $+inf$ *LOOP Process*

for $i = x - S_x$ to $x + S_x$ and $j = y - S_y$ to $y + S_y$ **do**

cost = abs(S[$i - S_x : i + S_x$,$j - S_y : j + S_y$] $\cdot T$ - target)

if ($cost < min$) **then**

min = cost

shift = abs($i - x$) + abs($j - y$)

end if

end for

Convolutions are computed using Scipy's FFTConvolve function for faster computation.

VI. Results and Observation

A. TASK 1

The algorithm works as expected, is general purpose, fast and very easily extendable. However, it does not generate the optimal path and has the drawbacks as mentioned in the previous section.

B. TASK 2

The chair at the center is detected to have the maximum disparity followed by the floor, the second chair and then the walls which seem to correspond with the geometry of the picture. Figures 9 shows the disparity map created by the OpenCV's inbuilt algorithm with tuned parameters and figure 10 shows the disparity map from custom algorithm. Figure 11 shows figure 10 after thresholding and repeated dilation followed by erosion to smoothen the output.

VII. Future Work

A. TASK 1

RRT is a very basic algorithm. It should be replaced by RRT* so that more efficient paths can be found. Also at the time of route obstruction, the current algorithm discards the

Fig. 9. OpenCV disparity map

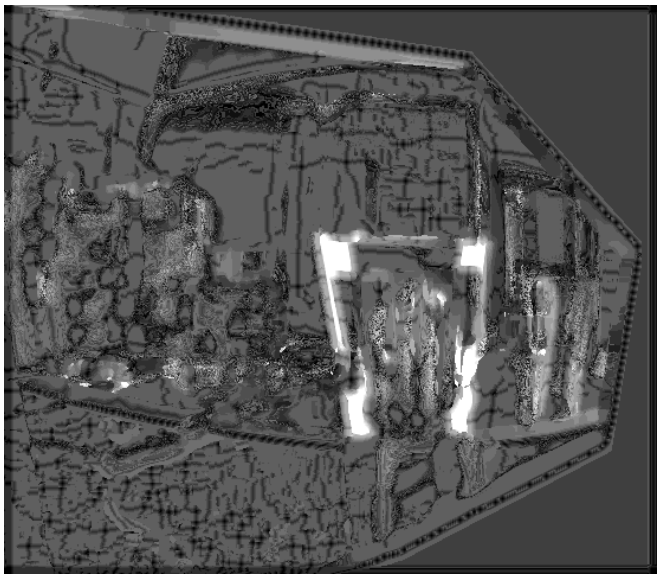

Fig. 10. Custom disparity map

entire previous tree where as this tree can still be useful in certain ways by re-rooting it. This will make computation faster. Finally, the algorithm tries to make no predictions about that movement of the obstacles and it will also not work if there are times when no paths exist to the final region.

B. TASK 2

The disparity map creation is a very active area of research. Current state-of-the-art implementations [7] use deep convolution neural network along with multiple layers of conventional image processing. The algorithm implemented for this task is very simplistic in comparison and arguably provides a worse output than the basic algorithm implemented in OpenCV. The optimisation is only local and suffers from bright patches and missing information at other places.

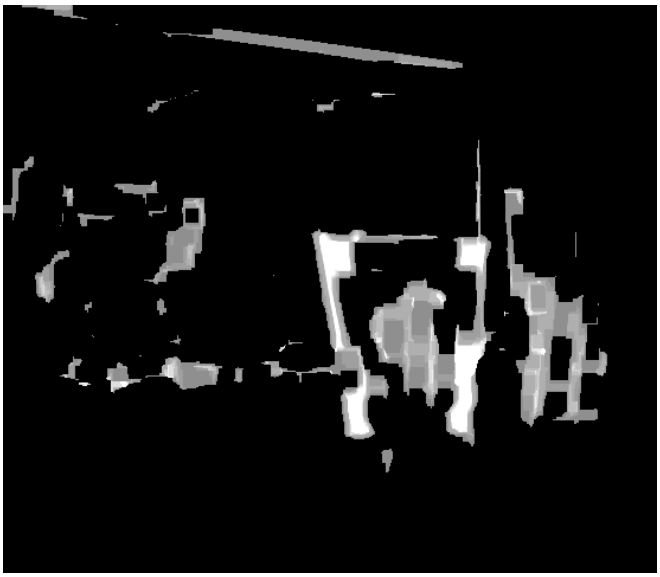

Fig. 11. Custom+Threshold disparity map

As such, there is a lot of room for improvement of algorithms in the disparity map generation part.

For the rectification of images, homographic projection could be improved by automatically finding lines that are supposed to be straight. The rectification will also improve by better feature matching of the SIFT features.

CONCLUSION

Task 1 was relatively very easy. It was made easier by the slow moving obstacles and the fact that a pre-existing algorithm could be used without much changes. The problem, however, is a very useful one and frequently faced. Path-planning is one of the most interesting areas of robotic research and this task was a good hands-on introduction to the same.

Task 2 turned out to be much more challenging. The first part of the challenge, stereo rectification was very hard and I had to learn a lot about stereo projection and cameras in general to be able to get the current solution. My solution ended up involving so many different concepts from SIFT to homography matrices. It was further complicated because of extensive use of custom function and not OpenCV. This task is very useful to all of robotics including ARK. Disparity maps can be used to generate depth information and this can help us in accurately maneuvering the drone as well as extracting extra information from the scene. This information will be particularly useful when trying to navigate close spaces.

REFERENCES

[1] Steven M. Lavalle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning", 1999

[2] David G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", International Journal of Computer November 2004, Volume 60, Issue 2, pp 91110, 2004

[3] Richard Hartley and Andrew Zisserman, "Multiple View Geometry", CVPR, 1999

[4] Devin Connell and Hung Manh La, ”Dynamic Path Planning and Replanning for Mobile Robots using RRT*”, arXiv:1704.04585, 2017
[5] Steven W. Smith, ”FFT Convolution”, The Scientist and Engineer’s Guide to Digital Signal Processing, 1997
[6] Rostam Affendi Hamzah and Haidi Ibrahim, ”Literature Survey on Stereo Vision Disparity Map Algorithms”, Journal of Sensors, Article ID 8742920, 23 pages, 2016
[7] Jure Zbontar and Yann LeCun, ”Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches”, Journal of Machine Learning Research 17 1-32, 2016