Final Project - Progress Report

Vamshi Kalavagunta, Wei-Li Su, Zidong Zhao, Yi-Chung Chen, Ji Liu

The goal of our project is to use optical flow to avoid obstacles using a single camera on a mobile robot. So far, we have achieved the following progress.

1. **Implementation and testing of both sparse and dense optical flow using OpenCV on the actual robot.** The optical flow can be computed using different methods such as Lucas-Kanade and Gunner Farneback approach. We recently wrote a Lucas-Kanade algorithm without pyramids and tested it on videos that were recorded from the robot's camera. Also, we compare the performance of the Lucas-Kanade method with pyramids (OpenCV built-in function). After consideration, the computation time of Lucas-Kanade with pyramids is too long if we wrote it from scratch. Therefore, we decided to implement the OpenCV function and keep the function we wrote as an alternative.

   With the physical robot, we first took videos while driving the robot on a fixed path and tested different optical flow algorithms to see which might offer better performance for the downstream control algorithm. Optical flow algorithms can be characterized as whether they are "sparse" or "dense". Sparse optical flow algorithms extract optical flow only for certain feature points while dense optical flow algorithms extract optical flow for every pixel on the image. Our control algorithm would depend on the left and right fields of view optical flow, and thus a dense algorithm is more preferable. Nevertheless, the dense algorithm might be computationally more expensive than the sparse algorithm and could be problematic for implementation on Raspberry Pi, which is the embedded computer for our mobile robot, to run in real-time. Therefore, it is worth comparing their performances. So far we have tested the Lucas-Kanade method (Figure 1, Video 1) as the sparse method. For dense methods, we tested the Dense Lucas-Kanade method (Figure 2, Video 2) and the Gunner Farneback method (Figure 3, Video 3). These methods offer quite qualitatively different results. The Lucas-Kanade method, as is based on feature points, heavily relies on distinct features in the images. As such, the floor optical flow is largely absent. On the contrary, Dense Lucas-Kanade and Grunner Farneback can more

reliably extract optical flow related to the floor. Comparing the Dense Lucas-Kanade method and the Grunner Farneback method, Grunner Farneback seems to be better in extracting optical flow further in the distance, while the Dense Lucas-Kanade method mostly extracts optical flow closer to the robot. Therefore, for the purpose of obstacle avoidance, the Grunner Farneback method might be the most suitable.

Thus, we used the Grunner Farneback method on one footage where the robot is approaching one obstacle (Figure 4, Video 4). The obstacle was on the left side of the robot, and correspondingly the left field of view optical flow shows areas due to the presence of the obstacle. The key assumption of the control method is that if an obstacle is close by, it would break the balance of the optical flow in the left and right field of view. To show this with Video 4, we computed the normalized strength of the optical flow in the left and right field of view (Figure 5), which indeed shows that the left side has a larger optical flow strength due to the obstacle. Thus, this information can be used to direct the robot away from the obstacle.

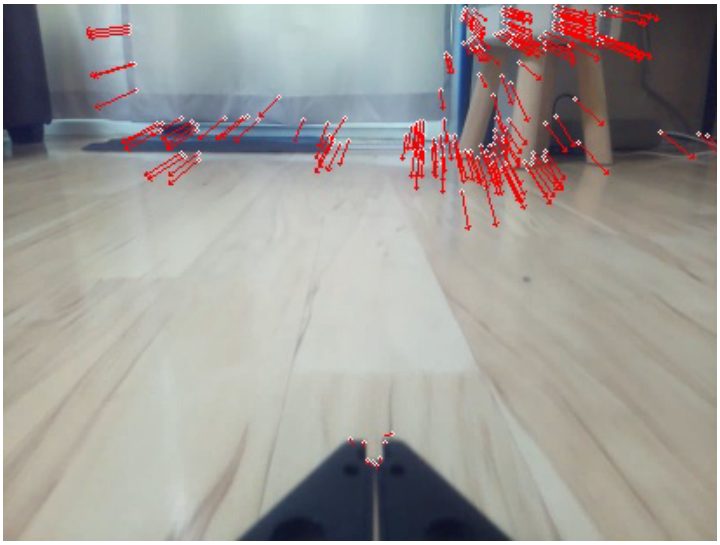

Figure 1. Lucas-Kanade method applied to the footage acquired on a mobile robot moving forward. The feature points are plotted in white, and the optical flow vectors are plotted in red (the length is exaggerated for plotting purposes).

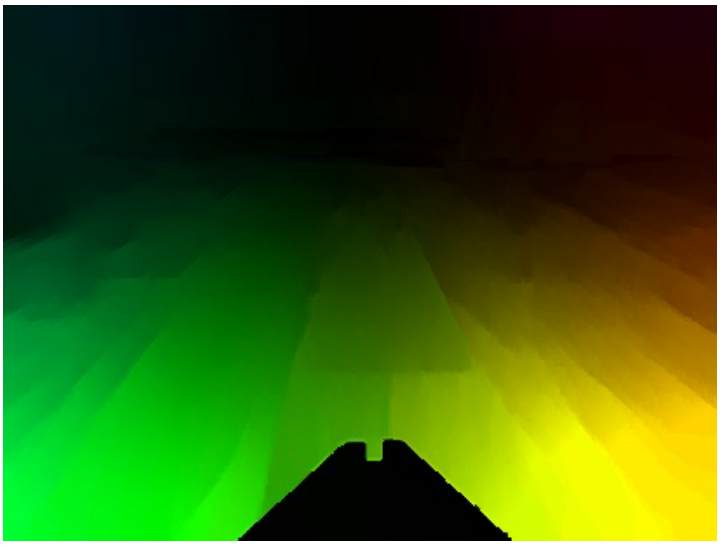

Figure 2. Dense Lucas-Kanade method applied on the same footage as in Figure 1.

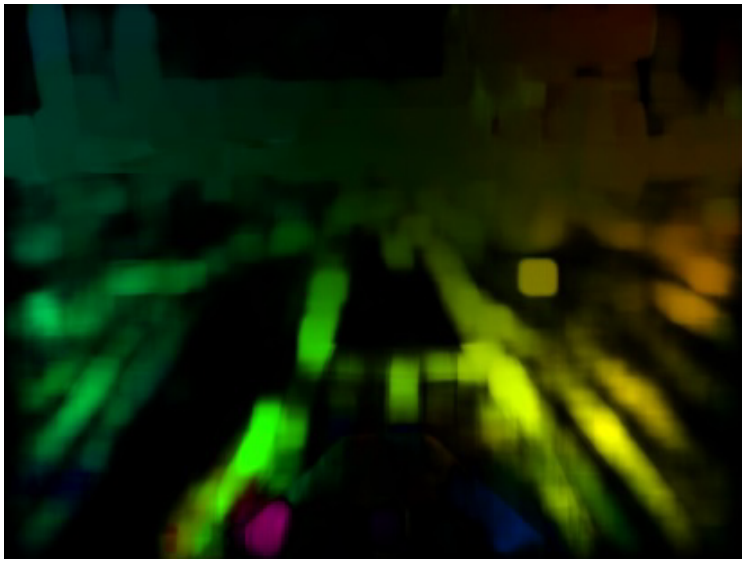

Figure 3. Gunner Farneback method applied to the same footage as in Figure 1.

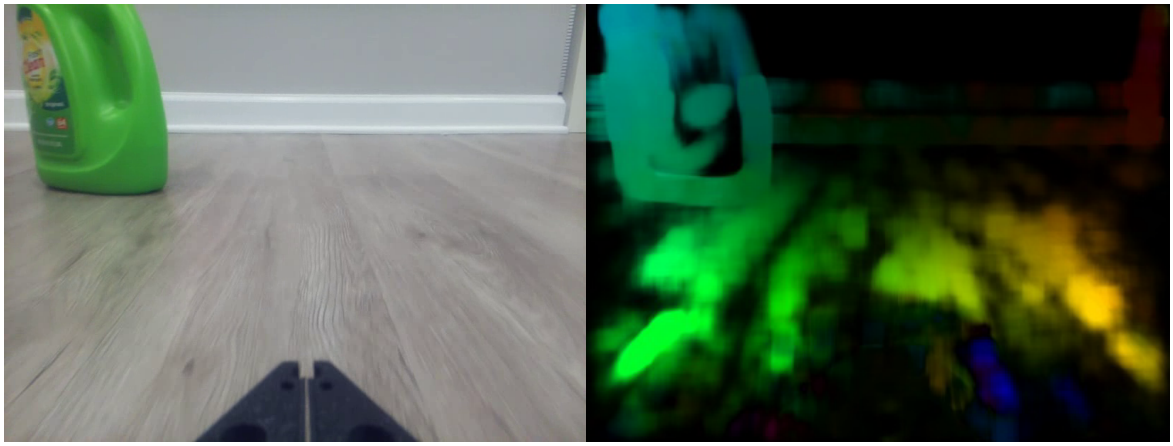

Figure 4. Gunner Farneback method applied to one footage where the robot is approaching an obstacle (green detergent bottle).

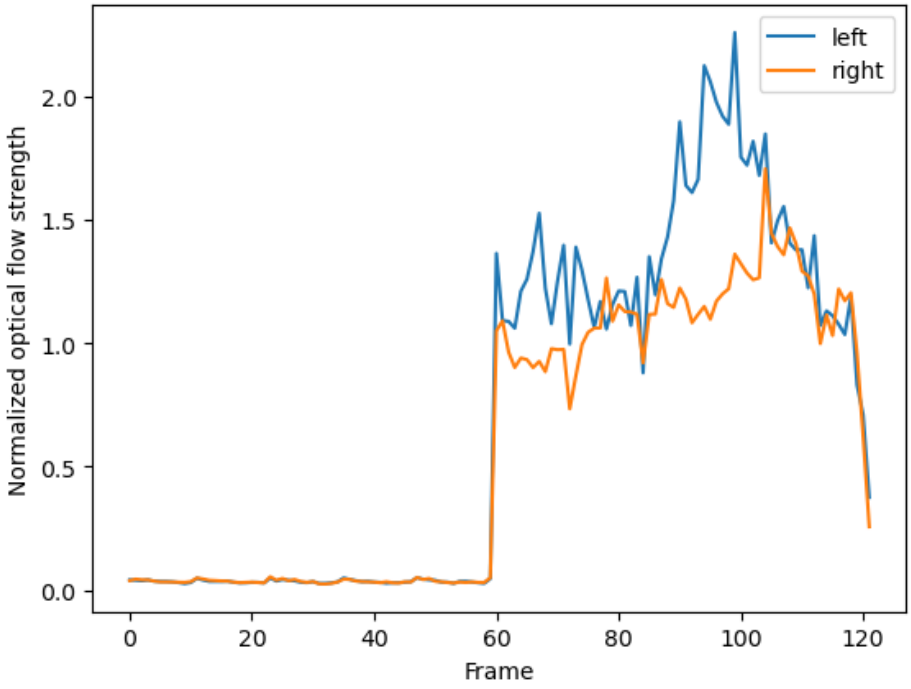

Figure 5. Calculation of optical strength as a function of the left and right field of view, using the same footage as in Figure 4.

2. **Implementation of the balanced optical flow control on the mobile robot.** We have used the Grunner Farneback method in real-time on the mobile robot as a first step to test the feasibility of implementing the optical flow control rule. Our robot has 2 control inputs for the left wheels and the right wheels respectively. For the optical flow control, we first constrain the 2 control inputs to have a constant sum, such that the robot moves approximately at a constant speed. However, the difference between the control inputs is modulated by the optical flow difference across the left and right field of view computed in real time. Two test runs with this control scheme can be found in Video 5 and Video 6, which showed that the robot can successfully steer away from the obstacle, which is the expected behavior. However, this method will not work for scenarios where the robot is for example approaching a wall head-on, as the optical flow for the left and right field of view is still balanced in this case. Therefore, in subsequent steps, we need to add the Focus of Expansion and depth calculation to the pipeline to augment the control law to deal with situations such as the aforementioned.

3. **Focus of Expansion calculation**

Finding the focus of expansion (FOE) using the optical flow feature points is to be implemented as follows:

1. Calculate the optical flow vectors for the feature points using the iterative Lucas-Kanade/Grunner Farneback algorithm.
2. Filter the optical flow vectors that are considered to have good quality.
3. Separate the optical flow vectors into two groups based on their direction, one for vectors pointing towards the left and the other for vectors pointing towards the right.
4. Calculate the center of mass for each group of vectors. This will give us two points, one for the left group and one for the right group.
5. Draw a line connecting the two centers of mass points. This line represents the direction of motion in the image.
6. Calculate the intersection of this line with the image plane. This intersection point is the FOE.

7. We can then use the FOE to determine the direction in which the robot should move to avoid obstacles.

4. Time to Contact and depth image is to be implemented as follows:

From the correspondence points across subsequent images, it is possible to find the essential matrix relating these points. From the essential matrix, the corresponding translation and rotation matrix can be extracted. Thus, given the subsequent two frames in the video stream, the velocity v of the camera origin and the distance Δ to the obstacles or surrounding objects can be computed. Given v and Δ. the time to contact can be computed as follows:

$$T = \Delta / v$$

To calculate the depth image, the Δ value computed above for each pixel can be thresholded as follows to produce a grayscale image:

$$\frac{\Delta_i - \Delta_{min}}{\Delta_{max} - \Delta_{min}} * 255$$

To recover the essential matrix, it is important that we calibrate the camera and we have finished this step (Figure 6). The essential matrix for the camera of our robot is as follows (the dimension of the image acquired was 820 by 616):

```
[[664.68   0.   413.48]
 [  0.   664.87 322.61]
 [  0.     0.     1.  ]]
```

For the calibration, more than 50 images of the checkerboard pattern were collected, and the average projection error was 0.041 pixel.

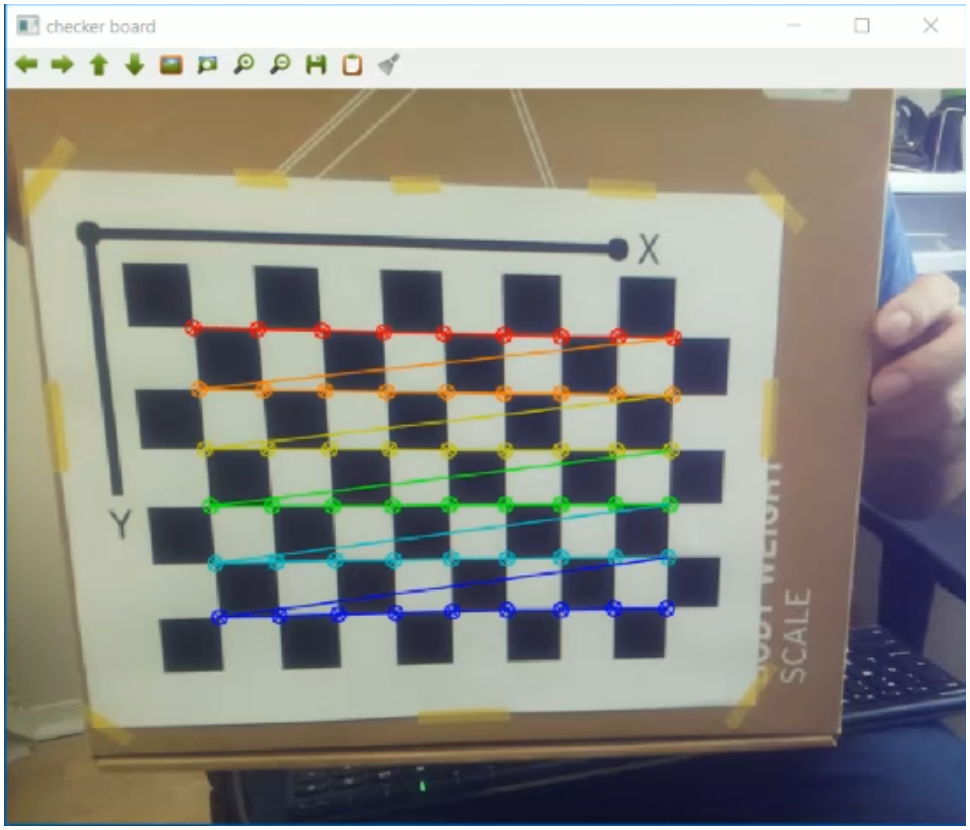

Figure 6. Camera calibration using the checkerboard pattern.

Together, we have made progress toward completing the final project and in the next stage, we will be focusing on augmenting the current control rule with the FoE and Time to Contact information.

Group members:

1. Vamshi Kalavagunta [119126332]
2. Ji Liu [112960186]
3. Wei-Li Su [117525298]
4. Yi-Chung Chen [119218990]
5. Zidong Zhao[117513646]