CS 3630 - Assignment 5

Alexander Huynh - James Liu

Data collection

To collect our data, we ran our robot using the same motion plan through 3 different environments. The robot took a roughly sinusoidal path, taking numerous images during each turn, and halting while taking each image. (We chose the included environment to analyze because it provided the best results.)

The path taken as interpreted by non-noisy odometry data only is down below in Figure 1. Of course, the actual path took by the robot will vary slightly due to imperfections.

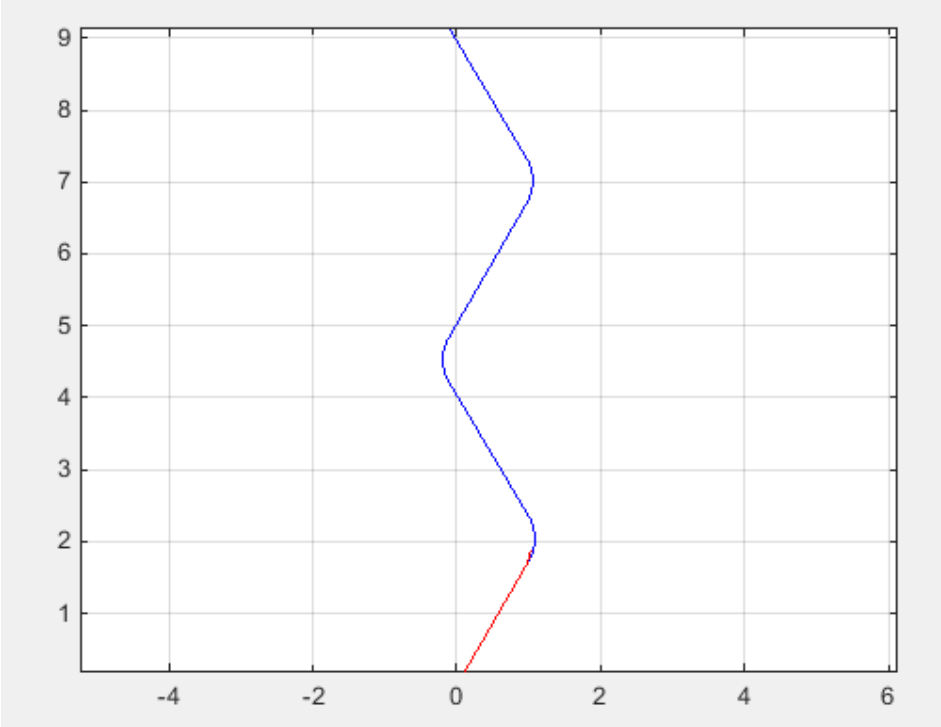

Figure 1. A perfect rendition of the path taken by our robot.

Feature matching

We make use of the RVC Toolbox extensively in our calculations.

We used SURF to identify features in our images. We chose this over Harris corner features because although it is more computationally expensive, the quality of SURF features is much better across the different images of the same features. This is the appropriate choice in our situation, as we have a moving robot taking images of a constant environment. An example of an image taken by the robot is shown in Figure 2.

Figure 2. A raw image taken by the Scribbler.

We use SURF in the RVC toolbox by calling:

```
surf = isurf(image);
```

on each of our images, and store it in a matrix for later use. We can see an example of the features found from the previous example image in Figure 3. Clearly, the features SURF found are reasonable; they are focused on the distinctive areas of the outlet and calculator, while avoiding non-distinct areas such as the wall or floor.

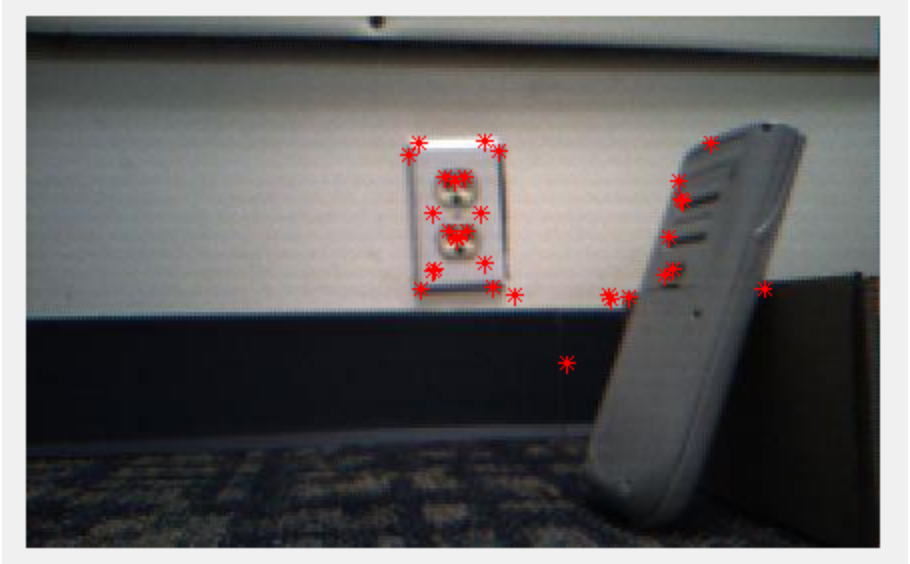

Figure 3. The features found by SURF.

Next, we match SURF features between two consecutive images. We again use the RVC Toolbox by calling:

```
match = current.match(next, 'thresh', 40);
```

We found an appropriate value for the distance threshold through trial and error. The results of this feature matching are shown in Figure 4, for the example image previously used as well as the previous image in the sequence. We see that while there are some good correspondences, the matches are quite noisy. We will rectify this by using RANSAC in the next section.

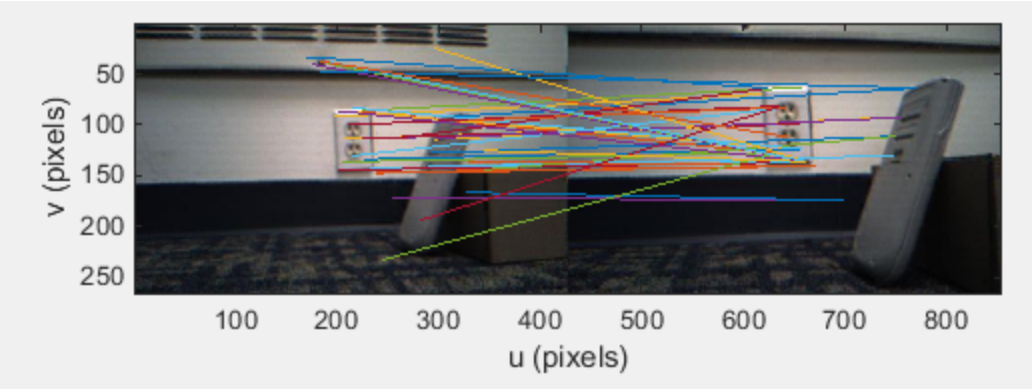

Figure 4. Raw feature matches between two consecutive images

Geometric verification using RANSAC

We next estimate the fundamental matrix F from the feature matches. To find the most appropriate model from our data, we will use RANSAC. RANSAC will find the best model of 8 corresponding points, chosen by majority consensus.

We can see in Figure 4 that most of the matches are fairly horizontal, with lines closer to the top sloping upwards, and vice versa. RANSAC will attempt to filter out the outliers.

Using the RVC toolbox, we call:

```
F = match.ransac(@fmatrix, 1e-1, 'verbose');
```

and find our fundamental matrix, F. We found our chosen threshold by trial and error, using the value that works reasonably across as many images as possible. The inliers from after running RANSAC are shown below in Figure 5. Although there are still some mistakes, this new model is significantly better than before RANSAC was used.

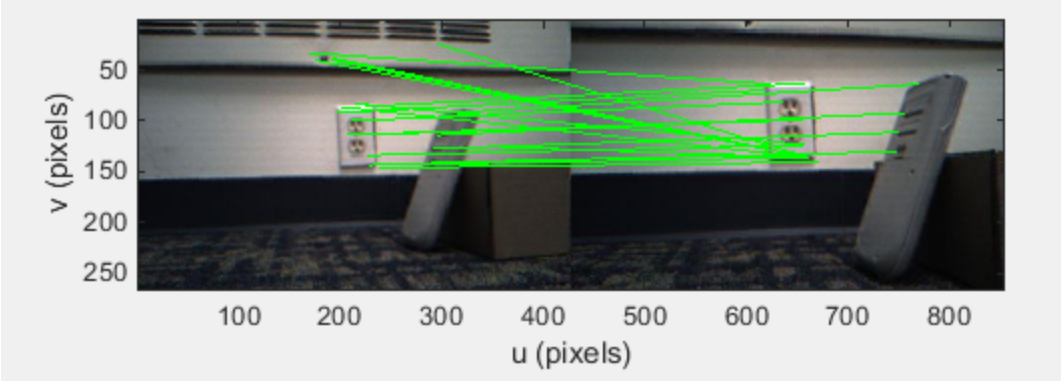

Figure 5. Feature matches between two consecutive images after filtering via RANSAC

Estimating essential matrix

We are given the intrinsic matrix, K, and we have found the fundamental matrix, F, in the previous step. To estimate the essential matrix, we simply compute:

```
E = K'FK
```

Estimating relative transformation from Essential Matrix

We decompose the essential matrix E using the RVC Toolbox. As there are two possible solutions, we test to see if a point in front of our camera, Q, is visible or not between the two images. This will give us the correct solution.

```
Q = [0 0 100]';
sol = camera.invE(E, Q);
```

Through this decomposition, we can recover both the rotation matrix, R, as well as the translation matrix, t.

```
R = sol(1:3, 1:3);
t = sol(1:3, 4);
```

We can then extract the euler angles for the rotation and X/Y translation.

```
euler = tr2eul(R);
```

Integrating relative transformations

We can then update the 2D movements of the robot using these values via euler integration.

```
x(3) = x(3) + euler(3);
x(1) = x(1) + sol(1) * cos(x(3) - sol(2) * sin(x(3)));
x(2) = x(2) + sol(1) * sin(x(3) + sol(2) * cos(x(3)));
```

Finally, we update the history and record the movements:

```
x_hist = [x_hist; x];
```

The results are seen below in Figure 6b.

Comparison of motor and visual odometry

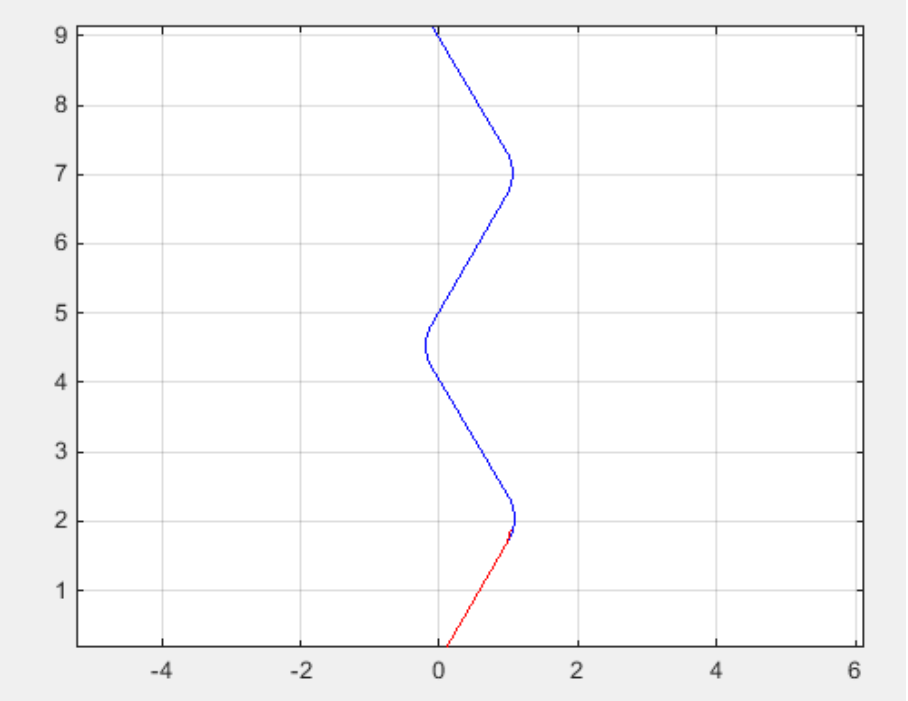

Figure 6a. The rendition of the path taken by our robot using motor odometry data.

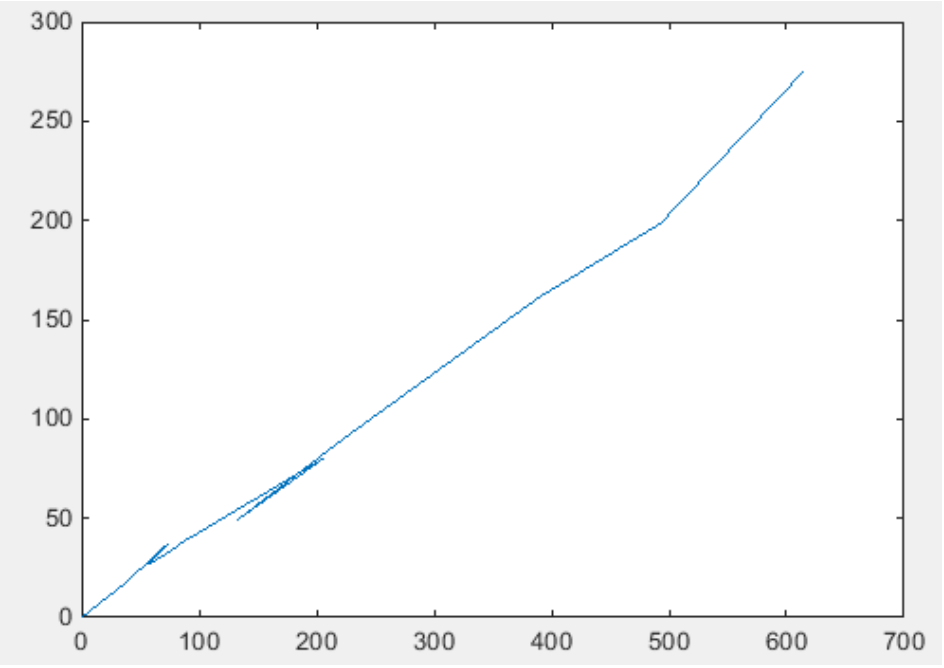

Figure 6b. The rendition of the path generated by our visual odometry algorithm.

The visual odometry output doesn't look as well-formed as the normal odometry only run. It is off by a large scale which could be fixed with a bit of scaling. Also the normal odometry run is handled in world coordinates while the visual odometry run is handled in relative to the robot's initial start frame, which is why the path is rotated by a fixed factor. Our visual odometry run also doesn't seem to handle robot's rotations well, either because there were not enough matches between frames taken by the robot or the change was too large for it to effectively handle.

Despite the differences, in general, the visual odometry does still slightly follow the predicted motor odometry data; apart from scaling, it does not deviate by an extreme amount. (That is, the path is not a wild jumbled mess.) This indicates that matching features of the images are, in fact, being identified and matched correctly to at least some degree.

To improve our results, calibration could be used. We elected to not use calibration for the sake of simplicity. However, because calculating the essential matrix is a very important step in recovering the transformation matrices, more accurate intrinsic parameters could have helped. Using more pictures may have also helped. By increasing the amount of images and decreasing the amount we transform between each image, we increase the chance that SURF and RANSAC will find good feature matches, therefore increasing the accuracy of the recovered transformation matrices. Finally, operating in an environment with more distinct features could have also helped. Despite the clarity of features such as the calculator and outlet, even more easily-distinguishable features would also help improve the SURF and RANSAC steps.