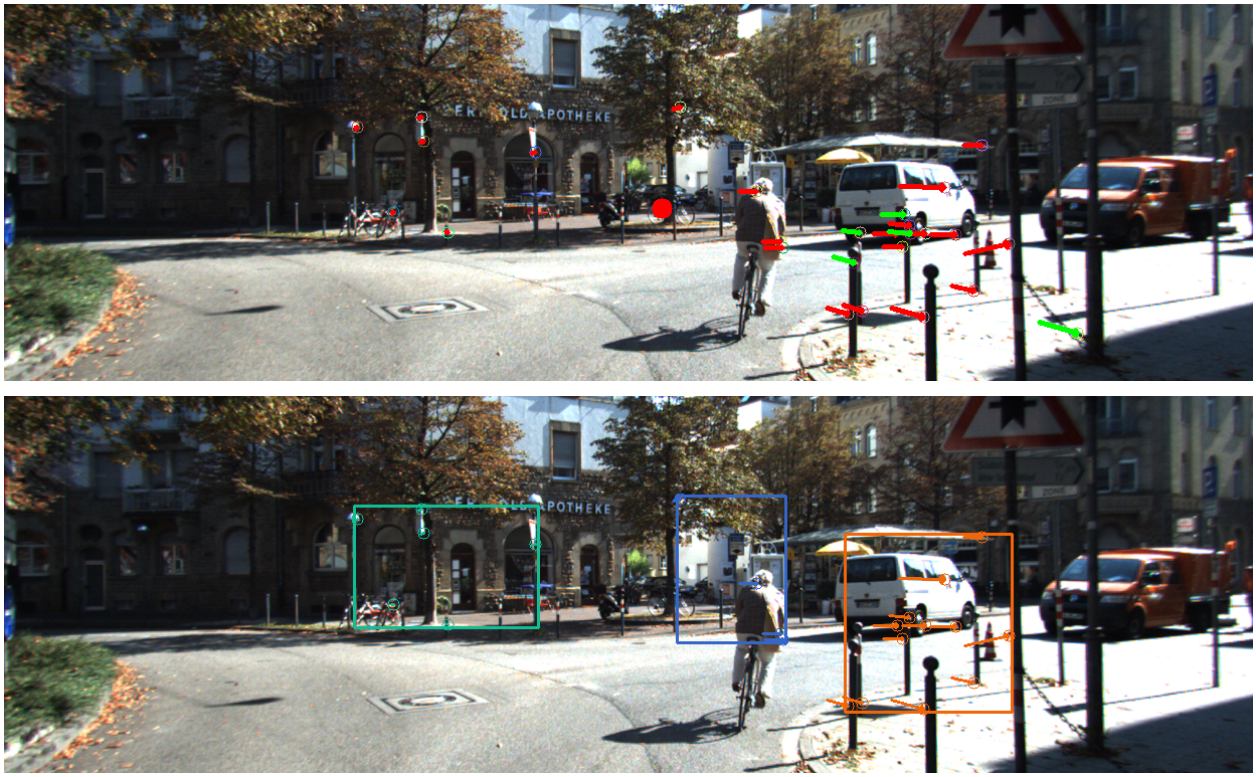
Jae Park
CV HW 7

(a) Design Decisions

- cv2.goodFeaturesToTrack() was used to detect keypoints.
    - https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_shi_tomasi/py_shi_tomasi.html
    - This algorithm is an improvement of Harris Corner detection algo.
- Lucas-Kanade Optical Flow in CV2 was used to estimate apparent motion from the keypoints
    - https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_lucas_kanade/py_lucas_kanade.html
- Motion vectors with magnitude less than 1 pixel was discarded (lower bound)
    - Also tried upper bound of pixels but didn't make much of a difference as no motion vectors that were extreme in length were found.
- Used RANSAC to find inliers/outliers
    - Fit potential FOEs across all motion vectors
    - Choose the best FOE with the largest number of motion vectors intersecting
- Decide that the camera is in motion if the number of inliers exceeds a certain number (4 does the best job out of 2,3 & 4)
    - Too few inliers means that the camera is probably not in motion.
- sklearn.cluster.KMeans was applied to outliers to find clusters of motions vectors that are from independently moving objects

(b) Results



Here is a case where it worked pretty well.

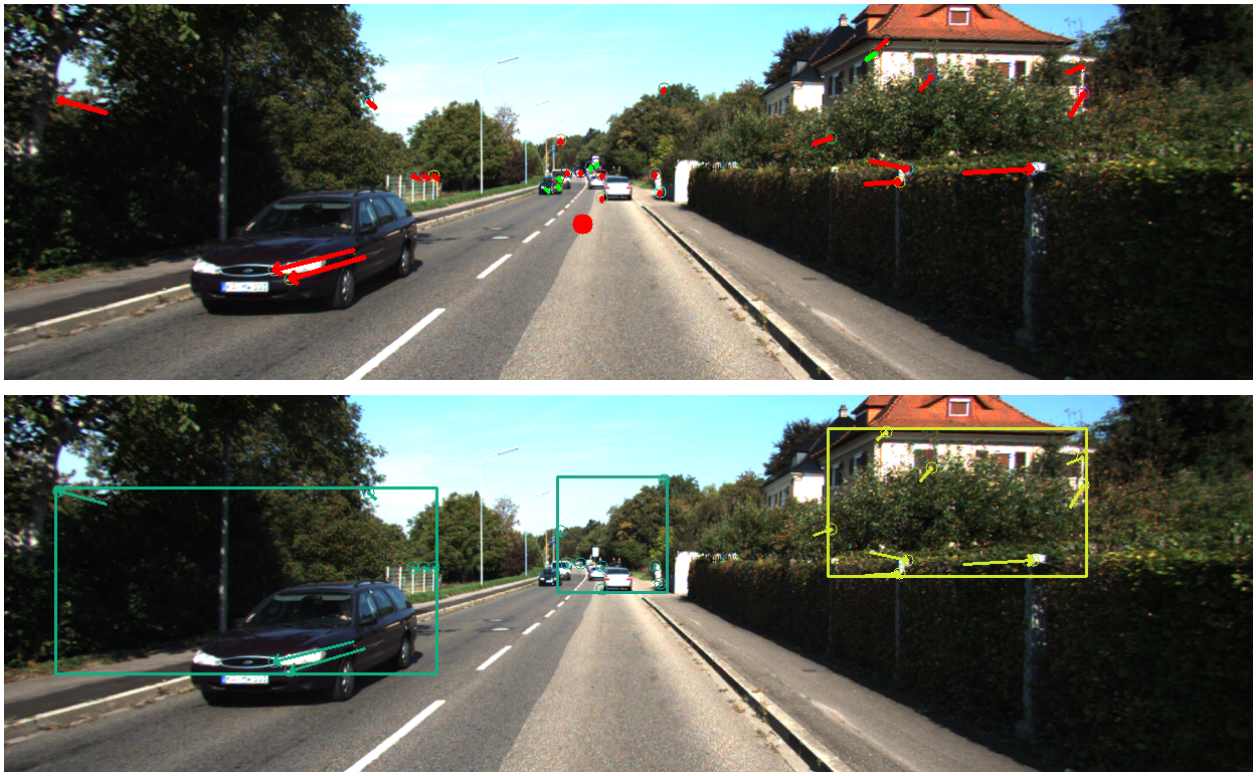
The red circle near the center is the FOE. Green arrow lines are inlier motion vectors, and red arrow lines are the outliers (errors or from independently moving objects).

The detection algorithm was able to roughly locate the bicycler and the white van that are moving. However, one false detection of a building in the turquoise box also exists. The red van is actually parked and stationary if you manually compare the two input images, so that was done correctly.

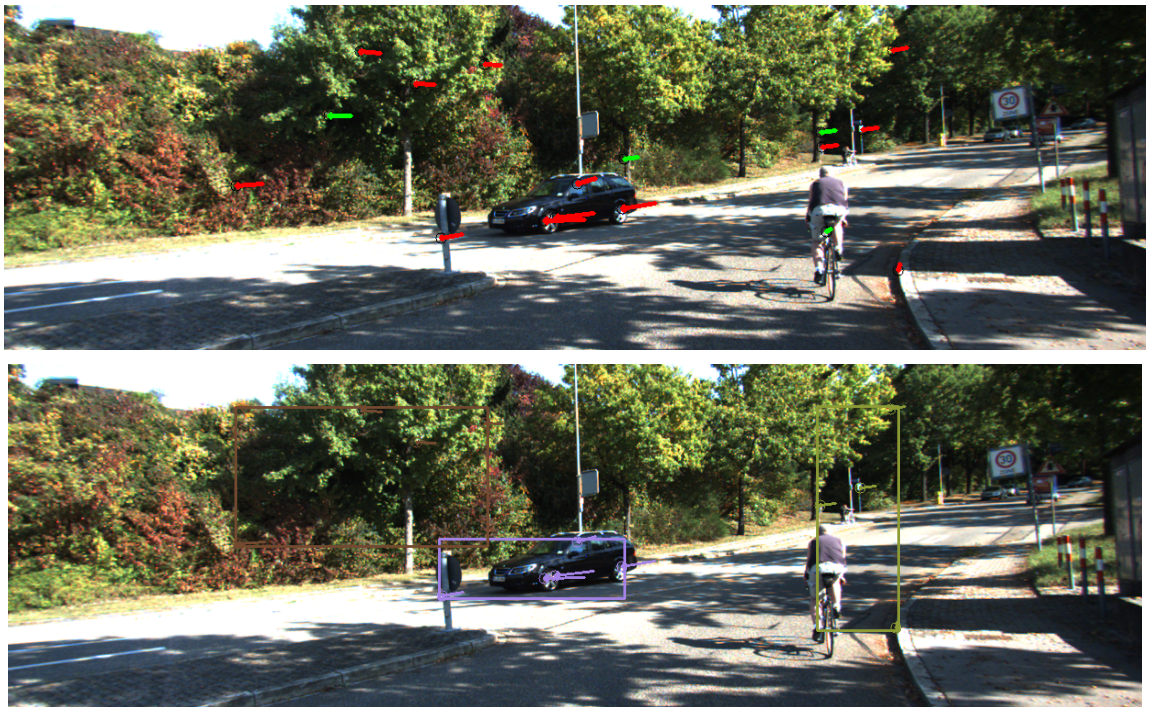
Within the correctly identified bounding boxes, the motion vectors are pretty much parallel, facing the same way, and similar in magnitude (by the length of them).

Although the point will be illustrated with more examples in the following, the major problem with my approach was that, overall, the bounding boxes don't work very well. Significant improvements will be needed to really have it working.

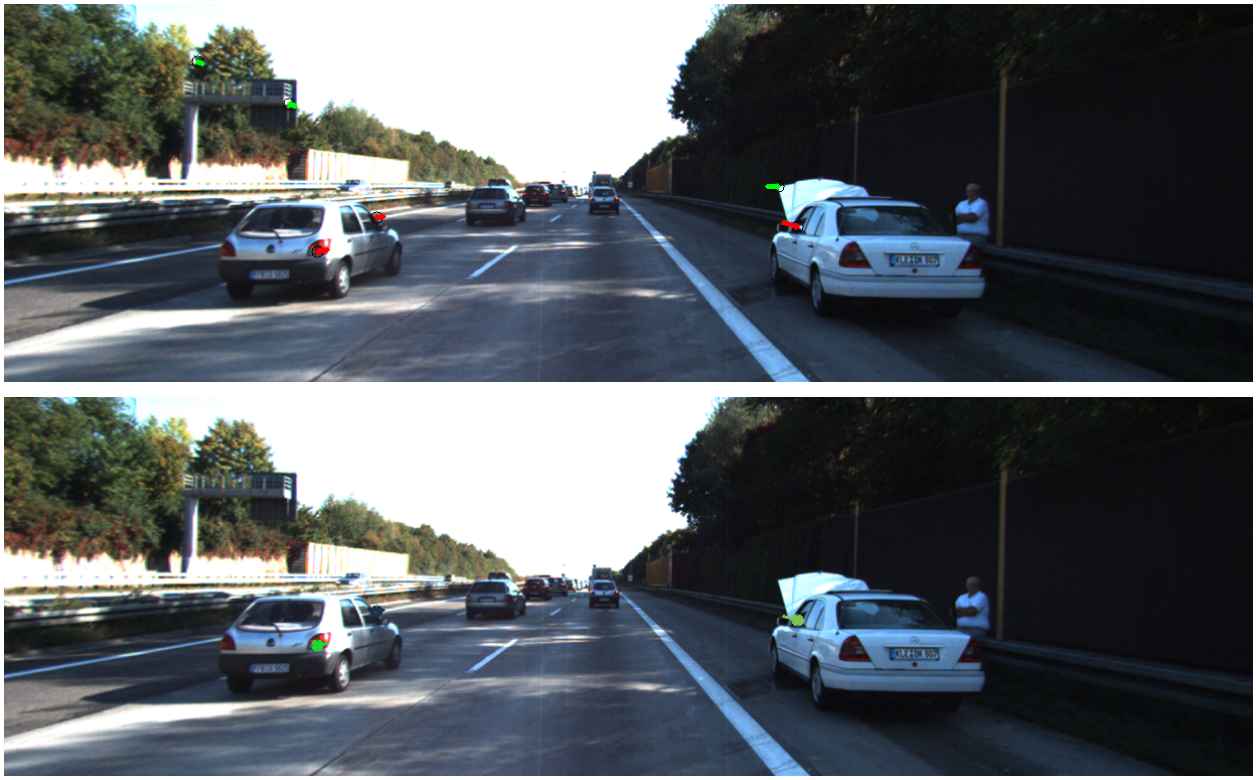
The input image was taken when the camera was not in motion but there were two vehicles that were moving in front: the white van and the bus. Some motion vectors were detected, but the bounding box didn’t quite get the bus, and didn’t find the white van at all.



Here's an example where the house was detected as an object in motion. The reason why many keypoints were detected on the house is probably because the detection algorithm is sensitive to corners. This lead to a false classification of it as an object in motion, and subsequently a bounding box was put around it.



Similar example of that where it found the tree on the left side as an moving object.



Lastly, here's an example where it failed badly. If you manually compare the two input images, what happened is that this car was stuck in traffic. And it's very likely that only this car moved just the slightest bit, and the other cars weren't moving between the moments when the two images were taken.
As other cars were nearly stationary, it makes sense that there are no bounding boxes (no objects of independent motion). However, due to low number of keypoints, it neither found a FOE. (no red dot towards the middle).

(c) Discussion
The biggest issue with the program is that it does not detect objects very well. Nevertheless, it can still find FOE pretty accurately. How this could be improved will be discussed below.

When the motion vectors are short (movements are small), the algorithms seem to get confused. It's hard to find a FOE when there are a bunch of short vectors scattered all over the place because that introduces more room for error.

High texture (trees), inconsistent brightness (due to shadows or other reasons), and other various factors can pose a challenge to the program.

With more time and help, I would like to try a different approach as good keypoint detection and bounding box detection didn't work quite well.

The suggestion is as follows:

Use variations of CNN to detect objects (cars/pedestrian) -> wrap the objects with bounding boxes and store the upper-left and lower-right coordinates of them -> use SIFT/ORB to find keypoints all over the (two) images -> only look at the keypoints that are inside the bounding boxes -> calculate motion vectors -> get FOE, independently moving objects