Final report

FP1 Match 3D objects

See implementation in code. Note that this function also populates the keypoints and keypoint matches in the bounding boxes, since it has to go through them anyway. (It doesn't filter out outliers.) A pair of bounding boxes (A in previous frame, B in current frame) is identified as a match if most keypoint matches from A to the current frame are to points in B, and most keypoint matches from B to points in the previous frame are to points in A; this two-way maximization is done to prevent situations like a car mirror being matched to the entire car, although in the current dataset they probably wouldn't have arisen.

FP2 Compute Lidar-based TTC

See implementation in code. I am discarding the two closest points (because those are often outliers), and using the mean of the next 30 as the distance estimate.

FP3 Associate keypoint correspondences with bounding boxes

See implementation in code. Note, the keypoints and matches are populated into their bounding boxes in FP1, so this function only filters out the outliers.

I felt this algorithm wasn't explained well in the course, so here's what I'm doing. I'm computing the pairwise distances between keypoints in the current frame, and die distances between corresponding pairs of matched keypoints in the previous frame. I am then taking the ratios of these distances. For any correct match, this ratio should be approximately constant. We could potentially detect its value by generating a histogram and picking the peak, but instead for each keypoint we compile the list of ratios for the distances from this keypoint to all the others, take the median (per keypoint), and filter out outliers from this list of medians. The outliers should indicate keypoints that were not matched correctly.

I would've appreciated a better background on why filtering out keypoints in this way is helpful: why not just take the median of all the ratios in the next step? We're relying on it being approximately correct for this method to work anyway.

FP4 Compute camera-based TTC

See implementation in code. As in FP3, we compute the distances between pairs of keypoints in the current frame, and their matched counterparts in the previous frame. Then we take the median of the ratios between these distances, and use it in the constant-velocity model. (I replace negative answers by large positive ones, on the theory that that is more intuitive; however, I leave infinite answers, which occur when the median ratio comes out to exactly 1, as infinite.)

FP5 Performance evaluation 1

See FP5.doc and FP5.xls.

FP6 Performance evaluation 2

See FP6.doc and ttcCamera.xls.