Mika Thormann 5465950

Progress report for week 1

Tasks:

1) Detect and correct poorly approximated corners caused by occlusion.

2) Better table orientation visualization by drawing a coordinate frame on the image.

3) Download and train TTNet

Implementation:

1) The idea for this task was to use the mask created for the table to further determine the middle line of the table. This line should be used to determine if a corner was poorly approximated and to calculate a better approximation. However, I realized that distances between corners and the middle line vary greatly due to differences in lens and perspective. In the following image, you can see that the distance to the bottom corner (red line) is almost twice as big as the distance to the right corner(green line).

That is why I came up with another idea. Every approximated point is given by 2 lines. In an ideal case, the 4 corner points are given by 4 lines (the tables edges) such that every line contributes to the approximation of exactly 2 points. If an occlusion now causes a fifth line, there will be 2 points that stem from one line that only contributed to the approximation of this single point and no other. These 2 points are now candidates for being poorly approximated due to occlusion(c1 and c2). If I can determine which points stem from occlusion-lines, I can simply determine the corrected position of those given the other lines. This is illustrated in the next image where c2 was replaced with c2* using the yellow line of c1 and the purple line of c2. If this is not possible (less than 3 lines contribute to 2 points simultaneously or both candidate points stem from occlusion), the program stops.

2) I simply applied cv.ProjectPoints on a scaled coordinate frame, using the translation vector and rotation vector.

3) I had various issues while installing the required resources(cuda etc.) and the data preparation took surprisingly long so I just now got the model ready to train. I will let the program run overnight and look at the results tomorrow.

Problems:
While writing this report, I realized that the method I used to determine if a candidate point stems from occlusion does not generally work in every case. I could do it with the middle line, but the earlier mentioned problem still persists. So I am still looking for a method to reliably determine which points stem from occlusion.

Side Note:
I reworked the code by including classes to make everything easier to work with.