Introduction to Computer Vision – Homework 3
RA192617 – Edgar Rodolfo Quispe Condori
RA192618 – Darwin Ttito Concha
Institute of Computing, University of Campinas (UNICAMP)
Campinas-SP, Brazil, 13083-852

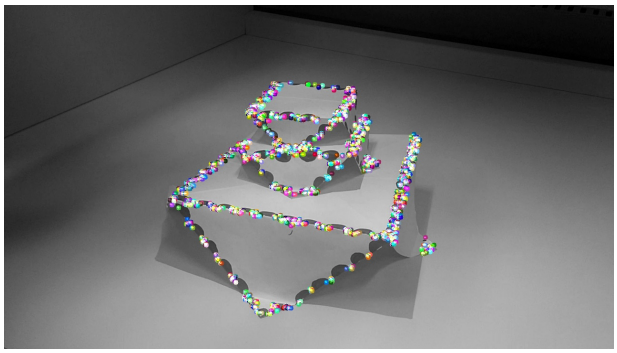
1 Input data and remarks

The video used for the experiments will be downloaded by command $make\ data$, but input along with outputs are available at this: *link*. This project has an extra directory named "my-output" that contains the $.ply$ file previously generated.

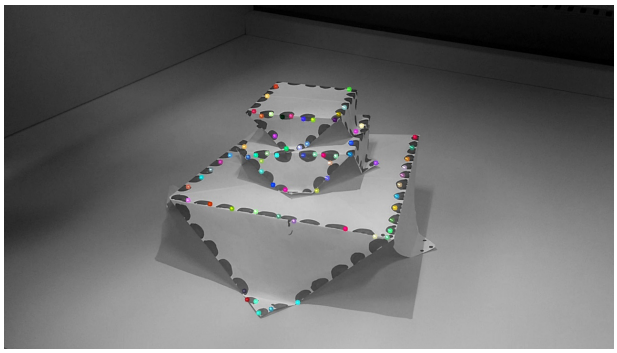
Command $make$ will download the input and run the project. Output tracking results of each frame as well as the $.ply$ is generated in directory $output$, note that this may take a while.

2 Keypoint Selection

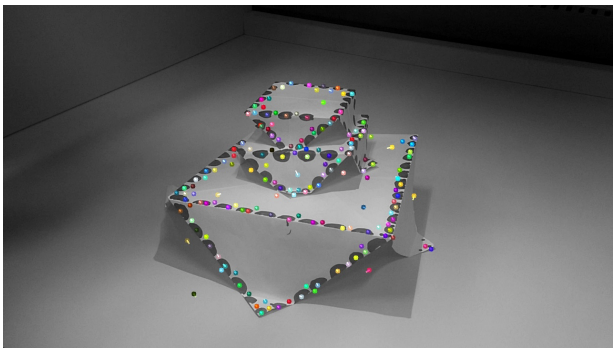
In order to decide the keypoints detector we evaluate three methods: Harris, Shi-Tomasi (Good Features to Track) and SIFT (Scale Invariant Features). Figure 1 shows some results for the feature extraction, we used OpenCV implementations for the detectors in this step.



(a) Harris



(b) Shi-Tomasi



(c) SIFT

Figure 1: Comparison of keyppoint detectors

As can be seen in the figure 1, Harris detects a lot of keypoints, most of them are overlapped, Shi-Tomasi detects similar points like Harris but removes the cases of overlap and on the other hand SIFT detects points that at first glance may look desirable but there are cases where these points are in the foreground and not in the object that we want to reconstruct, and this creates noise in the final reconstruction. Thus, we consider to use Harris because it has more points than Shi-Tomasi and all of them are in the object of interest close to the borders.

3 Feature tracking

For the feature step we decided to implement KLT with a pyramid, this was because the results with KLT were not good enough (figure 2). Our implementation of the pyramid may not follow a method proposed by an specific author but it is intuitive. Consider a video with frames I of some size w, h, and the same video with frames I' of size $w/2, h/2$. Then, if a keypoint has a "movement" of $[u, v]$ in frame I, it will have a movement of $[u/2, v/2]$ in frame I', this is util because KLT assumes that the movement between frames is small. Based on this idea we can build a pyramid such that we compute the motion in the highest level (smallest frame) and "translate" this motion to the smallest

level (original image). Note that intermediate levels of the pyramid are not used for motion computation. "Translate" means that if the motion at some level is $[u, v]$, then the motion in a lower level (bigger frame scale) the motion is $[2 \times u, 2 \times v]$.

Some implementation details of the tracking include: a bilinear interpolation in the computation of the derivatives, and a smoothing in the highest level of the pyramid. To compute the derivative of a pixel (x, y) we consider:

$$\begin{aligned} \frac{\partial f(x,y)}{\partial x} &= \frac{f(x+1,y) - f(x-1,y)}{2} \\ \frac{\partial f(x,y)}{\partial y} &= \frac{f(x,y+1) - f(x,y-1)}{2} \end{aligned} \tag{1}$$

The denominator 2 is important because without it the motion magnitude is divided by 2 in the final estimation.

A comparison between implementation with and without the pyramid is shown in figures 2 and 3. When pyramids are not used, most of the points are lost when a big movement occurs.

Figure 2: KLT without pyramid

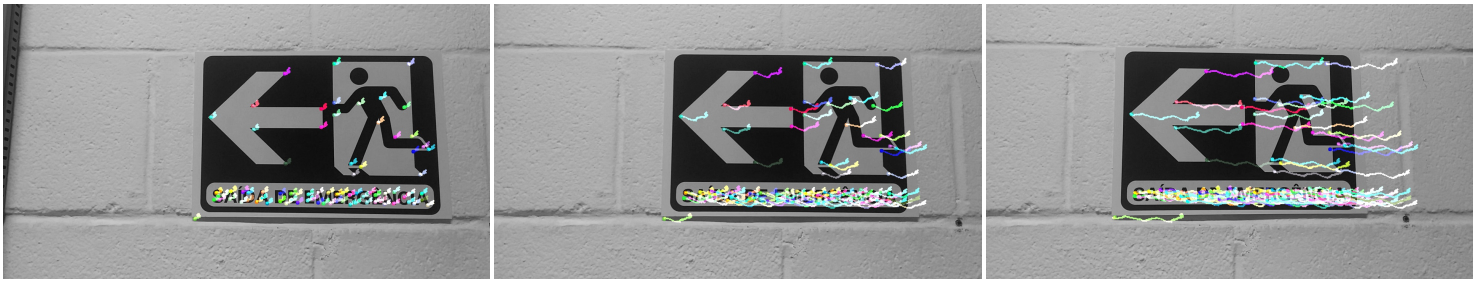


Figure 3: KLT with pyramid

The test of different sizes of neighborhood are shown in figures 4, 5, 6

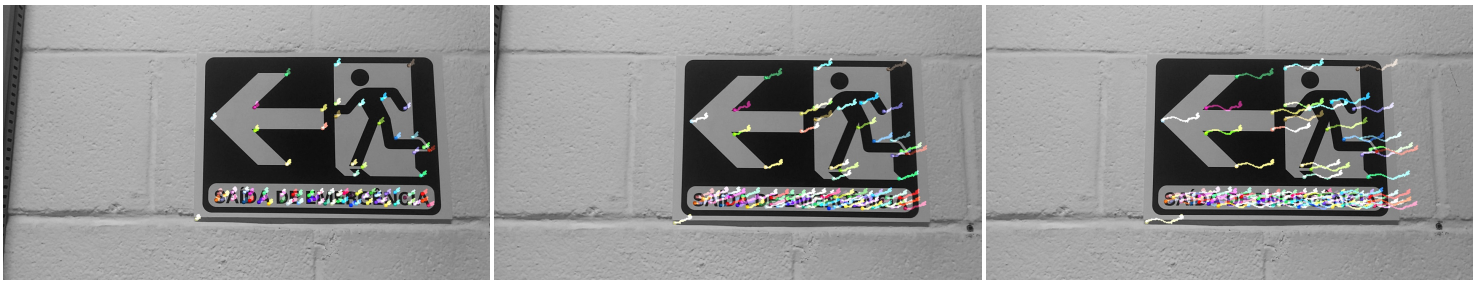


Figure 4: Neightborhood $= 5 \times 5$

Figure 5: Neightborhood $= 15 \times 15$

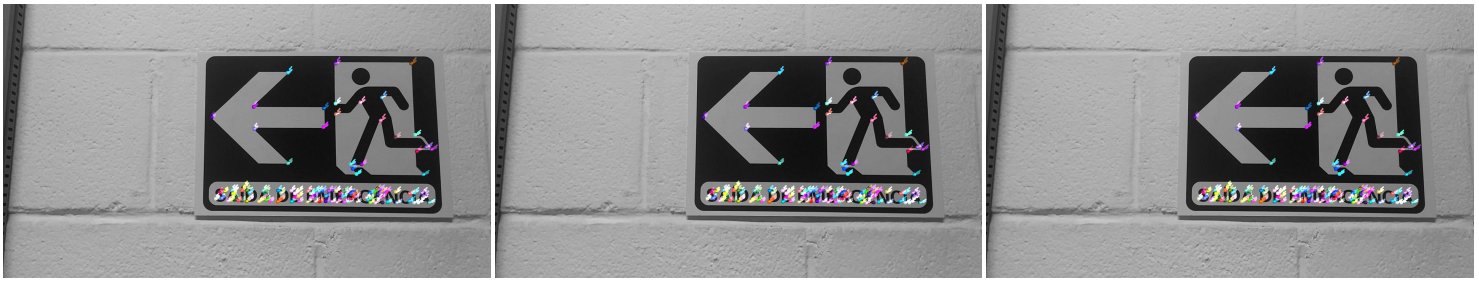


Figure 6: Neightborhood $= 20 \times 20$

As can be seen in figures 4, 5 and 6, there is not a dramatic improvement with the change of neighborhood, then, we decided to use a 15×15 neighborhood because the original authors suggest to use this size. Figure 7 shows the results of tracking of the input video with the 15×15 neighborhood.

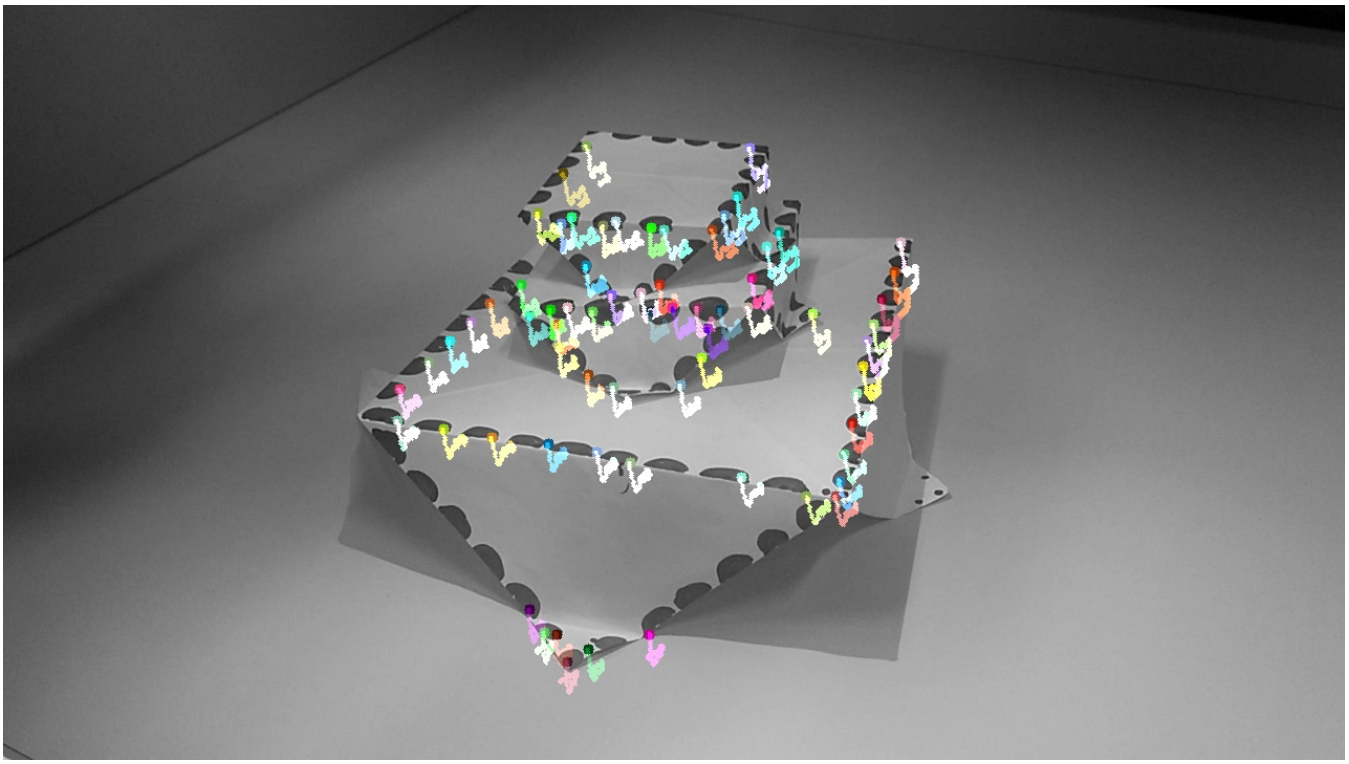
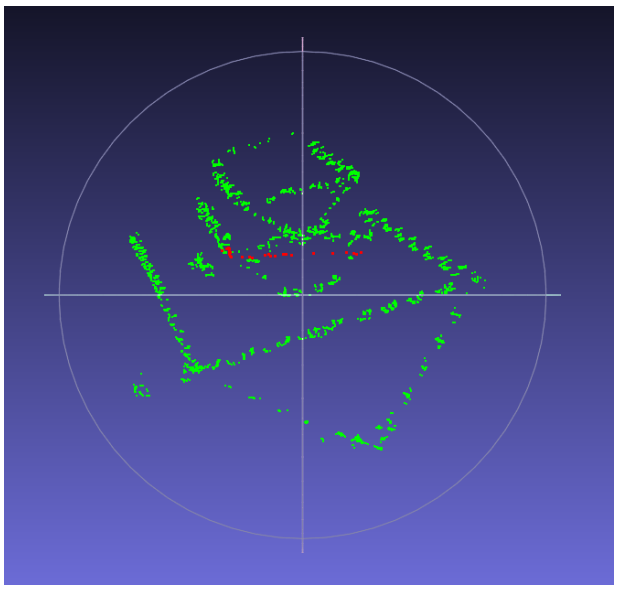


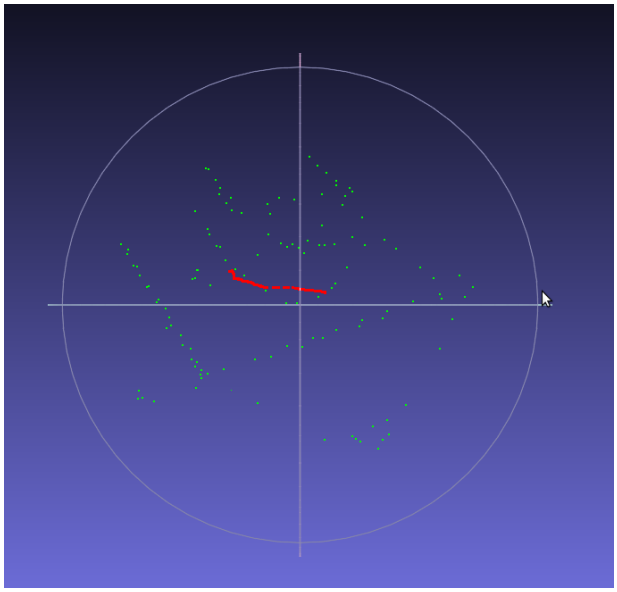
Figure 7: Results of tracking

4 Structure from Motion

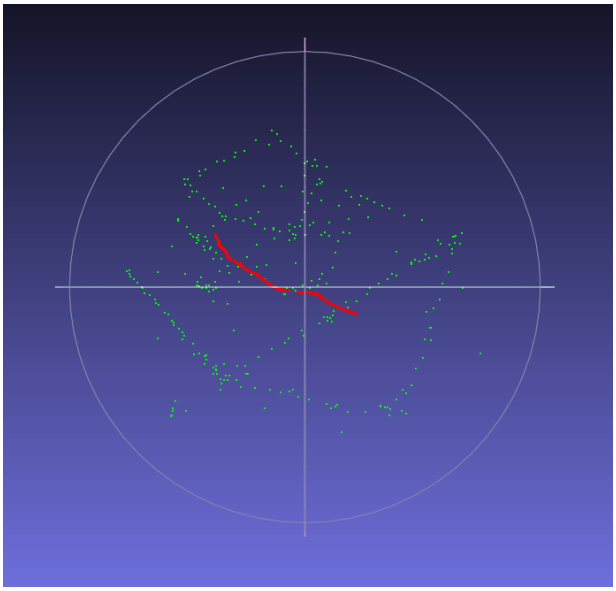
We ran experiments with 3 different keypoint detectors (a frame of the video is shown in figure 7) and this give as an idea of the best number of keypoints. As can be seen in figure 8.b, if the number of keypoints is low, it is difficult to notice the form, while if we consider more points (SIFT may consider foreground points) we can create some noise that is not desirable (figure 8.c), on the other hand, considering a lot of points but in specific parts of the object (specially in the borders) give us a better understanding of the structure (figure 8.a). Hence, a lot of points are desirable but their location is also important for better reconstruction.



(a) Harris



(b) Shi-Tomasi



(c) SIFT

Figure 8: Comparison of Structure from motion with different keypoint detectors.

Results of the structure from motion are shown in figure 9, note that the structure is represented by green points and the camera path with red points.

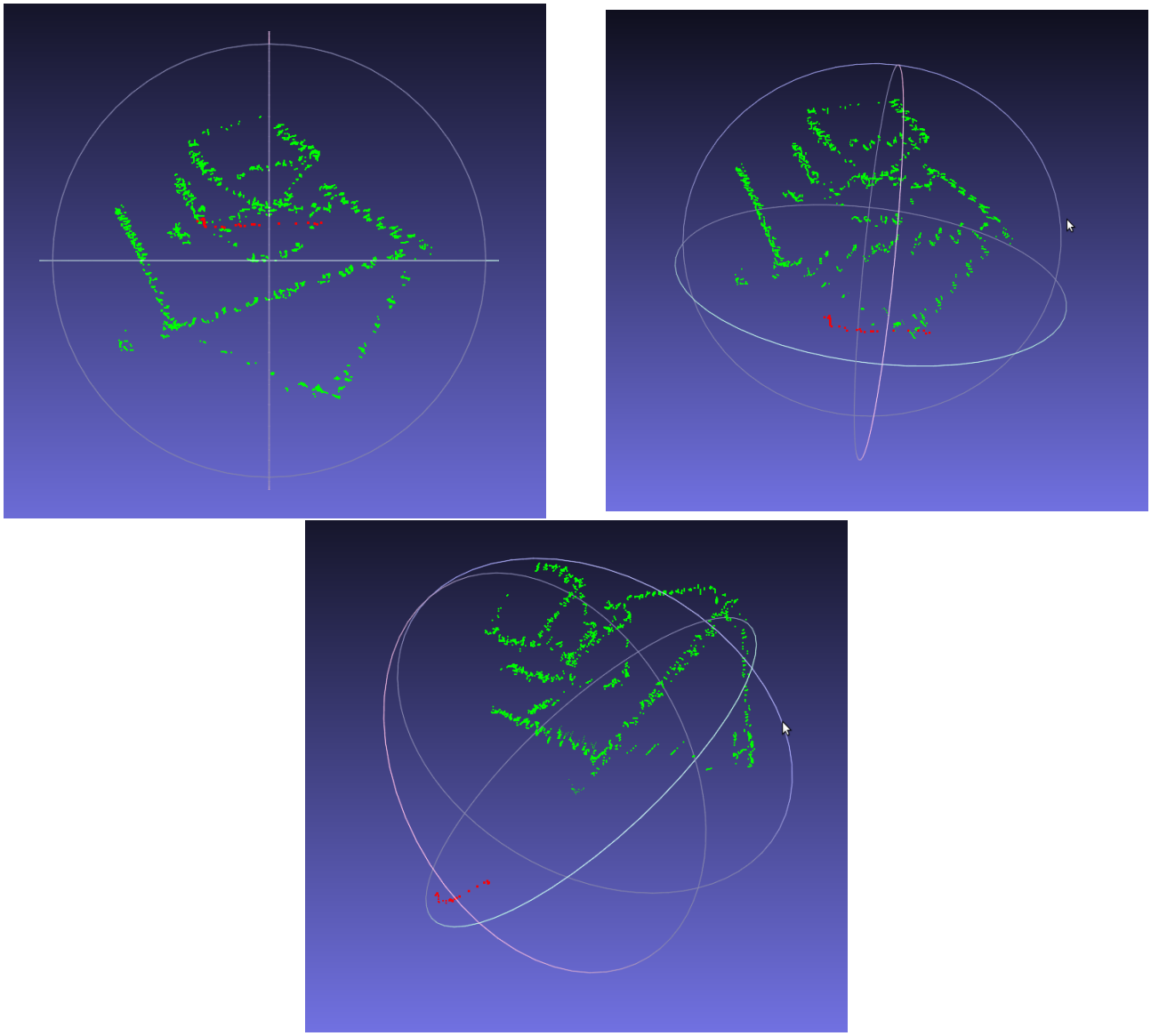


Figure 9: Structure from motion with Harris detector.

The expected technique to recover the camera position is to use homogeneous coordinates in the equation $X = MS$, such that when solving this system, M represents a transformation sub matrixes m of size 3×4 that maps a 3D point $(X, Y, Z, 1)$ to a 2D point $(x, y, 1)$, then m can be expressed as $[CR]$ where C is a 3×3 and $R = -CP$ is 3×1, then, the camera position P can be computed as $-C^{-1}R$, but we had problems working with homogeneous coordinates when solving with Cholesky decomposition(not positive definite matrix as input), we think this is because when constructing X in homogeneous coordinates and filling a third of the matrix with 1's the rank of the matrix remains three and the intermediate results rich in a non positive definite matrix, thus in order to recover the camera path we work over the 2×3 matrix m without homogeneous coordinates, let $m = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{bmatrix}$, then the camera position is given by the cross-product of vectors $[m_{11}, m_{12}, m_{13}]$ and $[m_{21}, m_{22}, m_{23}]$, the reason we use this product is that if the image plane is orthogonal to a vector V that goes from the origin of the 3D world [1] and $[m_{11}, m_{12}, m_{13}]$, $[m_{21}, m_{22}, m_{23}]$ defines the image plane, then, its cross product is also orthogonal to the image plane. The magnitude of this cross product is usually small, thus we multiply the result by a constant (experimentally setted to 200) before plotting.

[1] Note that center of the 3D world is also the centroit of the 3D object (assumtion of Factorization method)