Labb 3

Elias Berglin

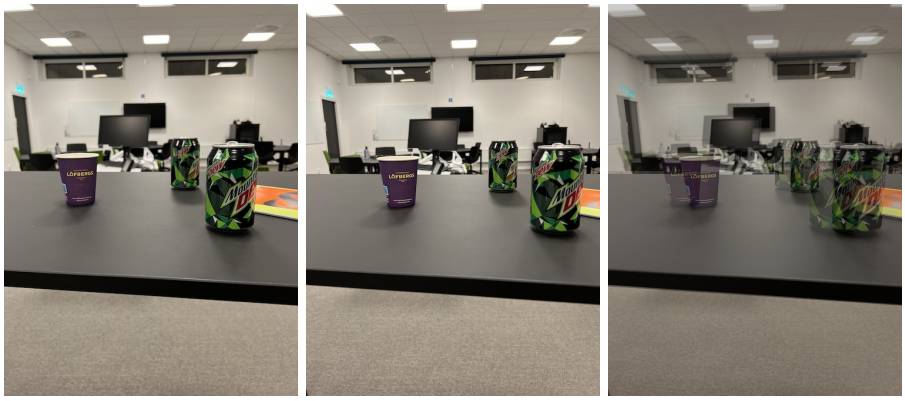
8th December 2021

1 Week 5

1.1 Assignment 5

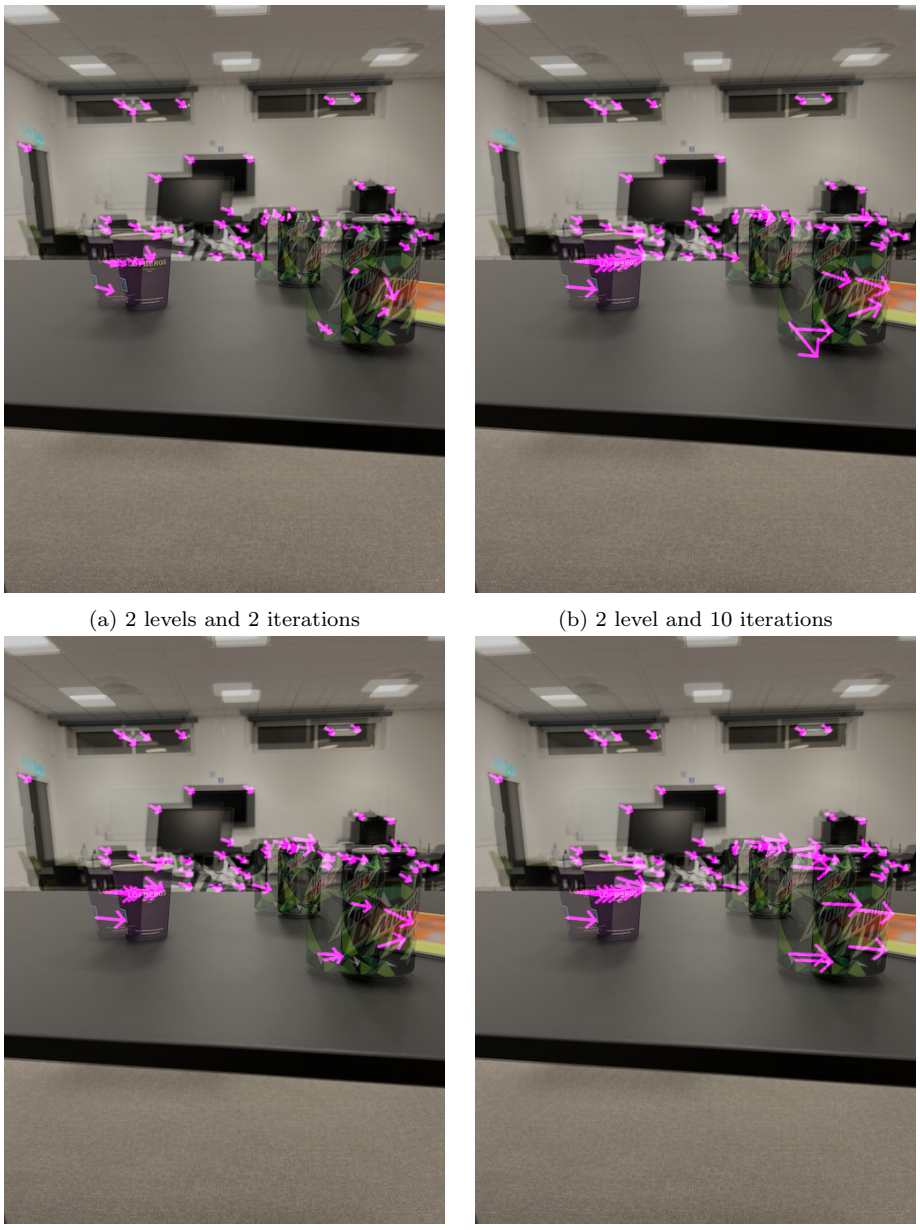
1.1.1 Optical Flow

I used OpenCV's implementation for Optical Flow. This was originally used for videos but was adopted to be used with images instead. Coarse-to-fine algorithm improves our result on larger deltas by creating different layers of the image in different sizes. The smaller sizes helps the algorithm find features points. The itterations helps the algorithm to improve our results by correcting errors. In figure 1 we can see the input images and in figure 2 we can see the different results. We can see that the arrows correct them selves with more iterations and the points getting more accurate with more layers. The Python code can be found in appendix A



(a) First image (b) Second image (c) Images overlayed

Figure 1: The two input images for Optical Flow and the the images overlayed



(a) 2 levels and 2 iterations

(b) 2 level and 10 iterations

(c) 3 layers and 2 iterations

(d) 3 layers and 10 iterations

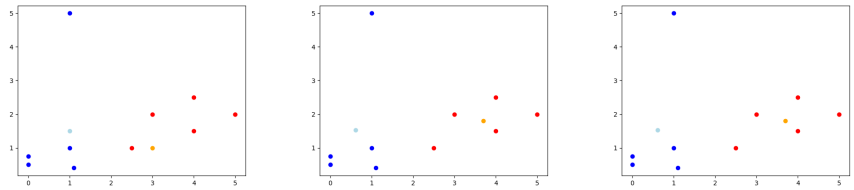
Figure 2: Resulting vectors from Lucas-Kanade method with different amount of pyramids and iterations

1.1.2 K-means

The K-means function can be found in equation 1. This algorithm assumes that all of the datasets belong to a category μ.

$$\min_{\mu,y} \sum_{i} \|x_i - \mu_{y_i}\|^2 \tag{1}$$

In figure 3 we can se the different iterations. In figure 3a is the original data categorized. Then first iteratio in figure 3b and then when we compare that to 3c we see that nothing changes and thus our final categorized are decided.



(a) 0 Iterations (b) 1 Iteration (c) 2 Iteration

Figure 3: Iterations of K-means

1.1.3 Segmentation

Appendix A Python code for optical flow

```
import numpy as np
import cv2
import os

frame1 = cv2.imread("im1.jpg")
frame2 = cv2.imread("im2.jpg")

frame1g = cv2.cvtColor(frame1, cv2.COLOR_BGR2GRAY)
frame2g = cv2.cvtColor(frame2, cv2.COLOR_BGR2GRAY)

combined = cv2.addWeighted(frame1, 0.3, frame2, 0.5, 0)
cv2.imwrite("pyramid/combined.png", combined)

parameters = dict(maxCorners=100, qualityLevel=0.3, minDistance=7,
    blockSize=7)

p0 = cv2.goodFeaturesToTrack(frame1g, mask=None, **parameters)

maxLevels = [2,3]
maxIterations = [2, 10]

for level in maxLevels:
    print("On Max Level: ", level)
    for iter in maxIterations:
        OFparams = dict(
            winSize=(15, 15),
            maxLevel=level,
            criteria=(cv2.TERM_CRITERIA_EPS |
                      cv2.TERM_CRITERIA_COUNT, iter, 0.03),
        )
        p1, st, err = cv2.calcOpticalFlowPyrLK(
            frame1g, frame2g, p0, None, **OFparams)
        good_new = p1[st == 1]
        good_prev = p0[st == 1]

        arrows = np.zeros_like(combined)

        for i, (n, p) in enumerate(zip(good_new, good_prev)):
            nx, ny = n.ravel()
            px, py = p.ravel()

            nx = int(nx)
            ny = int(ny)
            px = int(px)
            py = int(py)

            arrows = cv2.arrowedLine(arrows, (px, py), (nx, ny), [
                                255, 0, 255], 2, cv2.LINE_AA,
                                    tipLength=0.4)

            output = cv2.add(combined, arrows)
            if(not os.path.exists("pyramid/"+str(level))):
                os.mkdir("pyramid/"+str(level))
            outName = "pyramid/" + str(level)+"/" + \
                str(level) + "levels-" + str(iter)+"iterations.png"
            cv2.imwrite(outName, output)
```