INFOMCV Assignment 2
Authors (39)

Summary

(Briefly explain your background subtraction method, your postprocessing and how you build the voxel model. Approx. half a page.)

The background subtraction method creates a background image and saves the image to be processed for finding the foreground masks. I split this code into two main functions. We used averaging to improve the method. We used the cv2.morphology() function for noise reduction and post-processing. This function is the difference between dilation and erosion of an image (according to OpenCV documentation). The result will look like the outline of the object. For further post-processing, and to condense all of the images of the video into one mask, we used an 'accumulator', which increased the stability of the detected foreground mask. This method accumulates the cleaned mask from each frame of 'video.avi', and helps in distinguishing noisy pixels and elements that actually belong to the foreground. The voxel model was constructed by loading the mask and creating the look up table with the function is_voxel_in_foreground. This function is then used in the adapted set_voxel_position function to create the voxel reconstruction from the foreground masks rather than randomly generated. The lookup table is a dictionary which should be appending the voxels which are 'on'. In other words, if a voxel is 'visible', as dictated by the 'is_voxel_in_foreground' function, the voxel is appended to the lookup table dictionary. One of the many issues we were having was struggling with this specific function. The closest we got to realizing the assignment was producing a voxel reconstruction which rendered all the voxels as on.

Extrinsic parameters

(Include rotation matrix and translation for each of the four cameras. Approx. one third of a page.)

Cam 1:

$$R = \begin{bmatrix} \mathbf{0.72166976} & \mathbf{-0.69222514} & \mathbf{0.00413718} \\ \mathbf{0.10661145} & \mathbf{0.11704755} & \mathbf{0.9873874} \\ \mathbf{-0.68397862} & \mathbf{-0.71212656} & \mathbf{0.1582688} \end{bmatrix}$$

$$T = [\mathbf{230.38561877} \quad \mathbf{723.74131947} \quad \mathbf{4574.08603922}]$$

Cam 2:

$$R = \begin{bmatrix} \mathbf{0.9993751} & \mathbf{-0.03430174} & \mathbf{0.00853201} \\ \mathbf{-0.0091356} & \mathbf{-0.01747844} & \mathbf{0.9998055} \\ \mathbf{-0.03414595} & \mathbf{-0.99925867} & \mathbf{-0.01778089} \end{bmatrix}$$

$$T = [\mathbf{-216.15791448} \quad \mathbf{1480.97159024} \quad \mathbf{3583.36127291}]$$

Cam 3:

$$R = \begin{bmatrix} \mathbf{0.21667293} & \mathbf{-0.97603197} & \mathbf{0.02035751} \\ \mathbf{0.0409106} & \mathbf{0.02991252} & \mathbf{0.99871496} \\ \mathbf{-0.97538667} & \mathbf{-0.21556166} & \mathbf{0.04641129} \end{bmatrix}$$

$$T = [\mathbf{-300.33055773} \quad \mathbf{1212.55705836} \quad \mathbf{3313.40556956}]$$

Cam 4:

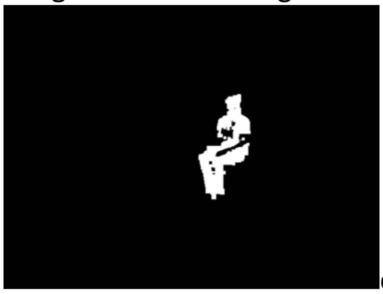
$$R = \begin{bmatrix} \mathbf{-0.6354288} & \mathbf{-0.77204512} & \mathbf{-0.01328779} \\ \mathbf{0.08178206} & \mathbf{-0.0844023} & \mathbf{0.99306996} \\ \mathbf{-0.76781634} & \mathbf{0.62993855} & \mathbf{0.11677109} \end{bmatrix}$$

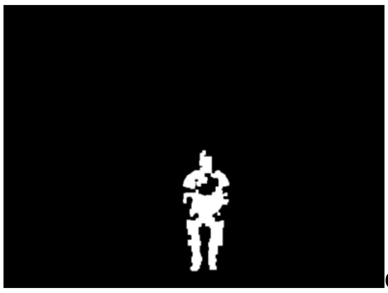
$$T = [\mathbf{-13.66370428} \quad \mathbf{907.21651387} \quad \mathbf{4321.60902664}]$$

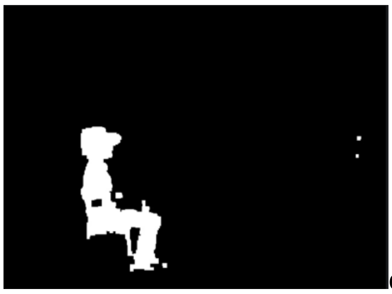
Background subtraction

(Mention how you set the thresholds for your background subtraction (or the description of other parameters in your approach), and how it is determined if a pixel is foreground or background. If you use an approach with training a background model, explain how that works. Also include a foreground image for each of the four cameras. Approx. half a page.)

The thresholds are hard-coded in this file subtractBackground.py for robustness. The hue value is set from 10 to 100 to ensure only significant changes are detected and error caused by reflections or shadows are discounted. The saturation has a lower limit of 30 to discount noise, which is often associated with low saturation. The value is set with a lower level of 40, to ignore very dark pixels which may indicate shadows or darker areas not contributing to the foreground. The background model image is used to detect whether an a pixel is foreground or background by comparing it to the particular video frame. The background model is created in the subtractBackground.py script with averaging employed to improve the background subtraction. We were unable to successfully attempt automatic thresholding due to time constraints, because we found the voxel reconstruction quite challenging. The foreground mask output as a result of hardcoding gives a satisfactory result however, due to its robustness and iterative accumulation which helps reduce noise. It may not, however, be as accurate as automatic thresholding. This is one of the restraints of our assignment. The foreground masks for cameras 1-4 respectively are shown below:


cam1


cam2


cam3

cam4

Choice tasks
(Indicate which ones you did, and how you did them; Approx. one third of a page.)
No choice tasks were implemented

Link to video
(Link to a video (Youtube, Vimeo, Wetransfer, etc.) clearly showing the 3D reconstruction of the input videos. Make sure the link is accessible by r.w.poppe@uu.nl and m.doyran@uu.nl.)