

# INFOMCV Assignment 2

## Nikola Grigorov and Ying Dang (Assignment pair: 1)

### Summary

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

We adjusted assignment 1 to use videos to perform the calibration for the cameras. We obtained the intrinsics and extrinsics via it. Background subtraction is based on a bruteforce adsdiff/threshold on all three HSV channels, then combined via bitwise\_or. To build the voxel model we added two arrays. One to hold all voxels positions, and one to hold per voxel all 4 pixel positions of each camera. These arrays share the same index. This allows us to match the voxel array ID to the pixel array ID. Once we construct the voxel space, and we project their coordinates on each view to generate the pixel array. We then make a loop for each voxel, then since we share the same id we can instantly get its 4 pixels, check if per view on foreground if they lay on white pixel. If they lay on all 4 cameras at the same time, we push the voxel coordinate to the renderer.

### Extrinsic parameters

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

#### Cam1

##### Rotation Matrix

$$\begin{bmatrix} 0.72306746 & -0.6907744 & 0.002041336 \\ 0.09565552 & 0.103053115 & 0.9900657 \\ -0.6841224 & -0.71568906 & 0.14059074 \end{bmatrix} \begin{bmatrix} 235.57646914672793 & 819.5515441115317 & 4596.107180994614 \end{bmatrix}$$

##### Tvecs

#### Cam2

##### Rotation Matrix

$$\begin{bmatrix} 0.99968517 & -0.024028005 & 0.0072295996 \\ -0.0073863855 & -0.0064427853 & 0.99995196 \\ -0.023980273 & -0.99969053 & -0.0066182367 \end{bmatrix} \begin{bmatrix} -259.7277307922695 & 1478.8132257826453 & 3581.0798157058616 \end{bmatrix}$$

##### Tvecs

#### Cam3

##### Rotation Matrix

$$\begin{bmatrix} -0.23149142 & 0.97243947 & 0.027805772 \\ -0.05945487 & -0.042670526 & 0.99731857 \\ 0.97101843 & 0.2292175 & 0.06769413 \end{bmatrix} \begin{bmatrix} -715.8035030990718 & 1237.446717476261 & 2405.18767545015 \end{bmatrix}$$

##### Tvecs

#### Cam4

##### Rotation Matrix

$$\begin{bmatrix} 0.6417071 & 0.76682794 & -0.013670536 \\ -0.08663384 & 0.090185516 & 0.9921498 \\ 0.76204103 & -0.63548523 & 0.12430593 \end{bmatrix} \begin{bmatrix} -950.9641136239776 & 927.7630271222954 & 4034.533778276294 \end{bmatrix}$$

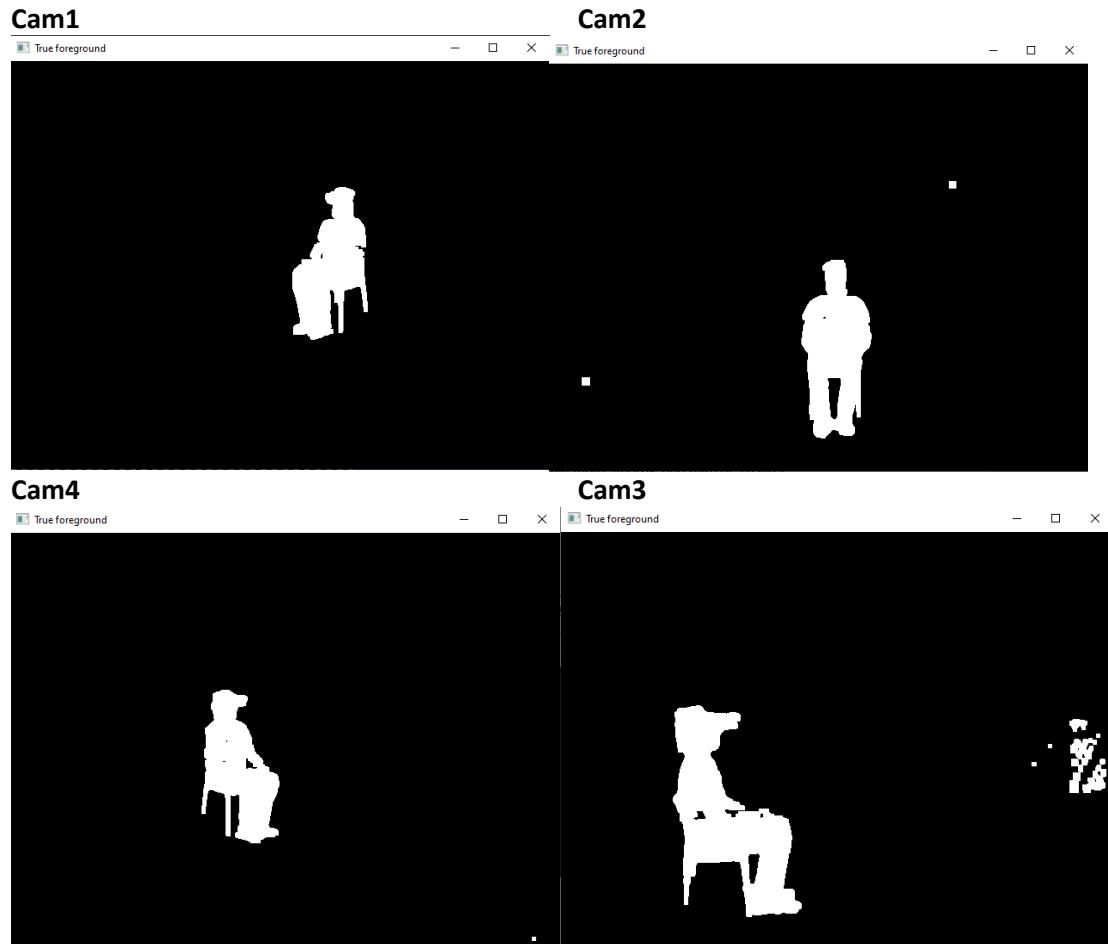
##### Tvecs

### 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.)

For the background, the first frame is taken from the video as it is, then each subsequent frame is added to it but the alpha value is increased between each iteration until there are no more frames left this is saved to a new image for each camera. During the live subtraction, we get the next frame via a counter then we convert it to HSV, blur it with gaussian blur and split it into 3 channels. The background image is loaded and also converted to HSV and split into 3 channels. The background subtraction is manual subtraction. First, the absdiff between each HSV channel on the precalculated background and the video frame is calculated. Second, the resulting **Mat** is thresholded on a

bruteforced value. If there is noise leftover, erode is used, otherwise dialite to fill the holes that erode mightve caused. Each channel is cleaned from any noise except the person then at the end the 3 channels are combined using a bitwise\_or operation.



Having noise on, on some images but not all is okay since those pixels won't match white color on all 4 cameras and won't be displayed.

```
threshold_h = 13  
threshold_s = 13  
threshold_v = 75
```

#### Choice tasks

(Indicate which ones you did, and how you did them; Approx. one third of a page.)

**Coloring the voxel models 10p:** The color of the voxel is determined by the average color value of the pixel from each view

#### 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](mailto:m.doyran@uu.nl).)

YouTube link: <https://youtu.be/hbFs1jyRzwM>