Computer Vision
and Geometry Lab

Computer Vision

Exercise Session 6 – Stereo matching

Assignment 6

- 3 Tasks:
  - Disparity computation
    - winner-takes-all
    - Graph-cut
  - Textured 3D model

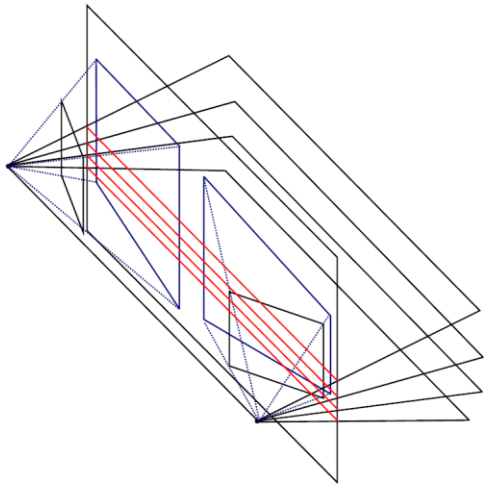
Stereo Setup

- Stereo setup

Left image

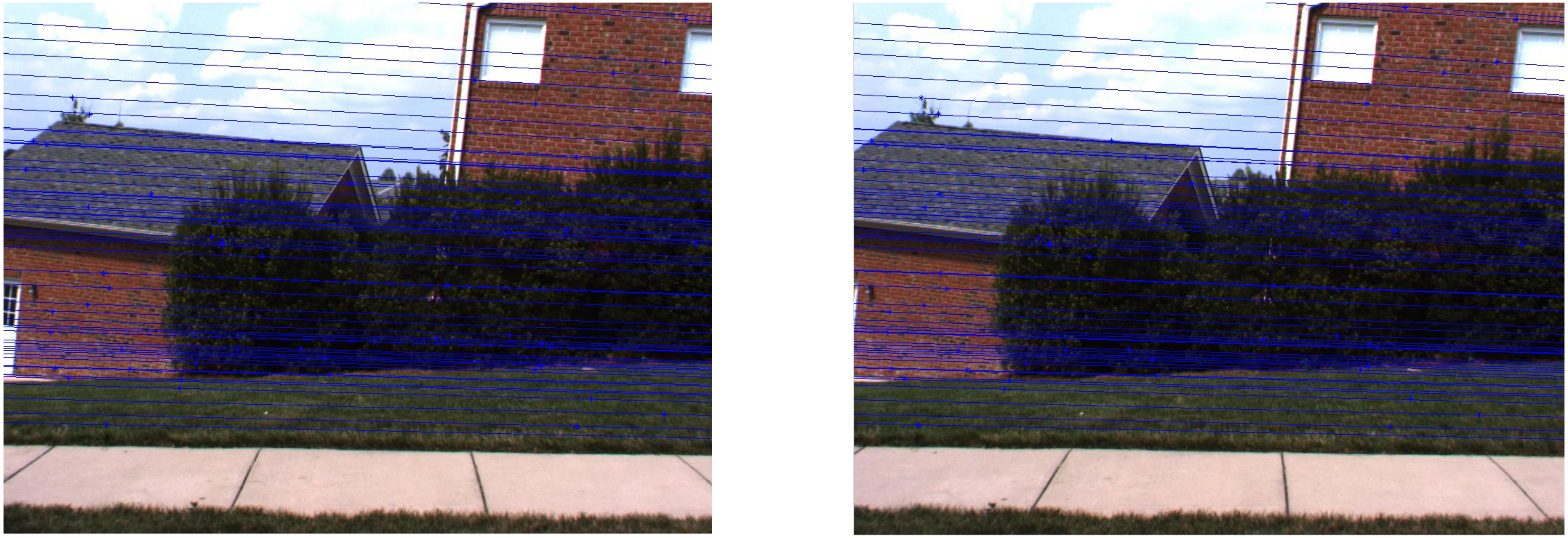
Right image

- Bring two views to standard stereo setup
  - Epipoles are at infinity
  - Epipolar lines are parallel



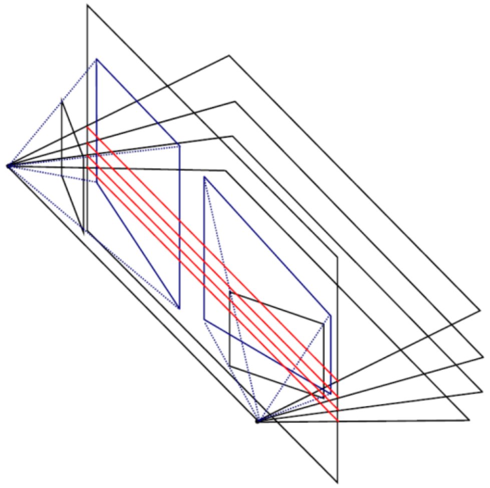
Planar rectification

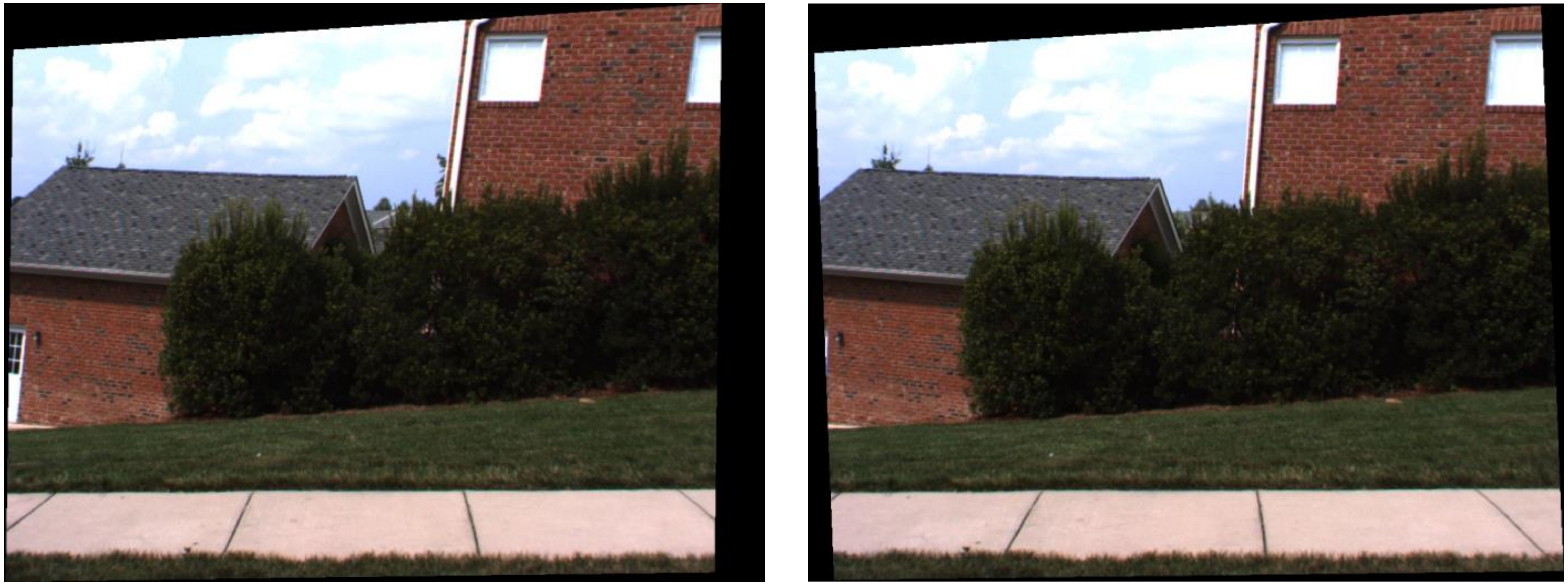
- Compute fundamental matrix
  - Use code from previous assignment or use code provided in the framework



Planar rectification

- Rectify images (code provided)





Disparity

- Find the offset d(x, y) of matching pixels

$$x' = x + d(x,y),\ y' = y$$

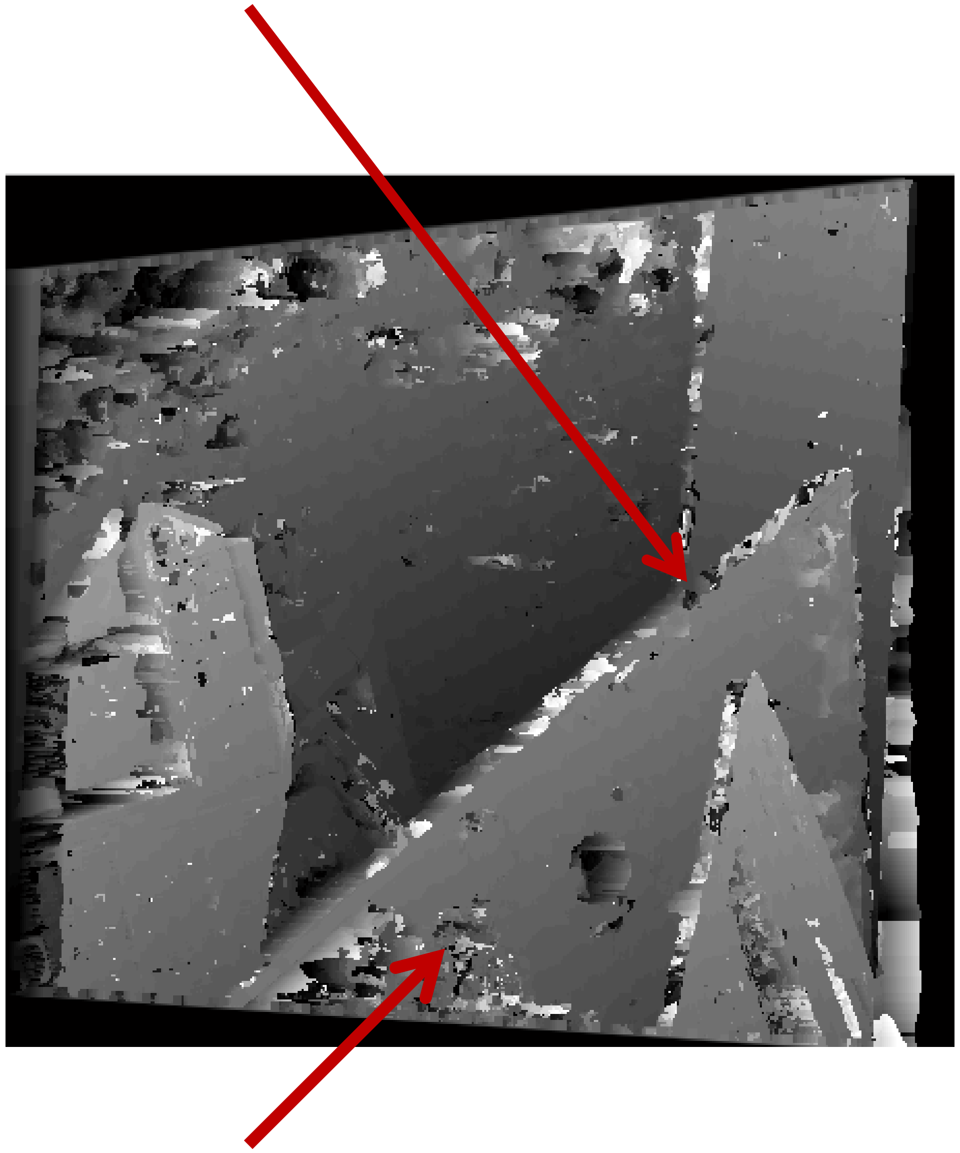
- Search algorithm (convert to gray scale *rgb2gray*)
  - For each pixel (x, y), for each disparity
    - SSD = 0
    - For each pixel (i, j) in window
      - SSD = SSD + (I1(x+i, y+j) - I2(x+i, y+j)).^2
    - Remember disparity with smallest SSD
- SLOW!

Disparity – faster version for Matlab

- For each disparity *d*
  - Shift entire image by *d* (code provided (*shiftImage*))
  - Compute image difference (SSD, SAD)
  - Convolve with box filter
    - Use conv2(..., ‘same’) and fspecial(‘average’,...)
  - Remember best disparity for each pixel
    - mask = Idiff < bestDiff
- Resize images if your stereo is taking too long

Disparity result

Depth discontinuities



Depth should be continuous

Disparity – Graph-Cut

- Stereo is a labeling problem
  - Assign each pixel the corresponding disparity (label)
  - Matching pixels should have similar intensities
  - Most nearby pixels should have similar disparities

$$f : P \to L$$

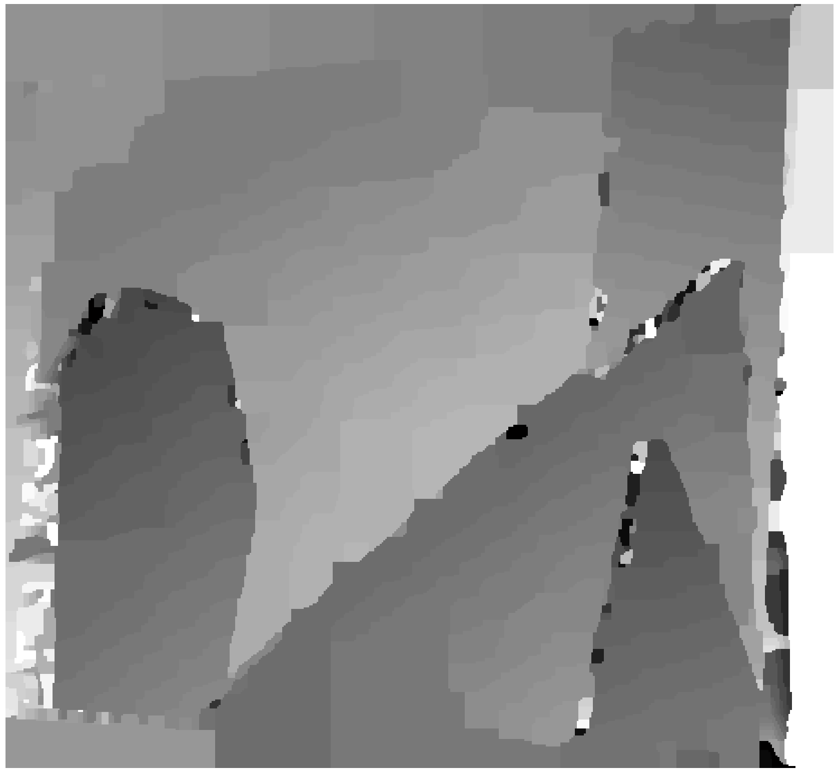
$$E(f) = E_{data}(f) + E_{smooth}(f) = \sum_{p \in P} D_p(f_p) + \sum_{p,q \in \mathcal{N}} S(f_p, f_q)$$

Disparity – Graph-Cut

- Familiarize yourself with the sample code
  - See the gc_example() file on color segmentation
- Adapt the code to compute the disparity
  - Change the data cost (Dc)
    - Compute for each pixel the SSD at each disparity
    - Store SSD values in a *m x n x r* matrix, where m x n is the image size and r is the number of disparities (labels)
  - The rest remains unchanged
- You may need to change the weighting of the terms

Graph-Cut - Results

- Result with simple cost function

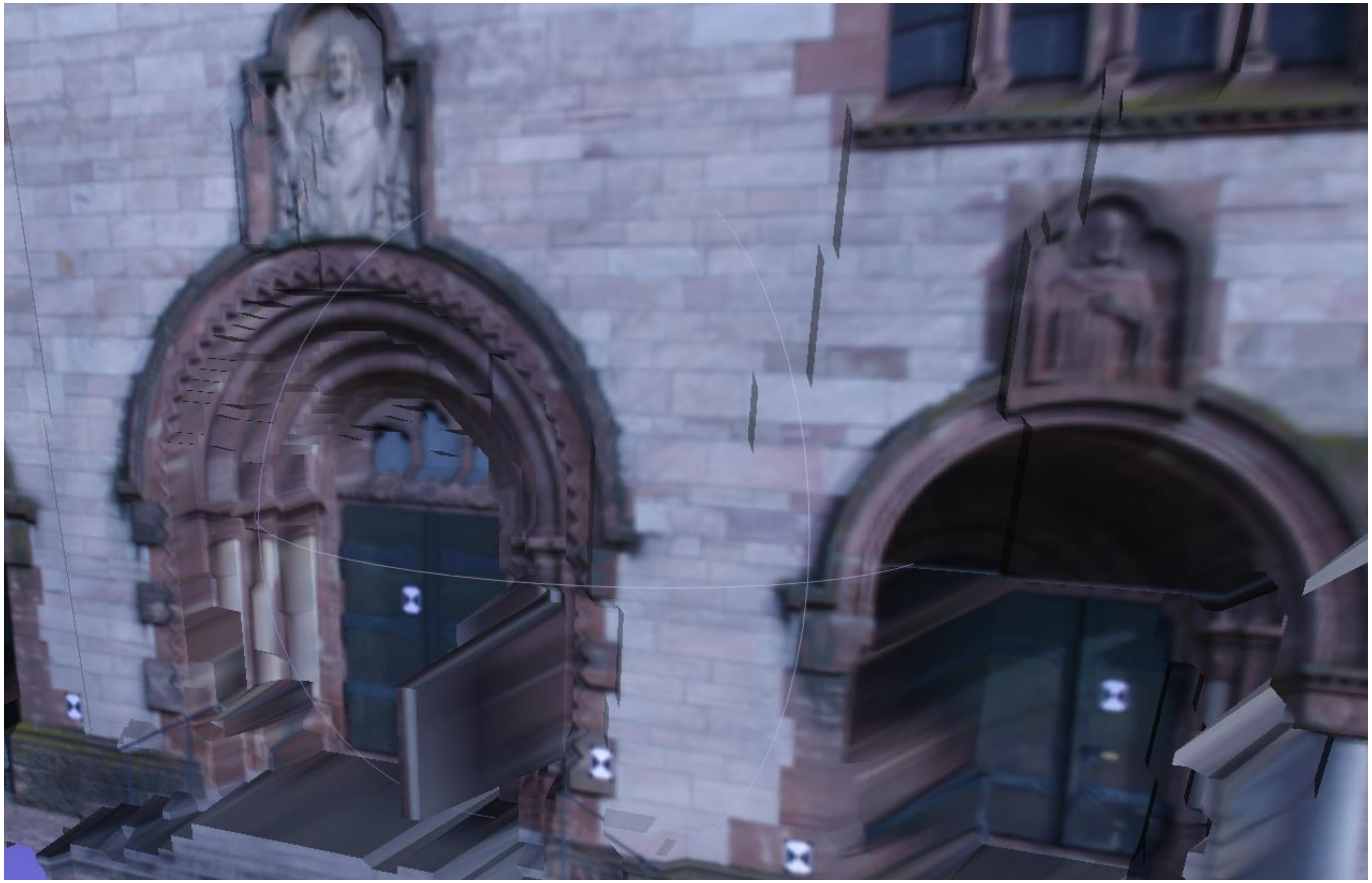


Textured 3D model

- Image pairs and camera parameters
- For each pixel find the corresponding 3D point
  - Disparity maps
  - Camera parameters
- Generate textured 3D model (code provided)
  - .obj-file
  - .mtl-file
  - Image file

Put everything in the same folder, load .obj-file with Meshlab.

Textured 3D model



Framework

- Functions that need to be completed/implemented (you can add functions, of course):
  - stereoDisparity.m
  - diffsGC.m
  - gcDisparity.m
  - generatePointCloudFromDisps.m
  - exercise6.m