Day 6 - Weekly Project (17. Feb)

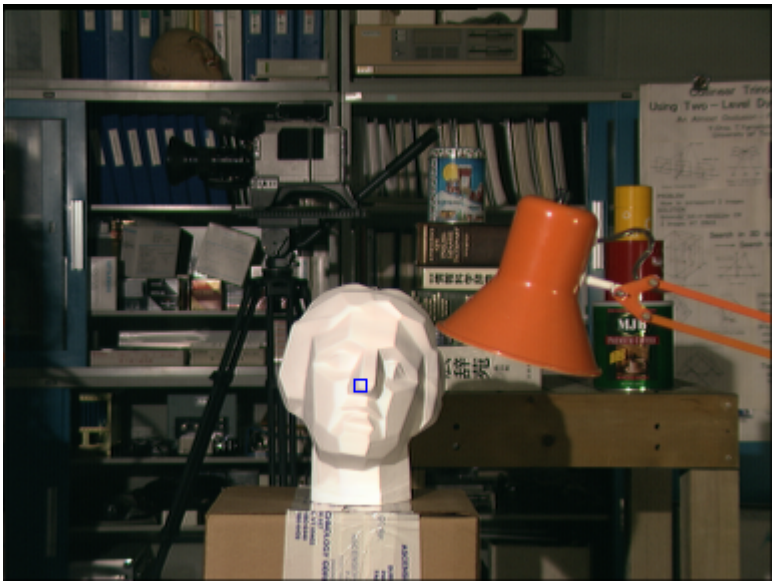
Implementation of stereo block matching

This week you'll be implementing your own block matching algorithm for stereo depth reconstruction. Thus you are ***not*** allowed to use *cv2.matchTemplate*, *cv2.stereoBM* or similar functions from opencv or other pre-existing libraries. You can find all necessary image-files in the attached zip-file.

It is recommended you use the following three intermediate steps before going directly to a full implementation:

Intermediate step 1:

Implement a function that takes two equal size images and calculates the sum of absolute differences. The two equal-sized images are *nose_left.png* (used as the template) and *nose_right.png*.

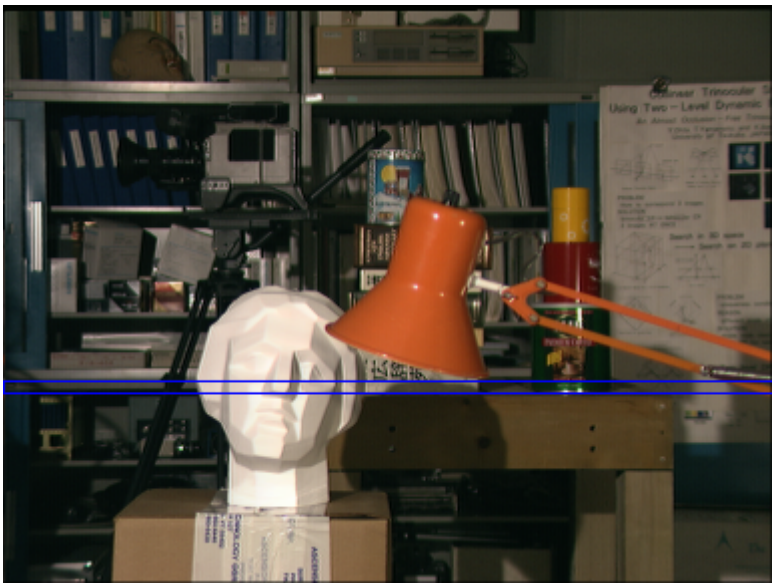


Intermediate step 2:

Using the function you just created, find which of the three images *nose1.png*, *nose2.png*, *nose3.png* gives the best match to *nose_left.png*?

Intermediate step 3:

Since these images are rectified, we can approximately assume that a pixel found in row **x** in the left image also lives in row **x** in the right image. *nose_span.png* contains the entire row containing the nose of the sculpture. The height of the span is 7 - the same height as your template. Write a function that loops through the image from left to right and computes the correlation using your function from step 1 to find where in the span the nose is located.



Full template matching implementation (Challenge):

Write a template matching function that iteratively takes a 7x7 subpart (numpy calls this slicing) of *tsukuba_left.png* and uses your function from step 3 to apply it to the corresponding row in *tsukuba_right.png* and stores the lowest/best matching disparity value for that template. Continue until all possible templates for a row has been used before moving down to the next row in the image.

0 % 0 of 1 topics complete

Weekly project

Zip Compressed File