COMS 3251 SU21 HW2

Due: Thurs July 22, 2021 at 11:59pm

This homework is to be done **alone**. No late homeworks are allowed. To receive credit, a typesetted copy of the homework pdf must be uploaded to Gradescope by the due date. You must show your work to receive full credit. Discussing possible solutions for homework questions is encouraged on course discussion board and with your peers, but you must write your own individual solutions and **not** share your written work/code. You must cite all resources (including online material, books, articlezs, help taken from specific individuals, etc.) you used to complete your work.

1 Numerical Problems

1. Let Q be an $n \times n$ matrix with columns $q_1, \dots, q_n$. Suppose Q^{T} equals Q^{-1}.
   - Show that each q_i is unit length.
   - Show that the vectors q_i's are orthogonal to each other.
   - Find a 2×2 example Q with first entry $q_{11} = \cos\theta$. (for some fixed θ)
2. Let A and B be $n \times n$ matrices with $AB = 0$. Give a proof or counterexample for each of the following.
   - Either $A = 0$ or $B = 0$ (or both).
   - $BA = 0$.
   - If B is invertible then $A = 0$.
   - There is a vector $v \neq 0$ such that $BAv = 0$.
3. Consider the system of linear equations
   $$\begin{aligned} kx + y + z &= 1 \\ x + ky + z &= 1 \\ x + y + kz &= 1. \end{aligned}$$
   For what value(s) of k does this have (i) a unique solution? (ii), no solution? (iii) infinitely many solutions? (Justify your assertions).
4. Find the triangular matrix R that can reduce the Pascal matrix. That is, find triangular R such that
   $$R \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 1 & 3 & 3 & 1 & 0 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 1 & 3 & 3 & 1 \end{bmatrix}$$

2 Theory

1. One property of invertible matrices is that the matrix equation $Ax = 0$ only has a trivial solution. That is, A is invertible if and only if the equation $Ax = 0$ only solvable for $x = 0$ (and no other vector). Using this fact, prove that for any $n \times n$ matrix A, if $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$, then A is necessarily invertible.

2. In this exercise we will prove the theorem:

   Let A be the adjacency matrix for a graph G with vertices $v_1, \ldots, v_n$, and let $l \in \mathbb{N}$. Then the number of walks of length l from v_i to v_j is $(A^l)_{ij}$.

   (a) Let $p(i, j, l)$ denote the number of walks of length l in G from v_i to v_j. Prove that for all $i, j = 1, \ldots, n$ and $l \geq 1$,

   $$p(i, j, l) = \sum_{k=1}^{n} p(i, k, l-1)p(k, j, 1).$$

   (Hint: Part of the trick is to parse this formula appropriately.)

   (b) Prove the theorem by induction on l, using the result from part (a).

3 Applications & Programming

One of the lasting markers of evolution on the human body is the parallel and (approximately) planar placement of our eyes. This setting of ”cameras”, which is optimized for depth reasoning and an ideal sense of visual perception in the 3D world, is known as **stereo**. In this problem, we will use pictures taken by a camera in two different locations, the poses of which are different only by some small translation, to simulate the setup of our eyes and observe that with some linear algebra, we can reconcile the 2D information from the two pictures to get a pixel-wise estimate for depth. In other words, we will create a 3D representation solely from two related 2D datapoints.

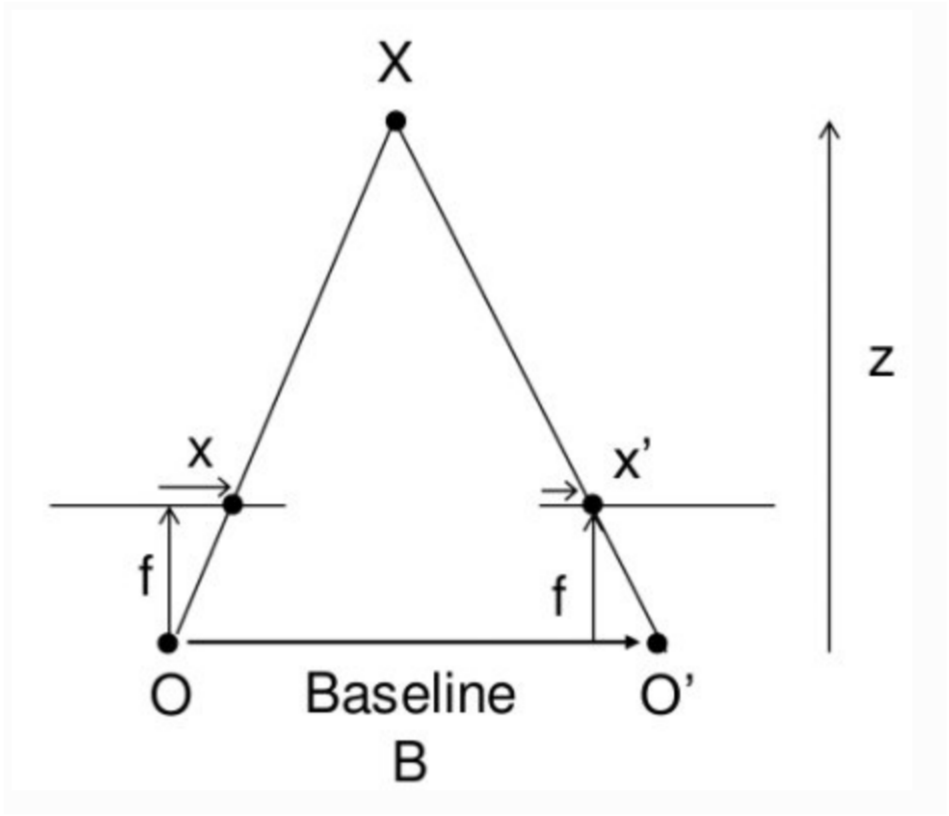
Preliminaries:

1. For this homework, you should build on the template and use the hyperparameters from **stereo.ipynb** but you are free to fill in your code elsewhere, should you feel comfortable. Download the provided image files **left.png** and **right.png** and read the same into your code (if you are doing this on Colab, the instructions on how to upload the same for use on Colab have been provided to you on **stereo.ipynb**). The images are from the “Cones” dataset created by Daniel Scharstein, Alexander Vandenberg-Rodes, and Rick Szeliski. You will notice that the two images almost appear visually indistinctive to each other: this is because they have been photographed by a camera that has ever so slightly been translated. We will see how, even from such a small translation, we are able to infer a surprising amount of information regarding the 3D structure of the scene

2. Read the attached **primer.pdf** (from the homework announcement) on image formation with a standard pin-hole model camera if you are unfamiliar with it. You will need the basics in it to answer the early parts of this question.

Question:

1. Consider the image below (**source:** OpenCV Tutorials) which shows a setup of two cameras picturing the point **X** a small distance from each other.



The cameras are separated by distance B and both have the same focal length f. The points **x** and **x'** correspond to the pixel locations of the point X in the respective image plane of each camera, and z is the depth information we wish to estimate for the (x, x') point correspondence. Use similar triangles (as in the primer) to show that z can be expressed as $C.(x-x')^{-1}$ where C is some scalar which can be calculated from known quantities or camera parameters. This means that it is enough to estimate $(x - x')$ at all points of correspondences: the output of this process is called the **disparity map**. Then, the **depth map** will simply be a function of the **disparity map**.

2. However, given two images determining the corresponding (x, x') pixels (the pixels in the images that photograph the same 3D point) is not always an easy problem, even in a planar translation case. Here, you will implement a simple algorithm for this problem, which treats windows in the given images as matrices and applies some measure of distance to establish correspondence. Implement **Algorithm 1** and use it to find the (x, x') correspondences in the given region (**Note**: all arrays used are 1-indexed, but this wont be the case in your code). Then use these correspondences to find the disparity map. Repeat this process with 3 different notions of a distance between matrices of your choice. Some examples to consider, given blocks A and B, include:

   (a) the sum of absolute differences (SAD): $\sum_{ij}(|A_{ij} - B_{ij}|)$

(b) the sum of squared differences (SSD): $\sum_{ij}(|A_{ij} - B_{ij}|)^2$

Use the provided template in **stereo.ipynb** to organize your code (and make use of the default block size and other parameters). Include in your homework solutions report:

(a) A description of each of the metrics you chose (and some text explaining your choice). You should label each metric as **1, 2 and 3** for further reference.

(b) For each of the three metrics, indicate clearly the functions in your code that given two matrices A and B, calculate the metric of the difference between them.

(c) For each chosen metric, include an image of the disparity map generated with it (labelled **1.png**, **2.png** and **3.png** respectively) in your homework report.

(d) Additionally, write a paragraph in your report explaining what you observe and analysis of the changes in the disparity map using your choice of metrics.

In your code, you may **not** use any library implementations (such as **OpenCV** and **Pillow**) **except** for reading in the images, converting to grayscale, and writing and visualizing the images. Everything else must be your own implementation. You may use $OpenCV$'s $StereoBM$ method to $check$ your answer, but you should **not** use it for your own implementation, and be aware that these methods are highly optimized for smoothness and performance so your image may not look exactly the same.

Algorithm 1: Block Matching (Parallel Image Planes)

```
**Data:**
$I_1$: Image of a scene, size $(h, w)$
$I_2$: Image of the same scene taken under translation of camera, size $(h, w)$
$r$: Disparity range (furthest distance between corresponding blocks)
$b$: Block size (size of the sub-region of image tested for disparity at one time, must be odd)
$m$: $((M_1, M_2) \mapsto \mathbb{R})$ a metric which takes in equal sized subregions of the image and
outputs a value indicating the similarity
**Result:**
$d$: Disparity map with pixel-wise correspondence
$I_1$ = **checkOrConvertGrayScale**($I_1$) ;
$I_2$ = **checkOrConvertGrayScale**($I_2$) ;
$d$ = **init_zero_2darray**(size=$(h, w)$) ;
$hb$ = $(b - 1)/2$
;
**for** $m$=1: $h$ **do**
    $min_m = \max(1, m - hb)$ ;
    $max_m = \min(h, m + hb)$ ;
    **for** $n$=1: $w$ **do**
        $min_n = \max(1, n - hb)$ ;
        $max_n = \min(w, n + hb)$ ;
        $min_d = \max(-r, 1 - min_n)$ ;
        $max_d = \min(r, w - max_n)$ ;
        template $= I_2(min_m : max_m, min_n : max_n)$ ;
        $numBlocks = max_d - min_d + 1$ ;
        diff = **init_zero_2darray**(size=$(numBlocks)$) ;
        **for** $i = min_d : max_d$ **do**
            block = $I_1(min_m : max_m, (min_n + i) : (max_n + i))$ ;
            blockIdx = $i - min_d + 1$ ;
            diff(blockIdx) = $m$(template, block)
        **end**
        sortedIdx = **argsort**(diff) ;
        bestIdx = sortedIdx[1] ;
        bestDisparity = $min_d$+ bestIdx - 1 ;
        $d(m, n) =$ bestDisparity; ;
    **end**
**end**
**return** $d$
```

3. Now, use your own camera or mobile device to click two pictures of any scene of your choice and repeat **Step 2**. You may use any of the three metrics you have used above for this part. Make sure you only translate your camera ever so slightly in the process of clicking pictures, as the above method will only work well under a small translation of the camera (and no rotation). Include three images in your homework solutions report: your **left image** titled **my_left.png**, your **right image** titled **my_right.png** and your disparity map estimation from running **Algorithm 1**, titled **my_stereo.png**

Submit all your code for parts **2** and **3** in a zip file to Gradescope and remember to follow the instructions to include the required analysis and images in your written work. You will be required to follow the typical code submission guidelines for file formats and README in preparing your zip file.