Introduction

In the data folder for the assignment, you are given five images, three rectified RGB images of a scene and two ground-truth disparity maps of view1 and view 5. In this homework you will be required to estimate the disparity maps of the view1 and view5 images using 2 methods: (1) basic block matching and (2) dynamic programming. You will also be required to implement *view interpolation* which is an interesting application of stereo vision.

Unlike the previous homeworks which emphasized code implementation, this homework is graded mainly on your evaluation framework and report. Your first two programs should output a disparity map, where each pixel represents the shift between the left and right images. If you know the length of baseline and the focal length of cameras, you can reconstruct depth, from disparity; but the focus of this homework is only limited to the computation of disparity d.

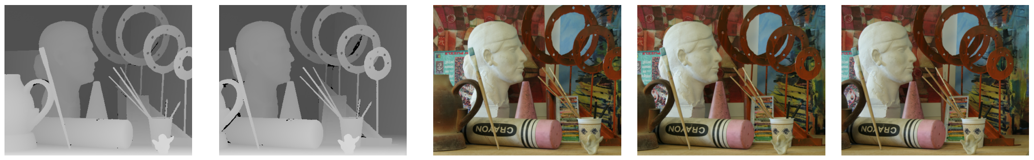


Figure 1: Two ground-truth disparity maps and three rectified RGB images

Requirements

You should perform this assignment in Matlab. It is due on **Saturday Nov 15th by 5pm**. You are strongly encouraged to start the assignment early and don't be afraid to ask for help from either the TA or the Instructor. You are also welcome to ask questions and have discussions about the homework on the course piazza but please do not post your solutions or any closely related material. If there are parts of the assignment that are not clear to you, or if you come across an error or bug please don't hesitate to contact the TAs or the Instructor. Chances are that other students are also encountering similar issues.

You are allowed to collaborate with other students as far as discussing ideas and possible solutions. However you are required to code the solution yourself. Copying others' code and changing all the variable names is not permitted! You are not allowed to use solutions from similar assignments in courses from other institutions, or those found elsewhere on the web. If you access such solutions YOU MUST refer to them in your submission write-up and points will be deducted accordingly. Your solutions should be uploaded to the CSE server using the CSE_SUBMIT command (There is a tutorial on how to use CSE_SUBMIT) on the course piazza page.

The instructions, data and starting code is provided in the zipped file **homework3.zip** which can be downloaded from Piazza. Your submitted zipped file for this assignment should be named **UBPersonNumber_hw3.zip**. Failure to follow this naming convention will result in your file not being being picked up by the grading script. Your zipped file should contain: (i) a PDF file named UBPersonNumber_hw3.pdf with your report, showing output images, explanatory text and the output of your evaluation where appropriate; (ii) the source code used to generate the solutions (with code comments). Please include a working runfile for this assignment **(runfile_hw3.m)**. You do not need to include any actual images with your final submission (but you should have images in your report). For grading, we will be testing your code on our own separate set of evaluation images and disparity maps.

Problem 1. Disparity Estimation using Block Matching (20 Points)
Use the basic block matching techniques from Programming Assignment 1 to estimate disparity between the RGB images view1 and view 5. You will generate two disparity maps, one for view1 and the other for view2.

(i) Try different block sizes and generate disparity maps. Report the best estimated disparity map and the calculated MSE (Mean Squared Error) with respect to the given ground truth. Report the results only for image set in Data folder. For images of **I** and **K** of dimension $m \times n$, MSE is defined as:

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \quad (1)$$

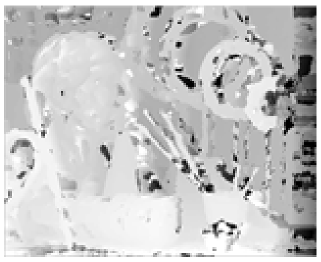


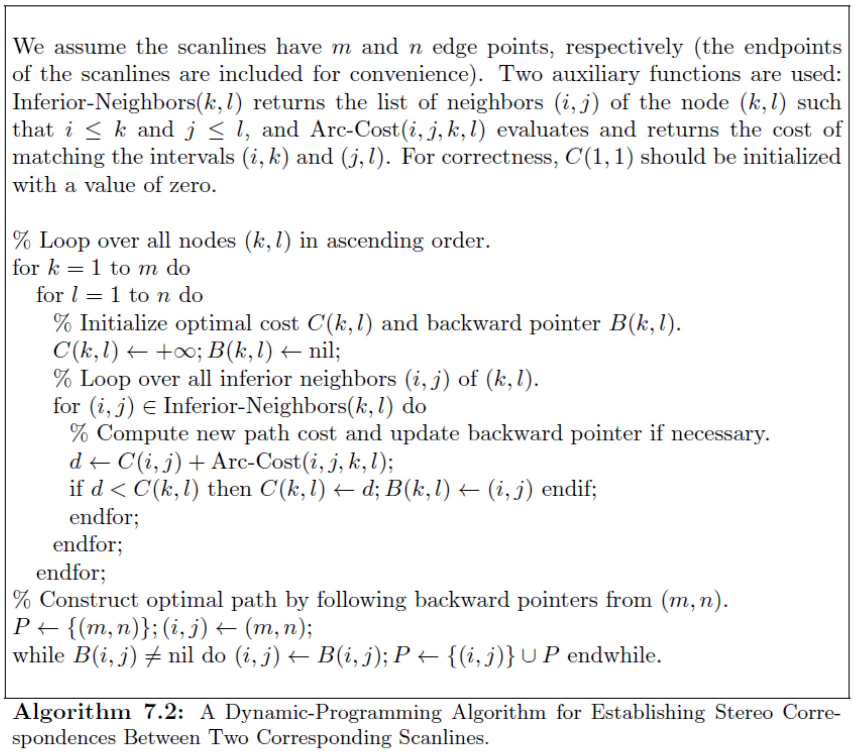
Figure 2: Example of the disparity map generated from SSD matching

(ii) Eliminate the noisy disparity estimates using consistency checking. Due to occlusion, we may have missing regions in the stereo image pair. This will result in random matches since a correct match does not exist. After obtaining your disparity maps for both the images, use back-projection to do a consistency check. Ideally, if the disparity maps are accurate, if you pick a pixel on left scan line and find its position on the right scan line, you would find the disparity value at that pixel to be consistent i.e. that pixel should back-project to the original pixel you chose on left scan line. Note: Implementing the consistency check requires that you have the disparity maps available for both the images.

Mark all the inconsistent pixel disparities as **NaN** (IEEE®arithmetic representation for Not-a-Number). Report the disparity map and re-evaluate the MSE with respect to the ground truth disparities by masking the NaN pixels from the computation.

Problem 2. Disparity Estimation using Dynamic Programming (30 Points)
The intention of this section is to give an understanding on how dynamic programming can be used to improve the disparity map and its implementation. You can use single or a combination of multiple constraints to estimate the disparity map. This section will

be graded based on correctness of your code and comparing your results with a benchmark disparity that is generated by posing ordering constraint on edges over scan lines as discussed in the Forsth and Ponce textbook.



We assume the scanlines have m and n edge points, respectively (the endpoints of the scanlines are included for convenience). Two auxiliary functions are used: Inferior-Neighbors(k, l) returns the list of neighbors (i, j) of the node (k, l) such that $i \leq k$ and $j \leq l$, and Arc-Cost(i, j, k, l) evaluates and returns the cost of matching the intervals (i, k) and (j, l). For correctness, $C(1, 1)$ should be initialized with a value of zero.

```
% Loop over all nodes $(k, l)$ in ascending order.
for $k = 1$ to $m$ do
  for $l = 1$ to $n$ do
    % Initialize optimal cost $C(k, l)$ and backward pointer $B(k, l)$.
    $C(k, l) \leftarrow +\infty; B(k, l) \leftarrow$ nil;
    % Loop over all inferior neighbors $(i, j)$ of $(k, l)$.
    for $(i, j) \in$ Inferior-Neighbors$(k, l)$ do
      % Compute new path cost and update backward pointer if necessary.
      $d \leftarrow C(i, j) +$ Arc-Cost$(i, j, k, l)$;
      if $d < C(k, l)$ then $C(k, l) \leftarrow d; B(k, l) \leftarrow (i, j)$ endif;
      endfor;
    endfor;
  endfor;
% Construct optimal path by following backward pointers from $(m, n)$.
$P \leftarrow \{(m, n)\}; (i, j) \leftarrow (m, n)$;
while $B(i, j) \neq$ nil do $(i, j) \leftarrow B(i, j); P \leftarrow \{(i, j)\} \cup P$ endwhile.
```

Algorithm 7.2: A Dynamic-Programming Algorithm for Establishing Stereo Correspondences Between Two Corresponding Scanlines.

Problem 3. Stereo Application - View Interpolation (30 Points)

View3 is what you would see if you were to place another camera exactly at the midpoint of the baseline of the two cameras that captured view1 and view5. Your task now is to generate view3 by using view1 and view5 images and their respective ground truth disparity maps. The synthesized view will probably have holes which can be marked as NaN. Compute the MSE between the synthesized view and view3 and report your result.

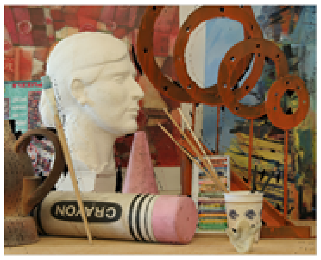


Figure 3: Example of what your interpolated view3 should look like.

Problem 4. Evaluation and Report (20 Points)

(a) You are provided with two folders - Data and Evaluation. You should implement your algorithms on the Data folder, working towards getting the best result on the images found there. Then try applying the same code on the images found in the Evaluation folder. What changes did you have to make to get your implementation working and generate disparity maps? Report the disparity maps and MSE for the images in the Evaluation folder using your basic block matching code without NaN masking.

(b) Discuss a technique for matching textureless regions in stereo images. Which of the evaluation images have the least texture and how did this affect the MSE results? What changes can you make to your implementation to get a better disparity estimate in textureless regions?

(c) Now plot a graph for each image set (4 graphs in total) with window size on x-axis and MSE on y-axis.

(d) Computer Vision algorithms can be computationally intensive. But MATLAB has been optimized to work on matrices so that functions like *sum(sum(X))* over a 2D image works more efficiently than performing a sum over all the pixels using for-loops. So, what in-built MATLAB functions have you used to make your program more efficient?

(e) Give a brief run-time analysis of your programs (see page - basic block matching and dynamic programming. How does the run time vary with image size and window size. Lastly, briefly mention two algorithms/articles from literature that have advanced the research in stereo disparity estimation and how they differ from our algorithms here.

You should turn in both your code and report discussing your implementations and results to get full credit.