# Stereo

EECE 5639: Computer Vision

Miquel Alberti Binimelis

Kevin Wu

## Introduction

In this project, we will explore the capabilities of stereo vision. Given a pair of images taken of the same scene from different locations, with stereo vision we can find correspondences between these images in an efficient way by constraining the search to just the epipolar line. This epipolar line can be computed from what's called the fundamental matrix, which relates corresponding points from each image by the equation:

$$p_r^T F p_l = 0$$

Where $p_r$ and $p_l$ are the corresponding points from the right and left images, using pixel homogeneous coordinates, and $F$ is the 3x3 fundamental matrix.

## General approach

The algorithm takes as input two images of the same scene from different perspectives (with some overlapping) and outputs three images, representing the vertical, horizontal, and vector disparity (with the color encoded by HUE). The disparity is the distance between the image coordinates of two corresponding points. We can break the algorithm in the following steps:

1. Detect features on each image using the Harris Corner Detector.
2. Find correspondences between these features using NCC to compare patches on both images.
3. Compute the fundamental matrix that explains the epipolar geometry between the two images, using the previous correspondences. However, we need to use RANSAC because outliers could have a really negative impact on the final result.
4. Compute the dense disparity map using the fundamental matrix obtained in the previous step to reduce the search space.

We will explain each of these steps after the next section.

CLARIFICATIONS ABOUT THE CODE

The first 3 steps are implemented using Python, but due to the requirements and plottings of the 4th step, we thought it was more convenient to use Matlab instead.

The way the Python code is organized is that there is a main.py that contains what's more similar to the general approach, and from there, we do the calls to the other Python files that contain each of the algorithms we are using. Those files are:

| *Filename* | *Description* |
|---|---|
| harris_corner_detector.py | Contains the code to find the features of a given image. |
| NCC.py | Given two images and their corresponding features, here we find the correspondences using NCC. |
| fundamental_matrix_RANSAC.py | Here we compute the fundamental matrix using |

| | |
|---|---|
| | the correspondences obtained in the previous script, using the RANSAC algorithm to remove outliers. |

Here are the MATLAB scripts used in the analysis of part 3:

| *Filename* | *Description* |
|---|---|
| runPart3.m | Main driver script that calls other functions and handles post-processing of feature matching for each pixel in A, over each corresponding line in B |
| computeNxc.m | Function that returns normalized cross-correlation using conv2() |
| searchAlongEpipolar.m | Function computing corresponding epipolar line, searching along it, then returning best matches |
| visualizeEpipolar.m | Function to generate side-by-side representation of a point in A to its corresponding epipolar line in B |

There are also a plethora of other MATLAB scripts used in testing and replicating the first two parts, but the ones listed above are the minimum scripts used to generate results contained in this report.

## Features detection: Harris Corner Detector

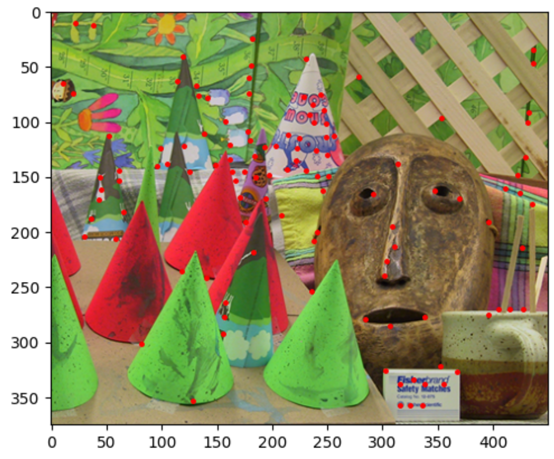
We first compute the gradients of the given images using a Sobel mask. The Sobel mask considers a 3x3 neighborhood to reduce the effect of noise, but gives more importance to the center pixel. We also apply a Gaussian smoothing mask, because gradients are very sensitive to noise.

Now we have all we need to compute the C matrix, considering neighborhoods of 7x7. However, instead of computing the eigenvalues, it is far more efficient to use the Harris R function instead. Finally, we extract the local maximums in R that are above a threshold (non-maximum suppression), which will be the coordinates of the desired features.

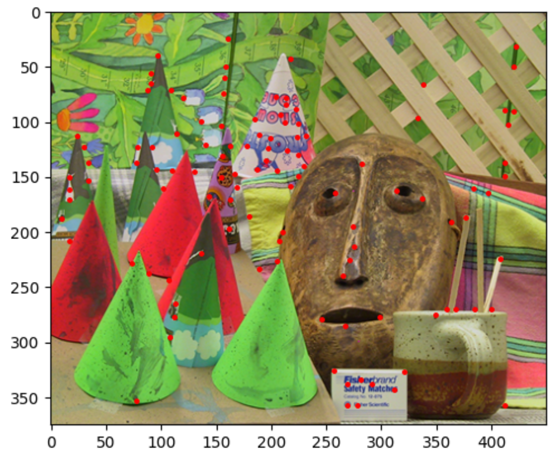
The parameters used are the following:

- Gaussian blurring uses $\sigma = 0.7$, and the window size is tied to $5\sigma$.
- The constant that we use for the Harris R function is $k = 0.05$. The higher it is, the more precision but less recall we get, but it should always be in a range of 0.04 - 0.06.
- We wanted the threshold value to depend on the values in the image rather than on some "magic number". What worked for us was considering the sum of the mean and the standard deviations of the matrix with all the R values. We also considered using the maximum value of R as a reference to compute the threshold, but a big outlier could impact the measure really badly in that case.

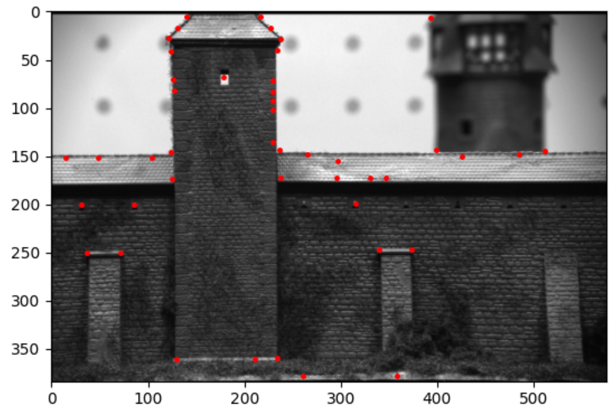
## Results



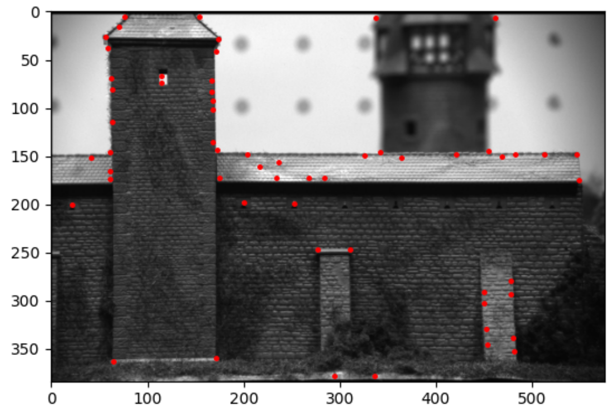
Features in input1\image-3.jpeg



Features in input1\image-4.jpeg



Features in input2/cast-left-1.jpg



Features in input2/cast-right-1.jpg

## Find correspondences: NCC

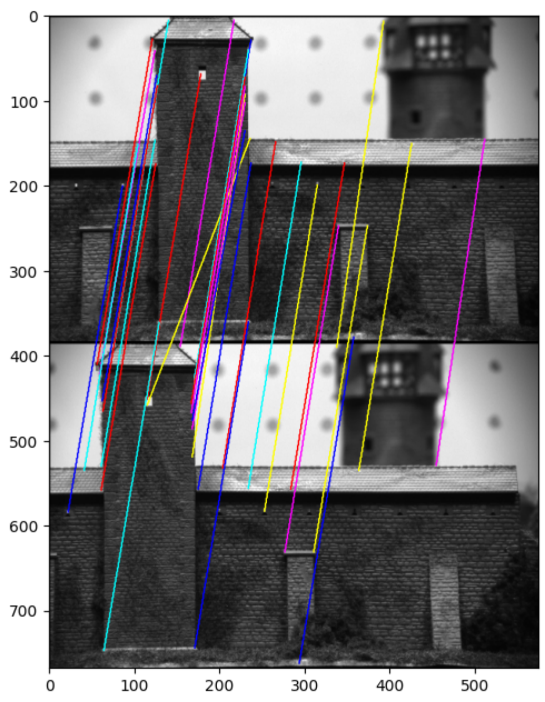
For each previously found feature, we consider a centered patch around it and compare it against all the other patches of the corresponding features in the other image. The comparison is done by computing the Normalized Cross Correlation: the bigger the NCC the more similar those patches are.

We end up with a bunch of good matches, but there can be points that belong to more than one good match. That's why in the end we remove duplicates by comparing NCCs and choosing the best ones for each duplicate.

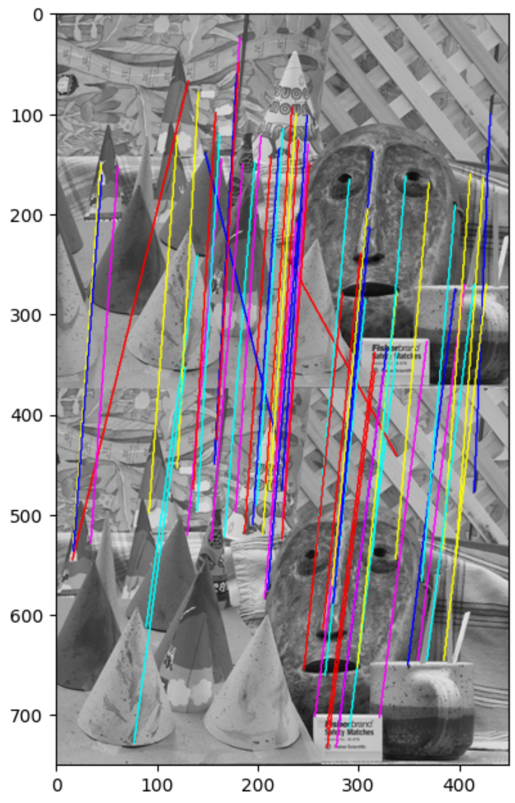
The parameters used are the following:

- The size of the patch is set to 5x5. The pros of using a small patch are the number of computations that it saves, but it has to be big enough to capture enough detail.
- The threshold to determine a good match is an NCC of 0.8. We still have some outliers, but they will be removed in the next step using RANSAC.

**Results**



input2 set of images
32 initial correspondances



input1 set of images
57 initial correspondences

## Estimate the *Fundamental Matrix* using RANSAC

Given a list of matches between features from the previous step, the goal here is to compute the fundamental matrix that capsules the epipolar geometry of the two images. Finding the matrix of the transformation is just a matter of setting a system of equations, writing it in matrix form as a homogeneous system, and using the SVD decomposition to obtain the result.

Another thing we have to take into account is that the fundamental matrix should have rank 2, but the previous computation could have rank 3 due to numerical issues. This is actually easy to fix, we can just compute the SVD of the fundamental matrix and set the smallest singular value to 0.

The fundamental matrix is a 3x3 matrix with 8 degrees of freedom, which means that we need 8 correspondences to solve the system of equations and get its coefficients. However, it is recommended to use much more to improve the accuracy of the calibration.

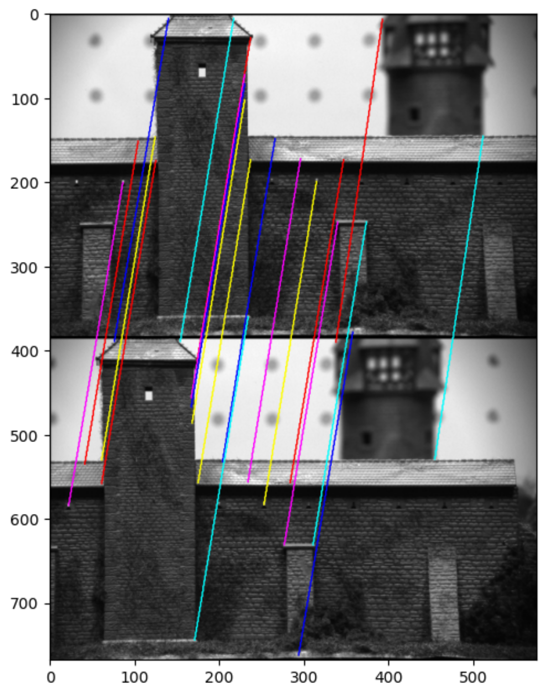
The problem is that our correspondences could have outliers, and that can have a really negative impact on the final computation. That's why we use RANSAC to find the inliers, and then we compute the fundamental matrix with **all** of them.

The parameters used are the following:

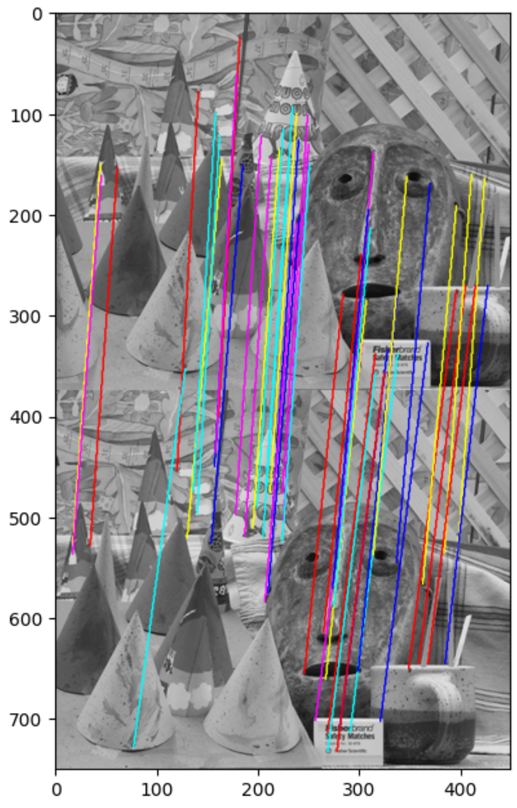
- To compute the fundamental matrix at each iteration of RANSAC, we use 8 correspondences which is the minimum needed to determine the fundamental matrix.
- To determine if a point is an inlier or not, we set the threshold of the distance from the point to the epipolar line to be lower than 0.001.
- The maximum number of iterations of RANSAC is set to 100. It is more than enough to obtain a good set of inliers with our correspondences because we never reach it in practice: we have set that if in one iteration we classify as inliers more than the 70% of the initial correspondences, it is good enough to assume that we have obtained the correct fundamental matrix (we will still compute it one more time using all the inliers).

## Results

These are the correspondences of the inliers we obtain after RANSAC:



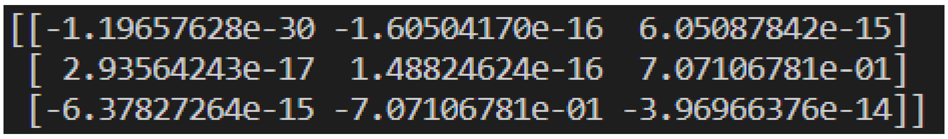
input2 set of images
21 inliers



input1 set of images
48 inliers

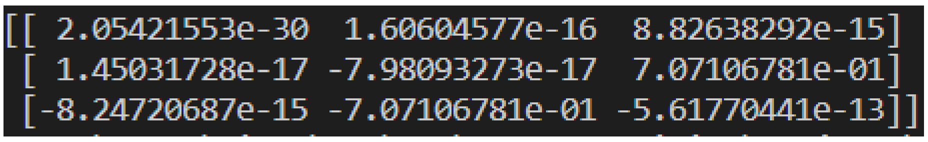
Judging by the correspondences we are getting, it looks like there is just a horizontal translation between both images. We can confirm that assumption by looking at the final fundamental matrix we are obtaining:

For input1 set:



```
[[-1.19657628e-30 -1.60504170e-16  6.05087842e-15]
 [ 2.93564243e-17  1.48824624e-16  7.07106781e-01]
 [-6.37827264e-15 -7.07106781e-01 -3.96966376e-14]]
```

For input2 set:



```
[[ 2.05421553e-30  1.60604577e-16  8.82638292e-15]
 [ 1.45031728e-17 -7.98093273e-17  7.07106781e-01]
 [-8.24720687e-15 -7.07106781e-01 -5.61770441e-13]]
```

In both cases, the fundamental matrix is approximately the same. Rounding to the 5 most significant decimals, it looks like:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0.70711 \\ 0 & -0.70711 & 0 \end{bmatrix}$$

Considering the equation $p_r^T F p_l = 0$, if we compute the product, we obtain the expression:

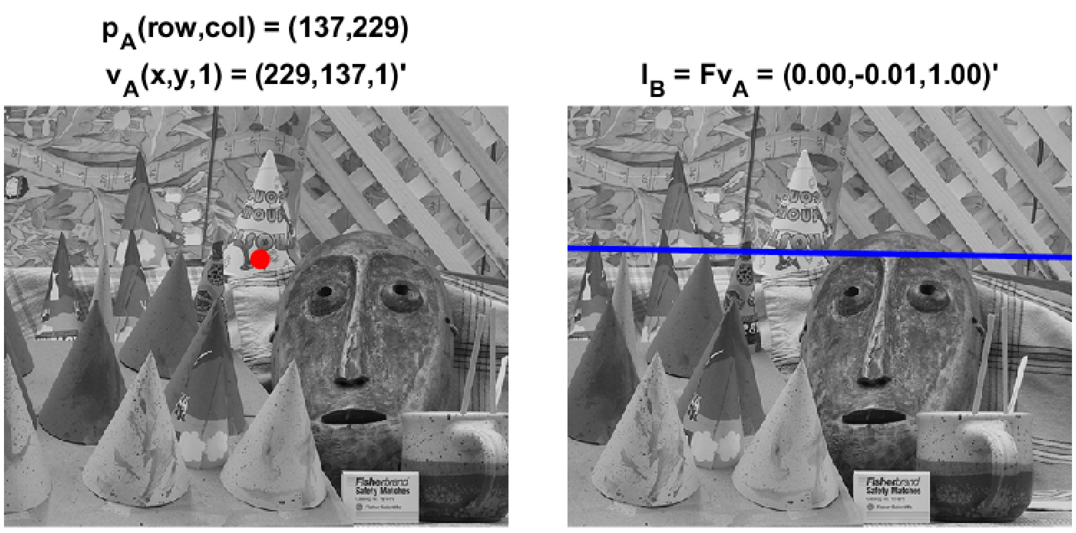
$$\begin{bmatrix} x_r & y_r & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0.70711 \\ 0 & -0.70711 & 0 \end{bmatrix} \begin{bmatrix} x_l \\ y_l \\ 1 \end{bmatrix} = 0.70711 y_r - 0.70711 y_l = 0$$

Which implies that:

$$y_r = y_l$$

And so the epipolar lines are horizontal! Therefore, our intuition was correct: there is just a horizontal translation between the camera location of the two pictures.

Below you can see the epipolar line (in blue) that corresponds to the red point, and is clearly horizontal.


$p_A$(row,col) = (137,229)
$v_A$(x,y,1) = (229,137,1)'
$l_B = Fv_A$ = (0.00,-0.01,1.00)'

If we now want to find a match for the red point in the other image, we know it has to be somewhere in the blue line.

## Compute a dense disparity map

This section presents the results of estimating dense disparity maps with the fundamental matrices from the previous section. Because the fundamental matrix relates a point in image A to an epipolar line in image B (or vice-versa), it constrains the search space for a match corresponding: given a point in image A, the corresponding point in image B should be somewhere in that line, so we only have to look for a match in that line (using normalized cross-correlation), instead of the whole image. This obviously makes the search much more efficient.

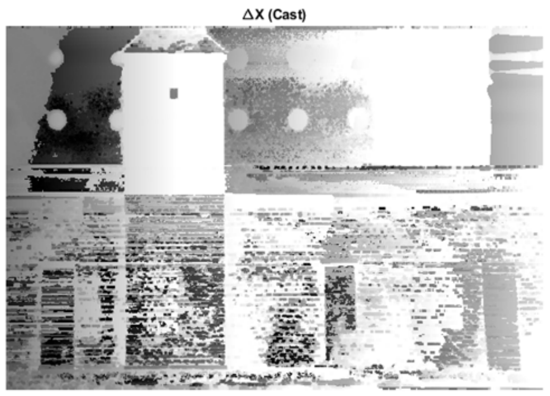
Remember that our goal is to find the disparity, which is the difference in pixel coordinates between all the points in image A and their match in B. An estimate of this disparity could constrain even more the search space in image B, by first searching along the points in the epipolar line that are: at the coordinates in A $\mp$ an estimate of the disparity. However, instead of using an estimate, we have set an upper bound for the disparity of half of the image size, which is appropriate for the given images.

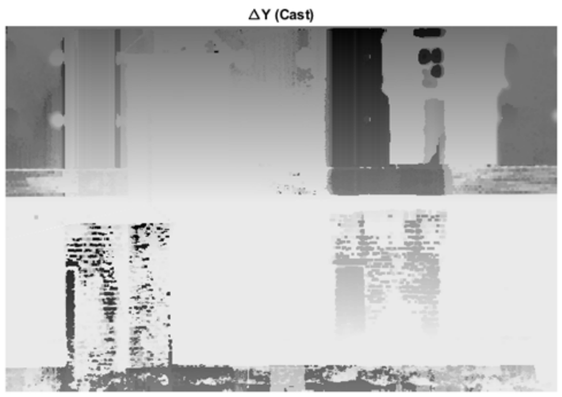
Both horizontal and vertical disparity maps encode a 3D representation of their scene, as you could recover depth information from them. We output 3 images to represent the disparity:

- Two images representing the vertical and horizontal disparity. The brightness at each pixel represents the magnitude of the disparity in the y and x directions, respectively. The brighter the pixel, the further away its correspondence is (bigger disparity).
- An image that uses coloring to represent the disparity vector, using HUE.

### Results

The first attempt at matching using the search space constrained by an epipolar line scores potential matches by summing the absolute differences in pixel grayscale intensities. This differencing considers an image patch centered on a point from A and compares to image patches centered on each point along the corresponding line in B, with the best match selected. This process is repeated for every pixel in A to produce a corresponding match from B, with the disparities computed from their difference.





Initial horizontal (*left*) and vertical (*right*) disparities in input2 set of images, in grayscale

Note that the "Cast" denomination for the images refers to the input2 set of images. Later on, the "Image" denomination refers to input1. These designations originate from the names of their files from Canvas.

The grayscale values are constructed by first shifting the original disparity so that the minimum value is 0, then they are scaled so that the maximum value is 255. Thus, closer objects should appear closer to white, whereas farther objects should appear closer to black. In the horizontal disparities, this appears to be the pattern for at least a small portion of the image: the outside of the leftmost tower appears white, whereas both the sky behind the tower and the window inside it appear as darker shades of gray.

However, this is the only major positive that the initial attempt has. The attempt has a plethora of problems to address:

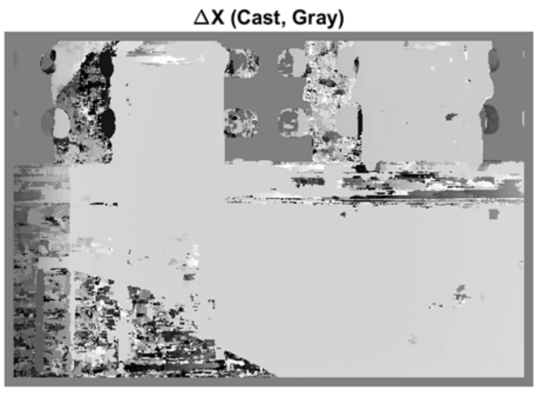
- There are some bricks that appear black, i.e. farther away than other parts of the wall that immediately surround them and yet appear white. Sensibly, these should be at the same depth, so there should be less contrast in the disparity there.
- The scene appears to be almost entirely a horizontal translation between the image pair, yet the vertical disparity map does not reflect this.

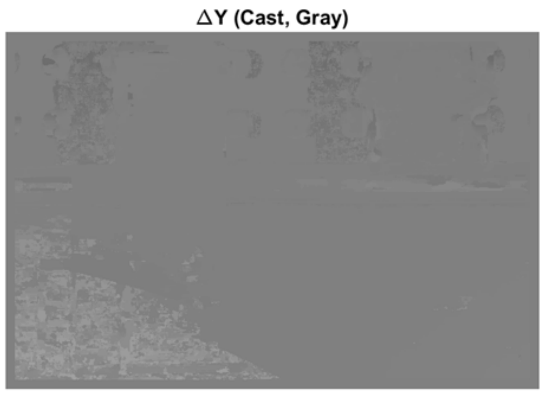
The latter point may be affected by the scaling methodology, but the scaling would not fully explain away the apparent gradient in the vertical disparities.

These above difficulties likely arise from the use of an overly simplistic method for matching, i.e. by just differencing grayscale values. Another attempt is made at generating the disparity maps by using a normalized cross-correlation (NCC) metric instead. Specifically, for each point from A and its corresponding epipolar line from B, the NCC is found between two partial images: an image patch centered on the point from A, and an image segment located around the epipolar line from B.

As design considerations, the patch size is set to **10-by-10** and the image segment with the epipolar line is bounded such that only possible matches at most **10 points** away from the line AND at most a disparity of **100** from the original point are considered. The first two are coupled and chosen after a series of tests attempting to find the right balance between accuracy and computational efficiency. Setting a maximum threshold on the disparity is also intended to reduce the computational load from the NCC computation and is justified as a heuristic by observing that not more than half the scene can be observed shifting.

The change in results from using NCC instead of the summation of absolute differences is discussed below.





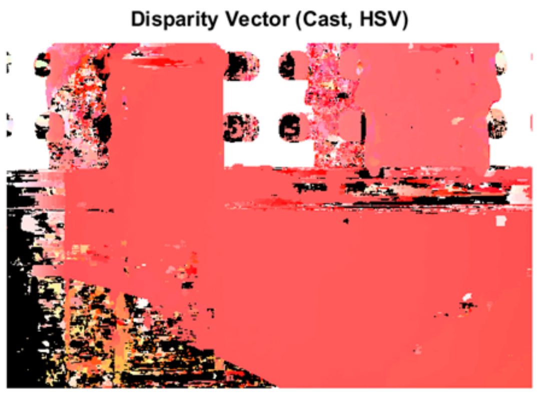
Horizontal (*left*) and vertical (*right*) disparities from using NCC matching, in grayscale

It is easy to appreciate how now the output reflects the displacement between the two images. On one hand, the vertical disparities are uniform on the whole image, because there is no vertical disparity at all (it looks grey instead of pure black because of the noise and the scaling used to plot the image).

On the other hand, we can clearly see in the horizontal disparity how the towers and the wall are brighter than the landscape. That's because the depth is inversely proportional to the disparity, and so objects that are closer should present a bigger disparity (represented by brighter pixels).

Another project requirement is the construction of a hue-saturation disparity map. First, the disparity vector is constructed in two parts: magnitude and orientation. Over each pixel, the magnitude is simply the 2-norm of both the horizontal and vertical disparities at that pixel. The orientation is simply the inverse tangent of the vertical disparity and the horizontal disparity.

Hues use orientations shifted from the original interval $[-\pi, \pi]$ over to $[0, 2\pi]$ then scaled to $[0, 1]$. Saturations use magnitudes shifted from the smallest value, then scaled by their maximum value into the interval $[0, 1]$. Values – the third component of the HSV color scheme – are set to a fixed 1 for ease of visualization. Then, all HSV codes are converted to RGB for plotting.



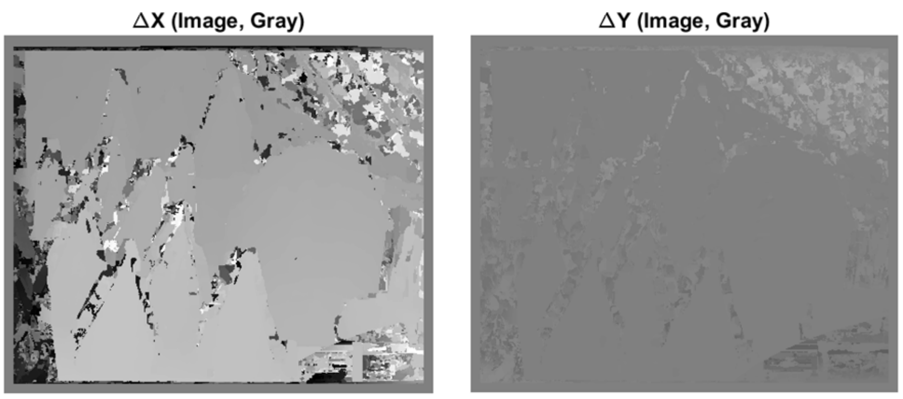
Disparity vector map of Cast; hue scaled by orientation and saturation scaled by magnitude

The disparity vector map for input2 yields interesting results:

- The building in the foreground shows a (mostly) reddish-pink color. This is explainable by the constant horizontal shift that persists throughout parts of the building. That is, the disparity here is almost entirely horizontal – evident in the previous grayscale figures – so the orientation is expected to be in one direction for the building covering most of the image. The result above is consistent with this expectation.
- Since the building is at a constant depth, the shade of the color appears uniform – as expected.
- There appears to be some artifacting that causes sharp changes in either the hue or the saturation, especially in the lower left of the image. This pattern also appears in the grayscale images, but most of it is probably due to noise.

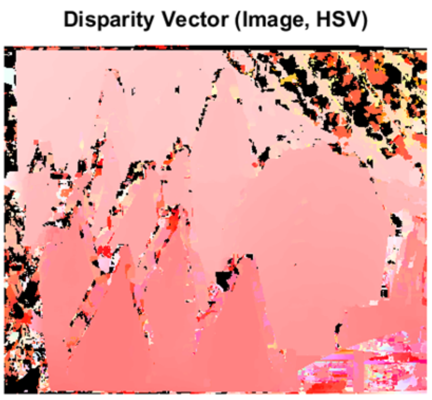
While further improvement can further diminish the artifacting in the disparity vector map, this result is still significantly better than the initial attempt using summation of absolute differences; instead, NCC yields much more accurate disparities consistent with farther objects exhibiting smaller disparities and closer objects exhibiting larger disparities.

Now consider the other pair of images. This is the output we get:



Horizontal (*left*) and vertical (*right*) disparities from NCC matching, in grayscale

Again, the vertical disparity looks uniform as it is very low in the whole image (looks grey and not black because of the scaling that the plotter does), and the horizontal disparities are brighter in the objects that are closer to the camera: cons in the front look slightly brighter than the ones in the bottom. The distinction is not as clear as in the first input because the difference in the depth of the cons is much lower than the one between the castle and the landscape.



Regarding the color output, again it shows a (mostly) reddish-pink color as the disparity is horizontal. However, we can now appreciate much more variety in the saturation as there is much more variety in the depth of the objects in the image. We have encoded the magnitude of the disparity vector as the saturation of the color, and so we have the relation:

$$\text{bigger saturation} \iff \text{bigger magnitude of the disparity} \iff \text{bigger depth}$$

## Conclusion

In this project, we had seen an application of what we can do with some knowledge of how epipolar geometry works. Without the constraint that the epipolar line provides, finding correspondences between these two images would be a much harder task, as the search space would be much bigger. It has also been really satisfying to appreciate the power of this mathematical model with a visual example, as the output we are getting clearly explains the displacement of both images.