COMP 408/508, Assignment #2

Feature Detection & Matching

Cakmak, Serike
Department of Computer Engineering
Koc University
Istanbul, TURKEY
scakmak13@ku.edu.tr

***Abstract*—This document is a report for the assignment of Computer Vision course of the Computer Engineering Dept., Koc University. Aim of this assignment was to be familiar with feature detection and marching using Matlab.**

***Keywords*— feature detection, matching, harris operator, sift descriptor, gradient,anti- aliasing, pca derivation**

I. INTRODUCTION

In this paper, antialiasing and filtering, feature detection, feature description and matching and their algorithm implementations are explained.

II. ANTIALIASING AND FILTERING

In signal processing and related disciplines, aliasing refers to an effect that causes different signals to become indistinguishable (or aliases of one another) when sampled. It also refers to the distortion or artifact that results when the signal reconstructed from samples is different from the original continuous signal. [3]

Here, we take 2048x2048 original image and reduce its size by a factor of 8. Stating the condition above, antialiasing will certainly occur if we directly downsample it.

In order to get a proper antialiased image I have firstly used Bartlett function to low pass filter the image. Before filtering I normalized the values of Bartlett by dividing to the sum of values. To avoid aliasing, I used this lowpass filter; then down-sampled.

```
H = bartlett(L);
filtered = imfilter(I,H);
down = filtered(1:n:end,1:n:end,:);
```

where n is the size of downsampling which is 8 in our case.

Below in Figure1 and Figure2, effects of downsampling is shown. Figure1 is the case when no law pass filtering is performed. The distortions can be seen in letters ‘M’ and ‘V’ clearly. In Figure2 distortions occur fewer but the image is blurred.

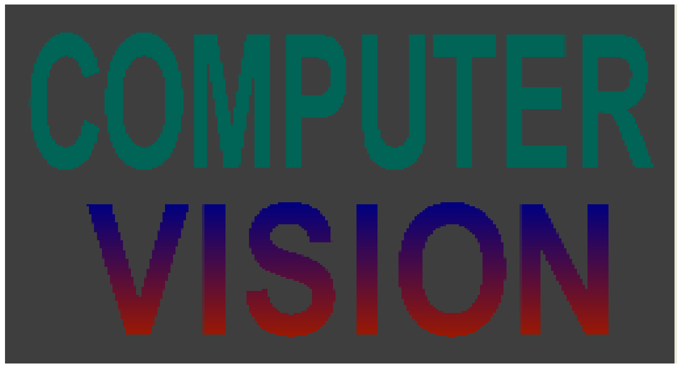

Figure 1. Downsampling original image by size 8 (Distortion occurs)

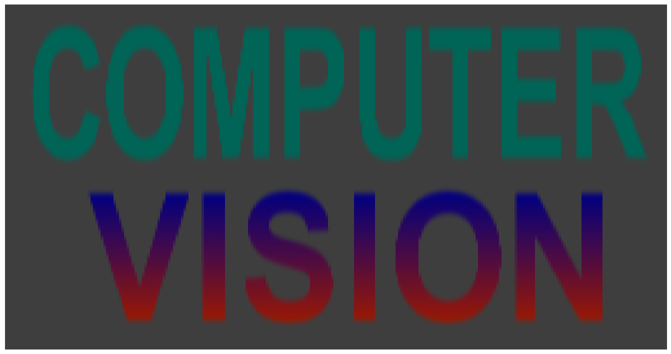

Figure 2. Antialiasing method is used (Blurred, some high-frequency details are lost but not distorted)

III. FEATURE DETECTION

A. *Edge Detection*

In this part, the edges in the image will be detect the corners in the reduced resolution. The gradient method from Matlab is used in the process of edge finding. First, color gradient was used in which I have separated three channels and compute the gradient for each channel. Second, intensity information was used to find the edges. (Intensity version: I = (R+G+B)/3). After computing each channel independently, I combined them using the formula from lecture slides:

$$\sqrt{|\nabla f_R(n_1,n_2)|^2 + |\nabla f_G(n_1,n_2)|^2 + |\nabla f_B(n_1,n_2)|^2}$$

The results of these two versions are compared in Figure 3.

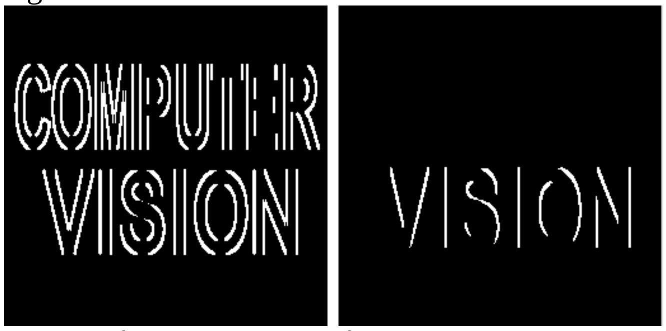

Figure3. Left image is the result of RGB version and right image is intensity version.

Figure 3 shows that using color gradient for this image performs better. Because if use only the intensity information, we lose edges related to “computer” word. But edges related to “vision” are preserved due to the high intensity difference in this part of image.

B. *Corner Detection*

In this part, I have found the corners on the same image by using Harris Detector. I used the Harris Operator from lecture notes.

$$h = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} = \frac{\det(\boldsymbol{C})}{\mathrm{trace}(\boldsymbol{C})}$$

Similar to edge detection part, I have found the corners both by using color gradient and intensity. The corner detection algorithm explained in lecture slides was followed and Gaussian mask in 3x3 size was used in my implementation:

```
g = fspecial('gaussian',[3 3]); % mask
```

To the whole range of detected points, I specified a threshold to keep only the ones which have a chance of being a corner. After that, I found local maximum among the points. Figure 4 and Figure 5 shows the results when color gradient and intensity gradient is used.

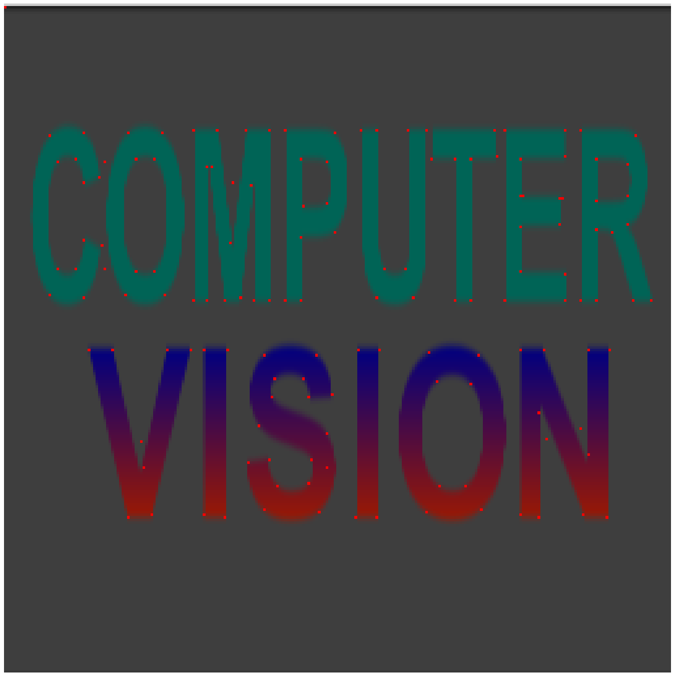

Figure 4. Harris Corner Detection with color gradient

When used separated color channels to compute the gradient, the result is pretty good. Almost all the correct corners are detected in both “computer” and “vision” words. There are some detected points which are unnecessary actually or that are wrong besides the letters in 2-3 locations. However, the general view was satisfying to me since it has detected most of the important corners in the image.

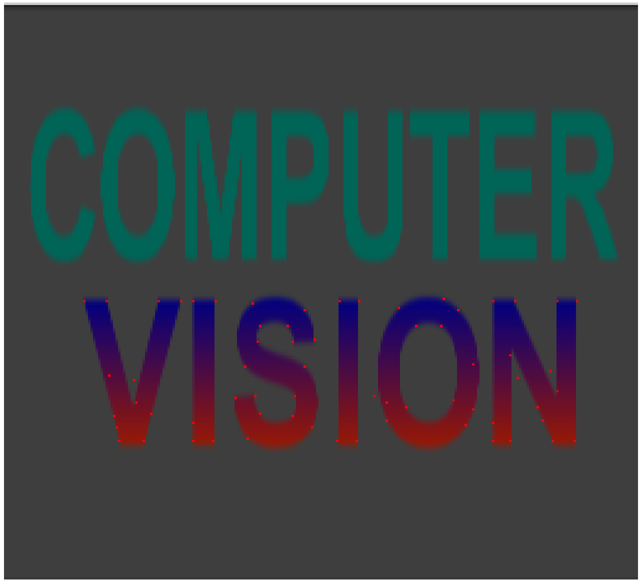

Figure 5. Harris Corner Detection with intensity gradient

The same Harris detector algorithm implemented once more with the intensity gradient. Figure 5 shows the detected corners when only intensity information is used. In this version the corners for “computer” word cannot be detected.

By comparing Figure 4 and Figure 5, I can say that using color gradient for this image is more suitable. However, in other images that will be demonstrated in feature description part, color gradient version performs better corner detection results than intensity version regarding the number of detected corners.

IV. FEATURE DESCRIPTION AND MATCHING

In this part, the Harris detector will be used to detect the points of interest in two given images. In fact, the two images are the same image from different perspectives. This part will define the points that we desire to match between two images. For this reason, I used the algorithm implemented above in corner detection part. In addition to the part above, I used morphologic operation dilation to visualize the interest points in a better way. By using dilation the boundaries of regions of foreground pixels are enlarged and holes within these regions become smaller. The result of corners (interest points to match) detection is shown in Figure 6 and Figure 7. As it performed better, I used color gradient.

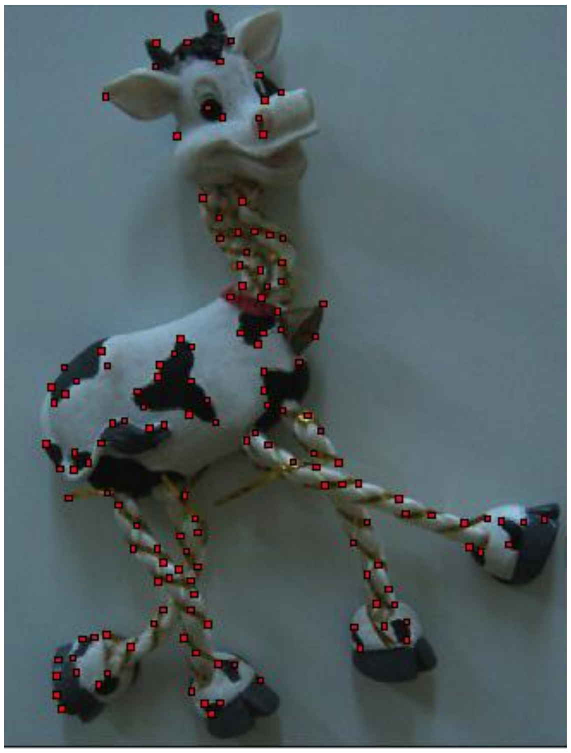

Figure 6. Harris Corners (interest points) of image1.png

Same algorithm was again used to find the interest points on a different perspective of the image. This image was taken from a different point of view and has a shaded area on the left up corner.

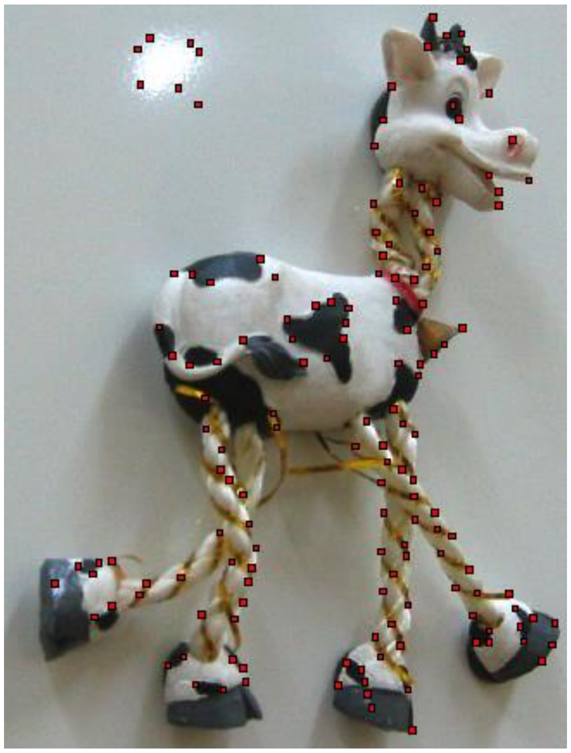

Figure 7. Harris Corners (interest points) of image2.png

A. *SIFT ALGORITHM*

The detected Harris Points were matched by using SIFT algorithm. I wrote the ***sift.m*** function which creates a 128 dimensional descriptor for a given interest point. In this function I tried to implement the SIFT descriptor algorithm given in lecture slides [2]. I have constructed 16x16 square around our harris interest point and found the gradient of that part to find edge orientations. However, there was some angles that are not exactly match to our 8 bin angles(45,90,135,..). Also, there were negative angles so I corrected them by adding 360 degrees. After that I have divided this square to 4x4 squares and calculated histogram for each of them and again found the edge orientations, but this time by normalizing according to the canonical orientation that I have found from 16x16 square.

Having thought that I created a descriptor, I tried to match a point from image 1 to the best point in image 2. Having a point from image 1, all SSD distances with the possible points from image 2 was compared and the most suitable two of these points were chosen. I tried to visualize my matching interest points from image 1 and image 2 in Figure 8.

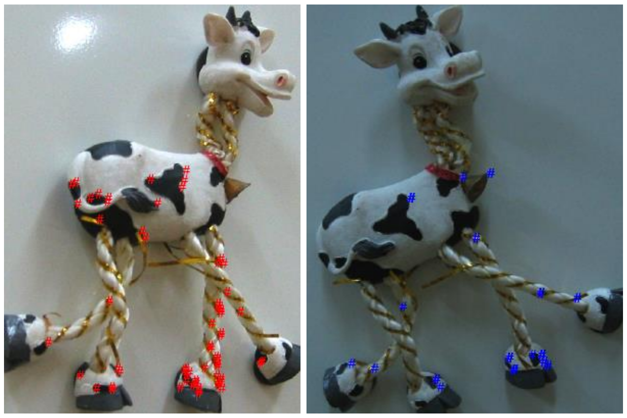
Figure 8. SIFT point matching (matching.m)

After that, I have used a plot_match.m from matlab submissions[6] to visualize the two images side by side and showing my matching points by connecting with different color lines. Although I have tried to set a threshold to visualize only the best matching points, there are unfortunately some wrong matches in my result. However algorithm works for some points correctly and I have recently got the idea behind. Following figures display my matching results on two 2 different image classes.

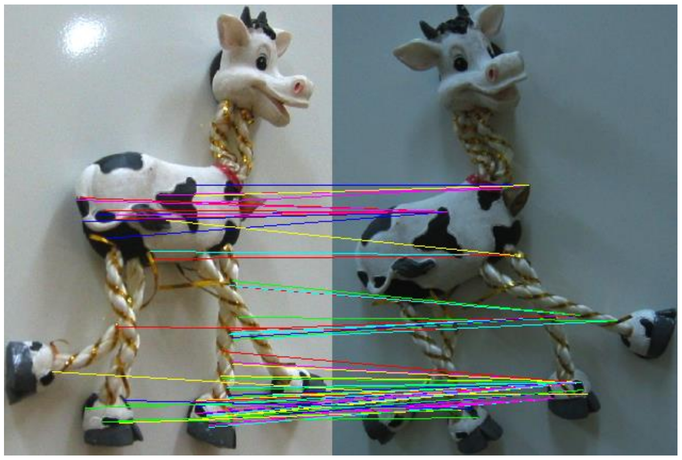
Figure 9. SIFT points visualization with lines (plot_match.m)

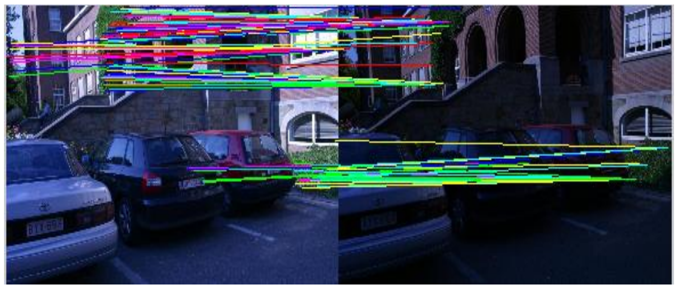
Figure 10. SIFT points visualization with lines (plot_match.m)

V. PCA FOR COLOR GRADIENT

In this part, Lagrange Multipliers Techniques was used to prove that the first principal eigenvector of correlation matrix. While solving the optimization problem, I have used the references [4] and [5]. My solution regarding the PCA derivation for color gradient is in Appendix Part.

REFERENCES

[1] "COMP 408/508 Computer Vision, Filtering and Image Enhancement " ,2013, http://home.ku.edu.tr/~yyemez/comp508/filtering.pdf

[2] Szeliski,Z. "Computer Vision: Algorithms and Applications," Springer, 2010, p.109.

[3] "Aliasing", http://en.wikipedia.org/wiki/Aliasing

[4] "Lagrange Multiplier", http://en.wikipedia.org/wiki/Lagrange_multiplier

[5] "PCA", Iyad Batal, http://people.cs.pitt.edu/~iyad/PCA.pdf

[5] "match_plot.m", Gooly,2011, http://www.mathworks.com/matlabcentral/fileexchange/31144-match-plot/content/match_plot.m

APPENDIX

PCA DERIVATION

f_i ➡ Random data vector of dimension 2, $f_i = (fx_1, fx_2)$
where f_1, f_2, f_3 correspond to gradient vectors in R,G,B.

Lagrange Multiplier Technique (wikipedia)

* consider the optimization problem
→ maximize $f(x,y)$
→ subject to $g(x,y) = c$
↳ Lagrange function is defined by;
$L(x,y,\lambda) = f(x,y) + \lambda(g(x,y) - c)$, λ : Lagrange multiplier

$v_1 = \underset{\|v\|=1}{\operatorname{argmax}} \sum_i (f_i \cdot v^T)^2$ ➡ write this in matrix form

$$v_1 = \underset{\|v\|=1}{\operatorname{argmax}} \|Fv^T\|^2 = \underset{\|v\|=1}{\operatorname{argmax}} vF^TFv^T$$

$$v_1 = \underset{\|v\|=1}{\operatorname{argmax}} \frac{vF^TFv^T}{vv^T}$$ since v is a unit vector

➡ Quotient's max possible value is the largest eigenvalue of the matrix which occurs when v is the corresponding eigenvector.

➡ $F^TF = \sum_i (f_i - \mu)^T (f_i - \mu) = var(F)$, if we say $F = [f_1 - \mu, \dots, f_n - \mu]$ where μ is the sample mean

➡ $v_1 = \underset{\|v_1\|=1}{\operatorname{argmax}} var(Fv_1)$ ➡ $\underbrace{var(Fv_1) = v_1^T D v_1}_{\text{maximize}}$ where D is the covariance matrix

Lagrange function: subject to $v_1^T v_1 = 1$

$$L(v_1, \lambda_1) = v_1^T D v_1 + \lambda_1 (v_1^T v_1 - 1)$$

$$\frac{dL(v_1,\lambda_1)}{dv_1} = 2Dv_1 - 2\lambda_1 v_1 = 0 \Rightarrow Dv_1 = \lambda_1 v_1 \quad \boxed{\text{Eigenvactor Problem}}$$

multiply by v_1^T ➡ $\lambda_1 = v_1^T D v_1$ $\boxed{\text{Eigenvalue Problem}}$

λ_1: largest eigen value
v_1: corresponding eigenvector