Amirkabir University of Technology (Tehran Polytechnic)

Assignment 2

When Cameras Fail: Enhancing Images in Spatial Domain

Homeworks Guidelines and Policies

- ***What you must hand in.*** It is expected that the students submit an assignment report (HW2_[student_id].pdf) as well as required source codes (.m or .py) into an archive file (HW2_[student_id].zip).
- ***Pay attention to problem types.*** Some problems are required to be solved *by hand* (shown by the 📝 icon), and some need to be implemented (shown by the icon).
  Please don't use implementation tools when it is asked to solve the problem by hand, otherwise you'll be penalized and lose some points.
- ***Don't bother typing!*** You are free to solve by-hand problems on a paper and include picture of them in your report. Here, cleanness and readability are of high importance. Images should also have appropriate quality.
- ***Reports are critical.*** Your work will be evaluated mostly by the quality of your report. Don't forget to explain what you have done, and provide enough discussions when it's needed.
- ***Appearance matters!*** In each homework, 5 points (out of a possible 100) belongs to compactness, expressiveness and neatness of your report and codes.
- ***Python is also allowable.*** By default, we assume you implement your codes in MATLAB. If you're using Python, you have to use equivalent functions when it is asked to use specific MATLAB functions.
- ***Be neat and tidy!*** Your codes must be separated for each question, and for each part. For example, you have to create a separate .m file for part b. of question 3. Please name it like p3b.m.
- ***Use bonus points to improve your score.*** Problems with bonus points are marked by the ⭐ icon. These problems usually include uncovered related topics or those that are only mentioned briefly in the class.
- ***Moodle access is essential.*** Make sure you have access to Moodle because that's where all assignments as well as course announcements are posted on. Homework submissions are also done through Moodle.

- ***Assignment Deadline.*** Please submit your work **before the end of May 12th**.
- ***Delay policy.*** During the semester, students are given 7 free late days which they can use them in their own ways. Afterwards there will be a 25% penalty for every late day, and no more than three late days will be accepted.
- ***Collaboration policy.*** We encourage students to work together, share their findings and utilize all the resources available. However you are not allowed to share codes/answers or use works from the past semesters. Violators will receive a zero for that particular problem.
- ***Any questions?*** If there is any question, please don't hesitate to contact me through the following email address: **ali.the.special@gmail.com**.

1. Mathematics of Image Point Processing Operations (8 Pts.)

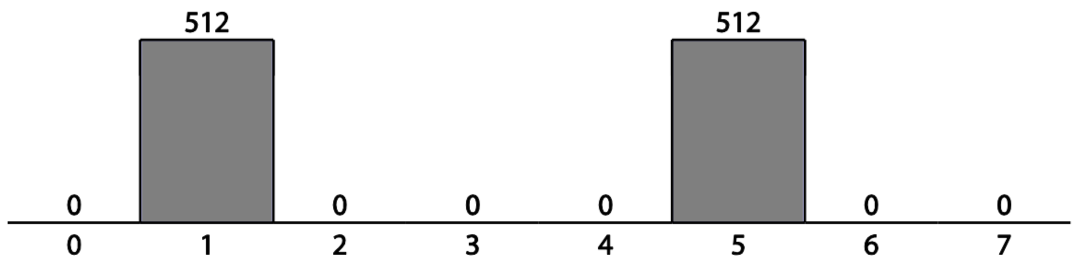
Keywords: *Image Point Processing, Image Histogram, Histogram Equalization, Histogram Matching (Specification), Image Negative, Bit-plane Slicing, Image Thresholding, Contrast Stretching*

The first task contains several problems in the topic of **Point Processing**. Point processing operations are an important group of **Image Enhancement** techniques in **Spatial Domain**, which focus on enhancing images solely with manipulating pixel intensity values. The problems here are aimed to make sure you've theoretically mastered those operations.

First, consider the following 3-bit image of the size 8×8:

1	2	3	3	3	5	6	6
2	2	2	4	6	6	7	7
2	2	3	4	5	7	7	6
2	3	3	3	6	7	6	6
3	4	5	6	5	4	4	5
2	3	5	5	4	4	4	5
2	4	4	4	4	2	3	4
3	4	4	3	3	2	1	4

a. Find the digital negative of the image.
b. Perform bit-plane slicing and write down all bit-planes representations of the image.
c. Compute the image histogram.
d. Determine a "valley" in the histogram, and threshold the image at this value. Write down the resultant image matrix.
e. Apply linear scaling for stretching the gray levels of the image to the range of 0-255. Write down the result matrix.
f. Apply histogram equalization to the image. Write the resultant matrix as well as the equalized histogram.
g. Suppose you want to shape the histogram of the above image to match that of a second image, given below:



Use histogram specification to devise a gray-level transformation that shapes the histogram of the original image to look like the second. Note that you have to adopt a strategy to handle certain cases, including when two input values map to the same output value, and no input values map onto a particular output value.

Now, assume an image with the following gray-level histogram:

$$p_r = \frac{e^r - 1}{e - 2}, \quad r \in [0,1]$$

h. Find a gray-level transformation $s = T(r)$ to make the transformed histogram p_s uniform, i.e.:

$$p_s = 1 \quad s \in [0,1]$$

Note: Make sure to show all intermediate steps in the calculations.

2. Practicing Image (De)Convolution Calculation (12 Pts.)

Keywords: *Image Spatial Filtering, Kernel (Mask), Image Convolution, Image Smoothing, Image Sharpening, Image Gradient*

Image Filtering is another way to enhance images in spatial domain. It is done through convolving a predefined **Kernel** over an image. So before jumping into the world of pixels and colors, it's good to practice some **2D Convolution** problems first.

First, a 3-bit image of the size 8×8 is given. Convolve each one of the given kernels (a) to (h) with this image and write the output image matrix. Also describe how each mask affects the input image.

0	7	7	0	0	3	3	3
0	7	0	0	0	3	7	3
0	7	3	3	3	3	3	3
0	7	0	0	0	7	0	0
0	0	7	0	3	3	3	0
3	0	7	0	3	3	3	0
7	0	7	0	3	3	3	3
3	3	0	0	0	7	0	7

-1	-1	-1
-1	8	-1
-1	-1	-1

(a)

-1	0	1
-1	0	1
-1	0	1

(b)

-1	-1	0
-1	0	1
0	1	1

(c)

0	-1	0
-1	4	-1
0	-1	0

(d)

1	1	1
1	1	1
1	1	1

(e)

-1	-1	-1
2	2	2
-1	-1	-1

(f)

0	-1	-1
1	0	-1
1	1	0

(g)

-1	2	-1
-1	2	-1
-1	2	-1

(h)

Now, let's do something more exciting. We're going to do the reverse process, i.e. **Image Deconvolution**. It's a technique widely used to restore the original image by estimating the convolution kernel. Here, you are asked to specify each one of the 3×3 kernel which was convolved with the original image to yield the outputs (i) to (p). Please clearly explain the strategy you used for each part.

Note: Rounded intensities are shown by blue, and out-of-bound values are shown by red.

2	2	2	1	1	2	2	2
2	3	4	2	2	3	3	2
2	3	3	1	2	3	3	2
2	3	3	2	2	3	2	1
1	3	2	2	2	3	2	1
1	3	2	3	2	3	2	1
2	3	2	2	2	3	3	2
1	2	1	1	1	2	3	1

(i)

7	7	7	7	3	6	7	6
7	7	7	0	3	7	7	7
7	7	7	7	7	7	7	6
7	7	7	0	7	7	7	0
0	7	7	7	6	7	6	3
3	7	7	7	6	7	6	3
7	7	7	7	6	7	7	6
6	6	3	0	7	7	7	7

(j)

0	5	2	0	0	2	3	2
0	7	3	1	1	3	4	3
0	7	1	1	1	4	3	2
0	5	3	1	2	4	2	1
1	2	5	0	2	4	2	0
3	0	7	0	3	3	3	1
4	1	5	0	2	4	2	3
3	1	2	0	1	3	1	3

(k)

0	7	7	0	0	7	2	7
0	7	0	0	0	2	7	2
0	7	5	7	7	0	2	7
0	7	0	0	0	7	0	0
0	0	7	0	7	0	7	0
7	0	7	0	6	3	6	0
7	0	7	0	7	0	6	5
5	7	0	0	0	7	0	7

(l)

7	0	0	7	6	0	0	0
7	0	7	7	7	0	0	0
7	0	0	0	0	0	2	0
7	0	7	7	7	0	7	7
7	7	0	7	0	0	0	6
0	7	0	7	0	0	0	7
0	7	0	7	0	0	2	0
0	0	7	7	7	0	7	0

(m)

1	4	3	1	1	2	3	2
2	4	3	1	1	3	4	3
2	4	3	2	2	3	3	2
1	3	3	1	2	3	2	1
1	2	3	2	2	3	2	1
2	3	4	3	2	3	2	1
3	3	3	2	2	3	3	2
2	2	1	1	1	3	3	2

(n)

0	0	0	0	0	0	0	0
0	4	5	0	0	0	0	0
0	0	0	0	0	0	7	7
7	7	2	2	3	0	3	6
1	4	0	0	0	2	0	0
0	0	0	0	0	0	0	0
0	1	7	7	2	0	0	0
7	7	7	7	7	7	7	7

(o)

2	0	0	0	1	0	0	0
4	0	4	3	2	0	0	0
4	0	1	0	0	0	0	0
2	0	3	4	2	0	2	2
2	3	0	3	0	0	0	1
0	5	0	5	0	0	0	2
0	3	0	3	0	0	1	0
0	1	0	3	1	0	2	0

(p)

3. Taking Part in a Noble Activity: Reviving Ancient Transcripts (10 Pts.)

Keywords: *Image Thresholding, Global Thresholding, Local Adaptive Thresholding, Otsu's Thresholding Method*

Project Gutenberg provides the first and one of the richest libraries of free eBooks, with over 60,000 documents available online in various formats such as HTML, PDF, EPUB and MOBI. The process of eBook preparation is mostly done by volunteers across the world, in which the books are first scanned and then digitized through an OCR process.

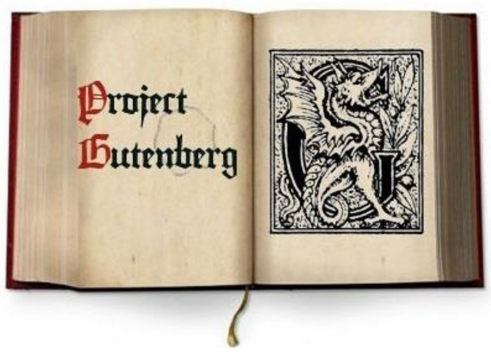


Figure 1 In Project Gutenberg, image processing and computer vision techniques play a major role in the process of digitizing and archiving cultural works

The focus of the project is on older books for which U.S. copyright has expired, which also includes old transcripts and priceless antique books. Although it is possible to remove the book spine in order to increase scanning quality, for books of this kind, however, destroying the original binding of the book is often undesirable. If the process of scanning is done with book spine left intact, the curvature of the pages causes uneven illumination in the scanned images. In Project Gutenberg, this problem is addressed via a post-processing procedure, namely **Image Thresholding**.

Here, we aim to investigate several thresholding techniques and evaluate their efficiencies in this application. Consider the following thresholding methods:

I. **Global thresholding.** Thresholding is performed using a fixed value, which is often chosen using histogram of the grayscale image.

II. **Otsu's method.** Another thresholding method which attempts to find the threshold that minimizes the weighted within-class variance (or maximizes the between-class variance). Here, the weighted within-class variance is:

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

Where the class probabilities are estimated as:

$$q_1(t) = \sum_{i=1}^{t} P(i), \quad q_2(t) = \sum_{i=t+1}^{I} P(i)$$

And the class means are given by:

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)}, \quad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)}$$

Finally, the individual class variances can be calculated as:

$$\sigma_1^2(t) = \sum_{i=1}^{t} \left[i - \mu_1(t)\right]^2 \frac{P(i)}{q_1(t)}, \quad \sigma_2^2(t) = \sum_{i=t+1}^{I} \left[i - \mu_2(t)\right]^2 \frac{P(i)}{q_2(t)}$$

Running through the full range of t values $[1,256]$ and picking the value that minimizes $\sigma_w^2(t)$ will give us the optimal threshold value. However, using the relationship between the within-class and between-class variance, one can generate a recursion relation that permits a much faster calculation. After some algebra, the total variance can be expressed as:

$$\sigma^2 = \sigma_w^2(t) + q_1(t)\left[1 - q_1(t)\right]\left[\mu_1(t) - \mu_2(t)\right]^2$$

in which the first term denotes within-class from before, and the second term indicates between-class $\sigma_B^2(t)$. Since the total is constant and independent of t, the effect of changing the threshold is merely to move the contributions of the two terms back and forth. Note that we can compute the quantities in $\sigma_B^2(t)$ recursively as we run through the range of t values.

Overall, the final algorithm steps can be written as below:

- Initialization: $q_1(1) = P(1), \quad \mu_1(0) = 0$
- Recursion: $$\begin{cases} q_1(t+1) = q_1(t) + P(t+1) \\ \mu_1(t+1) = \dfrac{q_1(t)\mu_1(t) + (t+1)P(t+1)}{q_1(t+1)} \\ \mu_2(t+1) = \dfrac{\mu - q_1(t+1)\mu_1(t+1)}{1 - q_1(t+1)} \end{cases}$$

III. **Adaptive mean thresholding.** Here, image is divided into blocks of the size B, and a constant C is assumed. Thresholding will be performed in each block separately, so that the threshold value will be the mean of the neighborhood area minus the constant C.

IV. **Adaptive Gaussian thresholding.** The method is similar to adaptive mean thresholding, only the threshold value is a Gaussian-weighted sum of the neighborhood values minus the constant C.

Now perform the following tasks using the provided images:

a. Perform thresholding using global thresholding method.
b. Perform thresholding using Otsu's method in its recursion form.
c. Perform adaptive mean thresholding. Find and report appropriate parameters by trial and error.
d. Perform adaptive Gaussian thresholding. Find and report appropriate parameters by trial and error.
e. Compare and justify the results obtained in the previous parts.

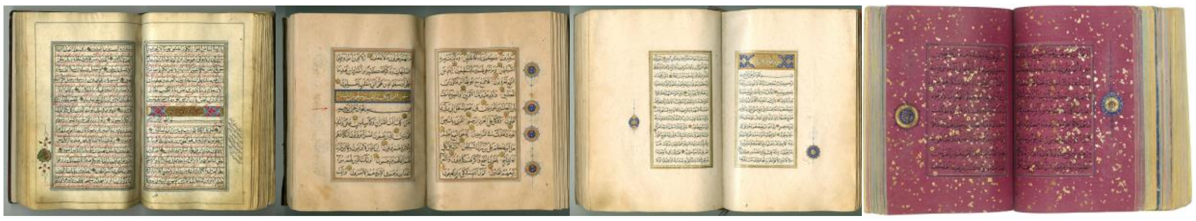


Figure 2 Different scanned pages of the Holy Quran. As can be seen, the books binding has caused uneven illumination in the middle parts, which would have a negative impact on the process of digitizing the books

4. Motion Detection Algorithm Based on Bit-plane Slicing (10 Pts.)

Keywords: *Motion Detection, Bit-plane Slicing, Image Difference*

If you think **Bit-plane Slicing** of an image is an old-fashioned image processing operation with no use, think again. Because it not only comes to use in various applications like **Image Compression**, but can even handle some high-level machine vision tasks, namely **Motion Detection**. Let's see how.

Imagine two consecutive frames are extracted from a video sequence, and the goal is to detect the moving object(s) inside these frames. One may quickly think of **Image Difference**, but there are more efficient ways available. Assume we first convert the frames into 8-bit grayscale images and then slice grayscale pixel values into eight constituting bits using:

$$a_k = \left\lfloor \frac{I}{2^k} \right\rfloor \bmod 2$$

where I is a grayscale pixel value, k is the bit number and a_k is the bit value of the corresponding bit number (Figure 3).

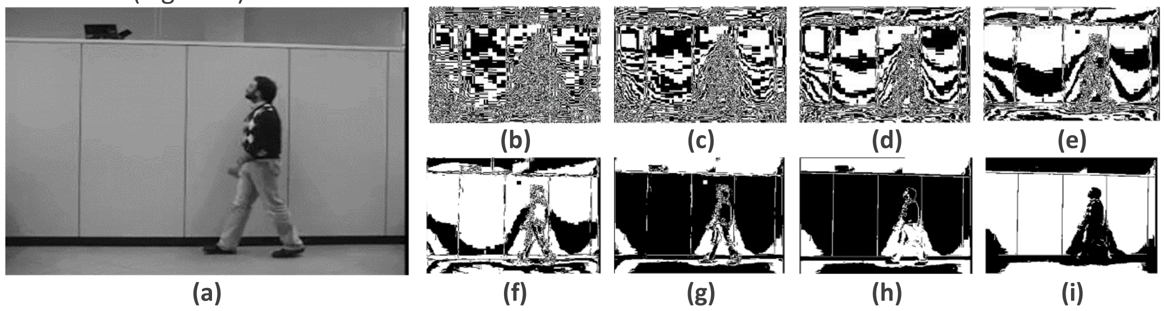


Figure 3 Slicing a frame into bit-plane representations (a) Original image (b) to (i) Bit-plane representations from least to most significant bits

Then, it would be easily possible to compare consecutive frames using XOR function of the corresponding bit-planes, i.e.:

$$c_k = a_k \oplus b_k$$

where a_k, b_k and c_k are bit values of frame 1 pixel, frame 2 pixel and resultant bit value respectively (Figure 4).

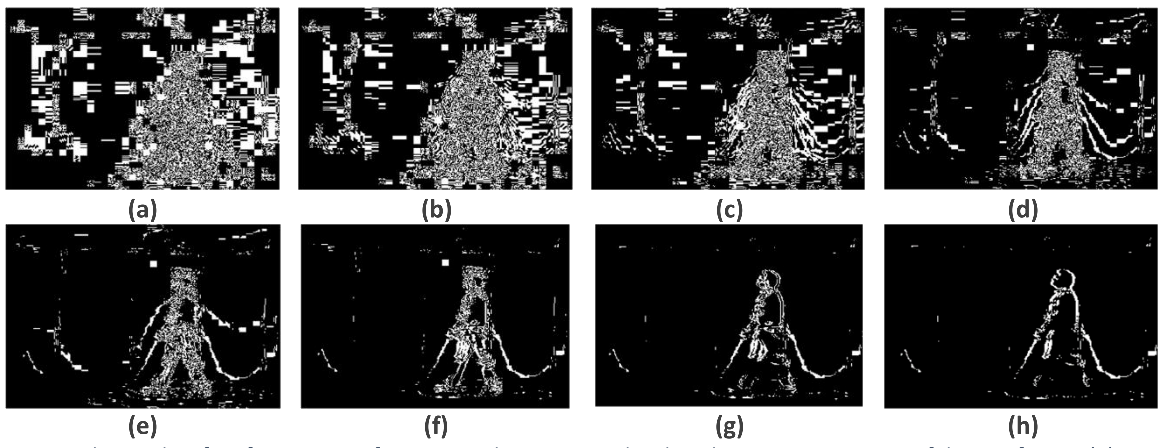


Figure 4 The results of performing XOR function on the corresponding bit-plane representations of the two frames (a) Least significant bit (b) to (g) intermediate bit-planes (h) Most significant bit

Finally, a grayscale image can be obtained by merging higher bit-planes of the resultant of the XOR operations:

$$Y = \sum_{k=4}^{7} 2^k \times c_k$$

This image is expected to highlight moving regions of the two frames. Some post-processing operations may also be needed to get a more clear result (Figure 5).

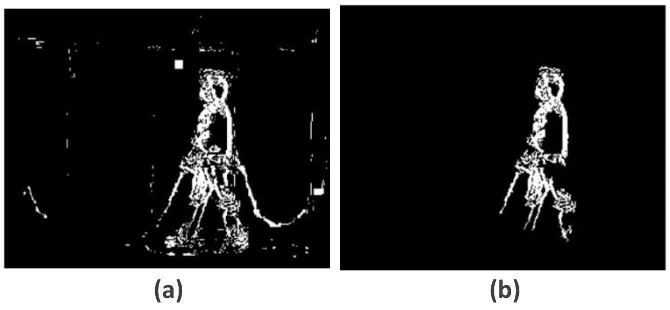


Figure 5 Reconstructed image using the four highest bit-planes (a) Before filtering (b) After filtering

Now, given two sets of consecutive frames, our goal is to implement a similar procedure on both.

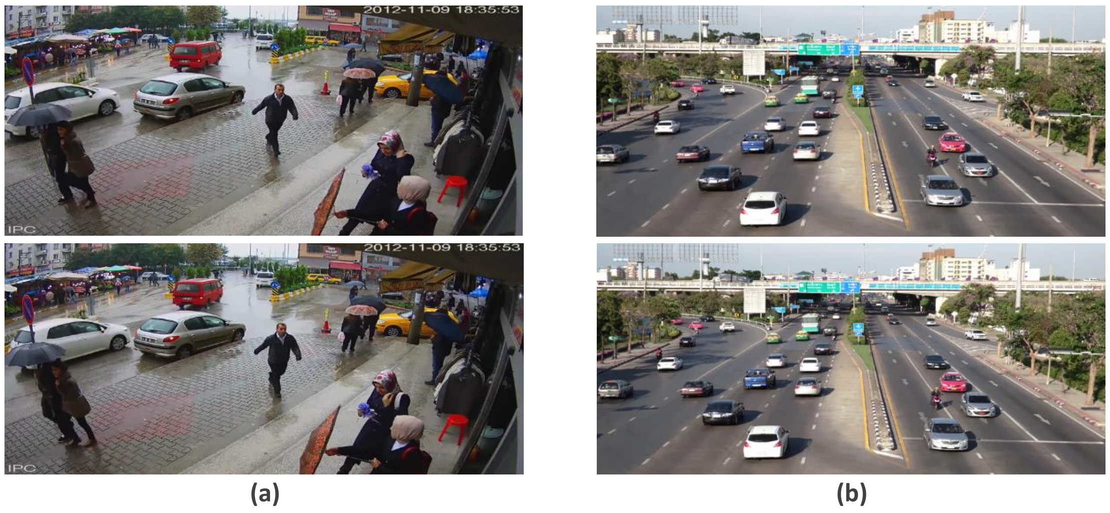


Figure 6 Inputs of this part contain two set of consecutive frames taken from stationary cameras (a) Footages taken from a pavement by a CCTV camera (b) Footages recorded by a traffic control camera in a highway

a. Implement a function `bitplane_slice()` which takes an input image, converts it into grayscale and then slices it into its bitplanes. Perform bit-plane slicing on the given two frames using your function, and display the results.
b. Perform XOR function on the corresponding bit-planes of the two frames, and display the results.
c. Use the 4 highest bit-planes to obtain moving parts of the two frames.
d. Apply proper image enhancement techniques to obtain a final clean image.

5. Afraid of Ghosts? Let's Meet Them! (10+3 Pts.)

Keywords: *Image Enhancement, Histogram Equalization, Adaptive Histogram Equalization, Bilinear Interpolation*

Despite centuries of investigation, no scientific study has ever been able to confirm the existence of ghosts. Also, there is no scientific evidence that any location is inhabited by spirits of the dead. However, over the history, ghost hunting has always been – and still is – a popular practice among both researchers and adventurers.

In general, a ghost-hunting team attempt to collect evidence supporting the existence of paranormal activity. This includes a variety of materials, from footages recorded by surveillance cameras to reports made by ordinary people. After verifying the documents, they travel to the spot equipped with various electronic devices including EMF meters, digital thermometers, digital video cameras, night vision goggles and so on.

In this problem, you are given a set of images which are claimed to be taken from ghosts. Your goal is to enhance these images and highlight their contents (including the ghosts!) using a variant of **Histogram Equalization** method, known as **Adaptive Histogram Equalization**. While ordinary HE algorithm works on the entire image, AHE operates on small regions in the image, called *tiles*.

a. Divide the image into tiles with the size T. Perform histogram equalization in each tile, and display the results.
b. Eliminate artificially induced boundaries in the images by combining neighboring tiles using **Bilinear Interpolation**.
c. Another variant of AHE, known as **Contrast Limited Adaptive Histogram Equalization (CLAHE)** attempts to limit the contrast amplification, so as to reduce the problem of noise amplification in AHE. Perform this method, and compare the results with the previous part.
d. Use any other image enhancement techniques so that the ghosts become clearly visible in white.

Figure 7 Images of the ghosts are taken in uncontrolled conditions and therefore suffer from various degradations including low contrast and brightness

6. Let's Keep the Ball Rolling! (10 Pts.)

Keywords: *Background Subtraction, Foreground Detection, Object Detection, Image Averaging, Image Thresholding*

Foreground Detection, also known as **Background Subtraction**, is a machine vision task in which the goal is to detect foreground objects in a given image. These techniques usually analyse video frames recorded with a stationary camera, and distinguish foreground from background based on the changes taking place in the foreground. They are widely used in various applications, such as traffic monitoring, object tracking, human action recognition, and so on.

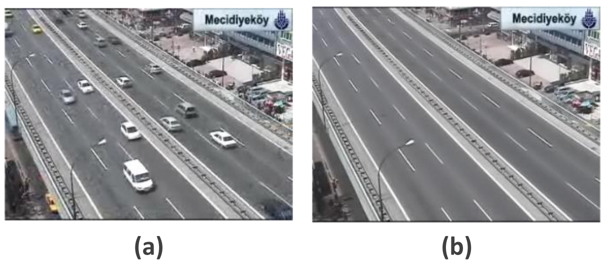


Figure 8 After detecting background image, it's easy to subtract the original image from the obtained background and find foreground image (a) Original image (b) Background image

As you learned in the class, **Image Averaging** is a technique used to reduce noises if multiple images of the same scene are available and noise has zero mean and be uncorrelated. In another word, assuming n different noisy images I_i, then

$$\bar{I}(x,y) = \frac{1}{n}\sum_{i=1}^{n} I_i(x,y)$$

can be shown to be less noisy.

Now based on this framework and applying **Image Thresholding**, we can introduce a simple background detection algorithm in a video sequence, where background is the mean of the previous n frames F_i :

$$B(x,y) = \frac{1}{n}\sum_{i=1}^{n} F_i(x,y)$$

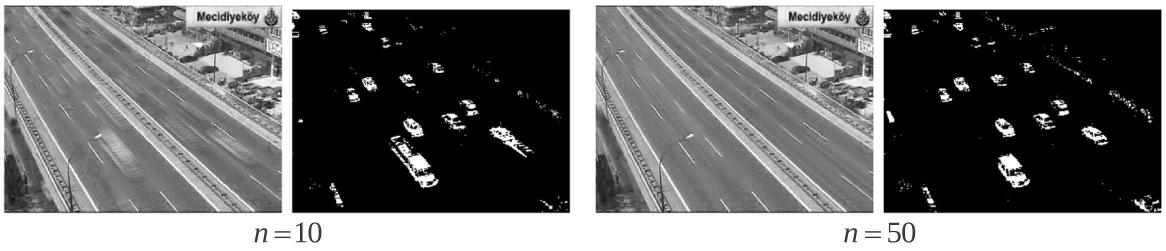


Figure 9 The number of images used in the averaging process considerably affects the "cleanness" of the background obtained by the algorithm, and therefore the output foreground image

Similarly, if the background is more likely to appear in a scene, we can use the median of the previous n frames as the background model:

$$B(x,y) = median\left[F_i(x,y)\right]_{i=1}^{n}$$

Here, your objective is similar. You are provided with 20 frames of a 4 seconds video sequence recorded from a football match, and another test frame of the same scene in a different time. The goal is to detect all players in the test image by first finding a clean image of the field (background) and then subtracting the test image from it.

a. Estimate the background image using the first n frames. Set $n = 2, 5, 10, 20$.
b. Subtract the test frame from the background image you obtained in the previous part.
c. Define appropriate thresholds and find foreground masks corresponding to each of the background images.
d. Use the resultant masks to extract players in the test frame. Assign black color $(0,0,0)$ to the remaining regions of the image.
e. Repeat this process using median operation, and compare the results.

Note: Make sure to display and save the intermediate results as well as including them in your report.

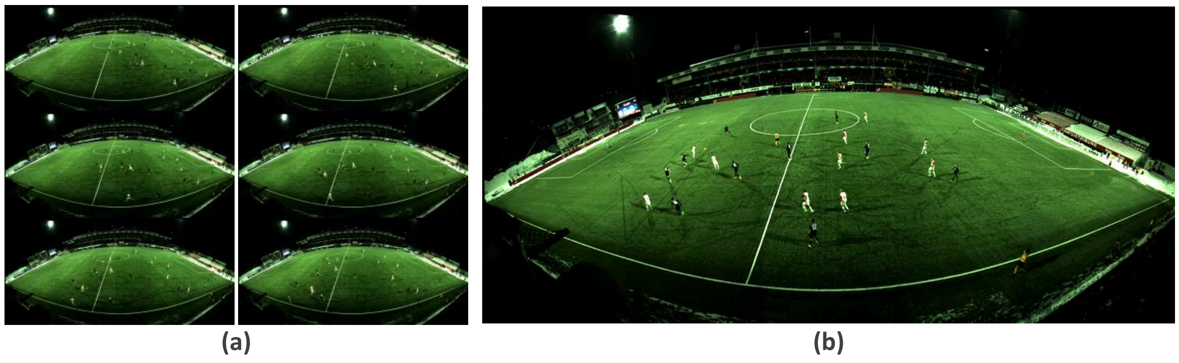


Figure 10 Input video sequence (a) Some extracted frames (b) Test frame

7. From Naïve Image Filtering to Smart Image Resizing (16+7 Pts.)

Keywords: *Image Filtering, Image Gradient, Image Resizing, Seam Carving, Dynamic Programming*

Have you ever experienced the dilemma of fitting an image into a report? On one hand, you have to reduce the image size as much as possible, on the other hand you prefer to keep it large enough to be visualized clearly. This issue stems from the fact that traditional image resizing methods ignore the details inside an image. In other words, they regard different parts of an image equally, while there are regions which we prefer to be displayed more vividly. The idea of cropping images is also not quite satisfactory, as it totally eradicates parts of an image which may contain valuable details.

Another simple, yet surprisingly smart image resizing technique was introduced in 2007, which enables us to resize images in a way that not only the image size is reduced, but also the main objects are preserved exactly the same. The intuition behind this method, known as **Seam Carving**, is that informative contents inside an image are indicated by higher energy values, which can be calculated through different **Image Filtering** methods. By using an *energy map*, one can simply detect horizontal or vertical *seams* (connected path of low-energy pixels in an image) with lowest energy, and remove them in order to obtain a final resized image.

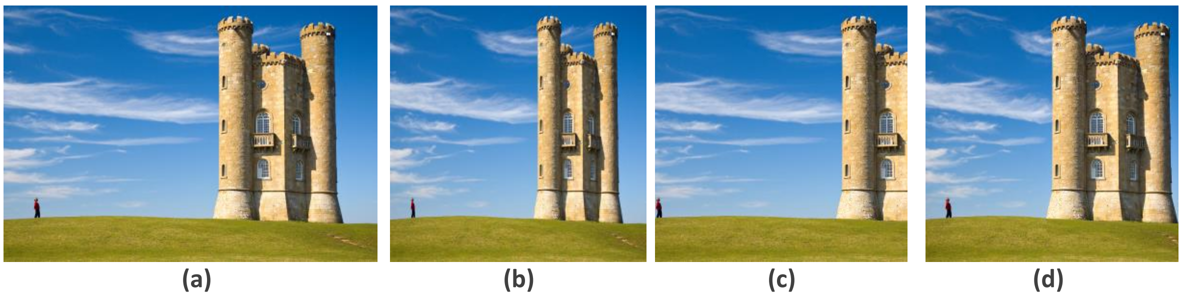


Figure 11 Different resizing strategies applied to an image of Broadway Tower in England (a) Original image (b) Normal image scaling which has led the castle to be distorted (c) The result of cropping, which has caused part of the castle to be deleted (d) Seam carving technique, which has not only reduced image width, but also kept the castle and the man intact

More precisely, Let I be an nxm image and a vertical seam is an 8-connected path in the image from the top to the bottom containing one pixel per row:

$$\mathbf{s}^{\mathbf{x}} = \left\{s_i^x\right\}_{i=1}^{n} = \left\{\left(x(i), i\right)\right\}_{i=1}^{n}, \quad \text{s.t.} \quad \forall i, \ \left|x(i) - x(i-1)\right| \le 1$$

Similarly, the horizontal seam is defined as:

$$\mathbf{s}^{\mathbf{y}} = \left\{s_i^y\right\}_{j=1}^{m} = \left\{\left(j, y(j)\right)\right\}_{j=1}^{m}, \quad \text{s.t.} \quad \forall j, \ \left|y(j) - y(j-1)\right| \le 1$$

Considering the above definitions, the algorithm can be written in the three main steps:

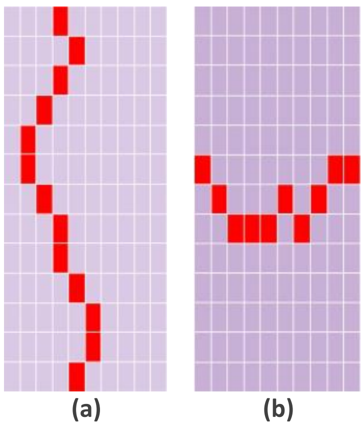


Figure 12 Seams are connected path of pixels in the image (a) Vertical seam (b) Horizontal seam

I. Finding the energy map

Energy function is defined as a kind of function that is related with the content in an image. One intuitive way is to use the **Image Gradient**, since the important contents in an image usually have strong local structures whose gradients are high, while contents that are less important often lie in plain area in the image whose gradients are small. Hence, the energy function is defined as:

$$e_1(\mathbf{I}) = \left|\frac{\partial}{\partial x}\mathbf{I}\right| + \left|\frac{\partial}{\partial y}\mathbf{I}\right|$$

II. Finding the seam with least energy

Given the energy function, the seam cost is defined as:

$$E(\mathbf{s}) = E(\mathbf{I}_{\mathbf{s}}) = \sum_{i=1}^{n} e\left(\mathbf{I}(s_i)\right)$$

in which $I(S_i)$ is one pixel of the seam. The goal is to find the optimal seam which minimizes seam cost:

$$s^* = \min_{\mathbf{s}} E(\mathbf{s}) = \min_{\mathbf{s}} \sum_{i=1}^{n} e\left(\mathbf{I}(s_i)\right)$$

The optimal seam can be found by dynamic programming. Dynamic programming is a programming method which stores the result of sub-calculations in order to simplify calculating a more complex result. In finding the optimal vertical seam, we first traverse the image:

$$M(i, j) = e(i, j) + \min\left(M(i-1), j-1), M(i-1, j), M(i-1, j+1)\right)$$

then we trace back the image from the minimal entry in the last row of the map. The operations for horizontal seam is also similar.

III. Image resizing

So far, a collection of seams, sorted from smallest to highest total energy values across their pixels, have been computed. Assume the goal is to change the size of an image from *n*x*m* to *n*x*m′*, where *m* - *m′* = *c*. This can be easily achieved by successively removing *c* vertical seams from the original image.

The more general situation in which the goal is to resize image from *n*x*m* to *n′*x*m′* requires additional calculations and will be skipped here.

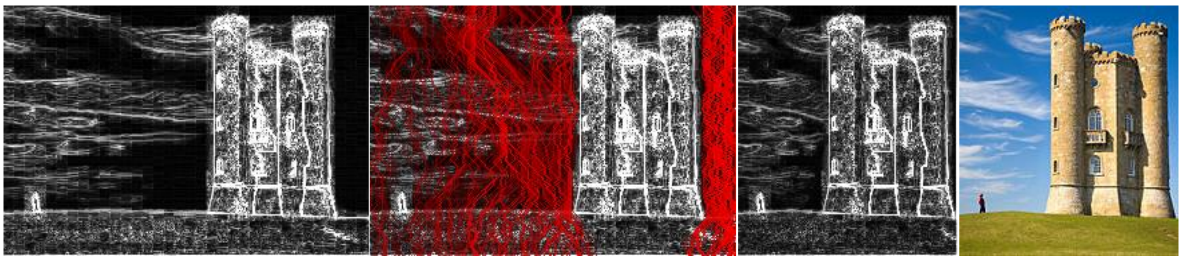


Figure 13 The process of seam carving technique (a) Gradient magnitude of the input image is calculated and used to determine the energy of pixels (b) Using the energy map, a list of seams are obtained and ranked by their energy. Lower energy seams are of lesser importance to the content of the image (c) Removing low-energy seams based on their total energy values across their pixels (d) Final resized image

Now, let's get our hands dirty. Here, your goal is to implement a seam carving algorithm with the following functions:

- `energy_map = find_energy_map(img)`: It takes an RGB image `img`, converts it to grayscale before computing its energy image `energy_map`, which is a double matrix of the same size as `img` and indicates the energy value at each pixel.

- `seams_list = find_seams_list(energy_map, seam_dir, seam_num)`: It takes a matrix `energy_map` as well as a string `seam_dir` which takes values `'horizontal'` and `'vertical'` which specifies seams direction, and a number `seam_num` which determines the number of seams required. The function must return a `seam_num x img_width` matrix (in case `seam_dir` is `'horizontal'`) with seams ordered by their pixel energy values in its rows, or a `img_height x seam_num` matrix (in case `seam_dir` is `'vertical'`) with seams ordered by their pixel energy values in its columns.
  Hint: Seams must be stored as `1 x img_width` (horizontal) or `img_height x 1` (vertical) arrays where, for example, if their 87th element is 136, it means that in the 87th row (or column) the pixel in row (or column) 136 is to be removed.
- `img_resized = remove_seams(img, seams_list, seam_dir)`: It takes color image `img`, matrix `seams_list` and string `seam_dir`, and attempts to remove pixels from the image according to the ordered list of seams and the direction given, to return a final color image `img_resized` which is the smaller version of the input image.

If you feel that you have to make additional assumptions (for example, regarding the overlapping seams), please do it by stating them clearly in your report. Except for the functions mentioned above, you are free to implement the remaining parts the way you want.

a. Reduce the width of the images in part (a) and (b) in Figure 14 by 10%, 25% and 50%. As well as displaying the final result, display the intermediate results including energy maps and selected seams (similar to Figure 13).
b. Reduce the height of the images in part (c) and (d) in Figure 14 by 10%, 25% and 50%. As well as displaying the final result, display the intermediate results including energy maps and selected seams (similar to Figure 13).
c. Find an input image of your own on the web in which this technique obtains interesting successful result (either horizontally or vertically). Display the image as well as the result, and describe why the algorithm seems to work well.
d. Find an input image of your own on the web in which this technique obtains interesting poor result (either horizontally or vertically). Display the image as well as the result, and describe why the algorithm seems to work poorly.
e. Use another energy function, and investigate how the results change when it is used to produce energy map.
f. Use a similar technique to increase the size of an image. You need to implement a function similar to `remove_seams()`, in which instead of removing seams, they are added. Test your algorithm on all the given images, and increase their size by 10%, 25% and 50%.

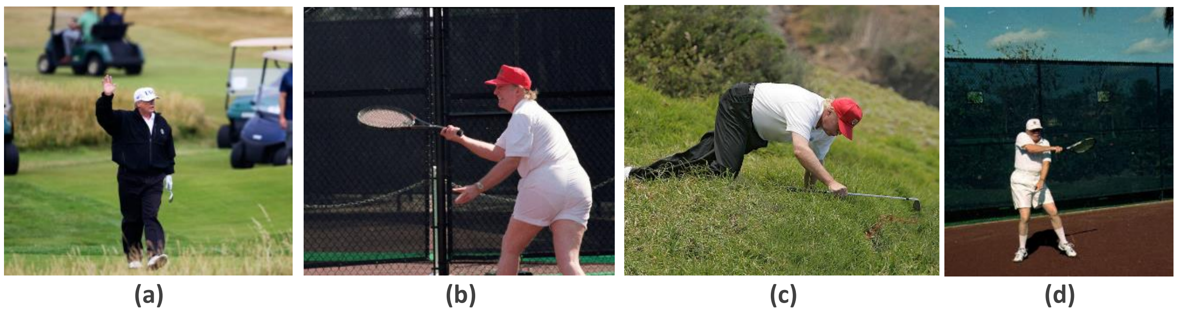


Figure 14 Samples provided for this problem contains regions that are of more importance than the others

Recommended MATLAB Functions: `rgb2gray()`, `gradient()`, `fspecial()`, `imfilter()`

8. Beyond Ordinary Image Filtering: Guided Filters (14 Pts.)

Keywords: *Image Enhancement, Image Filtering, Image Smoothing, Noise Reduction, Guided Image Filtering, Peak Signal to Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM)*

Image Smoothing is a double-edged sword: it efficiently helps reducing noise and artefacts, while it also negatively affects image edges and therefore its fine details. Over the past decades, various methods have been introduced in the literature to diminish noise while preserving as much details as possible.

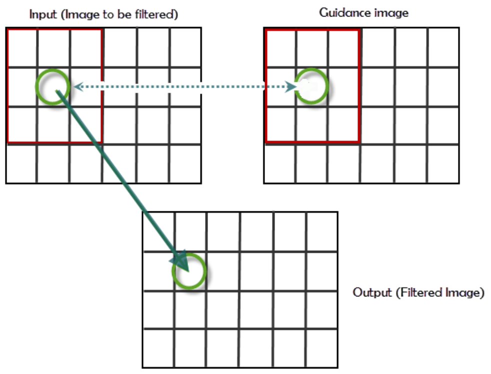


Figure 15 In guided image filtering, the idea is to improve ordinary image filtering by taking into account the statistics obtained from a second image, known as "guidance image"

One of the most recent and fairly successful algorithms of this kind, known as *edge-preserving smoothing algorithms,* is **Guided Image Filtering**, in which the content of a second image, called the *guidance image*, is used to improve filtering result. The guidance image can be the image itself, a different version of the image, or a completely different image. Similar to other image filtering methods, guided image filtering is also a neighborhood operation, yet it takes into account the statistics of a region in the corresponding spatial neighborhood in the guidance image when calculating the value of the output pixel in the filtered image.

The main assumption of the guided filter is a local linear model between the guidance I and the filtering output q. More precisely, q is a linear transform of I in a window w_k centred at the pixel k:

$$q_i = a_k I_i + b_k, \quad \forall i \in w_k$$

where (a_k,b_k) are some linear coefficients assumed to be constant in w_k.

Assuming input image p, guidance image I, square window of a radius r and regularization parameter ε, filtered output q will be obtained through the following steps:

I. $$\text{mean}_I = f_{\text{mean}}(I), \qquad \text{mean}_p = f_{\text{mean}}(p)$$
$$\text{corr}_I = f_{\text{mean}}(I \odot I), \quad \text{corr}_p = f_{\text{mean}}(I \odot p)$$

II. $$\text{var}_I = \text{corr}_I - \text{mean}_I \odot \text{mean}_I, \quad \text{cov}_{Ip} = \text{corr}_p - \text{mean}_I \odot \text{mean}_p$$

III. $$a = \text{cov}_{Ip} \oslash (\text{var}_I + \varepsilon), \quad b = \text{mean}_p - a \odot \text{mean}_I$$

IV. $$\text{mean}_a = f_{\text{mean}}(a), \quad \text{mean}_b = f_{\text{mean}}(b)$$

V. $$q = \text{mean}_a \odot I + \text{mean}_b$$

where $\odot$ and $\oslash$ denote element-wise multiplication and division respectively, and f_{mean} is a mean filter. You may refer to the paper for further explanations and details.

Please implement the above algorithm as a function, and use it in the following parts.

- **Edge-preserved smoothing**. First, we investigate guided image filtering performance in noise reduction and smoothing. Load the image in part (a) of Figure 16, convert it to grayscale, and add a zero-mean, Gaussian white noise with variance of 0.01 to it.

a. Apply guided image filtering to the noisy image with the guidance image identical to the input image. Set $r = 2, 4, 8$ and $\varepsilon = 0.1^2, 0.2^2, 0.4^2$. Display and compare all these 9 experiments in terms of PSNR and SSIM.
Hint: *PSNR (peak signal to noise ratio)* and *SSIM (structural similarity index measure)* are two measuring tools which are widely used in image quality assessment.

b. Investigate the effect of parameters (r and ε) on algorithm performance.

c. Compare (visually and quantitatively) the best result obtained in part (a) with the state in which the initial noise-free image is used as the guidance image, and the parameters are set to the same values.

d. Now apply Gaussian smoothing filter to the noisy image, and find a good set of parameters (*kernel size* and *sigma*) by trial and error. Compare the result with the best result obtained in part (a) both visually and quantitatively.

- **Color filtering.** Now we aim to extend this method to color images. You will work with the image in part (b) of Figure 16.

e. Convert the image to grayscale, and use it as the guidance image to filter the original color image. Find the best set of parameters within the ranges $r = \{2, 4, \ldots, 16\}$ and $\varepsilon = \{0.1^2, 0.2^2, \ldots, 0.4^2\}$.

f. Now use the input image itself as the guidance image to perform guided image filtering. Again, find the best set of parameters within the ranges $r = \{2, 4, \ldots, 16\}$ and $\varepsilon = \{0.1^2, 0.2^2, \ldots, 0.4^2\}$.

g. Compare the results obtained in part (e) and (f) both visually and quantitatively. Explain why the results of these two experiments are different.

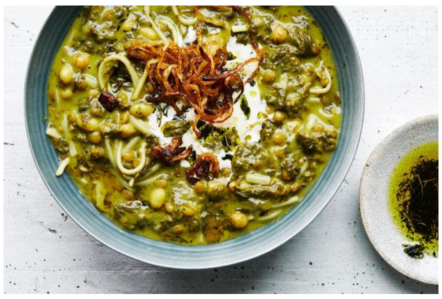
- **Detail Enhancement.** Still not impressed with guided image filtering? Then let's see a cool application of this algorithm. Given the input image p and its edge-preserving smoothed output q as a *base layer*, the difference $d = p - q$ is the *detail layer* and can be magnified to boost the details. The enhanced image is the combination of the boosted detail layer and the base layer:

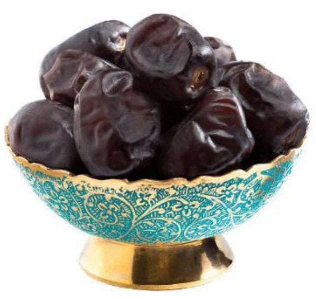
$$p_{\text{enhanced}} = q + \alpha d$$

The input can be found in part (c) of Figure 16.

h. Enhance the details of the input with r and ε set to 4 and 0.2^2 respectively, and $\alpha = 3, 5, 10$.

i. Investigate the effect of parameters (r, ε and α) on the result.


(a)


(b)

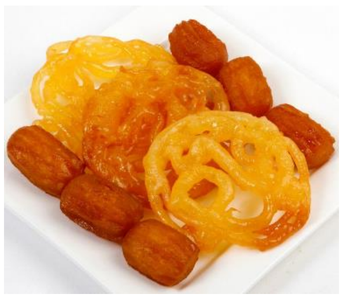

(c)

Figure 16 Input images given for different parts of the problem

Note: Both the input image and guidance image must be scaled in [0,1].

Recommended MATLAB Functions: `rgb2gray(), imnoise(), fspecial(), imfilter(), psnr(),ssim()`

9. Some Explanatory Questions (5 Pts.)

Please answer the following questions as clear as possible:

a. Does repeatedly applying a 3×3 sharpening filter to an image always converge to a certain state? Explain.
b. What is the effect of setting the LSB bit-plane to zero on the image histogram? What about setting the MSB bit-plane to zero?
c. Which one is more efficient: filtering an image with two 1D filters, or filtering it with one 2D filter. Justify your answer.
d. Is it possible to perform binary template matching using image filtering? If yes, design a 3x3 filter to find all plus signs (+) in an image. If no, explain why.
e. Imagine you want to produce a noisy grayscale image from a given color image. What is the difference between first converting the image into grayscale and then adding noise, and first adding noise and then converting to grayscale? Are these two states comparable? Assume the noise parameters equal in both experiments.

Good Luck!
Ali Abbasi