Advanced

Image Processing and Stochastic Modeling

TP1: Basic Image Processing

Prof. Sviatoslav Voloshynovskiy,
Olga Taran,
Maurits Diephuis.

Stochastic Information Processing Group

March 2, 2017

UNIVERSITÉ DE GENÈVE
FACULTÉ DES SCIENCES
Département d'informatique

SIP stochastic information processing

Submission

Please archive your report and codes in “Name_Surname.zip” (replace “Name” and “Surname” with your real name), and upload to “`Assignments/TP1: Basic Image Processing`” on `https://chamilo.unige.ch` before Thursday, March 15 2017, 23:59 PM. Note, **the assessment is mainly based on your report, which should include your answers to all questions and the experimental results**.

1 Building Blocks

Introduction

This exercise will run through some of the basic image commands from the Matlab imaging tool box.

Exercise

If haven’t done so, you should read the Matlab introduction on working with Matrices and Programming.

1. Read the image `'peppers.png'`. The image is present in Matlab.
2. Display the histogram of the image.
3. Convert the image to grayscale.
4. Determine the global mean and the global variance of the image.
5. Determine the local mean and variance of an image for window size 5×5 and display them. See Matlab `blockproc` and `colfilt`. Use distinct non-overlapping blocks.

Noise

This exercise will showcase two noise types, Additive White Gaussian Noise (AWGN) and *salt & pepper* noise.

Additive White Gaussian Noise

AWGN is a type of *channel* in which the discrete channel output Y_i at some time event with index i is the sum on the input X_i and noise Z_i, where Z_i is independently and identically-distributed (i.i.d.) from a zero-mean Gaussian distribution with variance σ^2. Formally:

$$\begin{aligned} &Z_i \sim \mathcal{N}(0, \sigma^2) \\ &Y_i = X_i + Z_i \sim \mathcal{N}(X_i, \sigma^2) \end{aligned}$$

Exercise

1. Write a function that generates an array of size $N \times M$ with Gaussian Noise, i.e. samples drawn from the distribution $\mathcal{N}(\mu, \sigma^2)$. See the Matlab function `randn`.

Salt & pepper Noise

Salt & pepper noise can be defined as follows. Let there be a $N \times M$ gray scale image whose datatype supports a value range of $\{s_{min}...s_{max}\}$. Let $\mathbf{y}$ denote the noisy image and $\mathbf{x}$ the original image. Then the observed gray level in image $\mathbf{y}$ at pixel location (i, j) is given by:

$$y_{i,j} = \begin{cases} s_{min} & \text{with probability } p \\ s_{max} & \text{with probability } q \\ x_{i,j} & \text{with probability } 1 - p - q \end{cases} \tag{1}$$

Exercise

1. Implement a function that generates *salt & pepper* noise with parameters p and q.

Metrics

We will now define and implement two important metrics used in signal and image processing. The Mean Squared Error (MSE) and the Peak Signal to Noise Ratio (PSNR).

Mean Squared Error

The Mean Squared Error (MSE) between two images $\mathbf{x}$ and $\mathbf{y}$ is defined as:

$$\text{MSE} = \frac{1}{N \cdot M} \sum_{i=1}^{N} \sum_{j=1}^{M} (y[i,j] - x[i,j])^2,$$

where N and M are the width and the height of image $\mathbf{x}$ and $\mathbf{y}$.

Exercise

1. Write a function that determines the Mean Squared Error (MSE) between two images $\mathbf{x}$ and $\mathbf{y}$.

2. Read in a new copy of the image `cameraman.tif`, keep it in its original datatype and range, i.e. `uint8` and $\{0..255\}$.

3. Now read in a second copy of the image `cameraman.tif` but map it to `double` and $\{0..1\}$. See Matlab `im2double`. Compare the two images using the MSE. Can you explain the result?

Peak Sigin annal to Noise Ratio

The Peak Signal to Noise Ratio (PSNR) is defined as:

$$\text{PSNR} = 10\log_{10}\left(\frac{\alpha^2}{\text{MSE}(\mathbf{x},\mathbf{y})}\right),$$

where α is the maximum value possible with the `type` of $\mathbf{x}$ and $\mathbf{y}$. The unit of the PSNR is dB.

Exercise

1. Write a function that implements the PSNR function.

Testing

We will test how various noise strengths influence the PSNR value between the original image and the noisy copy.

Exercise

1. Refractor the PNSR definition such that the PSNR is expressed as a function of the noise variance σ_z^2. You may assume that $\sigma_z^2 = MSE(\mathbf{x},\mathbf{y})$.
2. Add Gaussian noise to an image such that the PSNR ratio with the original image is 10dB, 20dB, 30dB and 40dB. Use `randn`, **not** `imnoise`.
3. Show the noisy images on the screen. How do they look?
4. Show the histograms for these noisy images, can you explain what you see?
5. Add salt & pepper Noise to an image until the PSNR ratio between the original and the noisy image is 40 dB. Visually compare it to the 40dB noisy image to which Gaussian noise was added. What can you conclude?

2 Singular Value Decomposition (SVD)

The goal of this exercise to showcase the low-rank approximation on the example of Singular Value Decomposition.

The singular value decomposition (SVD) is a factorization of a real or complex matrix. It is the generalization of the eigenvalue decomposition. Suppose A is a $n \times d$ matrix of real numbers. Then there exists a factorization of A in the form:

$$A = USV^T$$

where

- U is a $n \times n$ orthogonal matrix;
- S is a diagonal $n \times d$ matrix with non-negative real numbers on the diagonal;
- V is a $d \times d$ orthogonal matrix.

Low-Rank Approximations from the SVD

Given an $n \times d$ matrix A and a target rank $k \geq 1$, we produce a rank-k approximation of A as follows (see also Fig. 1):

1. Compute the SVD of A : $A = USV^T$;
2. Keep only the top k right singular vectors: set V_k^T equal to the first k rows of V^T (a $k \times d$ matrix);
3. Keep only the top k left singular vectors: set U_k equal to the first k columns of U (a $n \times k$ matrix);
4. Keep only the top k values: set S_k equal to the first k rows and columns of S (a $k \times k$ matrix);

The computed low-rank approximation is then:

$$A_k = U_k S_k V_k^T.$$

Exercise

1. Read the image `'peppers.png'` and convert it to grayscale. Perform its low-rank approximation for $k = 1, ..., n$. Plot the dependence between the k and MSE of k-rank approximation version of original image. Make a conclusion.
2. Read the image `'peppers.png'` and convert it to grayscale and add Gaussian noise $\mathcal{N}(0, 625)$. Perform its low-rank approximation for $k = 1, ..., n$. Plot the dependence between the k and MSE of k-rank approximation version of original image. Make a conclusion.

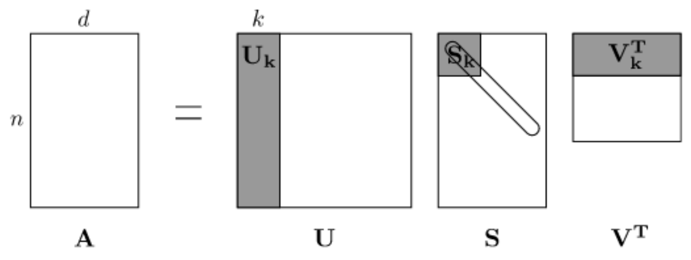


Figure 1 – Low rank approximation via SVD.

3 Noise Visibility Function (NVF)

The Noise Visibility Function (NVF) describes noise visibility in an image. The most known form of NVF is given as:

$$NVF(i,j) = \frac{1}{1+\theta\sigma_x^2(i,j)}$$

$$\theta = \frac{D}{\sigma_{x_{max}}^2}$$

where $\sigma_x^2(i,j)$ denotes the local variance of the image in a window centred on the pixel with coordinates (i,j), θ plays the role of contrast adjustment for every particular image, $\sigma_{x_{max}}^2$ is the maximum local variance for a given image and D is an experimentally determined parameter.

The final embedding equation is:

$$y_{i,j} = x_{i,j} + (1 - NVF)z_{i,j}$$

where $z \sim N(0, \sigma_z^2)$ denotes a watermark.

Exercise

1. Read the image `'lena.png'` and convert it to grayscale.
2. Add a watermark to the image with and without applying NVF function the different values of σ_z^2 (10, 25, 50, 75) and D. Choose the window size appropriate to used image. What can you say about the impact of NVF function?
3. Report the dependency between the parameters σ_z^2, D and original image.

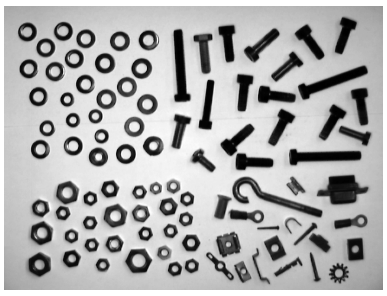
4 Nuts and bolts

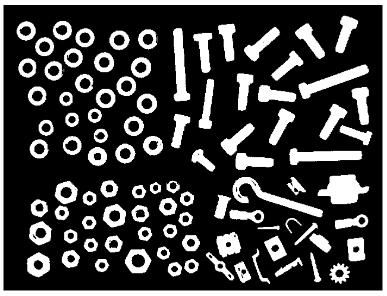
In this little project you will design and test a program that can recognize various nuts and bolts in an image using Matlab's *morphological* functions and a bit of statistics. The image can be seen in Figure 2. Matlab contains an excellent tutorial segmenting and counting rice in an image, which you can work through as preparation.[1]

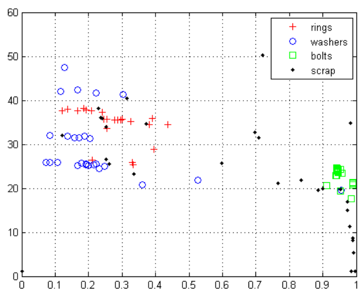
The principle steps that need to be done are the following:

- Segment the foreground which contains all parts, from the background. You can use morphological opening, e.g. `imopen` to ascertain background statistics or use the so called *Otsi's* method implemented by Matlab `graythresh`.
- Use morphology to remove any noise from the image
- Select all individual items using Matlab's `bwlabel` and `bwconncomp`.

[1]`http://www.mathworks.ch/ch/help/images/image-enhancement-and-analysis.html`


(a) Source picture


(b) Fore- and background



(c) Separation based on two `regionprops` statistics

Figure 2 – Recognizing nuts and bolts

- To gather statistics deploy Matlab's `regionprops` function. It is capable of collecting a vast amount of information on binary objects which in term can be used to distinguish the various parts from each other.
- Find a combination of metrics to separate the different parts as best as possible.

Exercise

1. Implement the image segmentation and statistics gathering functions
2. Report on what statistics work and why (not).