Image Processing Using Element-Wise Processes

Summery of chapter 2 and 3

Sara Awwad, Bhaskarjyoti Sarma, Dhananjay Mandalkar

Bergische Universität Wuppertal, Germany
December 9 2020

1. Introduction

In image processing we deal with digitised images presented by matrices, and we can extract useful pieces of information from images by applying simple operations on these matrices. In this paper we will discuss briefly some useful matrix operations and their applications, alongside a few examples.

2. Chapter2

2.1. The human perception

This section discusses functional components of visual systems, i.e. how human eyes see. **First** component is the translation or conversion of incident light into nerve impulses through the eye. Incident light travels from the object into eyes as electromagnetic wave in the visible spectrum. This light is then converted into nerve impulses through the eye. **Second** functional component is transmission and distribution of these nerve impulses along nerve fiber pathways through the brain to the occipital. **Third** component is the processing and interpretation of the information carried by impulses in visual cortex. This process also means that losing connection between any nerve fiber pathways leads to loss of sight.

The eye is one of the most important part in the visual perception system. In certain aspects, it can be considered similar to a camera. The most important elements in the eyes are the light receptors, distributed over the surface of the retina. Depending on their functionality and distribution location, they are are called **cones** or **rods**. Cones are highly sensitive to colors and give us the bright-light vision, while rods are sensitive to low levels of illumination and are responsible for black and white sight.

It is important to point out that there is a difference between what enters the eyes in reality and what our visual system sees or rather perceives. Perception of our visual system is not a simple measurable function. It is about how we perceive gray level edges in the context of colors, tones, and brightness etc. This is very important in the medical field because while presenting an image for diagnostics, we have to make sure that the image is free from perception-bias and the diagnosis is not affected by this.

2.2. Coordinate systems

The coordinate system of an image differs from the conventional system that we use in the linear algebra. In an image $(0, 0)$ starts at **the left-top corner**, although there are some certain software-packages that have the origin somewhere else. We always need to be careful at the beginning about where the origin is.

2.3. Representing digital images

A digital image is represented numerically as an $m \times n$, $2D-$matrix, where m is the number of rows and n is the number of columns. Each element in the matrix has gray values represented as $f(x, y)$, where x and y are the spatial coordinates, x being along the rows and y being along the columns. $f(x, y)$ can also be represented as $a(i, j)$ where i is the row-index and j is the column-index. For example, $a(i, j) = 0$ for a black pixel and $a(i, j) = 1$ for a white pixel (In black and white scale).

2.4. Element-wise versus Matrix Operations

Processing of an image, which is numerically a matrix,means doing operations on these matrices. Therefore, we need to define these operations. In image processing matrices are often processed element-wise unlike in matrix theory. An element-wise multiplication of two matrices A and B means that $a(i, j)$, an element of matrix A is multiplied with $b(i, j)$, an element of matrix B. However, there are many situations in which operations between images are carried out using matrix theory. Therefore we need to make a clear distinction between matrix inner operations, and element-wise operations.

2.5. Linear versus Non-linear Operations

Operations done on an image-matrix can be classified into either **linear** or **non-linear**. Let us consider two arbitrary images $f_1(x, y)$ and $f_2(x, y)$ and an operator $\mathscr{H}(x, y)$ operating on f_1 and f_2 such that the output images are g_1 and g_2 respectively, then the operator is linear if it holds the following condition for two arbitrary constants a and b:

$$\begin{aligned} \mathscr{H}[af_1(x, y) + bf_2(x, y)] &= a\mathscr{H}[f_1(x, y)] + b\mathscr{H}[f_2(x, y)] \\ &= ag_1(x, y) + bg_2(x, y) \end{aligned}$$

If the above condition is not fulfilled, then it is a non-linear operation. There are several possible element-wise operations between two images of the same size, such as: addition, subtraction, multiplication and division.

2.6. Masking

This method has several applications such as: eliminating noise or background in an image, distinguishing the difference between two images, and many other applications. It is based on element-wise subtraction, or product between two matrices. In order to demonstrate this method, we implemented two examples, the first one is very simple, and it uses masking to find the differences between two images. In figure 1, we can see two images with some small differences between them ($C = A - B^{-1}$).

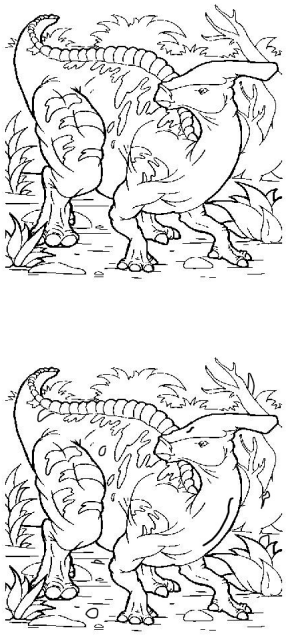


Figure 1: find the differences between two images (Matrix A B)

Using MATLAB, we implemented an algorithm that inverts one of the images and then subtracts them, which gives us the result as shown in figure 2.

```
[B,Map]=imread('dinasour1.jpg'); %reading the image
[C,Map]=imread('dinasour2.jpg'); %reading the image
B = rgb2gray(B); %converting to grayscale
C = rgb2gray(C); %converting to grayscale
figure; imshow(B) %showing the original image
figure; imshow(C) %showing the original image
B(525,:,:)=[];
black=B<150;
white=B>=150;
B(black)=1000;
B(white)=0; %inverting B
black1=C<150;
white1=C>=150;
C(black1)=0;
C(white1)=1000;
D=C+B;
figure; imshow(B)
figure; imshow(C)
figure; imshow(D)
```

With this simple method we easily extracted differences that are considered to be difficult to recognize by the human eye.

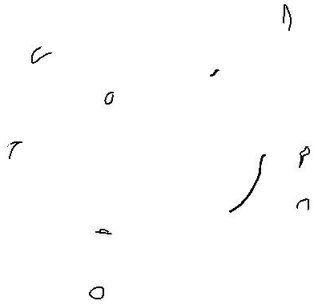
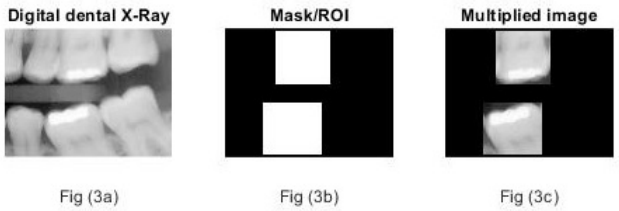


Figure 2: the differences between two images (Matrix C)

This method is widely used in the medical sector, where we can implement it to see some certain veins or organs, before and after an injection that includes a coloring agent.

Masking is also known as the region of interest (ROI). In Fig (3a) an image of a digital dental X-Ray is masked with Fig (3b) ROI for isolating teeth with fillings. Fig (3c) is the multiplication of Fig (3a) and Fig (3b). Below, a MATLAB code implemented to achieve this masking task.

```
A=imread('teeth.tif');
I1=uint16(A);
B=imread('mask.tif');
I2=uint16(B);
C=immultiply(I1,I2);
subplot(1,3,1);imshow(A);xlabel('Fig (3a)')
title('Digital dental X-Ray');
subplot(1,3,2);imshow(B);xlabel('Fig (3b)')
title('Mask/ROI');
subplot(1,3,3);imshow(C);xlabel('Fig (3c)')
title('Multiplied image');
```



2.7. Range scaling

One accompanying problem with arithmetic operations such as subtraction between images, is that it may lead to pixel values that are beyond the range permitted by an 8-bit or 24-bit image. When that allowed range is exceeded, operations such as clipping or scaling are necessary. Clipping or scaling can be avoided by the following approach, in order to ensure that the values lie in a predefined range $[0, K]$. Let us consider an image g, with a minimum pixel value of g_{min}. Then $g_m = g - g_{min}$ ensures that the minimum value of g_m is 0; where g_m represents an image g with a minimum pixel value of zero. In order to have the scaled image in the range $[0, K]$, the following operation is

performed:

$$g_s = K\left[\frac{g_m}{\max(g_m)}\right]$$

3. Chapter 3

3.1. Intensity Transformations and Spatial Filtering

In this chapter, we discuss how one aspect of an image e.g. contrast can be enhanced by doing intensity transformations in spatial regions. The word spatial means the *x*,*y* image plane, and pixels are directly manipulated in this category of transformation. Spatial transformations can be broadly classified into three categories depending on the regions that is considered for carrying out transformations in a particular pixel. They are (i) single-pixel operations, (ii) neighbourhood operations and (iii) geometric spatial transformations. In single pixel operations, output intensity in a pixel is a function of intensity of that pixel only. However, in neighborhood operations, output intensity is a function of the intensities of neighboring pixels around that pixel. Geometric spatial transformations are much more complex transformations and will be discussed in later lectures. Mathematically a general intensity transformation can be represented as $g(x, y) = T[f(x, y)]$; where $g(x, y)$ is the output image, $f(x, y)$ is the input image and T is an operator that acts on f defined over a neighborhood of points around (x, y). A neighborhood transformations can be carried out e.g. by having a 3×3 matrix where the pixel at the center of this matrix will be transformed and this transformation is a function of intensities in all the elements of the matrix. Then this 3×3 matrix (also called as *Kernel*) is scanned over the image, pixel by pixel to transform each pixel of the original image. An example of this transformation is an averaging filter. In an averaging filter output intensity in a pixel after transformation is the average of intensities in other elements in the kernel. An averaging filter gives a blurring effect to the image.

Another example of intensity transformations is the **contrast stretching function**, where an image of higher contrast is generated by darkening image the levels below a threshold and brightening above this threshold. A simple function for this transformation is the **thresholding function** that results in a binary image.

There are some specific functions that are frequently used for intensity transformations. They are: (i) linear, (ii) logarithmic and (iii) power law. A simple linear transformation is the transformation where output intensity remains the same as the input intensity. Another linear transformation can be a negative transformation where dark pixels are turned into bright and bright pixels are turned into dark (as done in the first example). Logarithmic transformations are used to widen the lower intensity ranges and to compress the higher intensity ranges. Exactly the opposite can be done with inverse logarithmic transformations. Power law (gamma) transformations with γ values less than 1, do the same as logarithmic transformations, while with γ values greater than 1, they do the same as inverse logarithmic transformation. Power law transformations are more versatile compared to logarithmic transformations because; here γ values can be adjusted as per our need. Gamma transformations can be used to enhance local contrast.

We now see Spatial operations performed on neighbourhood operations. We perform local averaging by applying neighbourhood procedure. This method is applied to an aortic angiogram Fig (4a) with $m = n = 41$ and results can be seen in Fig (4b). The net effect on Fig (4a) provides local blurring which eliminates minor details and provides pieces of information about clotting.

Below, you can find a MATLAB code implemented to achieve the blurring task.

```
org_img = imread('kidney.tif');
nbhood = 41; % Neighbourhood
kernel = ones(nbhood)/(nbhood^2);
blur_img = imfilter(org_img, kernel,'conv');
subplot(1,2,1);imshow(org_img);xlabel('Fig (4a)')
title('An aortic angiogram');
subplot(1,2,2);imshow(blur_img);xlabel('Fig (4b)')
title('The result of blurring');
```

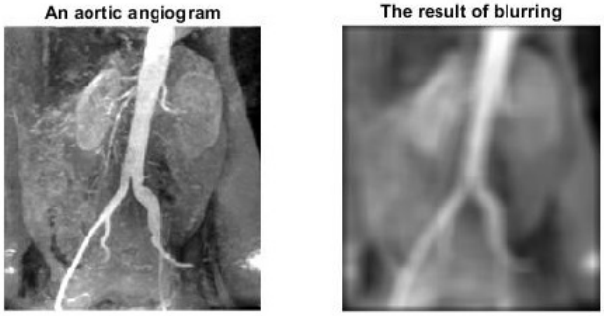


Fig (4a) Fig (4b)

3.2. Using piece-wise linear intensity transformations to highlight a kidney stone in an X-Ray

This method allows us to highlight a certain region in an image by boosting the intensity range of that region. We used an X-Ray of a kidney stone as an example, where the original image is shown in Figure(5).

The objective of the following implementation is to highlight the kidney stone and consider everything else as noise.

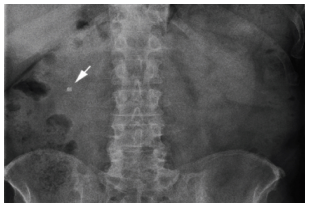


Figure 5: Original X-Ray image of a kidney stone

We determined the intensity range that the kidney stone image consists of, and then we boosted that range while keeping the other intensities unchanged as shown in **figure 5**.

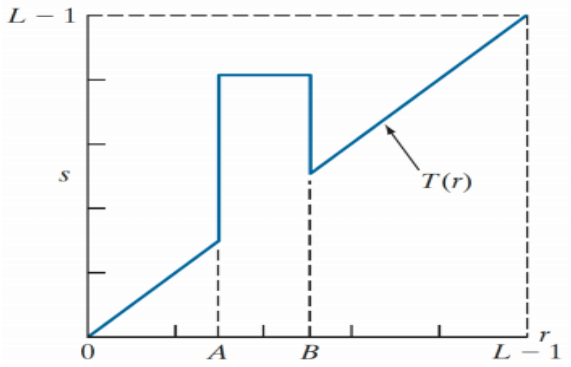
In the next stage we dimmed the intensities outside the range while boosting the intensities inside the range as shown in **figure 5**.

Again, a simple code in MATLAB was able to analyse this image and highlight the kidney stone.

```
[A,Map]=imread('StoneImage.jpg'); %reading the image
A = rgb2gray(A); %converting to grayscale
figure; imshow(A) %showing the original image
%% stage(1) peaking the values that are within the
    range only
color=A(:,:,1);
a=color>137 & color<155;
A(a)=1000;
figure; imshow(A)

%% stage(2) dimming the rest of the picture

A(~a)=100;
figure; imshow(A)
```



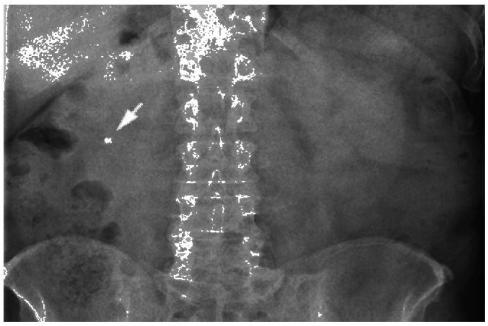
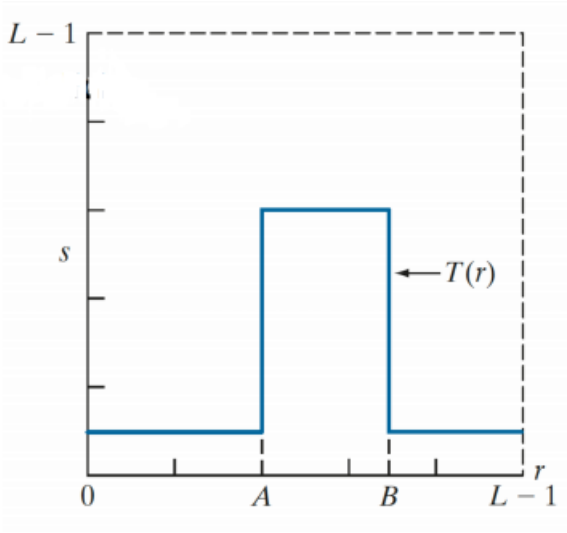


Figure 6: Boosting the intensity of a chosen range while keeping the intensities outside the range unchanged

3.3. Histogram processing

A histogram of an image gives us an overall first idea of it's characteristics and ideas on how to further optimize the image. A histogram is basically a plot where along horizontal axis we have the intensities from 0 to $L-1$ divided into bins and along the $Y-$axis we have the number of occurrences of these intensities in the pixels. A histogram can be normalized by dividing this number of occurrences by the total number of pixels. This means that the number of occurrences plotted along the vertical axis is the probability of an intensity, such that the summation of all probabilities is one.



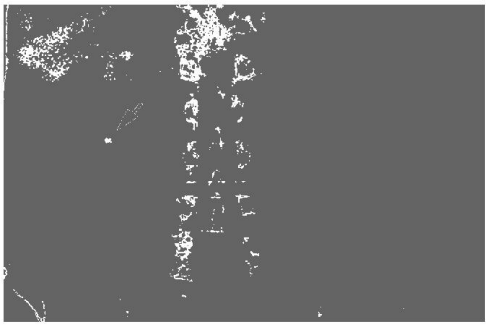


Figure 7: Boosting the intensity of a chosen range and reducing the intensities outside the range

4. Summery

In this paper we presented some of the basic concepts in image processing from our own prospective and were able to implement them in many examples. From our study, we concluded the following:

- Human perception plays an important role in the way we analyse images, and we should always make sure that our implementations are perception-bias free.
- There's a difference between linear algebra coordinates and digitised image coordinates
- Simple implementations of element-wise operations can be used as powerful tools to analyse images
- Images can be displayed based on intensity levels statistics using binning, where we can extract useful pieces of information about the intensity levels from this method.
- Manipulating the intensity levels in an image can allow us to highlight, dim or change some features in an image.