Introduction to digital images & Matlab, spatial and intensity resolution and thresholding

Contents

1 Assessment 3

2 Optional Exercise for Development Phase 1 4

3 Report Writing Instructions 4

4 Matlab fundamentals 4

5 Digital images in Matlab 4
   5.1 Quantization and dynamic range 4
   5.2 Spatial resolution 5

6 Point operations and lookup tables 5

7 Histogram-based operations 6
   7.1 Contrast enhancement 6
   7.2 Thresholding 7

8 Blurp Box Incorporated 8

9 Reporting Assessment 8
Background - An Industrial Need

You are a student intern engineer in a research and development company and your technical supervisor has given you the task of developing an image analysis system for detecting objects on an industrial conveyor belt.

Blurp Box Incorporated is an OEM manufacturer of wooden parts of different shapes and colors and they require an industrial vision system to identify and classify these parts correctly. Lately a number of misplaced tools have begun appearing on their conveyor belt such as hammers, spanners, and screwdrivers to name a few.

The company manager suspects sabotage and wants the imaging system to be able to detect and accurately classify these misplaced tools. Detection will allow the belt to be automatically stopped and the tool removed, and accurate classification might provide the necessary statistics to determine which work stations in the plant the tools are likely to be coming from, thus assisting in the investigation.

Production speed has had to be reduced so that manual inspection for tools can be performed. Whilst this has so far prevented any catastrophic damage to the machinery due to a machine swallowing a metal tool, production has had to be reduced to levels that will soon threaten their ability to deliver product. The manager estimates that if an automatic system cannot be developed in 8 weeks then they will have to start canceling orders and risk losing customers to their competitor.

It is your job to develop the algorithms and techniques to detect and classify the shapes. A local vision systems company, MUV Systems is installing the necessary hardware (cameras, computers) and performing the integration with the conveyor belt stop system. They need the analysis algorithms to be developed in MATLAB and in 8 weeks they expect to integrate your algorithms into their system for online operation.

MUV Systems has been working with Blurp Box Incorporated and has already collected images of all of the kinds of objects you are likely to have to detect and classify.

Your technical supervisor has broken the development task down into eight phases. The first seven phases are training and component development phases where you will both learn the necessary skills to develop the system components. The eighth phase involves the integration of all the developed components into a working analysis system. The development phases are scheduled to be completed one per week for eight weeks and you are expected to deliver the completed results of each development phase to your technical supervisor on time. It is critical that the system is delivered on-time.

For each development phase you need to deliver the source code and a clear written report that your supervisor can use to repeat exactly what you have done and generate similar results.

Good luck.
Introduction

Read all of the instructions in this document before starting.

The goal of this Development Phase 1 is get acquainted with digital images in Matlab, apply and learn various techniques in pixel operations and histograms, and begin to develop components so solve Blurp Box Incorporated’s needs.

The meaning of spatial and intensity resolution of digital images will be discussed and some histogram-based operations will be used.

See the course web page for information on how to setup MATLAB search paths to find source files that you will need.

You need to write a clear and reasoned report outlining how and what you have done. You technical supervisor will be reading your report and expects to be able to follow what you have done and use your instructions to recreate your results.

It will be helpful to collect your commands in a script file, or in functions, since you will need to reuse parts of your program and analyse many different parts with the same code.

Remember that in the end you will need to integrate all of the material from the seven different development phases into your final imaging system product.

1 Assessment

Please read this assessment criteria carefully. Ask questions if anything is unclear.

Each Development Phase will be graded fail (U), pass (3), good (4), or very good (5) based on the following criteria;

1. How well does the report answer all the questions outlined in the development plan. Does the student provide a thorough and complete answer?

2. How clearly written is the report and how well reasoned is the explanation? How easy would it be for a technically qualified person to understand what you have done and why, and then follow the development phase instructions and your report and repeat the results?

3. When you are asked to explain or discuss something Explain why your result is this way in relation to the theory from the course material (necessary in order to achieve a grade of 4 or 5)

4. Have the optional parts of the Development Phase been answered and how thoroughly? (necessary in order to achieve a 5 grade)

2 Optional Exercise for Development Phase 1

The optional exercise for Development Phase 1 is to spend extra effort getting familiar with MATLAB. Focus on understanding how to use MATLAB, follow the instructions for making
functions, using loops, writing code you can reuse for different images, and learn how to save your images and your figures. This will make it much easier to write your report. See how you can improve your work through better use of MATLAB.

3 Report Writing Instructions

Follow the guidelines in the report writing template, ReportTemplate.doc. If you want to write your report in latex or something else then that is fine too, just make the format similar and follow the guidelines.

4 Matlab fundamentals

Refer to the complimentary document MATLAB-Tips.pdf for help in getting started with MATLAB.

5 Digital images in Matlab

5.1 Quantization and dynamic range

Load and view the image apple2.tif.

\[ I_{apple} = \text{double}(	ext{imread('apple2.tif'))}; \]

Every pixel \((i, j)\) in the image has a possible grayscale value \(g\) in the range \([0\ldots255]\), i.e. the max value \(2^k - 1\) is where \(k\) is the number of bits in the digital grayscale representation, in this case 8.

In MATLAB you can access a given image pixel at row \(i\) and column \(j\) by simply typing \(I_{apple}(i,j)\) where \(I_{apple}\) is the image from the file apple.tif.

In order to study the effects of a reduced graylevel range (coarser quantization), construct and view the set of images

\[ \text{quantize}(I_{apple}, k) = \text{round} \left( I_{apple} \times \frac{1}{2^k} \right) \]

for \(k = 8, 7, \ldots, 1\). Note: this function cannot be directly written as it is expressed here. Create a MATLAB function \(\text{Out} = \text{quantize}(I_{apple}, k)\) to create a new image given an input of an image and a value of \(k\). Use a \textbf{for} loop in MATLAB to call the function \text{quantize} for all your values of \(k\).

\(\text{quantize}(I_{apple}, 8)\) will be a binary image (consisting of values 0 and 1) and cannot be viewed as it is. In order to see anything on the screen when an image consists of only low-valued graylevels we need to improve the contrast in the image before viewing. We can do this by contrast stretching, that is stretching the range of intensities values in the image.

Perform this contrast stretching by implementing a MATLAB function \(\text{Out} = \text{rescale}(\text{Im})\) that will spread out the intensities in the input image \(\text{Im}\) across the whole range of intensities
from 0 to 255.

The input image \( \text{Im} \) will have intensities in the range \([\text{minval}..\text{maxval}]\) where minval is the minimum value calculated by \(\min(\min(I))\), and maxval is the maximum value calculated by \(\max(\max(I))\). When the image is rescaled all the input intensities must be remapped to new values so that the output image has values spread out across the whole range of intensities from 0 to 255.

Refer to the document \textit{MATLAB-tips.pdf} for how to make a function.

You can calculate and view your images with \(\text{image(rescale(quantize(Iapple,k)))}\).

**Reporting**

Supply your (well commented) Matlab code for \texttt{rescale}.

What is the largest value of \(k\) that you can you not perceive any difference compared to the original image? Show the original image and the image for this value of \(k\).

### 5.2 Spatial resolution

Load and view the image \texttt{shapes.tif}.

\(\text{Ishapes = double(imread('shapes.tif'));}\)

Construct a binary image \texttt{bShapes} (consisting of values 0 and 1) from shapes through thresholding (as described in the document \textit{MATLAB-tips.pdf}). Use \(T_L = 50\) and \(T_U = 255\).

Extract the object borders through the operation \(\text{eShapes = double(bwperim(bShapes))}\).

We will discuss this operation later in the course and just apply it here without details.

In order to study the effects of a reduced spatial resolution (smaller \((\text{size}\_I, \text{size}\_J)\)), construct and view the object borders for the set of images \(\text{dShapes = eShapes(1:k:128,1:k:128)}\), for \(k = 2,3,4,...\).

**Reporting**

Which value of \(k\) can no longer clearly determine the original shapes. Show the image that corresponds to that \(k\) value.

### 6 Point operations and lookup tables

Load and view the image \texttt{pliers.tif}. A lookup table (LUT) is a fast and convenient way to perform grayscale mappings and is often realised in hardware on the digitizing board (frame grabber). The original grayscale \(\text{Im}(i,j)\) is mapped to a new grayscale \(\text{Out}(i,j)\) through the operation \(\text{Out}(i,j) = \text{LUT}(\text{Im}(i,j))\) as outlined in figure 1 below.

Implement a MATLAB function \(\text{Out = applyLUT2(Im,LUT)}\) where the argument \(\text{Im}\) is the original image, \(\text{LUT}\) is an arbitrary integer vector of length 256 and \(\text{Out}\) is the new grayscale mapped image.

Try your function using \(\text{Im}\) as the image loaded from \texttt{pliers.tif} and a LUT that performs the operation \(\text{Out}(i,j) = \text{round}(c \cdot \log(\text{Im}(i,j) + 1))\) (choose the constant \(c\) so that 255 is mapped on 255).
Reporting

Supply your (well documented) MATLAB code for applyLUT2, the mapped image and a plot of your LUT. Explain what this mapping actually does using the theory from the course material.

The document MATLAB-Tips.pdf tells you how to print a figure to a file so you can easily save it.

7 Histogram-based operations

7.1 Contrast enhancement

Load and view the image satim.tif

Isatim = double(imread('satim.tif'));

Construct and plot the image histogram as described in the document MATLAB-Tips.pdf. The very prominent 'hill' at the lower end of the histogram is typical for low-contrast images like this one. One common approach for enhancing the contrast in an image is histogram equalization (discussed in Digital image processing 3.3.1). The strategy here is to perform a grayscale mapping $\text{Out} = \text{histeq(Im)}$ (where the argument Im is the original image and Out is a new contrast enhanced image) so that the histogram of the new image Out resembles a uniform (or rectangular) distribution.

Create a histogram equalized image in MATLAB with

$\text{>> heSatim} = 255*\text{histeq(satim/255,256)}$

and study the resulting image and its histogram.

One other intuitively appealing strategy would be to stretch the 'hill' in the original histogram so that it covers the whole range 0-255. This can be performed with a histogram-stretching lookup procedure (discussed in Digital image processing 3.2.4 and lecture notes 1).
Construct a lookup table as described in figure 2 below. Chose the values for the parameters $\alpha$ and $\beta$ based on the 'hill' in the original histogram. Use applyLUT2 and study the resulting image and its histogram.

Figure 2: Contrast stretch lookup table.

Reporting
Supply your contrast image, a plot of its histogram and a plot of your lookup table. Explain the results of contrast enhancement using histeq and applyLUT2 using the theory from the course material.

7.2 Thresholding
Load and view the image block.tif.
Iblock = double(imread('block.tif'));
Construct and plot the image histogram as described in the document MATLAB-Tips.pdf. Use the 'hills' in the histogram to find threshold values and try to segment the original image into as many large connected regions as possible.

Construct an image where the different connected regions has been given different graylevels.

Reporting
Supply your segmented image and a plot of the image histogram where you have marked your choices of thresholds. Refer to the document MATLAB-Tips.pdf for how to draw lines on an existing figure.

8 Blurp Box Incorporated
Using the knowledge you have gained from the above exercises you are now ready to create your first component for the Blurp Box Incorporated problem.

Load images taken from a conveyor belt using these commands:
What you see is a color image frame containing a blurp box, captured by MUV-Systems with a camera mounted right above the conveyer belt. We will make use of the color information later, but for now we will only consider the gray scale representation of the image. This representation is found with the command (we will discuss this operation in lecture 7 later)

```plaintext
g >> boxG = 255*rgb2gray(box/255);
g >> imagesc(boxG/255), colormap(gray(256))
```

The data structure `conveyor` contains a colour frame of the conveyer belt without any blurp boxes or misplaced tools. Thus the histogram for its gray scale representation may be analysed separately. Find suitable thresholds $T_L$ and $T_U$ from the histogram and use this information to create a binary image $(\text{box}<T_L) + (\text{box}>T_U)$, that hopefully contains only the blurp box (and perhaps some noise).

**Reporting**

Try this strategy on 20 realisations of `box` and comment on the result. Once you can do this for one box, put your code inside a `for` loop so you can easily repeat it for 20. You can use all the suggestions in the document *MATLAB-Tips.pdf* to save display and even print to file each box image you create and threshold.

Is this a reliable method? If not, why?

**9 Reporting Assessment**

Using the criteria defined in the Assessment section, assess your own development report and write a short evaluation of how well it satisfies the criteria?

Are there any problems with your solution? That is, situations where it doesn’t work as expected? Your technical supervisor will need to know about these situations so that they can be solved and don’t cause unexpected problems later during system integration and installation in week 8.

The purpose of this assessment question is for you to evaluate your own work like you would have to in a real work situation before handing it over to your technical supervisor.