Simple Pixel Operations
4S1

Dr. Anil C. Kokaram,

Electronic and Electrical Engineering Dept.,

Trinity College, Dublin 2, Ireland,

anil.kokaram@tcd.ie

Overview

- Range of simple pixel operations allows fast enhancement/manipulation
- Contrast stretching, Gamma correction
- Image Histograms
- Image Comparision: Simple Change Detection

Point Operations

- Idea is to transform each pixel in the image to achieve some effect
- $g(h,k) = f(I(h,k))$ Where $f(\cdot)$ is some pointwise operation on the data at a single pixel site, and $g(\cdot)$ is the output of the operation at that site.
- A zero memory operation, input range $[0, 255]$ output range $[0, 255]$ for grey scale
- Contrast Stretching, 'Gamma' correction, thresholding, clipping are all pointwise operations

Digital Image as a Matrix

$$\mathbf{I} = \begin{bmatrix} I(0,0) & I(0,1) & \dots & I(0,M-1) \\ I(1,0) & I(1,1) & \dots & I(1,M-1) \\ \vdots & \vdots & \dots & \vdots \\ I(N-1,0) & I(N-1+1,1) & \dots & I(N-1,M-1) \end{bmatrix} \quad (1)$$

So using Matlab notation, $\mathbf{G} = 2.0 * \mathbf{I}$ scales image by a factor of 2. $\mathbf{G} = \mathbf{I} + 10$ increases all intensities by 10. Pointwise operations are easy (and fast!) in Matlab if you treat the image like a matrix.

Many companies build 'computational blocks' that perform functions like these in hardware and software. Alot of the times more complicated image processing algorithms can be *assembled* by joining up these blocks. Shifting, Clipping, Thresholding, Scaling are all very useful re-usable blocks.

Histograms

- An image histogram records the number of pixels which have a particular intensity value
- Typically displayed as a plot of frequency Vs. intensity (or level, or value).
- Shows the distribution of intensities across the WHOLE image. Could be normalised so that integrates to 1 (an approximation of the p.d.f. of the image intensities) by dividing by total pixels in image.
- Useful for working out where most of the information is and designing enhancement operations.
- Very useful for comparing images on the basis of overall 'look' or 'feel'. Content retrieval on basis of redness etc etc.

Thresholding

- Sometimes it is required to extract a part of an image which contains all the information. This is a *part* of the more general segmentation problem.
- **IF** this information is contained in an upper or lower band of intensities then we can threshold the image to expose the location of this data

$$g(h,k) = \begin{cases} 0 & \text{For } I(h,k) < t \\ 255 & \text{Otherwise} \end{cases} \qquad g(h,k) = \begin{cases} 0 & \text{For } I(h,k) < t \\ I(h,k) & \text{Otherwise} \end{cases} \tag{2}$$

Pixels having an intensity lower than t are set to 0 and otherwise the pixels are set to 255 or left at their original intensity depending on the effect that is required.

Intensity Slicing (Intensity based segmentation)

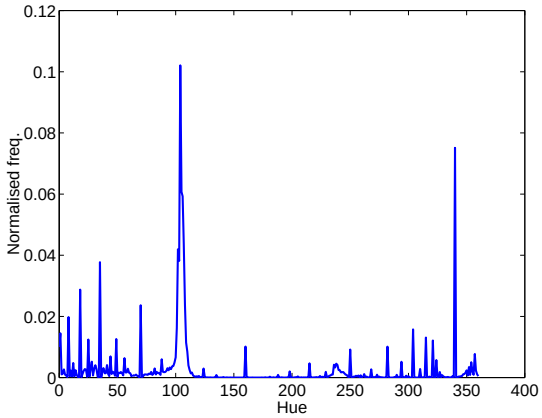
- **IF** relevant information is contained in a mid-band of intensities then we can *slice* the image to expose the location of this data

$$g(h,k) = \begin{cases} 0 & I(h,k) < t_1 \\ 255 & t_1 < I(h,k) < t_2 \\ 0 & \text{Otherwise} \end{cases} \qquad g(h,k) = \begin{cases} 0 & I(h,k) < t_1 \\ I_(h,k) & t_1 < I(h,k) < t_2 \\ 0 & \text{Otherwise} \end{cases} \tag{3}$$

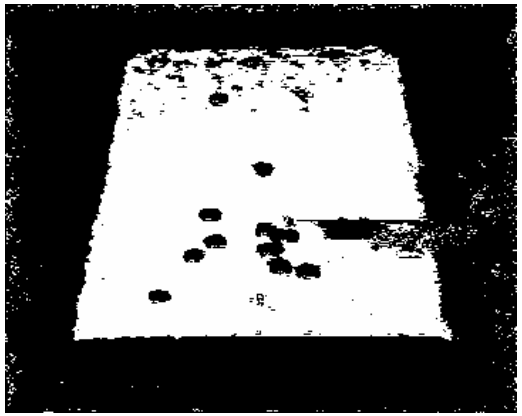
Pixels having an intensity between t_1 and t_2 are set to 255 or kept at their original values, and all other intensities are set to 0 (effectively discarded).

Intensity Slicing (Intensity based segmentation)

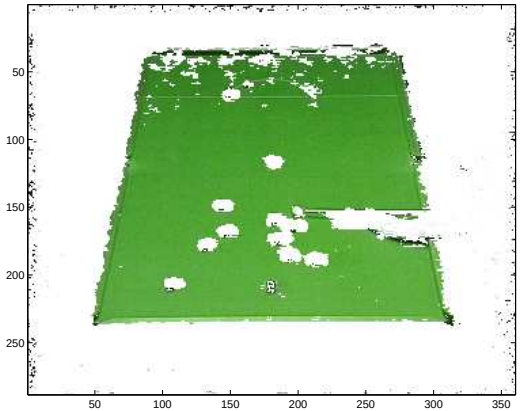
Snooker Frame



Histogram of Hue



$t_1 = 100 \quad t_2 = 110$



Extracted Pixels

Intensity Slicing: matlab

- Load up picture as image e.g. `pic = imread('pool.jpg','jpg');` `pic` is now a $N \times M \times 3$ data structure that stores red, green and blue components in `pic(:,:,1), pic(:,:,2), pic(:,:,3)`.
- Get the hue component `hsv = rgb2hsv(pic); hue = double(hsv(:,:,1))*360;`
- PIXELwise OP Make the mask for the *slice*. `mask = ((hue>100)&(hue<110) )` The MASK is now a matrix that is set to 1 everywhere HUE is in the range specified and it is 0 everywhere else. MASK is therefore the intensity *sliced* image.
- To see what has been *sliced*, extract the pels using the mask. e.g. `red = pic(:,:,1).*mask`
- Problems: lots of 'single' pels being flagged wrongly, segmented area not 'coherent'. This is due to noise in the image, and also due to the fact that you have not used a realistic model for the 'texture' of the table. [Using intensity slicing you have assumed that the table is just a constant (or near) constant intensity (hue).] Need to know about statistics and filtering. What about using H and S *at the same time*?

Clipping, Contrast Stretching

- Used when image contains a narrow band of intensities hence low contrast/brightness
- Idea is to 'stretch' a particular input range of values so that they occupy the full intensity range.
- Simplest is a linear transformation, non-linear transformation also possible.

$$g(h,k) = \begin{cases} 0 & \text{For } I(h,k) < I_{\min} \\ mI(h,k) + c & \text{For } I_{\min} < I(h,k) < I_{\max} \\ 255 & \text{For } I(h,k) > I_{\max} \end{cases} \quad (4)$$

  Here, pixels within the band $I_{\min} < I(h,k) < I_{\max}$ are *stretched* to occupy some range controlled by c and m; where m achieves the stretch, and c achieves a brightness shift. Further, all pixels below $I_{\min}$ are set to 0 and those above $I_{\max}$ are set to 255. This latter is a clipping operation.

- Clipping is typically used at the end of some image processing operation to keep the pixel data in the range of the allowed displayable levels.

Gamma correction

- Transformation from numerical values of pels to luminance of displayed pel is non-linear. $C(h,k) = I(h,k)^{\gamma}$. Applies to all film/image scanners, ccd's etc. Can mean that digitized intensities are not quite right. Sometimes just ageing of material does this.
- Can correct for this by inverse transformation. $I(h,k) = G(h,k)^{(1/\gamma)}$. In practice you may not know the γ, need to measure it or just use some non-linear coefficient until you get it 'right'.
- $I(h,k) = G(h,k)^{\gamma}$ Matlab syntax: `I =G.^0.5`
- Matlab Demo: imadjdemo

Histogram Equalisation

- A powerful technique for image enhancement. But a bit of a loose cannon.
- Idea is to redistribute the assignment of intensities to the pixels in the image so that they are all equiprobable. Thus if image contains alot of information in a narrow band of intensities, this is stretched out over all the usable intensity range.
- See ppt. example
- Problem is how to assign intensities to pels in the image. Got to choose! If you choose wrong you could be making matters worse, and you can't be consistent from image to image. Use Cumulative Distribution to design mapping instead.
- Say image is $N \times M$ pels. The p.d.f of an image with uniformly distributed pels will have a constant value at $\frac{1}{NM}$. This implies a Cumulative Distribution Function which is a straight line, having a gradient 1/256, a maximum of 1 and min of 0.

- Idea is now to transform the image intensities so that the CuF of the transformed image *matches* the target CuF i.e. as close as possible to the straight line.
- Works for *any* target distribution. See ppt example.
- There is a quantitative proof (involving prob. theory) that using the CuF will give the optimal transformation *for images with continuous grey scales* i.e. analogue pictures. But this will not be covered here.
- Matlab function `histeq`.

Summary

1. An image can be thought of as a matrix
2. There are some useful point wise or pixel wise operations for images which all are related to the function of scaling (contrast modification).
3. Histograms are useful for describing image content
4. Histogram modification can be used to perform non-linear image enhancement: Histogram Equalisation
5. These are very common image processing blocks. Often Pixel operations are assigned to specific hardware or software modules as they are used often.
6. *Most* of the pixel operations can be implemented as Look Up Tables since the input data is typically 8 bit. **This is very important for fast, simple and reprogrammable module design (whether hardware or software).**
7. References

- Lim Pages 453–458
- Jain [Fundamentals of Digital Image Processing] Pages 235–244