CS 376A
Digital Image Processing

02 / 20 / 2023

Instructor: Michael Eckmann

Today's Topics

- Questions / Comments?
- Histograms for Histogram Equalization for Contrast Enhancement
- HistogramEQ for Color images
- Consider converting to and from a color space that has intensity as a channel (e.g. YcbCr)
- Determining an appropriate threshold to select foreground pixels from background pixels (Otsu method)

Histograms

- For contrast enhancement via histogram equalization
  - we want to stretch the intensities to use a wider range (ideally all) of available intensities
  - should have approx. the same number of pixels per intensity (approx. a uniform distribution)
- Create a mapping from input intensity to output intensity based on the histogram (which tells us how frequent each intensity occurred in the input image).

Histograms

- For our output, since we should have approx. the same number of pixels per intensity and we want the range of intensities to be as large as possible, then we'd like to roughly have the same number of pixels in each output intensity.
- But this will not be perfect because all pixels that have the same input intensity must be mapped to the same output intensity
  - some input intensities will have more than #pixels / 256 in their histogram bin

Histograms

- Histogram equalization
  - compute histogram, h, of image intensities
  - from that histogram h, create a cumulative histogram, ch, defined to be
    - ch[i] = sum $_{j=0 \text{ to } i}$ of all h[j]'s
  - ch_{min} = lowest non-zero ch[i]
  - then mapping from input intensities (in) to output intensities is:

$$out(in) = 255 * (ch[in] - ch_{min}) / (numberOfpixels - ch_{min})$$

Notice: the i for ch[i]=ch_{min} gets mapped to 0, the fraction is 1 when ch[in] = numberOfpixels so the highest value of in gets mapped to 255 and all the other input values get mapped proportionally appropriately in between

Histogram Equalization for color

- Applying histogram equalization to each of the color channels independently could cause unwanted color changes
  - how is this different from us applying the mapping curve to each color channel for the images that were too light or too dark?

Histogram Equalization for color

- One better solution is to convert RGB to a color space that separates out the intensity/brightness/lightness/value into its own channel. Then do the histogram equalization on that channel and convert back to RGB.
- e.g. Y Cb Cr is a color space that has Y as the intensity channel.

Histogram Equalization for color

- Y Cb Cr is a color space that has Y as the intensity channel.
  - Y = 0.299*R + 0.587* G + 0.114*B
  - Cb = – 0.17*R – 0.33*G + 0.5*B + 128
  - Cr = 0.5*R – 0.42*G – 0.08*B + 128
- store Cb & Cr as doubles (and they could be positive or negative)
- To go back from Y Cb Cr to RGB:
  - R = (int) (Y + 1.4 * (Cr – 128))
  - G = (int) (Y – 0.344 * (Cb – 128) – 0.714 * (Cr – 128))
  - B = (int) (Y + 1.772 * (Cb – 128))
- could get negatives or > 255.
- in those cases set R,G,B channels to 0 or 255 respectively.

Histogram Equalization for color

- Let's get started writing code to convert RGB to and from YCbCr.
- Could write a method in RGBImage that converts the 2d array of RGBPixels into a 2d array of YCbCrPixels.
- Then can apply the histogram equalization to the Y channel.
- Then have another method that converts the 2d array of YCbCrPixels to RGB.
- This requires a new class YcbCrPixel. And in each of the Pixel classes, have a method that returns the other pixel type based on the calculations.

Histogram use to determine threshold

- Recall using a threshold to make a black and white image from a grayscale image (all pixels below some threshold are set to black and all others are set to white).
- Usually used (as a first step) to discriminating foreground objects from background.
- Let me draw some histograms on the board. Where do you think the ideal threshold is in each of these?

Histogram use to determine threshold

- Let me read from the introduction to Nobuyuki Otsu's 1979 paper “A Threshold Selection Method from Gray-Level Histograms” in IEEE Transactions on Systems, Man, and Cybernetics.

Histogram use to determine threshold

- Note: Histogram values now need to be proportions of how many pixels in the image correspond to each bin out of the total number of pixels. So, the sum of all the histogram values is 1.0
- This is called a *normalized histogram*

Background on Gaussian distribution

- Mean
- Variance – a measure of the spread or how similar / different a group is
  - high variance = wide spread, members of the group have high differences among them
  - low variance = tighter grouping, members of the group have lower differences among them
- Standard Deviation (square root of variance)
- Let's see two small examples of distributions and calculations of mean and variance of each of the distributions.

Otsu method

- Works well for bimodal histograms
- Finds the threshold such that the weighted sum of the within-group variances (to be defined shortly) is minimized.
- Consider a bimodal distribution with some threshold t dividing the groups.
- Example on the board.

Otsu method

- Consider a bimodal distribution with some threshold t dividing the groups

  the weighted sum of the within group variances =

  (sum of probabilities group 1) * (variance group 1) +

  (sum of probabilities group 2) * (variance group 2)

  - Compute the above for all t's and minimize
  - It turns out that minimizing the “within group variance” is the same as maximizing the “between group variance” and computing the between group variance is easier in an iterative fashion

Otsu method

- The "between group variance" =
  (sum of probabilities group 1) * (sum of probabilities group 2) *
  (mean of group 1 – mean of group 2)^2
  =
  (sum of probabilities group 1) * (1 - sum of probabilities group 1) *
  (mean of group 1 – mean of group 2)^2

- As we vary t from 1 to 255 we can update the terms of the above for a new t based on the values from the previous t.

  Note:
  (sum of probabilities group 1 at t+1) =
  (sum of probabilities group 1 at t) + probability at t+1

Otsu method

- As we vary t from 1 to 255 we can update the terms of the above for a new t based on the values from the previous t.

  Note:

  (sum of probabilities group 1 at t+1) =

  (sum of probabilities group 1 at t) + probability at t+1

  The mean of group 1 at t+1 is as stated on the handout (based on info at t and current info at t+1)

  The mean of group 2 at t+1 is as stated on the handout (based on info at t and current info at t+1 AND the mean of entire histogram)

  Note: We can precompute the mean of the entire histogram once.

  Computing the between group variance at each t is much more computationally efficient than computing the within group variances (can simply use the "update" functions for a new t based on values from previous t)

Otsu method

- sketch of the algorithm
  - bestT = 0, bgv = 0, currT = 0
  - compute the overall mean of the histogram
  - loop over all currT (0..254)
    - group 1 is the 0 to currT pixels
    - group 2 is the pixels with values: currT+1 .. 255
    - *compute sum of probabilities group 1
    - *compute mean of group 1
    - *compute mean of group 2
    - compute the between group variance currentBGV by:
      (sum of probabilities group 1) *
      (1 - sum of probabilities group 1) *
      (mean of group 1 – mean of group 2)^2
    - if currentBGV > bgv
      bestT = currT
      bgv = currentBGV

Otsu method

- The *'d lines on the previous slide --- compute first for currT = 0 and for all subsequent currT's use the update formulas (based on previous currT and overall mean):

  compute sum of probabilities group 1

  compute mean of group 1

  compute mean of group 2

Detailed example of Otsu method

- Let's review an example of Otsu's method with actual numbers (and images)

Adaptive methods

- Back to histogram equalization ...
- It may not be appropriate to apply histogram equalization to the entire image based on the entire image. In some cases it may be more beneficial to apply more localized histogram equalizations
  - where a histogram is computed of each portion of an image and histogram equalization on each portion is carried out separately.

Histogram matching motivation

- Suppose we have images taken from multiple cameras and we wish to alter them such that they conform in some way to one of them.
- Suppose we have images of the same scene over time, taken with the same camera but under different conditions (or the camera degrades over time) etc. we might want to alter the images such that they have some of the same characteristics of one of them.
- Another situation might be to map a color scheme from one image to another (possibly for artistic purposes).
- All of these situations may benefit from the idea of histogram matching.

Histogram matching (aka Specification)

- Recall what histogram equalization does
- Given an image and its histogram
- The resulting histogram equalized image is intended to have a cumulative histogram that is linear (approximately the line: y=x)
- For histogram matching, the idea instead is
  - using the histogram of a reference image and an image to be transformed, map the image to be transformed such that it has the same histogram as the reference image
- In general, in the discrete realm (bins are integer values corresponding to a gray level or a color channel) this can only result in an approximation of the resulting image histogram to "match" the reference histogram exactly.
- Why only an approximation?

ive histogram).

Histogram matching (aka Specification)

The approximation is for similar reasons to the result of histogram equalization not having a perfectly (hence being an approximation of) linear “y=x” cumulative histogram .

Histogram matching (aka Specification)

- A normalized histogram is one in which all of the bins add up to 1 and the value in each bin represents the proportion of total pixels that correspond to that bin.
- e.g. a histogram with values: 100, 150, 200, 50, 500
- Normalized would be: 0.1, 0.15, 0.2, 0.05, 0.5
- Bins would be numbered 0 .. 4

Histogram matching (aka Specification)

- Histogram matching can be done in the following way.
- Compute the cumulative normalized histogram *cNormHistRef* for the reference image *refImg* (the image whose histogram is to be mimicked).
- Compute the cumulative normalized histogram *cNormHistIn* for the image to be transformed *inImg*.
- Create a mapping from pixel value *inP* of *inImg* to *outP* of *outImg*

  map[inP] = outP

  in the following way:
- outP is the bin of *cNormHistRef* such that outP is the lowest bin number where *cNormHistRef[outP] >= cNormHistIn[inP]*
- Example on the board of this idea.

Histogram matching (aka Specification)

- Why do you think we are using normalized histograms instead of the regular histograms?

Histogram matching (aka Specification)

- Another way to understand how inP can be mapped to outP is to look at the handout.
- Left diagram is the cumulative histogram of the reference image
- Right diagram is the (not cumulative) histogram of the image to be transformed.
- It shows that a pixel with value a gets mapped to the value a'.