Image Processing Notes

Sasank

November 14, 2014

1 Grey Level Transformations

Evaluation is subjective.

input : image, $f(x, y)$

output : image, $g(x, y) = T(f(x, y))$

Want transformations to be monotonic.

Some transformations

1. Negative
2. Thresholding
3. Contrast stretching
4. Logarithm
5. Power and root transformation, Gamma correction: $y = l.(x/l)^{\gamma}$
   $\gamma > 1$: darkening, $\gamma < 1$: adding light
6. Histogram Processing

1.1 Histogram Equalization

Given histogram $f_x(x)$, design a function $y = T(x)$ such that $f_y(y)$ is uniform. Assume T to be monotonic.

$$P(x < x_o) = P(y < T(x_o))$$
$$F_x(x_o) = T(x_o)$$

So, required transformation is $y = F_x(x)$ where F is cdf of X. Because of discretization of space and intensity, output histogram is not exactly uniform.

1.1.1 Limitations

- "middle" intensities don't get contrast enhanced if lots of dark or bright pixels

1.2 Histogram Matching

Want histogram of y = T(x) to be G(y).

$$F_y(y = T(x)) = F_x(x)$$
$$T(x) = F_y^{-1}(F_x(x))$$

1.3 Adaptive Histogram Equalization - AHE

For each pixel p:

1. Construct NxN window around the pixel
2. Compute histogram of the window and map intensity of centre pixel p based on this histogram

Window size N has to be tuned. Small window brings out more detail but also amplifies noise in constant patches.

1.4 Contrast Limited Adaptive Histogram Equalization - CLAHE

Contrast amplification at a pixel value is given by slope of transformation at that pixel. For histogram equalization, slope of transformation, cdf = pdf. So, clip histogram and redistribute mass before computing cdf. This is very computationally expensive.

Approximation

1. Split image into MxM non overlapping tiles
2. Compute CDF for each tile
3. Bilinearly interpolate *transformation* for all pixels based on distance from tile centers

2 Spatial Filtering

Output pixel intensity is function of the pixel intensities in the neighbourhood. if the function is linear, linear spatial filtering. Such a filter performs convolution.

If mask is symmetric about x and y axes, then convolved image = cross-correlated image.

Normalized cross correlation: subtract mean from signal and make signal unit energy before correlating.

If a filter is **seperable** i.e, can be written as outer product $f = uv$ where u is a column matrix and v is a row matrix, convolution can be done faster.

$$f * H = (u * v) * H = u * (v * H)$$

Here, Last term can be computed in $Q(MN) + P(MN) = MN(P + Q)$ instead of $MNPQ$ steps

Mean Filter : Replace pixel value by mean of values in windows around pixel. Smoothens out edges. This is a separable filter.

Gaussian Filter : Window, $g(x, y) = e^{-(x^2+y^2)/2\sigma^2}$. Blurs less and preserves more edges but removes less noise in constant patches. Separable if covariance matrix is diagonal.

Median Filter : Replace pixel value by median of values in windows around pixel. Better for removing salt and pepper noise. On Gaussian noise, preserves edges better and can create artificial edges in smooth regions.

Image Derivative : Sharpening and very sensitive to noise. Convolution mask for first and second derivative.

Laplacian of Guassian : Apply Gaussian filter to smooth image, apply laplacian filter to smoothed image. "Blob detector" and gives strong positive and negative responses at edges.

Unsharp Masking

3 Bilateral Filtering

Problem with weighted mean filters is they fails to preserve edges while reducing noise. We want a filter which reduces noise but preserves edges.

For Bilateral filer, at pixel p weight of q depends on

1. spatial distance between p and q
2. dissimilarity between intensities at p and q

$$I^{filtered}(x) = \frac{1}{W_p} \sum_{x_i} I(x_i) f(||I(x_i) - I(x)||) g(||x_i - x||)$$
$$W_p = \sum_{x_i} f(||I(x_i) - I(x)||) g(||x_i - x||)$$

Here f and g usually are Gaussian. Observe that this defines different weighting mask for each pixel. So, bilateral filter is not convolution or seperable and thus slow.

3.1 Faster Implementation

We can compute bilateral filtering fast by rewriting it as linear filtering in different space.

$$\begin{aligned} I^{filtered}(x) &= \frac{1}{W_p} \sum_{x_i} I(x_i) f(||I(x_i) - I(x)||) g(||x_i - x||) \\ &= \frac{1}{W_p} \sum_{x_i} I(x_i) G_{sr}(||(x_i, I(x_i)) - (x, I(x))||) \\ W_p &= \sum_{x_i} G_{sr}(||(x_i, I(x_i)) - (x, I(x))||) \end{aligned}$$

Where G_{sr} is a higher dimensional Gaussian. This gives us motivation to construct at a higher dimensional space $(x, I(x))$ as following:

1. Value at $(p, I(p))$ is $I(p)$ for every pixel p in image
2. Assign 0 everywhere

We can apply separable Gaussian filtering in this space and get values at $(p, I(p))$ to get numerator. To get denominator create similar space with values 1 instead of $I(p)$.

Strategy to speed up computation is to downsample constructed space before applying Gaussian convolution and then upsample.

3.2 Cross/Joint Bilateral Filter

Get 2 kinds of weights from 2 different images.

3.3 Limitations

1. Texture softer than intensity kernel standard deviation is removed
2. "Staircase" artefacts: constant intensity pieces

4 Patch-Based Filtering

Problem with Bilateral filter is intensity based similarity is solely based on intensity difference; texture information is not best used. Texture is a repeated spatial pattern of intensities either geometric or stochastic.

$$I^*(k) = \sum_j w_h(j,k) I(j)$$

Design weights w based on patches around pixel; Similar patch has larger weight.

$$w_h(j,k) = \frac{\exp\left(-||I(R_j) - I(R_k)||^2/h^2\right)}{\sum_j \exp\left(-||I(R_j) - I(R_k)||^2/h^2\right)}$$

Here, R_j represent a patch around pixel j and $||.||$ is L_2 norm.

We are dealing with a regression problem where independent variable is intensities of the patch and dependent variable is intensity value at centre pixel of the patch. This is non linear regression.

4.1 Nonlinear Regression

Objective: Given n pairs (x_i, y_i) and x_0, find y_0.

K nearest neighbour : Given x_0, find K nearest neighbours $\{x_j\}$ and find average of these k neighbours.

Kernel Regression :

$$y_0 = \frac{\sum_i K_h(x - x_i) y_i}{\sum_i K_h(x - x_i)}$$

where K_h is a kernel function with bandwidth h.

In context of patch based filtering, we are doing kernel regression on patches to find centre pixel intensity.

1. Updates of adjacent pixels are interlinked
2. Initial neighbourhoods are noisy

To reduce artifacts, take a small step towards update and perform multiple passes over the image. Patches can be selected either in a window around pixel or randomly using a Gaussian distribution centred around pixel.

5 Filtering in Fourier Domain

5.1 Fourier Transform

Topics covered:

1. Fourier series
2. Fourier transform
3. Sampling theorem
4. Discrete time Fourier transform
5. Discrete Fourier transform

Be careful with convolution and FFT domain. You might have to pad zeros to vectors before convolution.

5.2 Filtering in Fourier Domain

Basic strategy:

1. Compute DFT of image Ff(u,v)
2. Modify values in DFT domain
3. Compute IDFT to these values

Some filters:

- Lowpass, Highpass, Bandpass filters
- Butterworth filter -"Maximally flat magnitude filter"
- Gaussian low pass filter

Applications

- Remove periodic patterns
- Some convolution filters are not separable. But DFT is!

6 Face Recognition

Given a database of face images of people, and a new test image, answer the following question - "The test image contains the face of which individual from the database?"

Challenges

- Face detection
- Pose variation
- Illumination variation
- Expression variation
- Variation of facial accessories/Occlusions
- Blur/noise/scanner artifacts

6.1 Naive Method

Choose closest match in database according to following distance measure:

$$||I - J||^2 = \sum_i \sum_j (I_{ij} - J_{ij})^2$$

Very bad algorithm.

6.2 PCA - Eigenfaces

Consider N vectors (or points) x_i, each containing d (usually $d \gg N$) elements, each represented as a column vector. Aim is to extract k most 'significant' features. i.e, we want to project vectors in $\mathbb{R}^d$ to $\mathbb{R}^k$. This is called **dimensionality reduction** in general.

Let us consider a linear transformation mapping the original d-dimensional image space into an k-dimensional feature space, where k < d. The new feature vectors $y_i \in \mathbb{R}^m$ are defined by the following *linear* transformation:

$$y_i = W^T x_i$$

where $W \in \mathbb{R}^{d \times n}$ is a matrix with orthogonal columns.

If the total scatter matrix S_T is defined as

$$S_T = \sum_{i=1}^{N} (x_i - \mu)(x_i - \mu)^T$$

where $\mu \in \mathbb{R}^n$ is the mean image of all samples. Then after applying the linear transformation W^T, the scatter of the transformed feature vectors $\{y_1, y_2, \ldots, y_N\}$is $W^T S_T W$.

In PCA, the projection W_{opt} is chosen to maximize the determinant of the total scatter matrix of the projected samples, i.e.,

$$W_{opt} = \arg\max_W |W^T S_T W| = [w_1 w_2 \ldots w_k]$$

where w_i are the set of d-dimensional eigenvectors of S_T corresponding to m largest eigenvalues. These are referred to as **eigenfaces**.

later

7 SVD

later