Outline

Representation

Linear operations

In-painting

Image de-blurring
Monochrome images

- a.k.a. *monochrome* or *gray-scale* image

- image represented by its brightness values at an array of $m \times n$ locations (pixels)

- typical sizes
  - thumbnail: $16 \times 16$, $64 \times 64$, $128 \times 128$
  - $4K \times 6K = 24M$ pixels
  - HD is $1280 \times 720$ pixels

- can represent by an $m \times n$ matrix $X$ or by a single vector $x \in \mathbb{R}^{mn}$ with some encoding of the pixel locations, e.g.,

\[
X_{ij} = x_k, \quad k = m(j - 1) + i, \quad k = 1, \ldots, mn
\]

(this stacks the columns of $X$, from left to right)
Brightness values

- $x_i$ is brightness of pixel $i$
- Typically $0 \leq x_i \leq 1$ where 0 is black and 1 is white
- Values outside $[0, 1]$ are clipped (so $x_i < 0$ shows up as black)
- If $x$ is an image, $-x$ is completely black
- *Negative image* is given by $1 - x$
- $\text{avg}(x)$ is average intensity (brightness) of image
- $\text{std}(x)$ corresponds to image *contrast*
Scaling, shifting, and adding images

- what does image

\[ y = a(x - \text{avg}(x))1 + (\text{avg}(x) + b)1 = ax + c1 \]

\((c = (1 - a)\text{avg}(x) + b)\) look like?
- \(a\) scale contrast
- \(b\) shifts brightness

- \(y_i = x^\gamma_i\) is called \(\gamma\)-correction (widely used)

- if \(x\) and \(y\) are images, \(x + y\) is perceived as composite or combination of the images (and isn’t natural, except in some cases)
Examples

original

original + 0.5

(original − 0.4) * 10

Representation
Examples

pumpkins

flowers

(pumpkins+flowers)/2
Color images

- humans perceive 3 colors, which can be represented in different ways (e.g., RGB, CMYK)
- most common is RGB (Red-Green-Blue)
- color represented as a 3-vector \((r, g, b)\), with \(r, g, b\) between 0 and 1
  - \((1, 0, 0)\) is bright red
  - \((1, 0, 1)\) is bright purple
  - \((0.2, 0.2, 0.2)\) is a gray
- \(m \times n\) image given by 3 \(m \times n\) matrices or one vector \(x \in \mathbb{R}^{3mn}\)
Colors

- $(1,0,0), (0,1,0), (0,0,1)$
- $(1,1,0), (0,1,1), (1,0,1)$
- $(0.2,0.2,0.2), (0.5,0.5,0.5), (0.75,0.75,0.75)$
Color images

original

red

green

blue

Representation
Converting color to monochrome

- color pixel values converted to monochrome using \( y_i = w^T (r_i, g_i, b_i) \)

  - obvious choice: \( w = (1/3, 1/3, 1/3) \)
  - another common choice: \( w = (0.299, 0.587, 0.114) \)
  - other choices used for special effects
Converting color to monochrome

original
weights: 0.2, 0.5, 0.3

equal weights
weights: 0.6, −0.4, 0.8
Video

- video is represented as a sequence of images captured periodically
- each image is called a frame
- typical frame rates: 24, 30, or 60 frames per second
Outline

Representation

Linear operations

In-painting

Image de-blurring
Linear image mappings

- for $y$ and $x$ images, linear mapping $y = Ax$ can represent many common operations on images
  - color to monochrome conversion
  - color correction
  - any mapping from original to distorted pixel locations (e.g., flipping, stretching)
  - blurring
  - changing to lower or higher resolution
  - vertical and horizontal differencing
Moving pixels

- pixel at location $i$ in image $y$ is the pixel at value $j = d(i)$ in image $x$
- $d(i)$ gives distortion map
- examples: flipping, zooming, rotating, shifting, key correction
- some issues/details:
  - we’ll need to approximate the location of the pixels
  - we need to do something with $y$ pixels that don’t correspond to any $x$ pixels
- $y = Ax$, where $i$th row of $A$ is $e^T_{d(i)}$
  (or 0, if $y_i$ doesn’t correspond to any $x$ pixel)
Flipping images

original image

horizontal flip

vertical flip

Linear operations
**Blurring images**

- represent image as $m \times n$ matrix $X$
- represent blur *point spread function* as $p \times q$ matrix $B$
- blurred image is given by $Y$ with

$$Y_{ij} = \sum_{k,l} X_{i-k+1,j-l+1} B_{k,l}$$

where

- the sum is over all integers $k, l$
- we interpret $X_{ij}$ and $B_{k,l}$ as zero when the indices are out of range

- called *2-D convolution* of $X$ and $B$, denoted $Y = X \ast B$ or $Y = A \ast B$
- blurring is model of effects of optical imperfections, motion blur, ...
Blurring images

original image

blurred image

point spread function
Horizontal and vertical differences

- $X$ is $m \times n$ image (matrix), $x$ its $mn$-vector representation
- horizontal first order difference is $m \times (n - 1)$ matrix $Y$ with
  \[
  Y_{ij} = X_{i,j+1} - X_{i,j}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n - 1
  \]
- vertical first order difference is $(m - 1) \times n$ matrix $Z$ with
  \[
  Z_{ij} = X_{i+1,j} - X_{i,j}, \quad i = 1, \ldots, m - 1, \quad j = 1, \ldots, n
  \]
- these are linear operations, so we have
  \[
  y = D^{\text{horiz}}x, \quad z = D^{\text{vert}}x
  \]
  for an $m(n - 1)$-matrix $D^{\text{horiz}}$ and an $(m - 1)n$-matrix $D^{\text{vert}}$
- each row contains one $+1$ and one $-1$

Linear operations
Horizontal and vertical differences

(shown for $3 \times 3$ image)

$$D_{\text{horiz}} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & +1 & -1 \\ +1 & -1 & +1 \end{bmatrix}$$

$$D_{\text{vert}} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & +1 & -1 \\ +1 & -1 & +1 \end{bmatrix}$$
Horizontal and vertical differences

original image  
horizontal difference  
vertical difference
The Laplacian function is

\[ \mathcal{L}(x) = \| D_{\text{horiz}} x \|^2 + \| D_{\text{vert}} x \|^2 \]

\[ = \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} \left( (X_{i+1,j} - X_{i,j})^2 + (X_{i,j+1} - X_{i,j})^2 \right) \]

(we also write \( \mathcal{L}(X) \))

- \( \mathcal{L}(X) \) is a measure of roughness of the image \( X \)
  - \( \mathcal{L}(X) \) is small when the image is smooth
  - \( \mathcal{L}(X) = 0 \) only if the image is constant

- \( \mathcal{L}(X) \) is used as a regularizer
Outline

Representation

Linear operations

In-painting

Image de-blurring
In-painting

- we are given an image with some pixels values unknown
- *in-painting* means to guess values of the unknown pixels so the recovered image looks good or natural
- in example below, unknown values are shown as black
Least-squares in-painting

- corrupted/damaged image is given by $m \times n$ matrix $X^{\text{corr}}$

- $\mathcal{K} \subset \{1, \ldots, m\} \times \{1, \ldots, n\}$ are the indices of known pixels

- we need to choose an image $X$ that agrees with the given image on known pixels: $X_{ij} = X_{ij}^{\text{corr}}$, $(i, j) \in \mathcal{K}$

- we'll choose $X$ to minimize $\mathcal{L}(X)$, the sum square deviation of all pixel values from their neighbors (small $\mathcal{L}(X)$ gives a smooth image)

- a least-squares problem (variables are unknown pixel values)
In-painting
Outline

Representation

Linear operations

In-painting

Image de-blurring
Corrupted image

- $y$ is a linear function of $x^{\text{true}}$, with noise:

$$y = Ax^{\text{true}} + v$$

- $y$ is the *corrupted* image, which we have
- $x^{\text{true}}$ is the original image, which we want to guess/recover
- $v$ is a noise, which we assume is small
- $A$ is a (known) matrix, often a blurring operator
- *Image de-blurring* is guessing $x^{\text{true}}$
- even if $A$ is invertible, the guess $x = A^{-1}y$ could look very bad
Least-squares de-blurring

- least-squares de-blurring: choose $x$ to minimize

$$\|Ax - y\|^2 + \lambda \mathcal{L}(x)$$

- first term is $\|v\|^2$
- $\lambda > 0$ is a regularization parameter
  - large $\lambda$ makes $x$ smooth
  - small $\lambda$ makes $\|Ax - y\|^2$ small
De-blurring example

original image

corrupted image

deburred image with lambda = 0.03
De-blurring: Effect of regularization

\[
\text{lambda } = 0.0003
\]

\[
\text{lambda } = 0.03
\]

\[
\text{lambda } = 3
\]