Digital Image Processing (CS/ECE 545) Lecture 11: Geometric Operations, Comparing Images and Future Directions

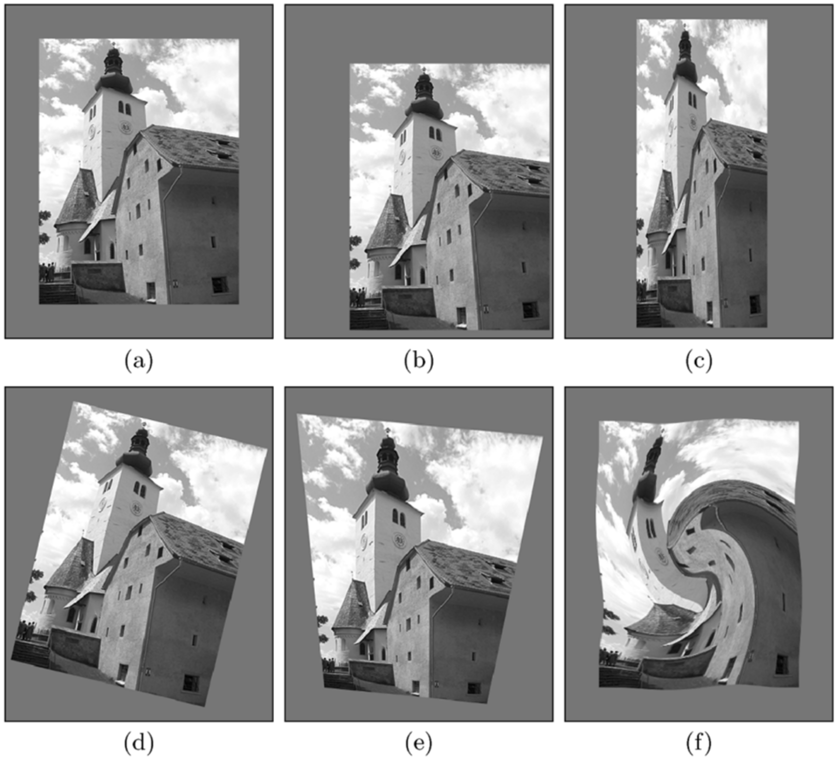
Prof Emmanuel Agu

Computer Science Dept.

Worcester Polytechnic Institute (WPI)

Geometric Operations

- Filters, point operations change intensity
- Pixel position (and geometry) unchanged
- Geometric operations: change image geometry
- **Examples:** translating, rotating, scaling an image



(a) (b) (c) (d) (e) (f)

Examples of Geometric operations

Geometric Operations

- Example applications of geometric operations:
  - Zooming images, windows to arbitrary size
  - Computer graphics: deform textures and map to arbitrary surfaces
- **Definition:** Geometric operation transforms image *I* to new image *I'* by modifying **coordinates of image pixels**

$$I(x, y) \rightarrow I'(x', y')$$

- Intensity value originally at (x,y) moved to new position (x',y')

Example: Translation geometric operation moves value at (x,y) to (x + d_x, y + d_y)

Geometric Operations

- Since image coordinates can only be discrete values, some transformations may yield (x',y') that's not discrete
- **Solution:** interpolate nearby values

Simple Mappings

- **Translation:** (shift) by a vector (d_x, d_y)

$$T_x : x' = x + d_x \qquad \text{or} \qquad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} d_x \\ d_y \end{pmatrix}$$
$$T_y : y' = y + d_y$$

- **Scaling:** (contracting or stretching) along x or y axis by a factor s_x or s_y

$$T_x : x' = s_x \cdot x \qquad \text{or} \qquad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$
$$T_y : y' = s_y \cdot y$$

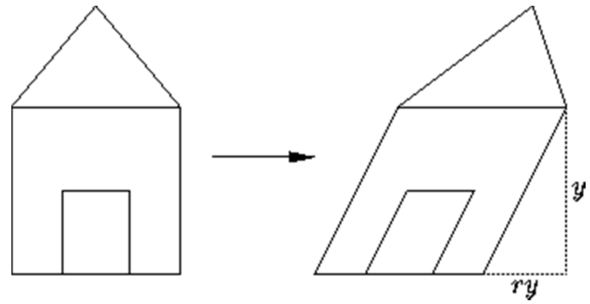
Simple Mappings

- **Shearing:** along x and y axis by factor b_x and b_y

$$T_x : x' = x + b_x \cdot y \qquad T_y : y' = y + b_y \cdot x$$

or

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & b_x \\ b_y & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$



- **Rotation:** the image by an angle α

$$T_x : x' = x \cdot \cos\alpha - y \cdot \sin\alpha$$
$$T_y : y' = x \cdot \sin\alpha + y \cdot \cos\alpha$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

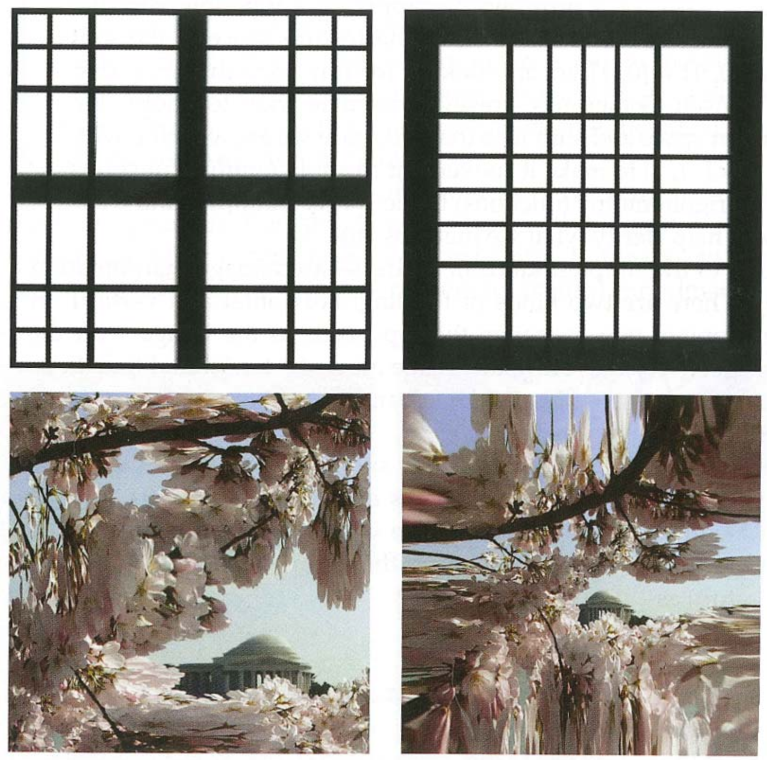
Image Flipping & Rotation by 90 degrees

- We can achieve 90,180 degree rotation easily
- Basic idea: Look up a **transformed pixel address** instead of the current one
- To flip an image upside down:
  - At pixel location *xy*, look up the color at location $x\ (1 - y)$
- For horizontal flip:
  - At pixel location *xy*, look up $(1 - x)\ y$
- Rotating an image 90 degrees counterclockwise:
  - At pixel location *xy*, look up $(y, 1 - x)$

Image Flipping, Rotation and Warping

- **Image warping:** we can use a function to select which pixel somewhere else in the image to look up
- For example: apply function on both texel coordinates (x, y)

$$x = x + y * \sin(\pi * x)$$



Homogeneous Coordinates

- Notation useful for converting scaling, translation, rotating into point-matrix multiplication
- To convert ordinary coordinates into homogeneous coordinates

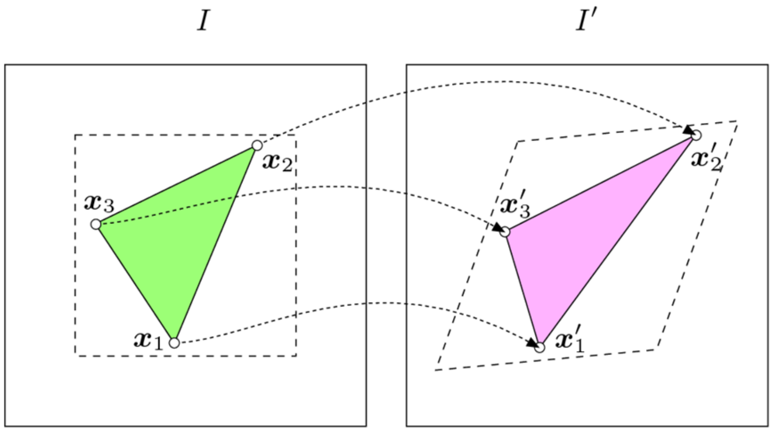
$$\boldsymbol{x} = \begin{pmatrix} x \\ y \end{pmatrix} \quad \text{converts to} \quad \hat{\boldsymbol{x}} = \begin{pmatrix} \hat{x} \\ \hat{y} \\ h \end{pmatrix} = \begin{pmatrix} h\,x \\ h\,y \\ h \end{pmatrix}$$

Affine (3-Point) Mapping

- Can use homogeneous coordinates to rewrite translation, rotation, scaling, etc as vector-matrix multiplication

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

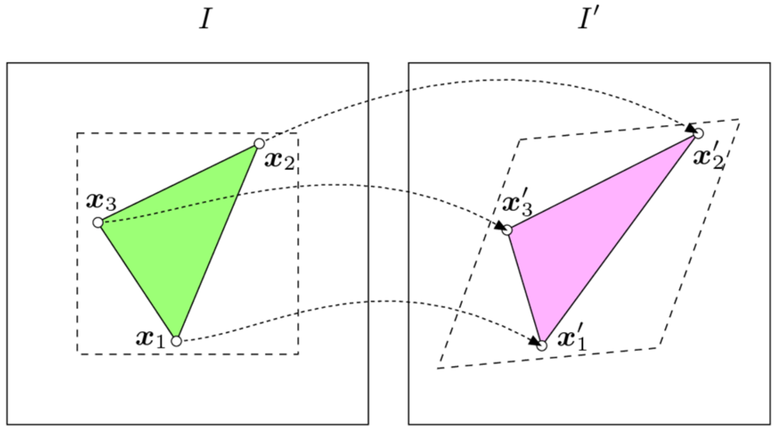
- **Affine mapping:** Can then derive values of matrix that achieve desired transformation (or combination of transformations)



- Inverse of transform matrix is **inverse mapping**

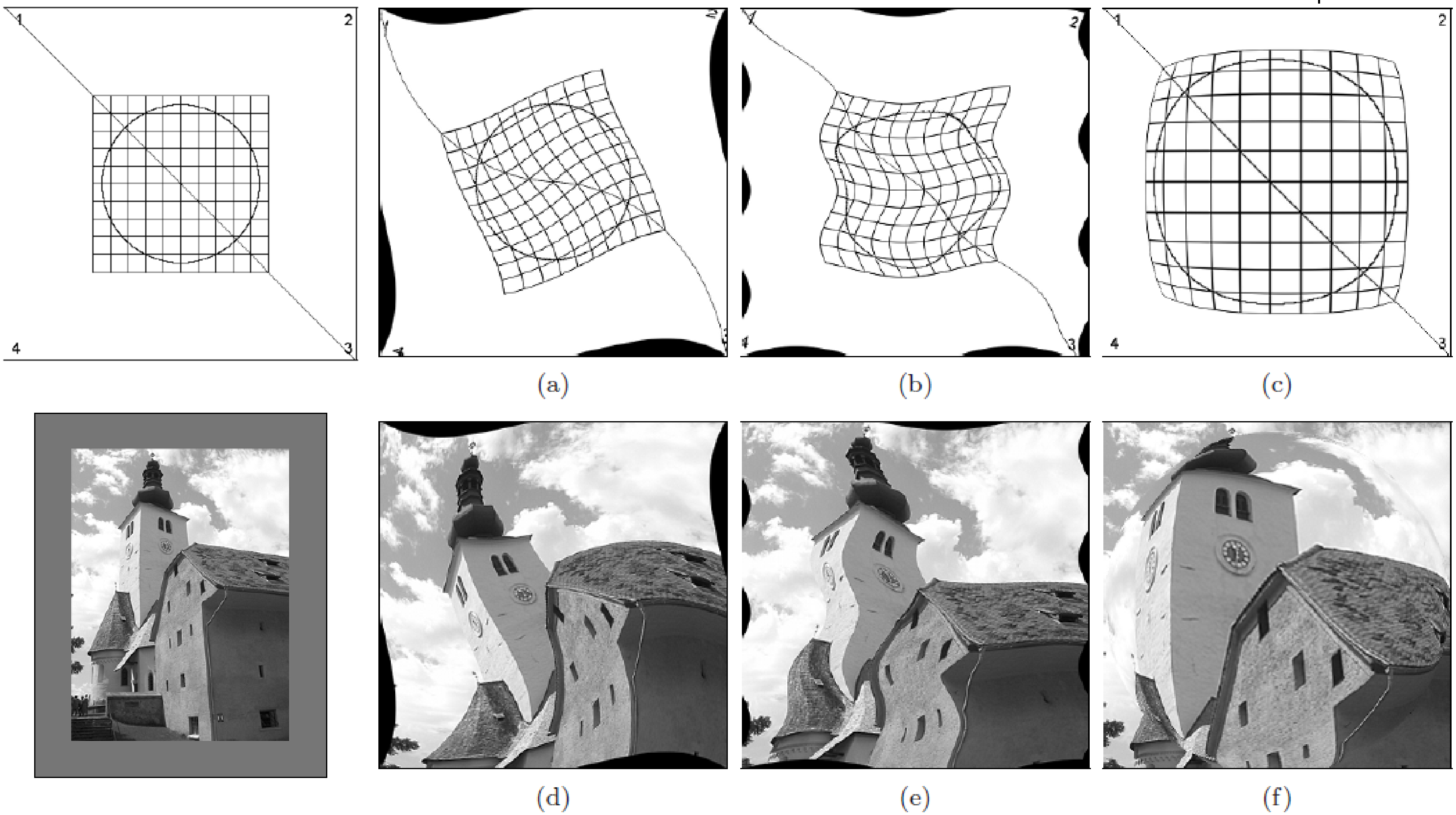
Affine (3-Point) Mapping

- **What's so special about affine mapping?**



- Maps
  - straight lines -> straight lines,
  - triangles -> triangles
  - rectangles -> parallelograms
  - Parallel lines -> parallel lines
- Distance ratio on lines do not change

Non-Linear Image Warps



Original

Twirl

Ripple

Spherical

Twirl

- **Notation:** Instead using texture colors at (x’,y’), use texture colors at twirled (x,y) location
- Twirl?
  - Rotate image by angle ***α*** at center or anchor point (x_c, y_c)
  - Increasingly rotate image as radial distance ***r*** from center increases (up to r_{max})
  - Image unchanged outside radial distance r_{max}

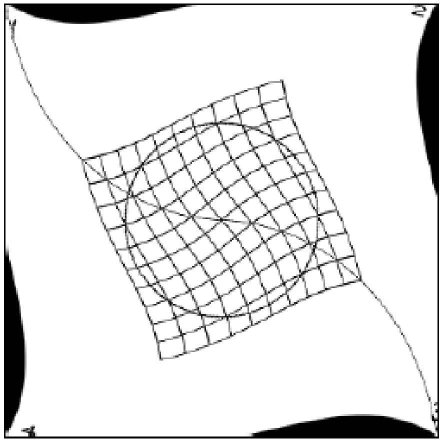
$$T_x^{-1}: \quad x = \begin{cases} x_c + r \cdot \cos(\beta) & \text{for } r \leq r_{\max} \\ x' & \text{for } r > r_{\max}, \end{cases}$$

$$T_y^{-1}: \quad y = \begin{cases} y_c + r \cdot \sin(\beta) & \text{for } r \leq r_{\max} \\ y' & \text{for } r > r_{\max}, \end{cases}$$

with

$$d_x = x' - x_c, \qquad r = \sqrt{d_x^2 + d_y^2},$$

$$d_y = y' - y_c, \qquad \beta = \operatorname{Arctan}(d_y, d_x) + \alpha \cdot \left(\tfrac{r_{\max} - r}{r_{\max}}\right).$$



(a)

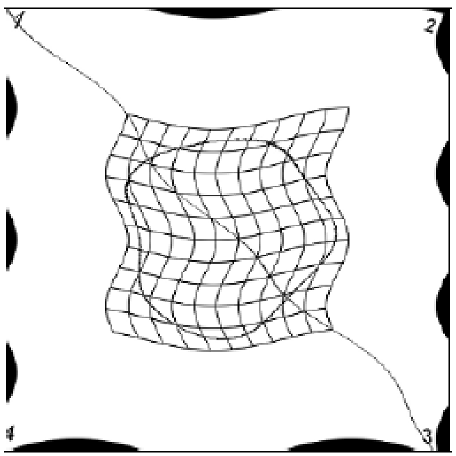
(d)

Ripple

- Ripple causes wavelike displacement of image along both the x and y directions

$$T_x^{-1}: \quad x = x' + a_x \cdot \sin\left(\frac{2\pi \cdot y'}{\tau_x}\right),$$
$$T_y^{-1}: \quad y = y' + a_y \cdot \sin\left(\frac{2\pi \cdot x'}{\tau_y}\right).$$

- Sample values for parameters (in pixels) are
  - τ_x= 120
  - τ_y= 250
  - a_x= 10
  - a_y= 15



(b)

(e)

Spherical Transformation

- Imitates viewing image through a lens placed over image
- Lens parameters: center (x_c, y_c), lens radius r_{max} and refraction index ρ
- Sample values ρ = 1.8 and r_{max} = half image width

$$T_x^{-1}: \quad x = x' - \begin{cases} z \cdot \tan(\beta_x) & \text{for } r \le r_{\max} \\ 0 & \text{for } r > r_{\max}, \end{cases}$$

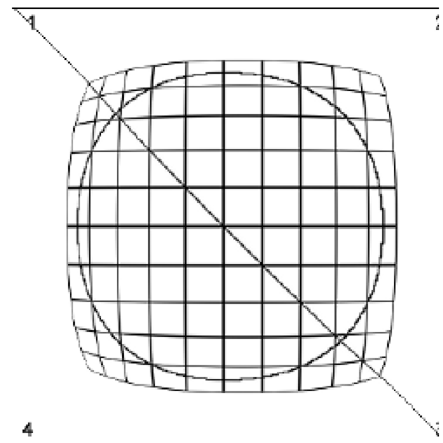
$$T_y^{-1}: \quad y = y' - \begin{cases} z \cdot \tan(\beta_y) & \text{for } r \le r_{\max} \\ 0 & \text{for } r > r_{\max}, \end{cases}$$

$$d_x = x' - x_c, \qquad r = \sqrt{d_x^2 + d_y^2}\,,$$

$$d_y = y' - y_c, \qquad z = \sqrt{r_{\max}^2 - r^2}\,,$$

$$\beta_x = \left(1 - \tfrac{1}{\rho}\right) \cdot \sin^{-1}\left(\tfrac{d_x}{\sqrt{(d_x^2 + z^2)}}\right),$$

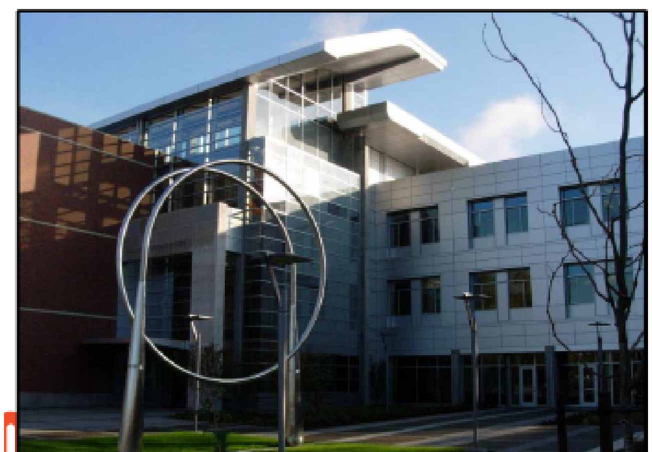
$$\beta_y = \left(1 - \tfrac{1}{\rho}\right) \cdot \sin^{-1}\left(\tfrac{d_y}{\sqrt{(d_y^2 + z^2)}}\right).$$

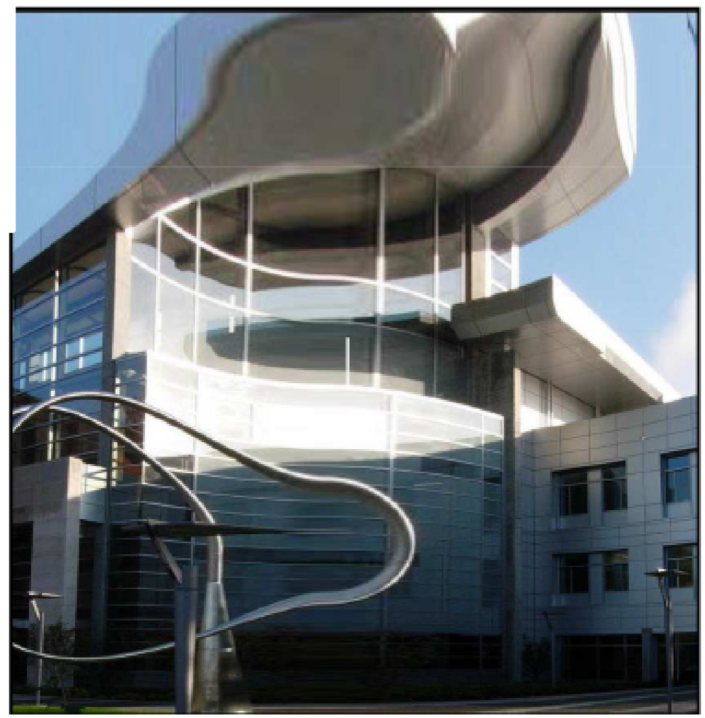


(c)

(f)

Image Warping





Digital Image Processing (CS/ECE 545)
Lecture 11: Comparing Images

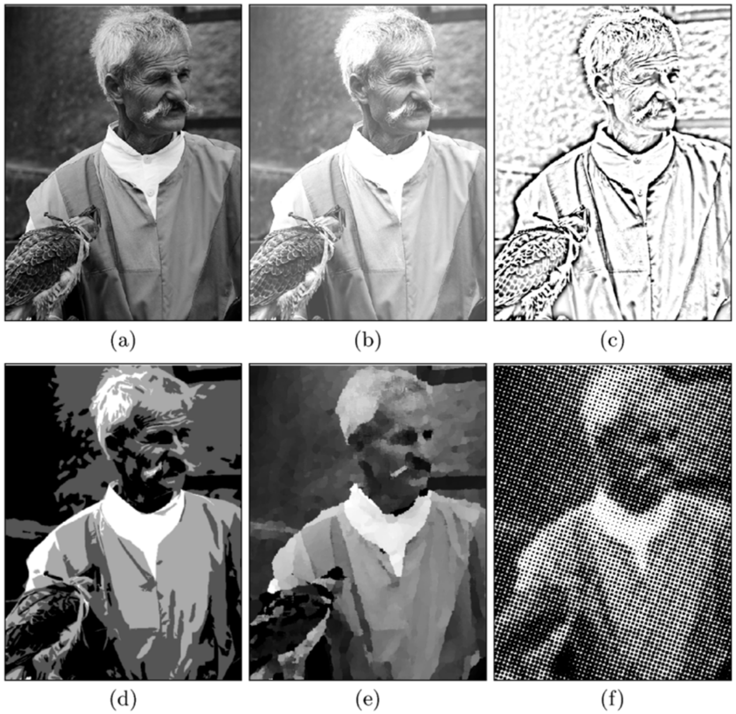
Prof Emmanuel Agu

Computer Science Dept.

Worcester Polytechnic Institute (WPI)

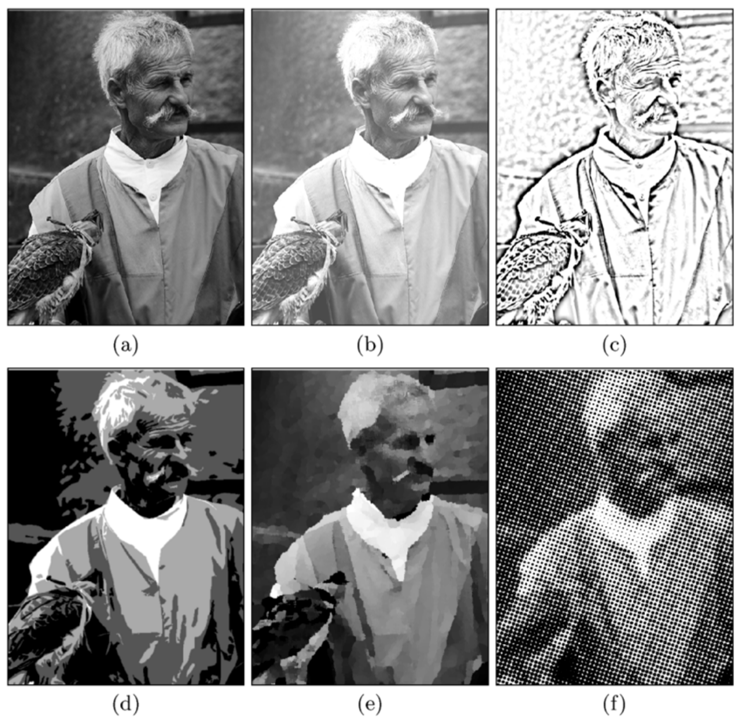
How to tell if 2 Images are same?

- Pixel by pixel comparison?
  - Makes sense only if pictures taken from same angle, same lighting, etc
- Noise, quantization, etc introduces differences
  - Human may say images are same even with numerical differences



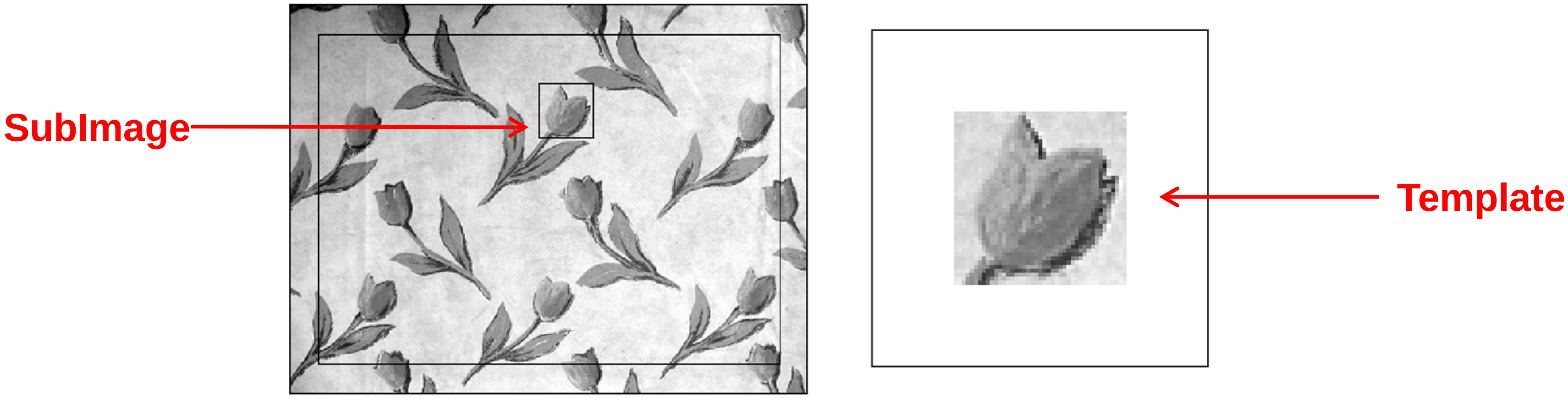
Comparing Images

- Better approach: Template matching
  - Identify similar sub-images (called template) within 2 images
- Applications?
  - Match left and right picture of stereo images
  - Find particular pattern in scene
  - Track moving pattern through image sequence



Template Matching

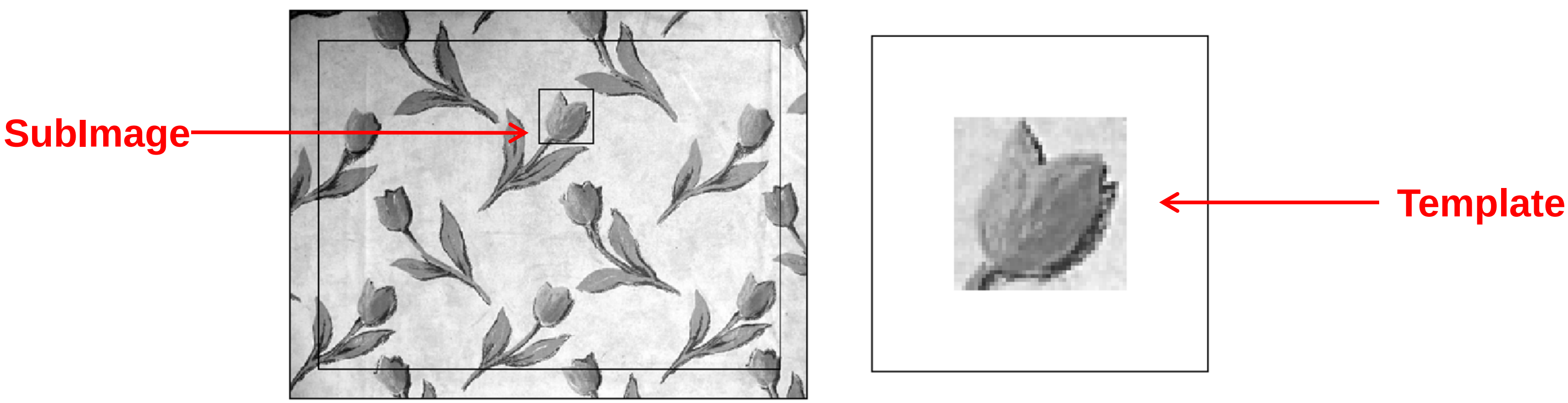
- Basic idea
  - Move given pattern (template) over search image
  - Measure difference between template and sub-images at different positions
  - Record positions where highest similarity is found



(a) original image I (b) reference image R

Template Matching

- Difficult issues?
  - What is distance (difference) measure?
  - What levels of difference should be considered a match?
  - How are results affected when brightness or contrast changes?

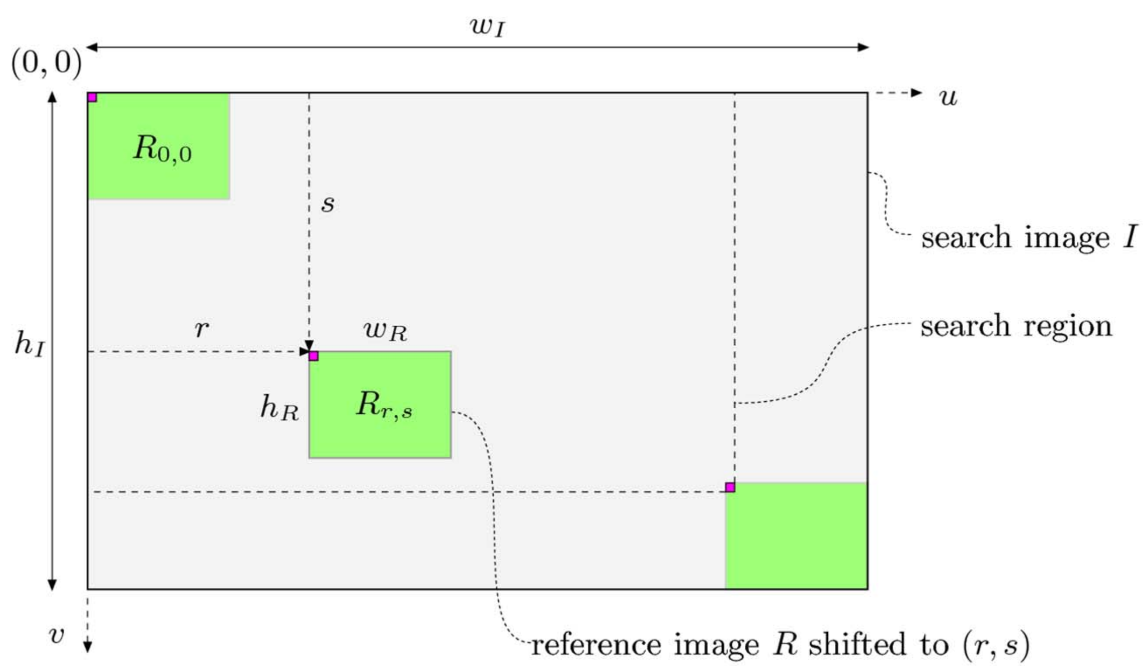


(a) original image I (b) reference image R

Template Matching in Intensity Images

- Consider problem of finding a template (**reference image**) **R** within a **search image**
- Can be restated as **Finding positions in which contents of R are most similar to the corresponding subimage of I**
- If we denote R shifted by some distance *(r,s)* by

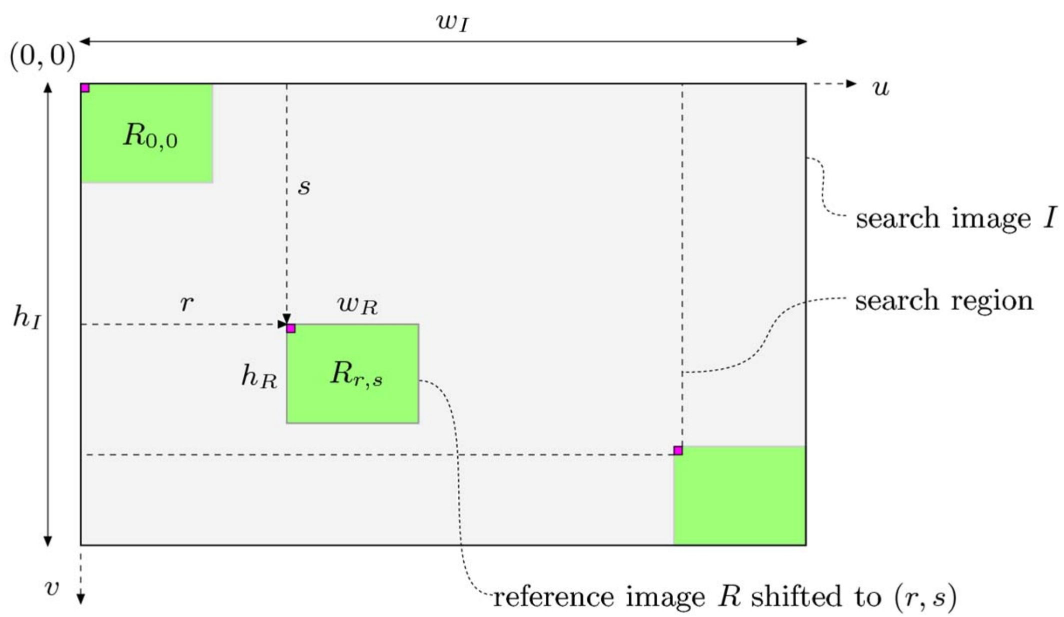
$$R_{r,s}(u, v) = R(u - r, v - s)$$



Template Matching in Intensity Images

- We can restate template matching problem as:
- Finding the offset *(r,s)* such that the similarity between the shifted reference image $R_{r,s}$ and corresponding subimage I is a maximum

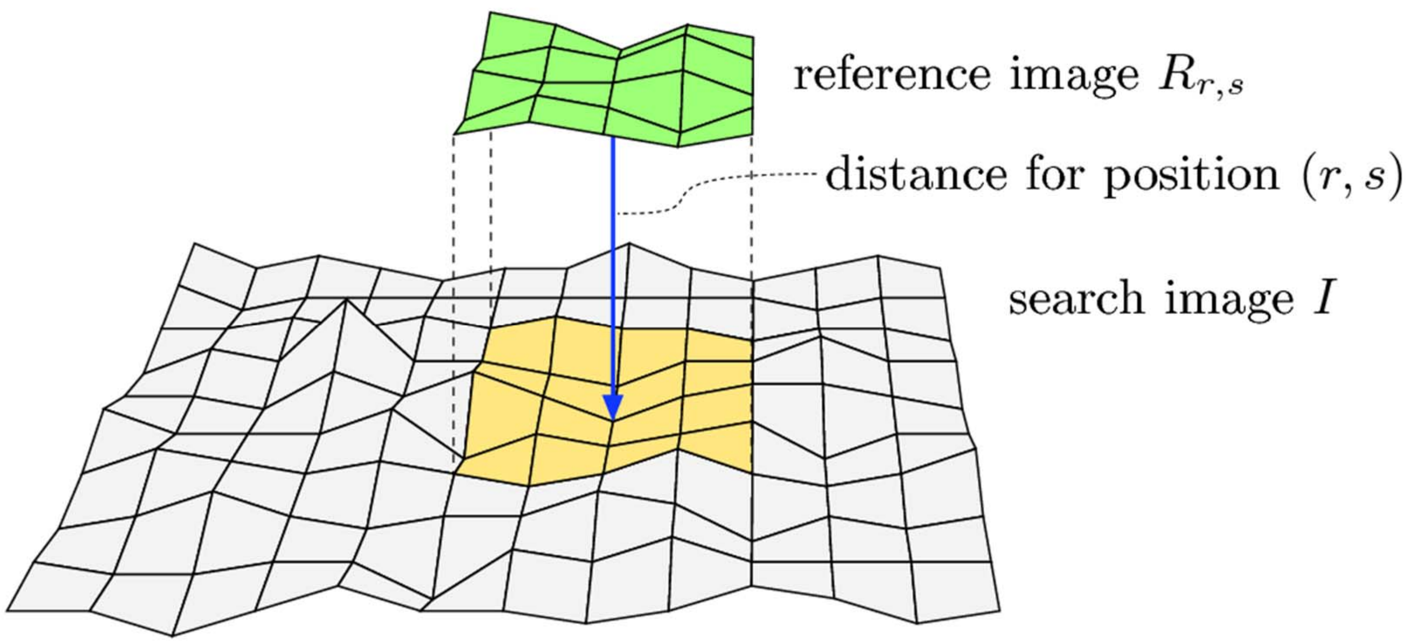
$$R_{r,s}(u,v) = R(u-r, v-s)$$



- Solving this problem involves solving many sub-problems

Distance between Image Patterns

- Many measures proposed to compute distance between the shifted reference image $R_{r,s}$ and corresponding subimage I



Distance between Image Patterns

- Many measures proposed to compute distance between the shifted reference image $R_{r,s}$ and corresponding subimage I
- Sum of absolute differences:

$$\mathrm{d}_A(r,s) = \sum_{(i,j)\in R} |I(r+i, s+j) - R(i,j)|$$

- Maximum difference:

$$\mathrm{d}_M(r,s) = \max_{(i,j)\in R} |I(r+i, s+j) - R(i,j)|$$

- Sum of squared differences (also called N-dimensional Euclidean distance):

$$\mathrm{d}_E(r,s) = \Big[\sum_{(i,j)\in R} \big(I(r+i, s+j) - R(i,j)\big)^2 \Big]^{1/2}$$

Distance and Correlation

- Best matching position between shifted reference image $R_{r,s}$ and subimage I minimizes square of d_E which can be expanded as

$$\mathrm{d}_E^2(r,s) = \sum_{(i,j)\in R} \big(I(r+i,s+j) - R(i,j)\big)^2$$

$$= \underbrace{\sum_{(i,j)\in R} I^2(r+i,s+j)}_{A(r,s)} + \underbrace{\sum_{(i,j)\in R} R^2(i,j)}_{B} - 2\underbrace{\sum_{(i,j)\in R} I(r+i,s+j)\cdot R(i,j)}_{C(r,s)}$$

- *B* term is a constant, independent of *r, s* and can be ignored
- *A* term is sum of squared values within subimage *I* at current offset *r, s*

Distance and Correlation

$$d_E^2(r,s) = \sum_{(i,j)\in R} \big(I(r+i, s+j) - R(i,j)\big)^2$$

$$= \underbrace{\sum_{(i,j)\in R} I^2(r+i, s+j)}_{A(r,s)} + \underbrace{\sum_{(i,j)\in R} R^2(i,j)}_{B} - 2\underbrace{\sum_{(i,j)\in R} I(r+i, s+j)\cdot R(i,j)}_{C(r,s)}$$

- *C(r,s)* term is **linear cross correlation** between *I* and *R* defined as

$$(I \circledast R)(r,s) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(r+i, s+j)\cdot R(i,j)$$

- Since *R* and *I* are assumed to be zero outside their boundaries

$$\sum_{i=0}^{w_R-1} \sum_{j=0}^{h_R-1} I(r+i, s+j)\cdot R(i,j) = \sum_{(i,j)\in R} I(r+i, s+j)\cdot R(i,j)$$

- **Note:** Correlation is similar to linear convolution
- Min value of $d^2_E(r,s)$ corresponds to max value of $(I \circledast R)(r,s)$

Normalized Cross Correlation

- Unfortunately, *A* term is not constant in most images
- Thus cross correlation result varies with intensity changes in image *I*
- **Normalized cross correlation** considers energy in *I* and *R*

$$C_N(r,s) = \frac{C(r,s)}{\sqrt{A(r,s)\cdot B}} = \frac{C(r,s)}{\sqrt{A(r,s)}\cdot\sqrt{B}}$$

$$= \frac{\sum_{(i,j)\in R} I(r+i,s+j)\cdot R(i,j)}{\left[\sum_{(i,j)\in R} I^2(r+i,s+j)\right]^{1/2}\cdot\left[\sum_{(i,j)\in R} R^2(i,j)\right]^{1/2}}$$

- $C_N(r,s)$ is a local distance measure, is in [0,1] range
- $C_N(r,s) = 1$ indicates maximum match
- $C_N(r,s) = 0$ indicates images are very dissimilar

Correlation Coefficient

- **Correlation coefficient:** Use differences between *I* and *R* and their average values

$$C_L(r,s) = \frac{\sum_{(i,j)\in R} \left(I(r+i,s+j) - \bar{I}(r,s)\right) \cdot \left(R(i,j) - \bar{R}\right)}{\left[\sum_{(i,j)\in R} \left(I(r+i,s+j) - \bar{I}_{r,s}\right)^2\right]^{1/2} \cdot \underbrace{\left[\sum_{(i,j)\in R} \left(R(i,j) - \bar{R}\right)^2\right]^{1/2}}_{S_R^2 = K \cdot \sigma_R^2}}$$

where the average values are defined as

$$\bar{I}_{r,s} = \frac{1}{K} \cdot \sum_{(i,j)\in R} I(r+i,s+j) \quad \text{and} \quad \bar{R} = \frac{1}{K} \cdot \sum_{(i,j)\in R} R(i,j)$$

- *K* is number of pixels in reference image *R*
- $C_L(r,s)$ can be rewritten as

$$C_L(r,s) = \frac{\sum_{(i,j)\in R} \left(I(r+i,s+j) \cdot R(i,j)\right) - K \cdot \bar{I}_{r,s} \cdot \bar{R}}{\left[\sum_{(i,j)\in R} I^2(r+i,s+j) - K \cdot \bar{I}_{r,s}^2\right]^{1/2} \cdot S_R}$$

Correlation Coefficient Algorithm

CorrelationCoefficient (I, R)

$I(u,v)$: search image of size $w_I \times h_I$
$R(i,j)$: reference image of size $w_R \times h_R$
Returns $C(r,s)$ containing the values of the correlation coefficient between I and R positioned at (r,s).

Step 1—initialize:

$K \leftarrow w_R \cdot h_R$
$\Sigma_R \leftarrow 0,\ \Sigma_{R2} \leftarrow 0$
for $i \leftarrow 0 \ldots (w_R - 1)$ **do**
  for $j \leftarrow 0 \ldots (h_R - 1)$ **do**
    $\Sigma_R \leftarrow \Sigma_R + R(i,j)$
    $\Sigma_{R2} \leftarrow \Sigma_{R2} + \big(R(i,j)\big)^2$

$\bar{R} \leftarrow \Sigma_R / K$ ▷ Eqn. (17.8)
$S_R \leftarrow \sqrt{\Sigma_{R2} - K \cdot \bar{R}^2} = \sqrt{\Sigma_{R2} - \Sigma_R^2 / K}$ ▷ Eqn. (17.10)

Step 2—compute the correlation map:

$C \leftarrow$ new map of size $(w_I - w_R + 1) \times (h_I - h_R + 1)$, $C(r,s) \in \mathbb{R}$
for $r \leftarrow 0 \ldots (w_I - w_R)$ **do** ▷ place R at position (r,s)
  for $s \leftarrow 0 \ldots (h_I - h_R)$ **do**
    Compute correlation coefficient for position (r,s):
    $\Sigma_I \leftarrow 0,\ \Sigma_{I2} \leftarrow 0,\ \Sigma_{IR} \leftarrow 0$
    for $i \leftarrow 0 \ldots (w_R - 1)$ **do**
      for $j \leftarrow 0 \ldots (h_R - 1)$ **do**
        $a_I \leftarrow I(r+i, s+j)$
        $a_R \leftarrow R(i,j)$
        $\Sigma_I \leftarrow \Sigma_I + a_I$
        $\Sigma_{I2} \leftarrow \Sigma_{I2} + a_I^2$
        $\Sigma_{IR} \leftarrow \Sigma_{IR} + a_I \cdot a_R$

    $\bar{I}_{r,s} \leftarrow \Sigma_I / K$ ▷ Eqn. (17.8)
    $$C(r,s) \leftarrow \frac{\Sigma_{IR} - K \cdot \bar{I}_{r,s} \cdot \bar{R}}{\sqrt{\Sigma_{I2} - K \cdot \bar{I}_{r,s}^2} \cdot S_R} = \frac{\Sigma_{IR} - \Sigma_I \cdot \bar{R}}{\sqrt{\Sigma_{I2} - \Sigma_I^2 / K} \cdot S_R}$$

return C. ▷ $C(r,s) \in [-1, 1]$

Correlation Coefficient Java Implementation

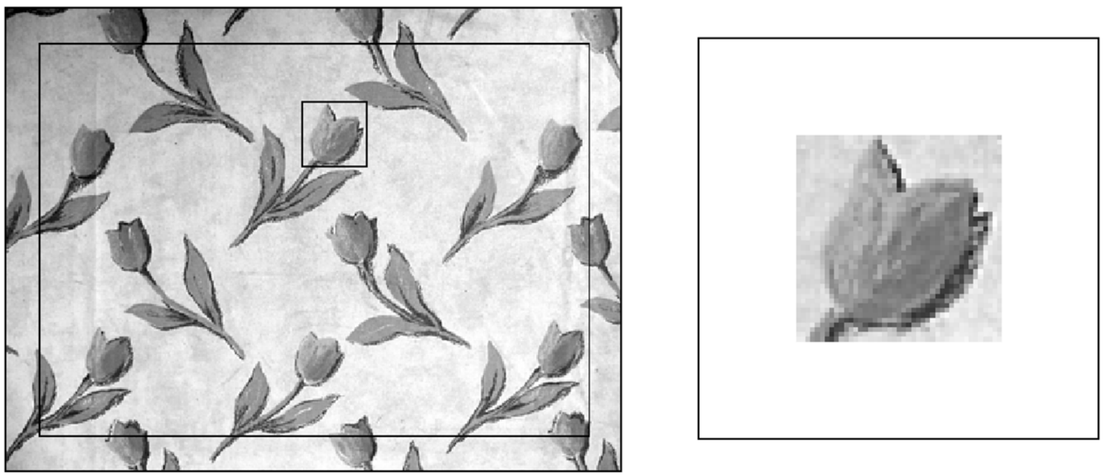
```java
class CorrCoeffMatcher {
  FloatProcessor I; // image
  FloatProcessor R; // template
  int wI, hI;       // width/height of image
  int wR, hR;       // width/height of template
  int K;            // size of template

  float meanR;      // mean value of template (R̄)
  float varR;       // square root of template variance (σ_R)

  public CorrCoeffMatcher( // constructor method
          FloatProcessor img,  // search image (I)
          FloatProcessor ref)  // reference image (R)
  {
    I = img;
    R = ref;
    wI = I.getWidth();
    hI = I.getHeight();
    wR = R.getWidth();
    hR = R.getHeight();
    K = wR * hR;

    // compute the mean (R̄) and variance term (S_R) of the template:
    float sumR = 0;     // Σ_R = ∑ R(i,j)
    float sumR2 = 0;    // Σ_R2 = ∑ R²(i,j)
    for (int j = 0; j < hR; j++) {
      for (int i = 0; i < wR; i++) {
        float aR = R.getf(i, j);
        sumR  += aR;
        sumR2 += aR * aR;
      }
    }
    meanR = sumR / K;   // R̄ = [∑ R(i,j)]/K
    varR =              // S_R = [∑ R²(i,j) − K·R̄²]^(1/2)
          (float) Math.sqrt(sumR2 - K * meanR * meanR);
  }

  // continued...
```

Correlation Coefficient Java Implementation

```
public FloatProcessor computeMatch() {
  FloatProcessor C = new FloatProcessor(wI-wR+1, hI-hR+1);
  for (int r = 0; r <= wI-wR; r++) {
    for (int s = 0; s <= hI-hR; s++) {
      float d = getMatchValue(r,s);
      C.setf(r, s, d);
    }
  }
  return C;
}

float getMatchValue(int r, int s) {
  float sumI = 0;    // $\Sigma_I = \sum I(r+i, s+j)$
  float sumI2 = 0;   // $\Sigma_{I2} = \sum (I(r+i, s+j))^2$
  float sumIR = 0;   // $\Sigma_{IR} = \sum I(r+i, s+j) \cdot R(i, j)$

  for (int j = 0; j < hR; j++) {
    for (int i = 0; i < wR; i++) {
      float aI = I.getf(r+i, s+j);
      float aR = R.getf(i, j);
      sumI  += aI;
      sumI2 += aI * aI;
      sumIR += aI * aR;
    }
  }
  float meanI = sumI / K;          // $\bar{I}_{r,s} = \Sigma_I / K$
  return (sumIR - K * meanI * meanR) /
    ((float)Math.sqrt(sumI2 - K * meanI * meanI) * varR);
}

} // end of class CorrCoeffMatcher
```

Examples and Discussion

- We now compare these distance metrics
- **Original image *I*:** Repetitive flower pattern
- **Reference image *R*:** one instance of repetitive pattern extracted from *I*



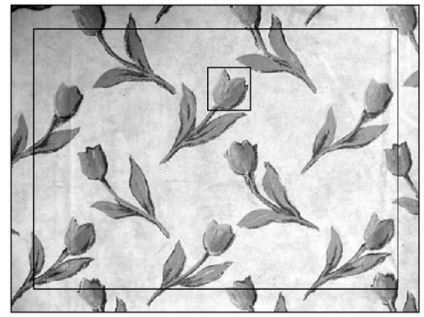
(a) original image I (b) reference image R

- Now compute various distance measures for this *I* and *R*

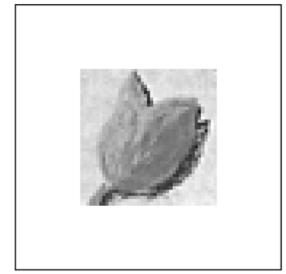
Examples and Discussion

(c) sum of absolute differences

(d) maximum difference



(a) original image I



(b) reference image R

- **Sum of absolute differences:** performs okay but affected by global intensity changes

$$d_A(r,s) = \sum_{(i,j)\in R} |I(r+i, s+j) - R(i,j)|$$

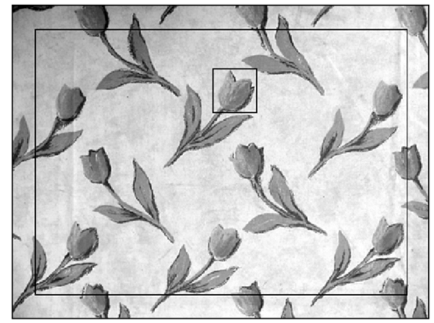
- **Maximum difference:** Responds more to lighting intensity changes than pattern similarity

$$d_M(r,s) = \max_{(i,j)\in R} |I(r+i, s+j) - R(i,j)|$$

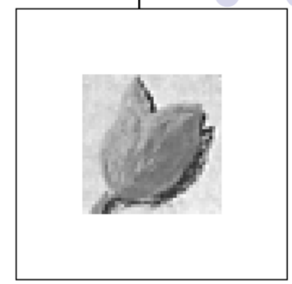
Examples and Discussion

(e) sum of squared distances

(f) global cross correlation



(a) original image I



(b) reference image R

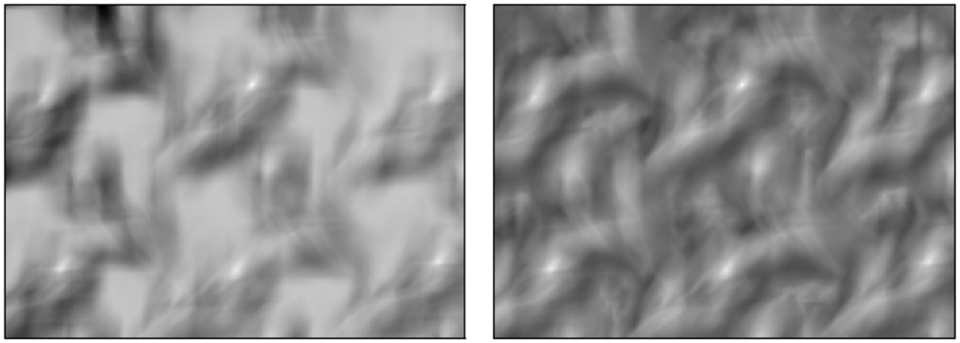
- **Sum of squared (euclidean) distances:** performs okay but affected by global intensity changes

$$\mathrm{d}_E(r,s) = \Big[\sum_{(i,j)\in R} \big(I(r+i, s+j) - R(i,j)\big)^2 \Big]^{1/2}$$

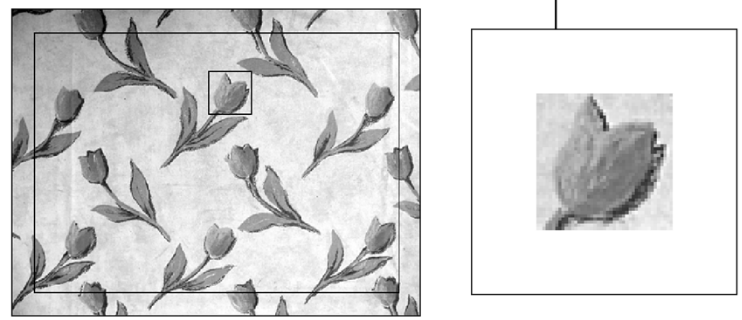
- **Global cross correlation:** Local maxima at true template position, but is dominated by high-intensity responses in brighter image parts

$$(I \circledast R)(r,s) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(r+i, s+j) \cdot R(i,j)$$

Examples and Discussion



(g) normalized cross correlation (h) correlation coefficient



(a) original image I (b) reference image R

- **Normalized cross correlation:** results similar to euclidean distance (affected by global intensity changes)

$$\frac{\sum_{(i,j)\in R} I(r+i, s+j)\cdot R(i,j)}{\left[\sum_{(i,j)\in R} I^2(r+i,s+j)\right]^{1/2} \cdot \left[\sum_{(i,j)\in R} R^2(i,j)\right]^{1/2}}$$

- **Correlation coefficient:** yields best results. Distinct peaks produced for all 6 template instances, unaffected by lighting

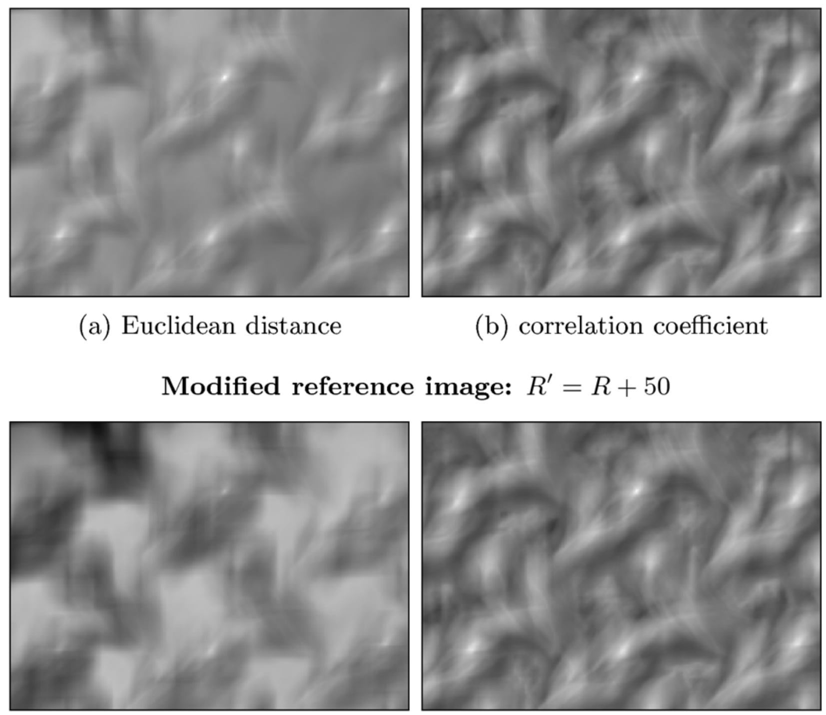
$$C_L(r,s) = \frac{\sum_{(i,j)\in R} \big(I(r+i,s+j)\cdot R(i,j)\big) \; - \; K\cdot \bar{I}_{r,s}\cdot \bar{R}}{\left[\sum_{(i,j)\in R} I^2(r+i,s+j) \; - \; K\cdot \bar{I}^2_{r,s}\right]^{1/2} \cdot S_R}$$

Effects of Changing Intensity

- To explore effects of globally changing intensity, raise intensiy of reference image *R* by 50 units
- Distinct peaks disappear in **Euclidean distance**
- **Correlation coefficient** unchanged, robust measure in realistic lighting conditions

(a) original image I (b) reference image R

Original reference image: R

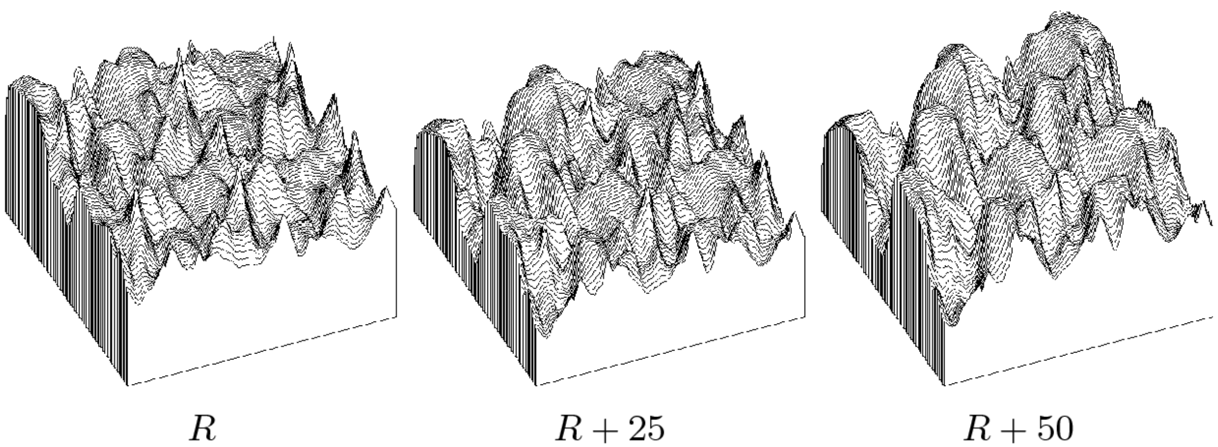


(a) Euclidean distance (b) correlation coefficient

Modified reference image: $R' = R + 50$

(c) Euclidean distance (d) correlation coefficient

Euclidean Distance under Global Intensity Changes



Distance function for original template *R*

Distance function with intensity increased by 25 units

Distance function with intensity increased by 50 units

- Local peaks disappear as template intensity (and thus distance) is increased

Shape of Template

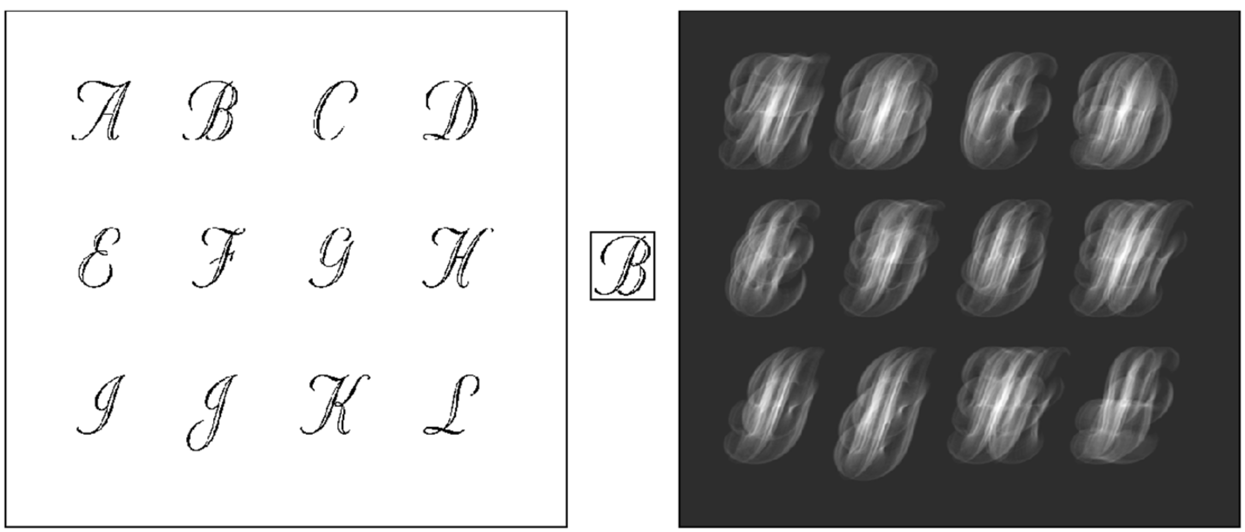
- Template does not have to be rectangular
- Some applications use circular, elliptical or custom-shaped templates
- Non-rectangular templates stored in rectangular array, but pixels in template marked using a mask
- More generally, a weighted function can be applied to template elements

Matching under Rotation and Scaling

- **Simple Approach:**
  - Store multiple rotated and scaled versions of template
  - Computationally prohibitive
- Alternate approaches:
  - Matching in logarithmic-polar space (complicated!)
  - Affine matching use local statistical features invariant under affine image transformations (including rotation and scaling)

Matching Binary Images

- **Direct Comparison:**
  - Count the number of identical pixels in search image and template
  - Small total difference when most pixels are same
- Problem: Small shift, rotation or distortion of image create high distance
- Need a more tolerant measure



(a) (b) (c)

The Distance Transform

- For every position (u,v) in the search image *I*, record distance to closest foreground pixel
- So, for binary image

$$FG(I) = \{\boldsymbol{p} \mid I(\boldsymbol{p}) = 1\}$$
$$BG(I) = \{\boldsymbol{p} \mid I(\boldsymbol{p}) = 0\}$$

- Distance transform is defined as

$$D(\boldsymbol{p}) = \min_{\boldsymbol{p}' \in FG(I)} \mathrm{dist}(\boldsymbol{p}, \boldsymbol{p}')$$

- Examples of distance measures are **Euclidean distance**

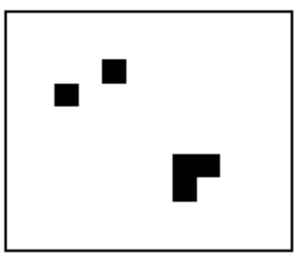
$$\mathrm{d}_E(\boldsymbol{p}, \boldsymbol{p}') = \|\boldsymbol{p} - \boldsymbol{p}'\| = \sqrt{(u-u')^2 + (v-v')^2} \quad \in \mathbb{R}^+$$

- Or **Manhattan distance**

$$\mathrm{d}_M(\boldsymbol{p}, \boldsymbol{p}') = |u-u'| + |v-v'| \quad \in \mathbb{N}_0$$

Distance Transform Example

- Example using Manhattan distance



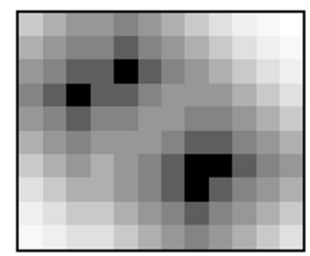
binary image

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	**1**	0	0	0	0	0	0	0
0	0	**1**	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	**1**	**1**	0	0	0
0	0	0	0	0	0	0	**1**	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

$\longrightarrow$

distance transform

5	4	3	3	2	3	4	5	6	7	8	9
4	3	2	2	1	2	3	4	5	6	7	8
3	2	1	1	**0**	1	2	3	4	5	6	7
2	1	**0**	1	1	2	3	3	3	4	5	6
3	2	1	2	2	3	3	2	2	3	4	5
4	3	2	3	3	3	2	1	1	2	3	4
5	4	3	4	3	2	1	**0**	**0**	1	2	3
6	5	4	4	3	2	1	**0**	1	2	3	4
7	6	5	5	4	3	2	1	2	3	4	5
8	7	6	6	5	4	3	2	3	4	5	6



Chamfer Algorithm

- Efficient method to compute distance transform
- Similar to sequential region labeling
- Traverses image twice
  - First, starting at upper left corner of image, propagates distance values downward in diagonal direction
  - Second traversal starts at bottom right, proceeds in opposite direction (bottom to top)
- For each traversal, the following masks is used for propagating distance values

$$M^L = \begin{bmatrix} m_2^L & m_3^L & m_4^L \\ m_1^L & \times & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \qquad M^R = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \times & m_1^R \\ m_4^R & m_3^R & m_2^R \end{bmatrix}$$

Chamfer Distance

- Specifically, for masks for Manhattan distance

$$M_M^L = \begin{bmatrix} 2 & 1 & 2 \\ 1 & \times & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \qquad M_M^R = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \times & 1 \\ 2 & 1 & 2 \end{bmatrix}$$

- And masks for Euclidean distance

$$M_E^L = \begin{bmatrix} \sqrt{2} & 1 & \sqrt{2} \\ 1 & \times & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \qquad M_E^R = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \times & 1 \\ \sqrt{2} & 1 & \sqrt{2} \end{bmatrix}$$

- Floating point-operations can be avoided using distance masks with scaled integer values for Euclidean distance such as

$$M_{E'}^L = \begin{bmatrix} 4 & 3 & 4 \\ 3 & \times & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \qquad M_{E'}^R = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \times & 3 \\ 4 & 3 & 4 \end{bmatrix}$$

Chamfer Matching

- Uses distance transform for matching binary images
- Finds points of maximum agreement between binary search image *I* and binary reference image *R*
- Accumulates values of distance transform as match score *Q*
- At each position, *(r,s)* of the template *R*, distance values to all foreground pixels are accumulated

$$Q(r,s) = \frac{1}{K} \cdot \sum_{(i,j) \in FG(R)} D(r+i,\, s+j)$$

where *K* = |*FG(R)*| is number of foreground pixels in template *R*

- Zero *Q* score = maximum match
- Large *Q* score = large deviations
- Best match corresponds to global minimum of *Q*

Chamfer Matching

```
ChamferMatch (I, R)
    I: binary search image of size w_I × h_I
    R: binary reference image of size w_R × h_R
    Returns a two-dimensional map of match scores.

    Step 1—initialize:
    D ← DistanceTransform(I)                                   ▷ see Alg. 17.2
    K ← number of foreground pixels in R
    Q ← new match map of size (w_I−w_R+1) × (h_I−h_R+1), Q(r, s) ∈ ℝ

    Step 2—compute the match score:
    for r ← 0...(w_I−w_R) do                                   ▷ place R at (r, s)
        for s ← 0...(h_I−h_R) do
            Get match score for template placed at (r, s):
            q ← 0
            for i ← 0...(w_R−1) do
                for j ← 0...(h_R−1) do
                    if R(i, j) = 1 then                        ▷ foreground pixel in template
                        q ← q + D(r+i, s+j)
            Q(r, s) ← q/K

    return Q.
```

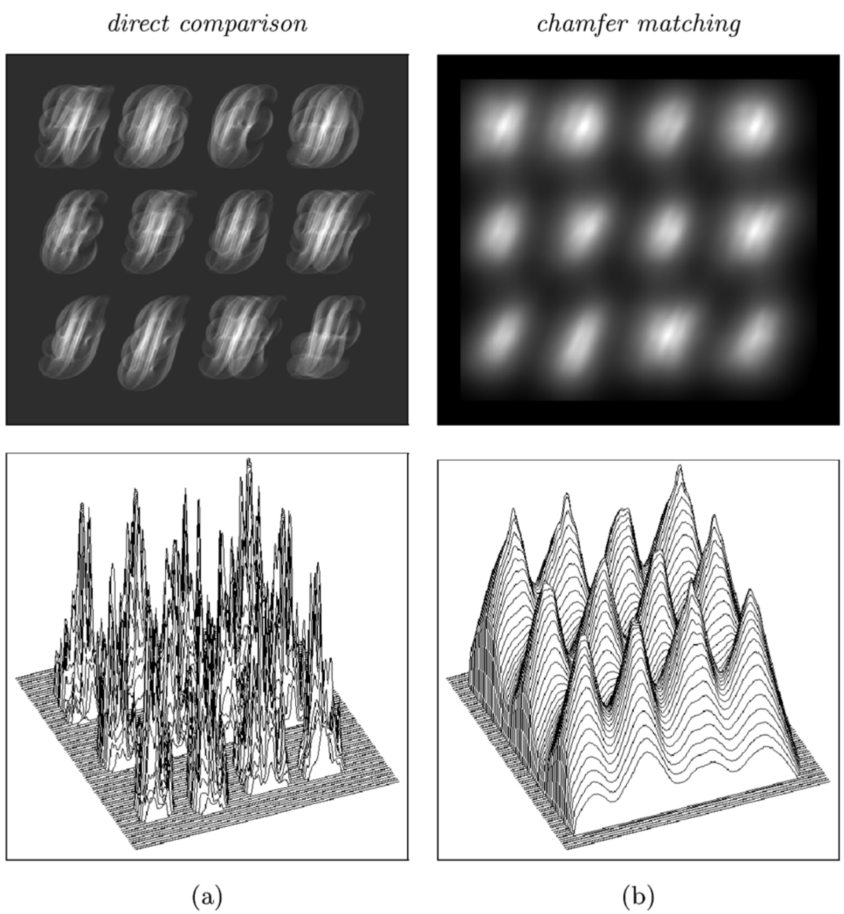
Compute distance transform D of image using Chamfer algorithm

Accumulate sum of distance values For all foreground pixels in template *R*

Results stored in 2D match map *D*

Comparing Direct Pixel comparison and Chamfer Matching

- Chamfer match score *Q* much smoother than direct comparison
  - Distinct peaks in places of high similarity



Distance Transform using Chamfer Algorithm

DistanceTransform (I)

I: binary image of size $M \times N$.

Returns the distance transform of image I.

Step 1—initialize:

$D \leftarrow$ new *distance map* of size $M \times N$, $D(u,v) \in \mathbb{R}$
for all image coordinates (u,v) **do**
    if $I(u,v) = 1$ **then**
        $D(u,v) \leftarrow 0$ ▷ foreground pixel (zero distance)
    else
        $D(u,v) \leftarrow \infty$ ▷ background pixel (infinite distance)

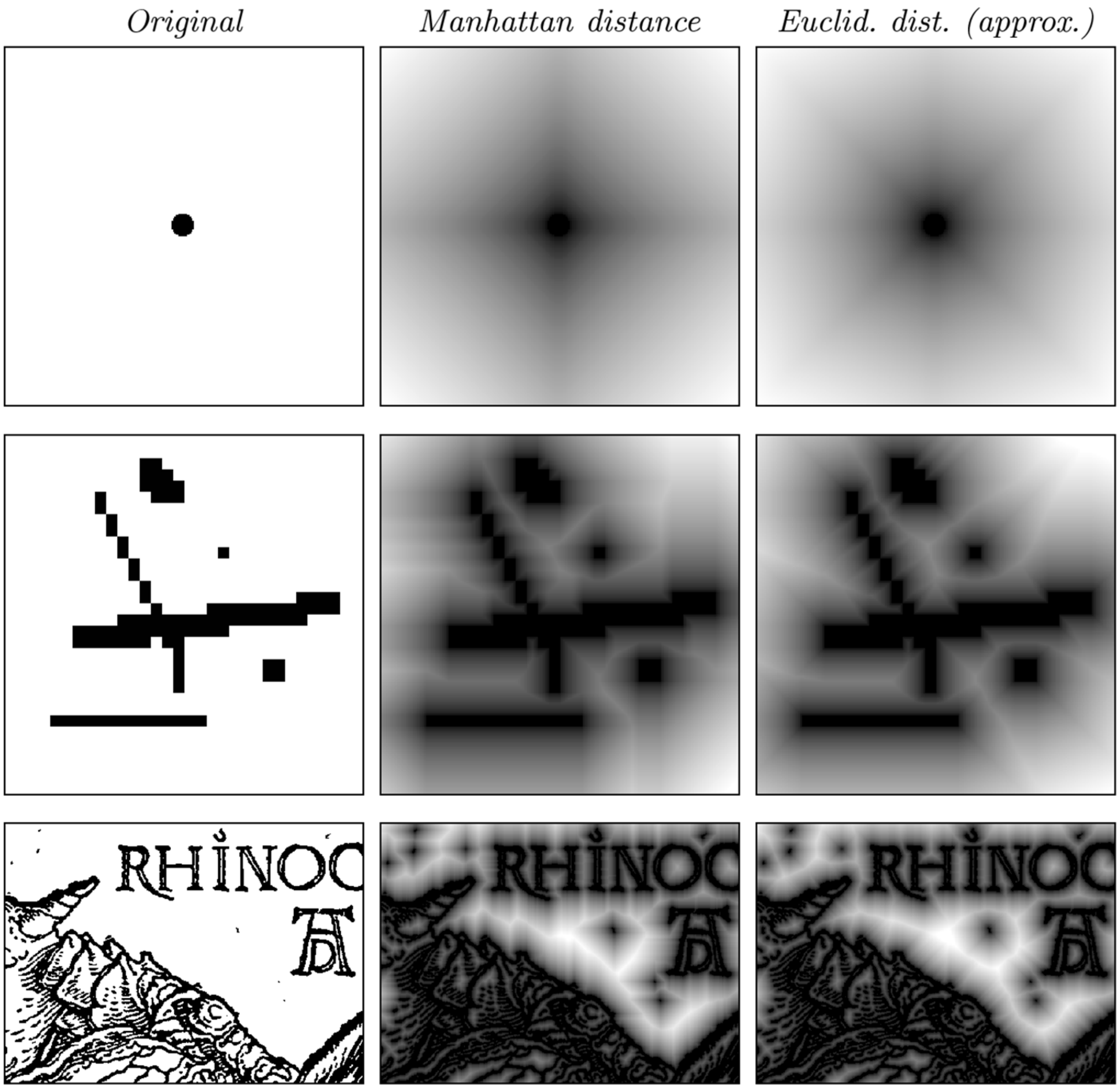
Step 2—L→R pass (using distance mask $M^L = m_i^L$):

for $v \leftarrow 1, 2, \ldots, N{-}1$ **do** ▷ top → bottom
    for $u \leftarrow 1, 2, \ldots, M{-}2$ **do** ▷ left → right
        if $D(u,v) > 0$ **then**
            $d_1 \leftarrow m_1^L + D(u{-}1, v)$
            $d_2 \leftarrow m_2^L + D(u{-}1, v{-}1)$
            $d_3 \leftarrow m_3^L + D(u, v{-}1)$
            $d_4 \leftarrow m_4^L + D(u{+}1, v{-}1)$
            $D(u,v) \leftarrow \min(d_1, d_2, d_3, d_4)$

Step 3—R→L pass (using distance mask $M^R = m_i^R$):

for $v \leftarrow N{-}2, \ldots, 1, 0$ **do** ▷ bottom → top
    for $u \leftarrow M{-}2, \ldots, 2, 1$ **do** ▷ right → left
        if $D(u,v) > 0$ **then**
            $d_1 \leftarrow m_1^R + D(u{+}1, v)$
            $d_2 \leftarrow m_2^R + D(u{+}1, v{+}1)$
            $d_3 \leftarrow m_3^R + D(u, v{+}1)$
            $d_4 \leftarrow m_4^R + D(u{-}1, v{+}1)$
            $D(u,v) \leftarrow \min(D(u,v), d_1, d_2, d_3, d_4)$

return D.



Distance Transform using Chamfer Algorithm

Digital Image Processing (CS/ECE 545)
Lecture 11: Future Directions

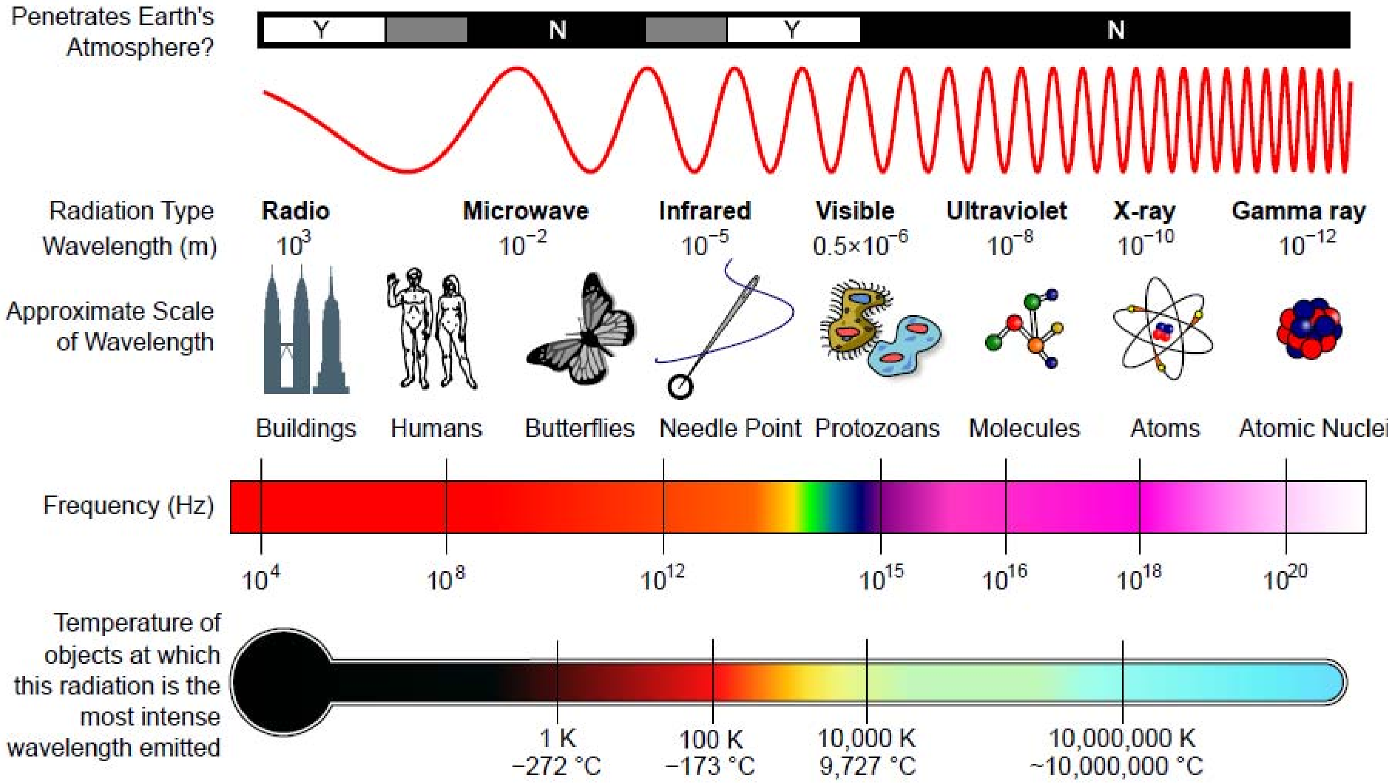
Prof Emmanuel Agu

Computer Science Dept.

Worcester Polytechnic Institute (WPI)

Recall: Electromagnetic Spectrum and IP

- Images can be made from any form of EM radiation



From Wikipedia

Recall: Images from Different EM Radiation

- Radar imaging (radio waves)
- Magnetic Resonance Imaging (MRI) (Radio waves)
- Microwave imaging
- Infrared imaging
- Photographs
- Ultraviolet imaging telescopes
- X-rays and Computed tomography
- Positron emission tomography (gamma rays)
- Ultrasound (not EM waves)

Non-visible Wavelengths Used for Medical imaging

Medical Imaging Example Technologies

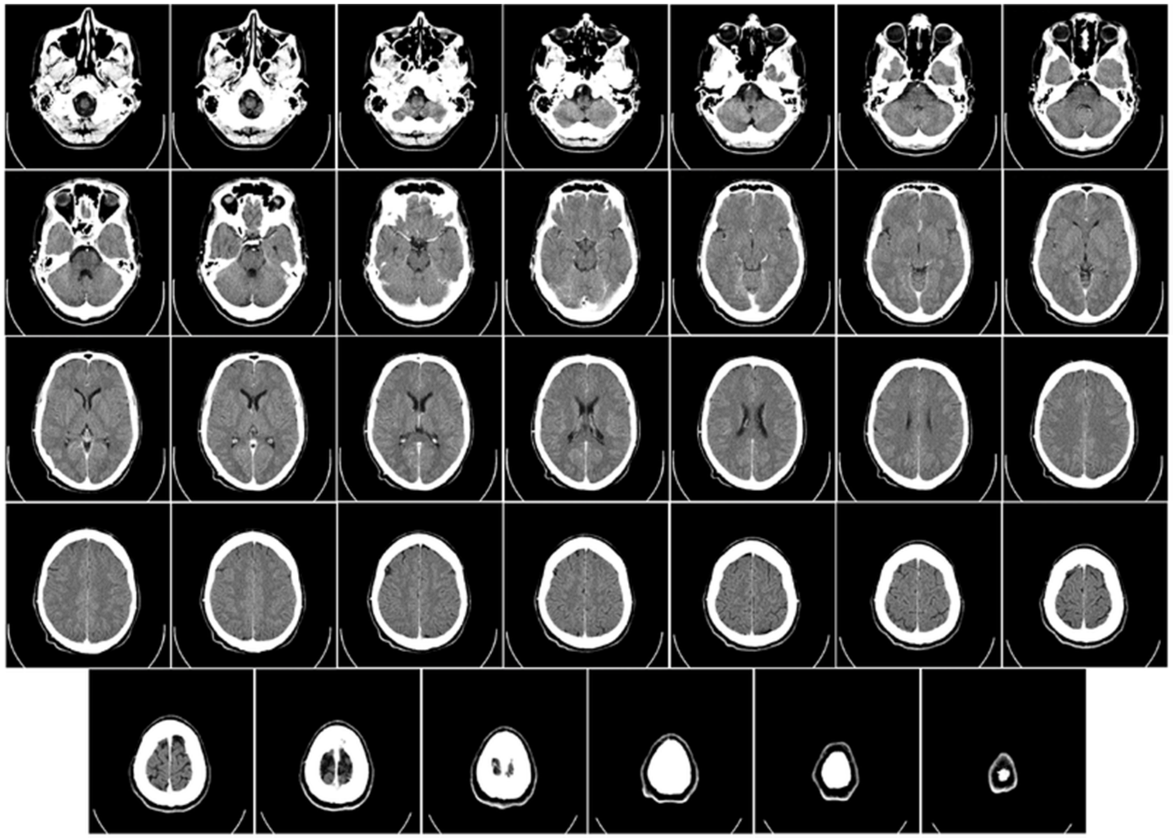
- XRay
- Computerized tomography
- Mammogram
- Nuclear magnetic resonance
- Positron Emission Tomography
- Single Photon Emission Computerized Tomography
- Ultrasound imaging

XRay

- Imaging body internals using electromagnetic waves of wavelength 0.01 to 10 nanometers

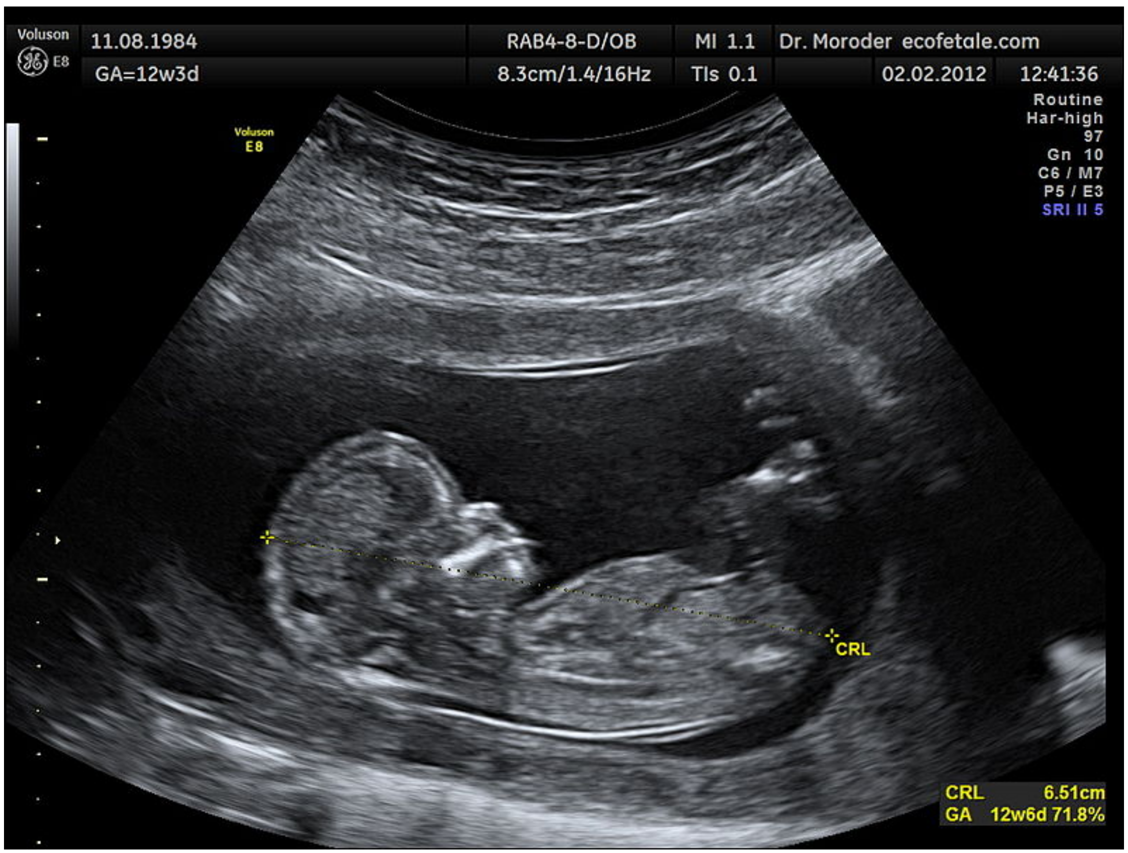
Computerized Tomography

- **Tomography:** Cross-sectional image formed from projections
- **Example:** XRay Computerized tomography of human brain
- Virtual slices allow human to see inside without cutting open



Ultrasound

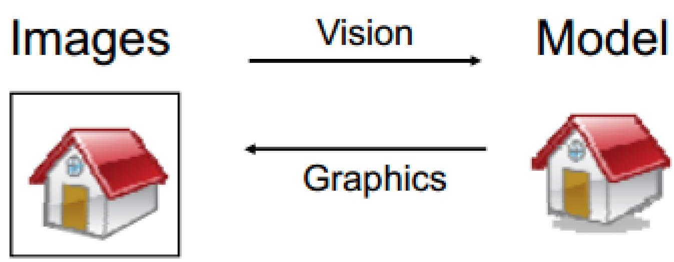
- Uses sound waves undetectable by human ear
- Non-invasive imaging, used for imaging unborn babies



Computer Vision

- Vision builds on Image processing
- Inverse problem to computer graphics

Vision and graphics



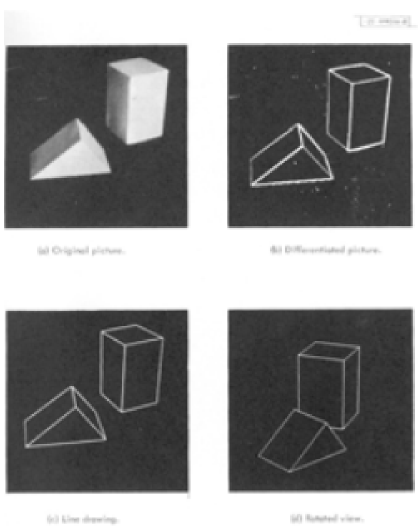
Inverse problems: analysis and synthesis.

Courtesy
Grauman
U of Texas

Why do we need Computer Vision?

- Explosion of visual content
- Let computers help humans with "easy" tasks

Visual data in 1963



L. G. Roberts, *Machine Perception of Three Dimensional Solids*, Ph.D. thesis, MIT Department of Electrical Engineering, 1963.

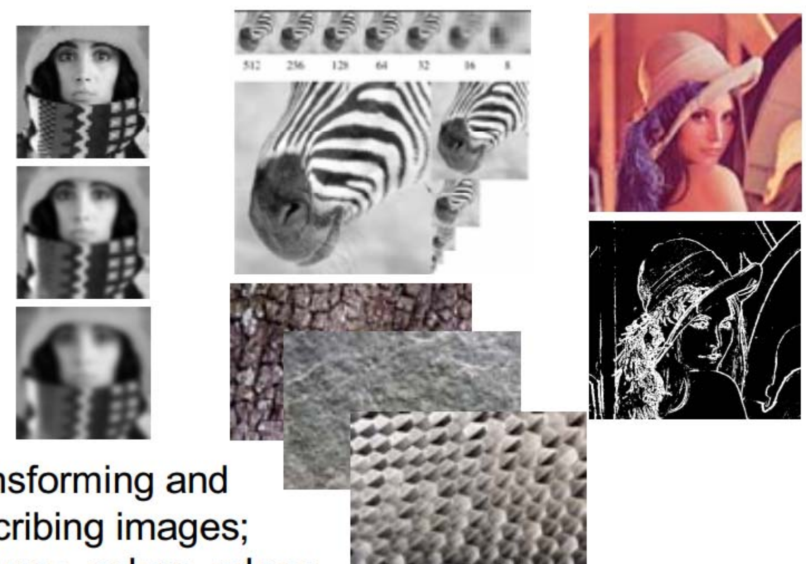
Visual data in 2011

Slide credit; L. Lazebnik

Computer Vision

- **Classic CV task:** Recognize objects in image
- **First step:** Describe images using distinct features (textures, colors, edges, etc)
- Outputs of image processing = inputs for CV

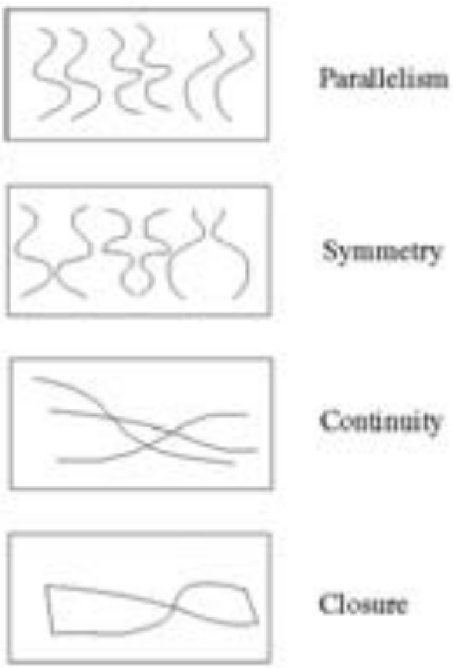
Features and filters

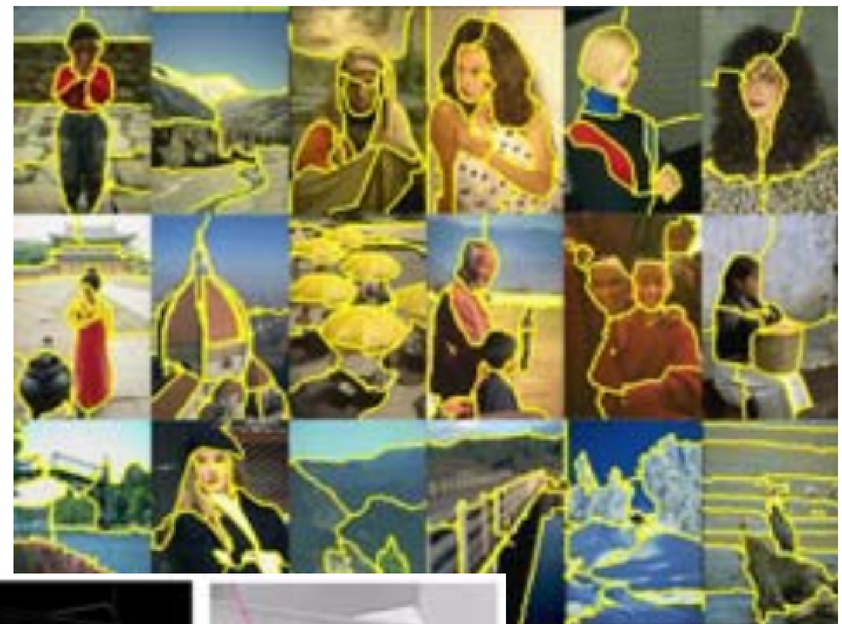


Transforming and
describing images;
textures, colors, edges

Courtesy Grauman U of Texas

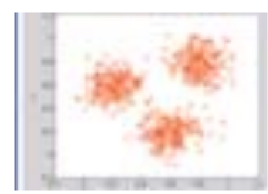
Grouping & fitting





[fig from Shi et al]

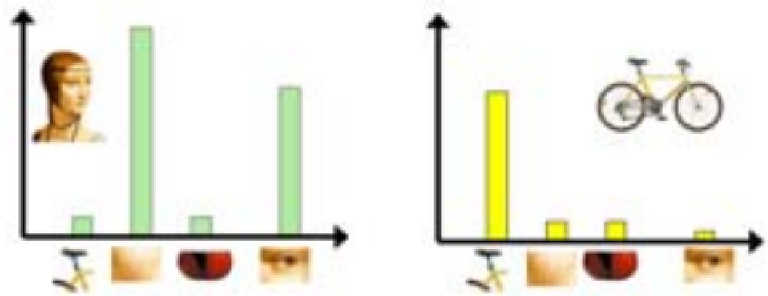
Clustering, segmentation, fitting; what parts belong together?



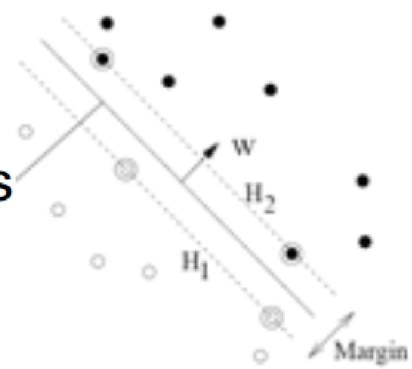
Courtesy Grauman U of Texas

Hough transform, etc

Recognition and learning



Recognizing objects and categories, learning techniques



Courtesy Grauman U of Texas

Digital Forensics

- Detecting when images have been tampered with
- Has been around for a long time
- Example: 1961 Grigoriy Nelyubov, one of astronauts removed from image of Russian astronauts on moon for misbehavior

Computational Photography

- Traditional camera: only configurable settings
- Computational camera: More parts programmable
  - Programmable illumination: complex flash patterns
  - Programmable apertures, shutter, etc
  - Programmable image processing
- What's possible?

Tone Mapping, Color Correction on Camera

- Courtesy Fredo Durand MIT

Depth from Image using programmable aperture

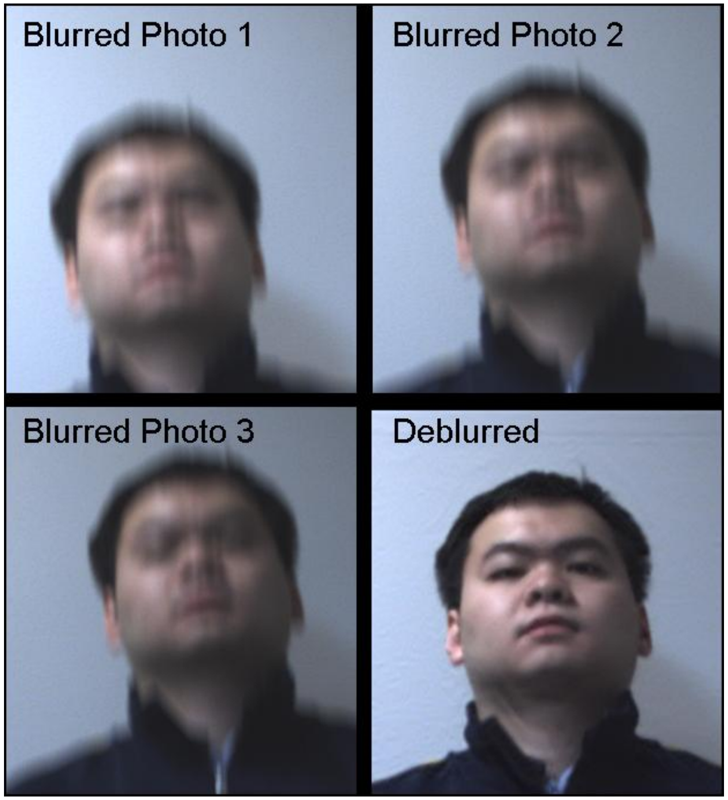
- Courtesy Bill Freeman, MIT

ReFocus badly focussed Images

- Courtesy Fredo Durand MIT

Computational Photography

- What's possible?
  - Deblurring: Take out motion blur artifacts



Computer Graphics
CS/ECE 545 – Final Review

Prof Emmanuel Agu

Computer Science Dept.
Worcester Polytechnic Institute (WPI)

Exam Overview

- Wednesday, April 30, 2014, in-class
- Midterm covered up to lecture 5 (Corner Detection)
- Final covers lecture 6 till today's class (lecture 11)
- Can bring:
  - One page cheat-sheet, hand-written (not typed)
  - Calculator
- Will test:
  - Theoretical concepts
  - Mathematics
  - Algorithms
  - Programming
  - ImageJ knowledge (program structure and some commands)

What am I Really Testing?

- Understanding of
  - concepts (NOT only programming)
  - programming (pseudocode/syntax)
- Test that:
  - you can plug in numbers by hand to check your programs
  - you did the projects
  - you understand what you did in projects

General Advise

- **Read your projects** and refresh memory of what you did
- **Read the slides**: worst case – if you understand slides, you're more than 50% prepared
- Focus on **Mathematical results, concepts, algorithms**
- Plug numbers: calculate by hand
- Try to **predict subtle changes** to algorithm.. What ifs?..
- **Past exams**: One sample final will be on website
- All lectures have references. Look at refs to focus reading
- Do all readings I asked you to do on your own

Grading Policy

- I try to give as much partial credit as possible
- In time constraints, laying out outline of solution gets you healthy chunk of points
- Try to write something for each question
- Many questions will be easy, exponentially harder to score higher in exam

Topics

- Curve Detection
- Morphological Filters
- Regions in Binary Images
- Color Images
- Introduction to Spectral Techniques
- Discrete Fourier Transform
- Geometrical Operations
- Comparing Images
- Future Directions

References

- Wilhelm Burger and Mark J. Burge, Digital Image Processing, Springer, 2008
- University of Utah, CS 4640: Image Processing Basics, Spring 2012
- Rutgers University, CS 334, Introduction to Imaging and Multimedia, Fall 2012
- Gonzales and Woods, Digital Image Processing (3rd edition), Prentice Hall
- CS 376 Slides, Computer Vision, Fall 2011