T-61.5070 COMPUTER VISION, Exercise 3/08

1.

Scene matching is represented in Sec. 5.4, pp. 190–194. Anil K. Jain represented hierarchical search in his book *Fundamentals of Digital Image Processing*, p. 407, as follows:

> If the observed image is very large, we may first search a low-resolution-reduced copy using a likewise reduced copy of the template. If multiple matches occur, then the regions represented by these locations are searched using higher-resolution copies to further refine and reduce the search area. Thus the full-resolution region searched can be a small fraction of the total area. This method of *coarse-fine search* is also logarithmically efficient.

A coarse search is performed by reducing both the image and the template and by matching the reduced template to the image. The reductions are accomplished by replacing every 2×2 neighborhood by its average. Values are rounded to the nearest integer. The reduced template is

$$h_r(i,j) = \begin{array}{|ccc|} \hline 0 & 0 & 1 \\ 0 & 7 & 0 \\ 2 & 0 & 0 \\ \hline \end{array}$$

The reduced image is

$$f_r(u,v) = \begin{array}{|cccccc|} \hline 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 7 & 1 & 1 & 0 \\ 0 & 2 & 0 & 3 & 3 & 0 \\ 0 & 2 & 5 & 0 & 0 & 0 \\ 0 & 0 & 5 & 2 & 0 & 0 \\ 0 & 0 & 2 & 5 & 5 & 0 \\ \hline \end{array}$$

Squared distance of the template from the image blocks yields a measure for their similarity

$$C(u,v) = \sum_{(i,i)\in V} [f(i+u, j+v) - h(i,j)]^2 = \tag{1}$$

$$\begin{array}{|cccccc|} \hline 54 & 103 & 108 & 53 & 56 & 51 \\ 58 & 107 & \mathbf{11} & 113 & 52 & 52 \\ 62 & 94 & 136 & 84 & 32 & 64 \\ 58 & 84 & 49 & 100 & 68 & 63 \\ 54 & 102 & 71 & 126 & 88 & 59 \\ 54 & 73 & 80 & 67 & 38 & 79 \\ \hline \end{array}.$$

The best-matching point, indicated in bold in the distance array, is suggestive for further evalution. A finer search is done by matching the original template to this site in the original image.

$$\begin{array}{ccccc} & \vdots & \vdots & \vdots & \\ \cdots & 602 & 530 & 450 & \cdots \\ \cdots & 512 & \mathbf{91} & 662 & \cdots \\ \cdots & 422 & 555 & 752 & \cdots \\ & \vdots & \vdots & \vdots & \end{array}$$

The smallest squared distance for the template match, 91, was obtained when the left upper corner of the template was at position $(2, 0)$ in the image.

2.

Quadtrees are represented in Sec. 3.3.2, pp. 51–52. Two sample images A and B, their intersection, and the corresponding quadtrees are depicted in Fig. 1.
The algorithm starts from the root node by processing its child nodes from left to right and by creating nodes in the intersection tree according to the following cases

1. Create a black leaf node if both counterpoint nodes are black leaf nodes.

2. Create a white leaf node if either or both counterpoint nodes are white leaf nodes.

3. Create a parent node if counterpoint nodes are a parent node, denoted by n, and a black leaf node. The child nodes of the parent node n are copied to the intersection tree.

4. If both counterpoint nodes are parent nodes then create a parent node and process iteratively the child nodes. If all child nodes result in white leaf nodes then replace the parent node with a white leaf node.

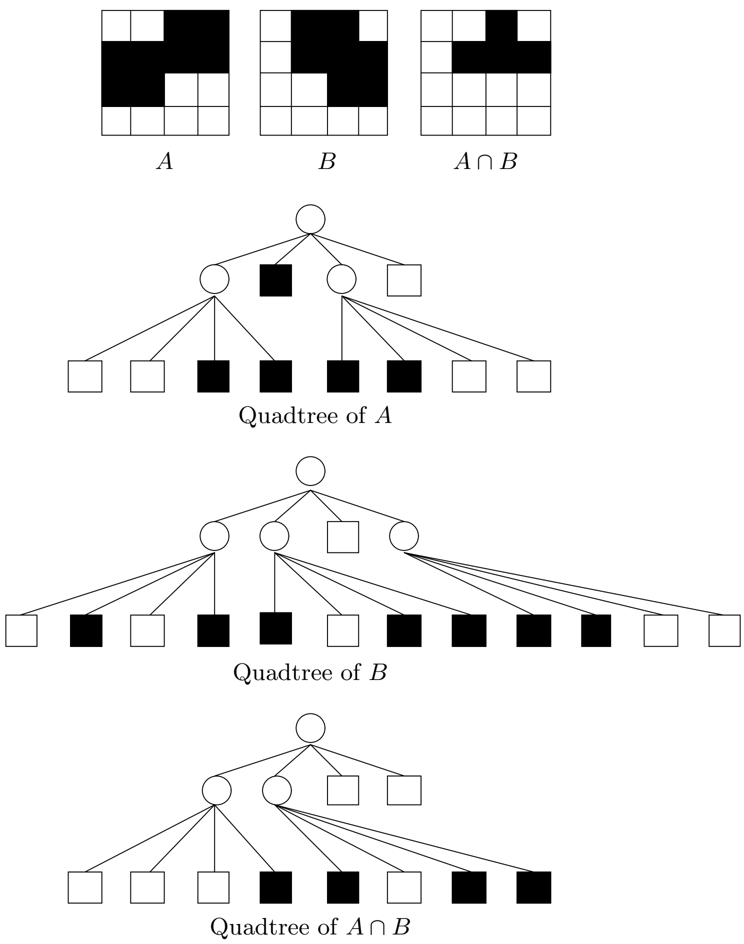


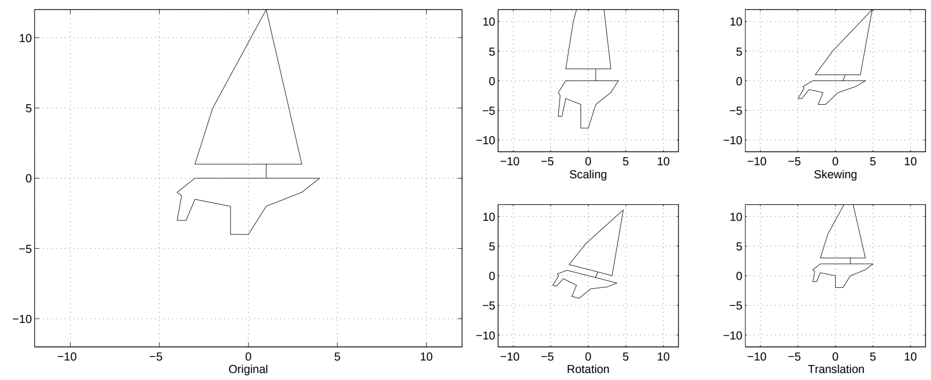
Figure 1: Two sample images, their intersection, and the quadtrees.

3.

a) Affine transformations (Sec. 4.2.1 p. 64) can be expressed in homogeneous coordinates by matrix product $\boldsymbol{p'} = \boldsymbol{Ap}$, where

$$\boldsymbol{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \ \boldsymbol{p'} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}, \text{and } \boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

The matrix $\boldsymbol{A}$ is a combination of scaling $\boldsymbol{A_{sc}}$, skewing $\boldsymbol{A_{sw}}$, rotation $\boldsymbol{A_{rt}}$, and translation $\boldsymbol{A_{tr}}$, $\boldsymbol{A} = \boldsymbol{A_{sc}A_{sw}A_{rt}A_{tr}}$.



$$\boldsymbol{A_{sc}} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ \boldsymbol{A_{sw}} = \begin{bmatrix} 1 & \tan\phi & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\boldsymbol{A_{rt}} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \text{and } \boldsymbol{A_{tr}} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{3}$$

The order of different transformations is significant. Rotation followed by translation does not necessarily give the same result as translation followed by rotation.
Control points $\boldsymbol{p_1}, \boldsymbol{p_2}, \cdots, \boldsymbol{p_N}$ and their counterpoints $\boldsymbol{p'_1}, \boldsymbol{p'_2}, \cdots, \boldsymbol{p'_N}$ are selected to determine $\boldsymbol{A}$.

$$\boldsymbol{P'_{3\times N}} = \boldsymbol{A_{3\times 3}P_{3\times N}},$$

$$\boldsymbol{P} = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \\ y_1 & y_2 & \cdots & y_N \\ 1 & 1 & \cdots & 1 \end{bmatrix}, \ \boldsymbol{P'} = \begin{bmatrix} x'_1 & x'_2 & \cdots & x'_N \\ y'_1 & y'_2 & \cdots & y'_N \\ 1 & 1 & \cdots & 1 \end{bmatrix}. \tag{4}$$

Six elements in matrix $\boldsymbol{A}$ are unknown. Therefore, three control points must be selected to solve for $\boldsymbol{A} = \boldsymbol{P'P^{-1}}$. For instance, the following points may be chosen

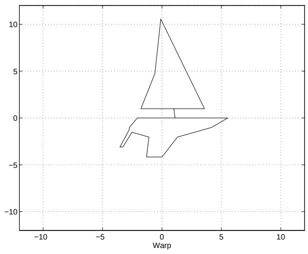
```
P  = [ -58 38 12; -4   10 72; 1 1 1];
Pc = [ -40 40 40; -40 -40 40; 1 1 1];
A = Pc * inv(P)

A =

    0.7853    0.3293    6.8651
   -0.1773    1.2160  -45.4212
   -0.0000    0.0000    1.0000
```

b) In the previous exercise, affine transforms were used for representation of geometric distortion functions. In this exercise, the functions are approximated by Nth order 2-D polynomials, yielding the so-called *polynomial warp model.* Notice that no assumptions about the viewing geometry or about the character of geometric distortion are necessary. The degree of the 2-D polynomial is often selected empirically, or it is restricted by the number of available control points. The polynomial warp relationship is determined by

$$x' = \sum_{i=0}^{N} \sum_{j=0}^{N} \hat{k}^1_{ij} x^i y^j \tag{5}$$

$$y' = \sum_{i=0}^{N} \sum_{j=0}^{N} \hat{k}^2_{ij} x^i y^j. \tag{6}$$

An example of polynomial warp:



$$\begin{aligned} x' &= x + \tfrac{1}{10}x^2 - \tfrac{1}{1000}x^3 - \tfrac{1}{10}xy \\ y' &= y - \tfrac{1}{100}y^2 \end{aligned}$$

In our case, N is 2. The 18 coefficients $\hat{k}^{1,2}_{ij}$ of the 2-D polynomial are estimated with two estimation equations from each control point. Therefore, at least nine control points are required. The transformation is given by $\boldsymbol{p'} = \boldsymbol{K}\boldsymbol{w}$, where

$$\boldsymbol{p'} = [x' \; y']^T,$$

$$\boldsymbol{K} = \begin{bmatrix} \hat{k}^1_{00} & \hat{k}^1_{10} & \hat{k}^1_{01} & \cdots & \hat{k}^1_{22} \\ \hat{k}^2_{00} & \hat{k}^2_{10} & \hat{k}^2_{01} & \cdots & \hat{k}^2_{22} \end{bmatrix}, \text{ and}$$

$$\boldsymbol{w} = [1 \; x \; y \; xy \; x^2 \; y^2 \; x^2y \; xy^2 \; x^2y^2]^T. \tag{7}$$

Matrix $\boldsymbol{K_{2\times9}} = \boldsymbol{P'_{2\times9}}\boldsymbol{W^{-1}_{9\times9}}$, where

$$\boldsymbol{P'} = \begin{bmatrix} x'_1 & \cdots & x'_N \\ y'_1 & \cdots & y'_N \end{bmatrix} \text{ and } \boldsymbol{W} = \begin{bmatrix} 1 & \cdots & 1 \\ x_1 & \cdots & x_N \\ \vdots & & \vdots \\ x_1^2y_1^2 & \cdots & x_N^2y_N^2 \end{bmatrix}.$$

The following points may be chosen, for instance

```
Px  = [ -58 -70 -82 -24 -35  7  38 25 12]';
Py  = [  -4  27  58  34  65 38  10 40 72]';
Pc = [ -40 -40 -40   0   0 24  40 40 40; ...
       -40   0  40   0  40  0 -40  0 40];
one = [1 1 1 1 1 1 1 1 1]';
W =[one Px Py Px.*Py Px.*Px Py.*Py Px.*Px.*Py ...
    Px.*Py.*Py Px.*Px.*Py.*Py]';
K = Pc * inv(W)
```

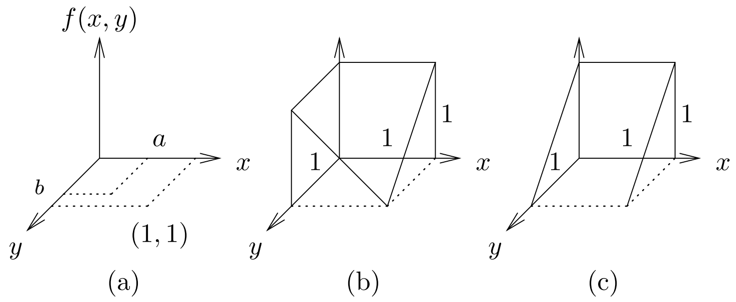


Figure 2: Bilinear interpolation

```
K =

  Columns 1 through 6

   57.5536     0.2388    -1.8436     0.0287    -0.0243     0.0212
  -44.5636    -0.2115     1.1901     0.0033    -0.0008     0.0002

  Columns 7 through 9

  0.0012      -0.0003    -0.0000
  0.0001      -0.0000    -0.0000
```

4.

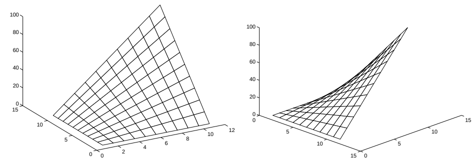
a) Bilinear interpolation is represented as linear interpolation in Sec. 4.2.2, p. 67. Assume that values of function f are known at points $(0,0)$, $(0,1)$, $(1,0)$, and $(1,1)$, Fig. 2. Bilinear interpolation yields the values of $f(a,b)$, $a \in [0,1]$ and $b \in [0,1]$

$$\begin{aligned} f(a,b) = \quad & (1-a)(1-b)f(0,0) \\ & +a(1-b)f(1,0) \\ & +(1-a)bf(0,1) \\ & +abf(1,1) \end{aligned} \tag{8}$$

The Eq. 8 is a plane equation if it has the form $f(a,b) = f_0 + f_1 a + f_2 b$, i.e., the equation

$$f(a,b) = f(0,0) + [f(1,0) - f(0,0)]a + [f(0,1) - f(0,0)]b$$
$$+[f(0,0) - f(1,0) - f(0,1) + f(1,1)]ab \tag{9}$$

is a plane equation only if the factor of ab is equal to zero. For instance, if $f(0,0) = f(1,0) = f(0,1) = 1$ and $f(1,1) = 0$, then $f(a,b) = 1 - ab$, which is not a plane, Fig. 2 (b). On the other hand, if $f(0,0) = f(1,0) = 1$ and $f(0,1) = f(1,1) = 0$, then $f(a,b) = 1 - b$, which is a plane, Fig. 2 (c).



Bilinearity: linear performance if either variable is fixed!

b) The Eq. 9 is a plane equation if

$$f(0,0) + f(1,1) = f(1,0) + f(0,1), \tag{10}$$

i.e., when the control points $f(0,0)$, $f(1,1)$, $f(1,0)$, and $f(0,1)$ are in the same plane.