Basic Algorithms for Digital Image Analysis

Dmitrij Csetverikov   Collaborators

Eötvös Loránd University, Budapest, Hungary
csetverikov@sztaki.hu
http://visual.ipan.sztaki.hu/mitya/

Faculty of Informatics

Lecture 4 – Matching

Lecture 4 – Matching

Lecture 4 – Matching

Lecture 4 – Matching

Lecture 4 – Matching

Tasks of computer vision related to matching 1/3

- Given images of a scene taken by **different sensors**, bring them into registration
  - this is multimodal **image registration**
  - in medical imaging, images obtained by sensors of different types are called *modalities*
  - interfaces that use different sensors, such as images, video, sound, haptics (tactile), are also called *multimodal*
  - when data structures to be registered are different, the term **data fusion** is used
- Given images of a scene taken at **different times**, find correspondences, displacements, or changes
  - this is **motion analysis**
  - typical example: motion tracking

Tasks of computer vision related to matching 1/3

- Given images of a scene taken by **different sensors**, bring them into registration
  - this is multimodal **image registration**
  - in medical imaging, images obtained by sensors of different types are called *modalities*
  - interfaces that use different sensors, such as images, video, sound, haptics (tactile), are also called *multimodal*
  - when data structures to be registered are different, the term **data fusion** is used
- Given images of a scene taken at **different times**, find correspondences, displacements, or changes
  - this is **motion analysis**
  - typical example: motion tracking

Tasks of computer vision related to matching 2/3

- Given images of a scene taken from **different positions**, find correspondent points to obtain 3D information about the scene
    - this is stereopsis, or simply **stereo**
    - matching provides *disparity*: shift of point between two views
    - by triangulation, disparity and baseline (distance between cameras) provide *depth*: 3D distance to point
    - generalised stereo is called *3D scene reconstruction* from multiple views

Tasks of computer vision related to matching 3/3

- Find places in image or on contour where it **matches a given pattern**
  - *template matching*: pattern is specified by template
  - *feature detection*: feature is specified by description
- Match **two contours** for object recognition, measurement, or positioning
  - this is *contour matching*
- Only the above two tasks are considered in this course

Tasks of computer vision related to matching 3/3

- Find places in image or on contour where it **matches a given pattern**
  - *template matching*: pattern is specified by template
  - *feature detection*: feature is specified by description
- Match **two contours** for object recognition, measurement, or positioning
  - this is *contour matching*
- Only the above two tasks are considered in this course

Tasks of computer vision related to matching 3/3

- Find places in image or on contour where it **matches a given pattern**
  - *template matching*: pattern is specified by template
  - *feature detection*: feature is specified by description
- Match **two contours** for object recognition, measurement, or positioning
  - this is *contour matching*
- Only the above two tasks are considered in this course

Critical issues of matching

- Widespread opinion in computer vision: Solving the correspondence problem is of key importance
  - opens way to solution of many other problems
  - however, hard to tackle because of numerous critical issues
- Invariance under imaging transformations
  - spatial (3D)
  - photometric (illumination, intensity)
- Sensitivity to noise, distortions, occlusion

Critical issues of matching

- Widespread opinion in computer vision: Solving the correspondence problem is of key importance
  - opens way to solution of many other problems
  - however, hard to tackle because of numerous critical issues
- Invariance under imaging transformations
  - spatial (3D)
  - photometric (illumination, intensity)
- Sensitivity to noise, distortions, occlusion

Critical issues of matching

- Widespread opinion in computer vision: Solving the correspondence problem is of key importance
  - opens way to solution of many other problems
  - however, hard to tackle because of numerous critical issues
- Invariance under imaging transformations
  - spatial (3D)
  - photometric (illumination, intensity)
- Sensitivity to noise, distortions, occlusion

Transformations considered in this course

- **Spatial**
  - 2D shift and rotation in image plane
- **Photometric**
  - intensity shift and scaling

  $\Rightarrow$ $I' = aI + b$
- Meaning of intensity shift and scaling
  - a: change of *direct illumination*

  $\Rightarrow$ illumination directed at object
  - b: change of *ambient light*

  $\Rightarrow$ overall illumination of scene

  $\Rightarrow$ lights coming from all directions

Template matching

- Compare subimage (**template**) $w(x', y')$ with an image $f(x', y')$ for all possible displacements (x, y)
  - in other words, **match** $w(x', y')$ against $f(x + x', y + y')$ for all (x, y)
- Template matching: Varying r and c, search for locations of
  - *low dissimilarity* (mismatch) between image and template, *or*
  - *high similarity* (match) between image and template

Template matching

- Compare subimage (**template**) $w(x', y')$ with an image $f(x', y')$ for all possible displacements (x, y)
  - in other words, **match** $w(x', y')$ against $f(x + x', y + y')$ for all (x, y)
- Template matching: Varying r and c, search for locations of
  - *low dissimilarity* (mismatch) between image and template, *or*
  - *high similarity* (match) between image and template

Outline

Sum of Square Differences (SSD)

$$SSD(x,y) = \sum \left\{ f(x+x', y+y') - w(x',y') \right\}^2$$

where for simplicity

$$\sum \quad \text{denotes} \quad \sum_{\substack{(x',y')\in W \\ (x+x',y+y')\in F}}$$

- Here
  - W is set of pixel positions in template w (template coord.)
  - F is set of pixel positions in image f (image coord.)
- $SSD(x,y)$ is *not* invariant under
  - 2D rotation $\Rightarrow$ cannot find significantly rotated pattern
  - intensity changes $\Rightarrow$ can't cope with varying illumination

Sum of Square Differences (SSD)

$$SSD(x,y) = \sum \left\{ f(x+x',y+y') - w(x',y') \right\}^2$$

where for simplicity

$$\sum \quad \text{denotes} \quad \sum_{\substack{(x',y')\in W \\ (x+x',y+y')\in F}}$$

- Here
  - W is set of pixel positions in template w (template coord.)
  - F is set of pixel positions in image f (image coord.)
- $SSD(x,y)$ is *not* invariant under
  - 2D rotation $\Rightarrow$ cannot find significantly rotated pattern
  - intensity changes $\Rightarrow$ can't cope with varying illumination

Intensity shift-corrected SSD

$$SSD_{SC}(x,y) = \sum \left\{ \left[f(x+x',y+y') - \bar{f}(x,y) \right] - \left[w(x',y') - \overline{w} \right] \right\}^2$$

- $\bar{f}(x,y)$ is average value of image in region covered by template
  - computed in each position (x,y)
  - $\Rightarrow$ use running box filter
- $\overline{w}$ is average value of template
  - computed only once
- $SSD_{SC}(x,y)$ is used to compensate for *intensity shift* due to varying illumination
  - handles changes in average level of signal
  - does not handle changes in amplitude of signal

Intensity shift-corrected SSD

$$SSD_{SC}(x,y) = \sum \left\{ \left[f(x+x', y+y') - \bar{f}(x,y) \right] - \left[w(x', y') - \overline{w} \right] \right\}^2$$

- $\bar{f}(x,y)$ is average value of image in region covered by template
  - computed in each position (x,y)
  - $\Rightarrow$ use running box filter
- $\overline{w}$ is average value of template
  - computed only once
- $SSD_{SC}(x,y)$ is used to compensate for *intensity shift* due to varying illumination
  - handles changes in average level of signal
  - does not handle changes in amplitude of signal

Outline

Unnormalised cross-correlation (CC)

$$CC(x,y) = \sum f(x+x',y+y') \cdot w(x',y')$$

- Properties of cross-correlation and convolution already studied
- $CC(x,y)$ is formally the same as *filtering* image f with mask w
  - $\Rightarrow$ our knowledge of filters is applicable: normalisation, separability, fast implementation
- $CC(x,y)$ is *not* invariant under intensity shift and scaling
- When $w > 0$ and f is large, $CC(x,y)$ is large, independently from similarity between w and f
  - $\Rightarrow$ to compensate for this, *normalised* version is used

Unnormalised cross-correlation (CC)

$$CC(x,y) = \sum f(x+x', y+y') \cdot w(x', y')$$

- Properties of cross-correlation and convolution already studied
- $CC(x,y)$ is formally the same as *filtering* image f with mask w
  - $\Rightarrow$ our knowledge of filters is applicable: normalisation, separability, fast implementation
- $CC(x,y)$ is *not* invariant under intensity shift and scaling
- When $w > 0$ and f is large, $CC(x,y)$ is large, independently from similarity between w and f
  - $\Rightarrow$ to compensate for this, *normalised* version is used

Unnormalised cross-correlation (CC)

$$CC(x,y) = \sum f(x+x', y+y') \cdot w(x', y')$$

- Properties of cross-correlation and convolution already studied
- $CC(x,y)$ is formally the same as *filtering* image f with mask w
  - $\Rightarrow$ our knowledge of filters is applicable: normalisation, separability, fast implementation
- $CC(x,y)$ is *not* invariant under intensity shift and scaling
- When $w > 0$ and f is large, $CC(x,y)$ is large, independently from similarity between w and f
  - $\Rightarrow$ to compensate for this, *normalised* version is used

Normalised cross-correlation (NCC)

$$NCC(x,y) = \frac{1}{N_1} \sum \left[f(x+x', y+y') - \bar{f}(x,y) \right] \cdot \left[w(x',y') - \overline{w} \right],$$

where normaliser

$$N_1 = \sqrt{S_f(x,y) \cdot S_w}$$

$$S_f(x,y) = \sum \left[f(x+x', y+y') - \bar{f}(x,y) \right]^2$$

$$S_w = \sum_{(x',y') \in W} \left[w(x',y') - \overline{w} \right]^2$$

- $S_f(x,y)$ is computed in each position (x,y), S_W only once
- $NCC(x,y)$ is invariant to any *linear intensity transformation* $g(x,y) = \alpha f(x,y) + \beta$

Normalised cross-correlation (NCC)

$$NCC(x,y) = \frac{1}{N_1}\sum\left[f(x+x',y+y') - \bar{f}(x,y)\right]\cdot\left[w(x',y') - \overline{w}\right],$$

where normaliser

$$N_1 = \sqrt{S_f(x,y)\cdot S_w}$$

$$S_f(x,y) = \sum\left[f(x+x',y+y') - \bar{f}(x,y)\right]^2$$

$$S_w = \sum_{(x',y')\in W}\left[w(x',y') - \overline{w}\right]^2$$

- $S_f(x,y)$ is computed in each position (x,y), S_W only once
- $NCC(x,y)$ is invariant to any *linear intensity transformation* $g(x,y) = \alpha f(x,y) + \beta$

Modified normalised cross-correlation (MNCC)

$$MNCC(x,y) = \frac{1}{N_2} \sum \left[f(x+x', y+y') - \bar{f}(x,y) \right] \cdot \left[w(x',y') - \overline{w} \right],$$

where normaliser

$$N_2 = S_f(x,y) + S_w$$

- MNCC differs from NCC only in *normalisation*
- MNCC is used to avoid numerically unstable division by small number when $S_f(x,y)$ is small
  - small image variation
- Formally, MNCC is only shift-corrected
  - in practice, insensitive to scaling: $S_f(x,y) + S_w \propto S_f(x,y)$, approximately

Modified normalised cross-correlation (MNCC)

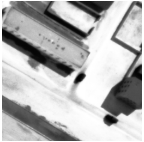
$$MNCC(x,y) = \frac{1}{N_2} \sum \left[f(x+x', y+y') - \bar{f}(x,y) \right] \cdot \left[w(x',y') - \overline{w} \right],$$

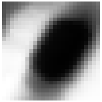
where normaliser

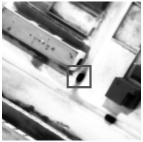
$$N_2 = S_f(x,y) + S_w$$

- MNCC differs from NCC only in *normalisation*
- MNCC is used to avoid numerically unstable division by small number when $S_f(x,y)$ is small
  - small image variation
- Formally, MNCC is only shift-corrected
  - in practice, insensitive to scaling: $S_f(x,y) + S_w \propto S_f(x,y)$, approximately

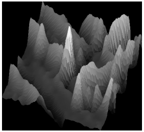
Examples of matching a stereo pair


left image

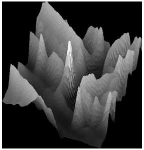

template, zoomed


right image


NCC image


NCC surface


SDD image


SDD surface

- Pattern from right image is searched in left image
  - NCC is normalised cross-correlation
  - SSD is sum of square differences

Numerical example of matching

template

0	0	0
1	1	1
0	0	0

input image

0	0	0
1	1	1
0	0	0

output of CC

1	2	3	2	1

output of NCC

1.0	1.4	1.7	1.4	1.0

1	1	1
1	1	1
1	1	1

output of CC

1	2	3	2	1
1	2	3	2	1
1	2	3	2	1

output of NCC

1.0	1.2	1.0
	1.0	
1.0	1.2	1.0

- (N)CC is (normalised) cross-correlation
  - input image is surrounded by 0's
  - in output, values below 1 are set to 0 and *not shown*
- Perfect match close to near misses in position and shape

  $\Rightarrow$ match is **not sharp**

Interior matching versus outline matching 1/2

template

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

input image

1	1	1
1	1	1
1	1	1

output of NCC

		1.2		
	1.3	2.0	1.3	
1.2	2.0	3.0	2.0	1.2
	1.3	2.0	1.3	
		1.2		

0	0	0	0	0
0	1	1	1	0
0	1	0	1	0
0	1	1	1	0
0	0	0	0	0

1	1	1
1	0	1
1	1	1

		1.3		
		1.4		
1.3	1.4	2.8	1.4	1.3
		1.4		
		1.3		

- Matching of outlines yields sharper matches
  - interior matching: ratio perfect match/near miss is 1.5
  - outline matching: 2

Interior matching versus outline matching 1/2

template

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

input image

1	1	1
1	1	1
1	1	1

output of NCC

		1.2		
	1.3	2.0	1.3	
1.2	2.0	3.0	2.0	1.2
	1.3	2.0	1.3	
		1.2		

0	0	0	0	0
0	1	1	1	0
0	1	0	1	0
0	1	1	1	0
0	0	0	0	0

1	1	1
1	0	1
1	1	1

		1.3		
		1.4		
1.3	1.4	2.8	1.4	1.3
		1.4		
		1.3		

- Matching of outlines yields sharper matches
  - interior matching: ratio perfect match/near miss is 1.5
  - outline matching: 2

Interior matching versus outline matching 1/2

template

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

input image

1	1	1
1	1	1
1	1	1

output of NCC

		1.2		
	1.3	2.0	1.3	
1.2	2.0	3.0	2.0	1.2
	1.3	2.0	1.3	
		1.2		

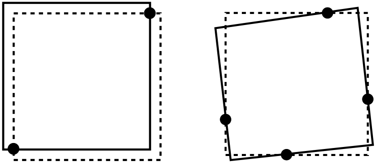
0	0	0	0	0
0	1	1	1	0
0	1	0	1	0
0	1	1	1	0
0	0	0	0	0

1	1	1
1	0	1
1	1	1

		1.3		
		1.4		
1.3	1.4	2.8	1.4	1.3
		1.4		
		1.3		

- Matching of outlines yields sharper matches
  - interior matching: ratio perfect match/near miss is 1.5
  - outline matching: 2

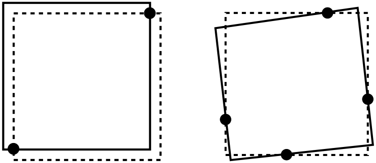
Interior matching versus outline matching 2/2



- *dashed rectangle: template*
- *solid polygon: object*
- *circles: overlapping contour points*

- For **ideal** object, small shift of template results in
  - drastic decrease of contour overlap
  - negligible descrease of area overlap
  - $\Rightarrow$ outline matching is sharper
- For **distorted** (or rotated) object,
  - outline overlap is small $\Rightarrow$ likely to miss object
  - area overlap is large $\Rightarrow$ likely to find object
  - $\Rightarrow$ outline matching is less robust

Interior matching versus outline matching 2/2



ideal object distorted object

- *dashed rectangle: template*
- *solid polygon: object*
- *circles: overlapping contour points*

- For **ideal** object, small shift of template results in
  - drastic decrease of contour overlap
  - negligible descrease of area overlap
  - $\Rightarrow$ outline matching is sharper
- For **distorted** (or rotated) object,
  - outline overlap is small $\Rightarrow$ likely to miss object
  - area overlap is large $\Rightarrow$ likely to find object
  - $\Rightarrow$ outline matching is less robust

Localisation accuracy versus robustness

- Contours matching
  - sharper matches: higher localisation accuracy
  - less robust: objects may be missed
  - faster
- Interior matching
  - less sharp matches
  - more robust
  - slower
- In general, one trades localisation accuracy for robustness
  - higher localisation accuracy $\Rightarrow$ less robust

Localisation accuracy versus robustness

- Contours matching
  - sharper matches: higher localisation accuracy
  - less robust: objects may be missed
  - faster
- Interior matching
  - less sharp matches
  - more robust
  - slower
- In general, one trades localisation accuracy for robustness
  - higher localisation accuracy $\Rightarrow$ less robust

Localisation accuracy versus robustness

- Contours matching
  - sharper matches: higher localisation accuracy
  - less robust: objects may be missed
  - faster
- Interior matching
  - less sharp matches
  - more robust
  - slower
- In general, one trades localisation accuracy for robustness
  - higher localisation accuracy $\Rightarrow$ less robust

Critical issues in template matching

- Invariance to **changes in size and rotation**
- Robustness to **pattern distortion**
  - for example, because of varying viewing angle
- Robustness to **'noisy' matches**
  - unexpected patterns that produce high matching values
- **Computational load**

Handling variations in object size and orientation

- Image normalisation
  - transform image to **standard size and orientation**
  - assumes no size/orientation variation within image
  - requires definition of orientation
- Adaptive solutions
  - spatially **scale and rotate** template in each position
  - select best matching scale and rotation
  - very slow if number of scales and rotations is large

  $\Rightarrow$ used only for small number of scales and rotations
- Invariant solutions
  - use scale and rotation **invariant description**
  - compare descriptions instead of comparing patterns

Handling variations in object size and orientation

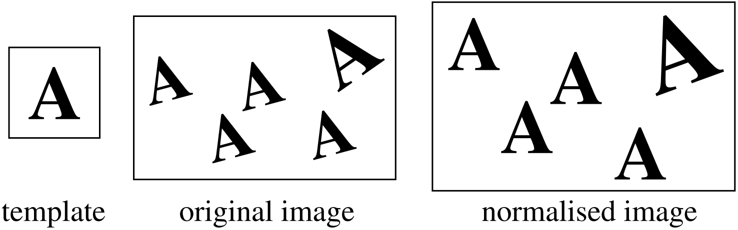
- Image normalisation
  - transform image to **standard size and orientation**
  - assumes no size/orientation variation within image
  - requires definition of orientation
- Adaptive solutions
  - spatially **scale and rotate** template in each position
  - select best matching scale and rotation
  - very slow if number of scales and rotations is large

  $\Rightarrow$ used only for small number of scales and rotations
- Invariant solutions
  - use scale and rotation **invariant description**
  - compare descriptions instead of comparing patterns

Handling variations in object size and orientation

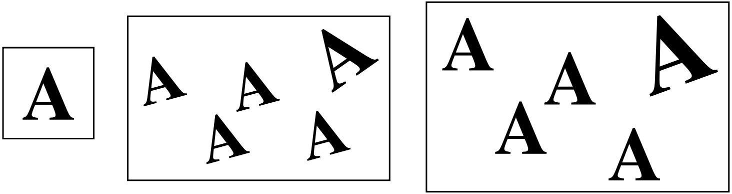
- Image normalisation
  - transform image to **standard size and orientation**
  - assumes no size/orientation variation within image
  - requires definition of orientation
- Adaptive solutions
  - spatially **scale and rotate** template in each position
  - select best matching scale and rotation
  - very slow if number of scales and rotations is large
  - $\Rightarrow$ used only for small number of scales and rotations
- Invariant solutions
  - use scale and rotation **invariant description**
  - compare descriptions instead of comparing patterns

Normalising image for size and orientation



- Letter **A** in top right corner differs in size and orientation
  - $\Rightarrow$ this letter *will not match*
- The other four letters will match
- How to define **image orientation**?

Normalising image for size and orientation



template original image normalised image

- Letter **A** in top right corner differs in size and orientation

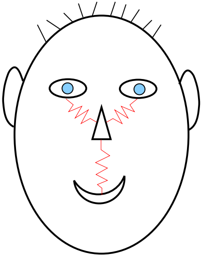
  $\Rightarrow$ this letter *will not match*

- The other four letters will match
- How to define **image orientation**?

Distortion-tolerant matching

- Use **flexible templates**
  - subtemplates connected by flexible links (‘springs’)
- Springs allow for moderate spatial variation of template
  - cost function introduced to penalise large variations

  $\Rightarrow$ larger variation means larger penalty
- Works when subtemplates are *characteristic enough* for reliable matching



Face template as set of flexibly connected subtemplates

Fast impementation of matching

- Work with **local features** of images and templates rather that patterns themselves
  - edges, contours
  - useful for sparse and reliable features
  - may be sensitive to distortion (recall outline matching)
- For large templates ($> 13 \times 13$ pixels), implement cross-correlation via **Fast Fourier Transform** (FFT)

$$f \otimes w = IFFT\left[FFT\left[f(x,y)\right]^* \cdot FFT\left[w(x,y)\right]\right]$$

  - $IFFT$ is inverse FFT, X^* is complex conjugate of X
  - FFT needs $O(N^2 \log N)$ operations for $N \times N$ image
  - direct implementation needs $O(N^4)$ operations

Fast impementation of matching

- Work with **local features** of images and templates rather that patterns themselves
  - edges, contours
  - useful for sparse and reliable features
  - may be sensitive to distortion (recall outline matching)
- For large templates ($> 13 \times 13$ pixels), implement cross-correlation via **Fast Fourier Transform** (FFT)

$$f \otimes w = IFFT\left[FFT\left[f(x,y)\right]^* \cdot FFT\left[w(x,y)\right]\right]$$

  - $IFFT$ is inverse FFT, X^* is complex conjugate of X
  - FFT needs $O(N^2 \log N)$ operations for $N \times N$ image
  - direct implementation needs $O(N^4)$ operations

Fast selection and rejection of candidates

1. Select match candidates, reject mismatches rapidly
2. Carefully test selected candidates only

- Use coarse **grid of template positions**, rectify candidates
  - *coarse-to-fine* sampling for cross-correlation
  - works if peaks of cross-correlation has no spikes
- Compute **simple properties** of template and image region
  - reject region if properties differ from template
- Use **subtemplates**
  - reject candidate region if a subtemplate does not match
- Set **threshold** on cumulative measure of mismatch
  - reject candidate when mismatch exceeds threshold

Fast selection and rejection of candidates

1. Select match candidates, reject mismatches rapidly
2. Carefully test selected candidates only

- Use coarse **grid of template positions**, rectify candidates
  - *coarse-to-fine* sampling for cross-correlation
  - works if peaks of cross-correlation has no spikes
- Compute **simple properties** of template and image region
  - reject region if properties differ from template
- Use **subtemplates**
  - reject candidate region if a subtemplate does not match
- Set **threshold** on cumulative measure of mismatch
  - reject candidate when mismatch exceeds threshold

Fast selection and rejection of candidates

1. Select match candidates, reject mismatches rapidly
2. Carefully test selected candidates only

- Use coarse **grid of template positions**, rectify candidates
  - *coarse-to-fine* sampling for cross-correlation
  - works if peaks of cross-correlation has no spikes
- Compute **simple properties** of template and image region
  - reject region if properties differ from template
- Use **subtemplates**
  - reject candidate region if a subtemplate does not match
- Set **threshold** on cumulative measure of mismatch
  - reject candidate when mismatch exceeds threshold

Fast selection and rejection of candidates

1. Select match candidates, reject mismatches rapidly
2. Carefully test selected candidates only

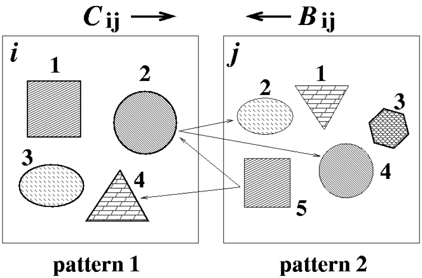
- Use coarse **grid of template positions**, rectify candidates
  - *coarse-to-fine* sampling for cross-correlation
  - works if peaks of cross-correlation has no spikes
- Compute **simple properties** of template and image region
  - reject region if properties differ from template
- Use **subtemplates**
  - reject candidate region if a subtemplate does not match
- Set **threshold** on cumulative measure of mismatch
  - reject candidate when mismatch exceeds threshold

Fast selection and rejection of candidates

1. Select match candidates, reject mismatches rapidly
2. Carefully test selected candidates only

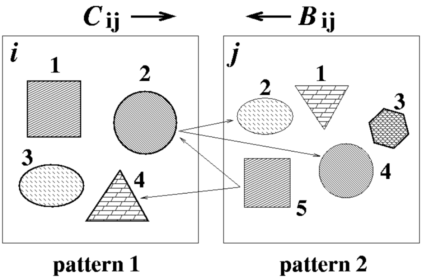
- Use coarse **grid of template positions**, rectify candidates
  - *coarse-to-fine* sampling for cross-correlation
  - works if peaks of cross-correlation has no spikes
- Compute **simple properties** of template and image region
  - reject region if properties differ from template
- Use **subtemplates**
  - reject candidate region if a subtemplate does not match
- Set **threshold** on cumulative measure of mismatch
  - reject candidate when mismatch exceeds threshold

Matching images segmented into regions 1/2



- Segment images into regions and find correspondences by comparing **region properties**
  - define *distance measure* between properties
  - handle regions having no pair (e.g., because of occlusion)
- Works when segmentation is reliable

Matching images segmented into regions 1/2



- Segment images into regions and find correspondences by comparing **region properties**
  - define *distance measure* between properties
  - handle regions having no pair (e.g., because of occlusion)
- Works when segmentation is reliable

Matching images segmented into regions 2/2

Algorithm: Stable Matching of Two Segmented Images

1. Compute **distance matrix** D_{ij} for all i, j:
   i^{th} region of image 1, j^{th} region of image 2
2. Calculate **forward matching matrix** C_{ij}:
   $C_{ij} = 1$ if $D_{ij} < D_{ik}$ for all $k \neq j$; otherwise, $C_{ij} = 0$
3. Calculate **backward matching matrix** B_{ij}:
   $B_{ij} = 1$ if $D_{ij} < D_{kj}$ for all $k \neq i$; otherwise, $B_{ij} = 0$
4. Match regions i and j if $C_{ij} B_{ij} = 1$
5. Remove established correspondences from D_{ij}
6. Iterate until no further matching is possible

Matching images segmented into regions 2/2

Algorithm: Stable Matching of Two Segmented Images

1. Compute **distance matrix** D_{ij} for all i, j:
   i^{th} region of image 1, j^{th} region of image 2
2. Calculate **forward matching matrix** C_{ij}:
   $C_{ij} = 1$ if $D_{ij} < D_{ik}$ for all $k \neq j$; otherwise, $C_{ij} = 0$
3. Calculate **backward matching matrix** B_{ij}:
   $B_{ij} = 1$ if $D_{ij} < D_{kj}$ for all $k \neq i$; otherwise, $B_{ij} = 0$
4. Match regions i and j if $C_{ij}B_{ij} = 1$
5. Remove established correspondences from D_{ij}
6. Iterate until no further matching is possible

Matching images segmented into regions 2/2

Algorithm: Stable Matching of Two Segmented Images

1. Compute **distance matrix** D_{ij} for all i, j:
   i^{th} region of image 1, j^{th} region of image 2
2. Calculate **forward matching matrix** C_{ij}:
   $C_{ij} = 1$ if $D_{ij} < D_{ik}$ for all $k \neq j$; otherwise, $C_{ij} = 0$
3. Calculate **backward matching matrix** B_{ij}:
   $B_{ij} = 1$ if $D_{ij} < D_{kj}$ for all $k \neq i$; otherwise, $B_{ij} = 0$
4. Match regions i and j if $C_{ij}B_{ij} = 1$
5. Remove established correspondences from D_{ij}
6. Iterate until no further matching is possible

Matching images segmented into regions 2/2

Algorithm: Stable Matching of Two Segmented Images

1. Compute **distance matrix** D_{ij} for all i, j: i^{th} region of image 1, j^{th} region of image 2
2. Calculate **forward matching matrix** C_{ij}: $C_{ij} = 1$ if $D_{ij} < D_{ik}$ for all $k \neq j$; otherwise, $C_{ij} = 0$
3. Calculate **backward matching matrix** B_{ij}: $B_{ij} = 1$ if $D_{ij} < D_{kj}$ for all $k \neq i$; otherwise, $B_{ij} = 0$
4. Match regions i and j if $C_{ij}B_{ij} = 1$
5. Remove established correspondences from D_{ij}
6. Iterate until no further matching is possible

Matching images segmented into regions 2/2

Algorithm: Stable Matching of Two Segmented Images

1. Compute **distance matrix** D_{ij} for all i, j:
   i^{th} region of image 1, j^{th} region of image 2
2. Calculate **forward matching matrix** C_{ij}:
   $C_{ij} = 1$ if $D_{ij} < D_{ik}$ for all $k \neq j$; otherwise, $C_{ij} = 0$
3. Calculate **backward matching matrix** B_{ij}:
   $B_{ij} = 1$ if $D_{ij} < D_{kj}$ for all $k \neq i$; otherwise, $B_{ij} = 0$
4. Match regions i and j if $C_{ij}B_{ij} = 1$
5. Remove established correspondences from D_{ij}
6. Iterate until no further matching is possible

Matching images segmented into regions 2/2

Algorithm: Stable Matching of Two Segmented Images

1. Compute **distance matrix** D_{ij} for all i, j:
   i^{th} region of image 1, j^{th} region of image 2
2. Calculate **forward matching matrix** C_{ij}:
   $C_{ij} = 1$ if $D_{ij} < D_{ik}$ for all $k \neq j$; otherwise, $C_{ij} = 0$
3. Calculate **backward matching matrix** B_{ij}:
   $B_{ij} = 1$ if $D_{ij} < D_{kj}$ for all $k \neq i$; otherwise, $B_{ij} = 0$
4. Match regions i and j if $C_{ij}B_{ij} = 1$
5. Remove established correspondences from D_{ij}
6. Iterate until no further matching is possible

Stable matching as consistency check

- Forward-backward matching is consistency check
  - standard way to discard invalid matches
  - simple solution to the Stable Marriage Problem

left image

right image

original ME

consistent ME

- Matching of stereo pair in presence of occlusion
  - ME is matching error: lighter pixel shows larger error
  - Consist. check removes wrong matches due to occlusion