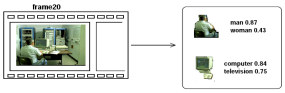
Shape

- Segmentation/tiling (based on color or illumination)

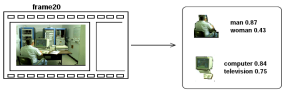


- A *segmentation* of image I is a set $\{R_1,\ldots,R_n\}$ of regions such that:
  - $R_1 \cup \ldots \cup R_n = I$
  - $R_i \cap R_j = \{\}$ for all $1 <= i,j <= n$
  - $H(R_i)$ = true for all $1 <= i <= n$
  - If R_i,R_j are adjacent, then $H(R_i \cup R_j)$ = false.

  H is an homogeneity predicate

Shape

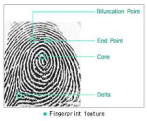
- Segmentation/tiling (based on color or illumination)



- Hough transform of edges
  - Global features
    - Roundness
    - Aspect ratio (ratio between the with of the image and the height of the image)
    - Major axis orientation
  - Local features
    - Size and orientation of consecutive boundary segments

Spatial queries

- ….fingerprint detection
  - spatial distribution
    - positions of features relative to each other
      - ..their angles
      - ..their distances from each other



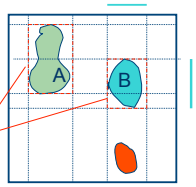
http://www.sphinxtech.com.sg/fingerfeatures.jpg

http://www.informatik.htw-dresden.de/~iwe/Belege/Poetzsch/

Spatial relationships

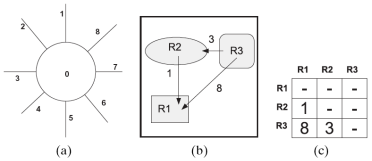
- How do we represent the spatial relationships among these objects?
  - to the left, to the right
    A is to the left of B
  - above of, below of
    A is above/below B?

Minimum Bounding Rectangles (MBR)



Maria Luisa Sapino (BDM 2018)

Nine-directional Lower Triangular Matrix (9DLT)



	R1	R2	R3
R1	-	-	-
R2	1	-	-
R3	8	3	-

(a) (b) (c)

Figure 2.35. (a) The nine directions between two regions (0 means "at the same place") [Chang, 1991]. (b) An image with three regions and their relationships (for convenience, the relationships are shown only in one direction). (c) The corresponding 9DLT matrix.

Maria Luisa Sapino (BDM 2018)

Nine-intersection Matrix

$$\begin{pmatrix} {o_1}^o \cap {o_2}^o & {o_1}^o \cap \delta o_2 & {o_1}^o \cap {o_2}^- \\ \delta o_1 \cap {o_2}^o & \delta o_1 \cap \delta o_2 & \delta o_1 \cap {o_2}^- \\ {o_1}^- \cap {o_2}^o & {o_1}^- \cap \delta o_2 & {o_1}^- \cap {o_2}^- \end{pmatrix}$$

Maria Luisa Sapino (BDM 2018)

Nine-intersection Matrix

$$\begin{pmatrix} {o_1}^o \cap {o_2}^o & {o_1}^o \cap \delta o_2 & {o_1}^o \cap {o_2}^- \\ \delta o_1 \cap {o_2}^o & \delta o_1 \cap \delta o_2 & \delta o_1 \cap {o_2}^- \\ {o_1}^- \cap {o_2}^o & {o_1}^- \cap \delta o_2 & {o_1}^- \cap {o_2}^- \end{pmatrix}$$

$$\begin{pmatrix} \geq 1 & \geq 1 & \geq 1 \\ 0 & \geq 1 & \geq 1 \\ 0 & 0 & \geq 1 \end{pmatrix}, \quad \begin{pmatrix} \geq 1 & \geq 1 & \geq 1 \\ \geq 1 & \geq 1 & \geq 1 \\ \geq 1 & \geq 1 & \geq 1 \end{pmatrix}$$

Spatial orientation graph

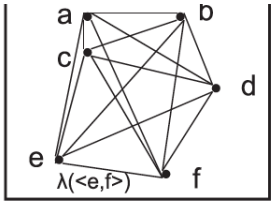


Figure 2.36. A sample spatial orientation graph.

Converting a region into a set of points

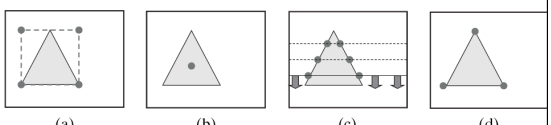
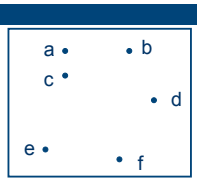
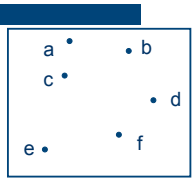


Figure 2.37. Converting a region into a set of points: (a) minimum bounding rectangle (b) centroid, (c) line sweep, and (d) corners.

Compare space = compare string



(e<a=c<f=b<d; a=b<c<d<e<f)



(e<c<a<f=b<d; a<b<c<d<f<e)

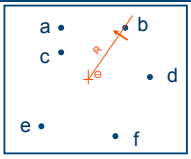
- Instead of comparing spatial composition directly, we can compare the corresponding 2Dstrings

How do we compare strings??

(e<a=c<f=b<d; a=b<c<d<e<f) vs. (e<c<a<f=b<d; a<b<c<d<f<e)

- Edit distance:
  - "table" vs. "cable": 1 (replace "t" with "c")
  - "table" vs. "bale": 2 (delete "t"; swap "a" and "b")

ΘR string

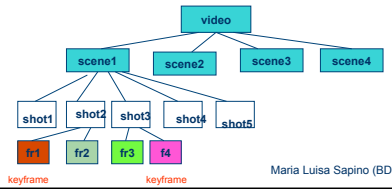


String representations of space

- 2D string
  - Use centroids (points)
  - 3 operators
- 2D Estring
  - Use intervals
  - Compare intervals
  - More operators
    - Contain, meet, begin, end, overlap, equal, less than, overlap, inverse

Video (temporal structure!!!!)

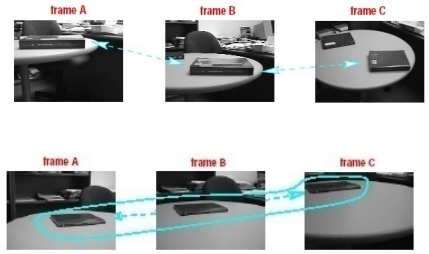
- Time (as in audio) + objects and motion
- Frame by frame representation is too costly (30f per second)
- Shot detection (video segmentation) gives a set of frames which are atomic



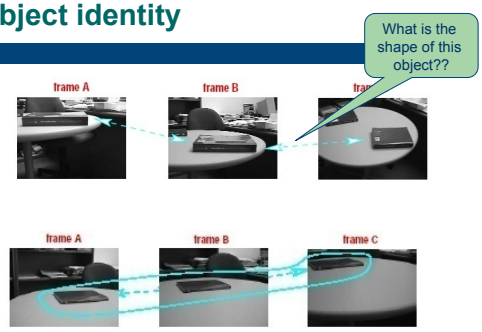
Motion

- Camera motion
  - Zooming (varying the focus distance)
  - Tilting (down/up - camera vertical rotation)
  - Panning (right/left - camera horizontal rotation)
  - Tracking (horizontal transverse movement)
  - Booming (vertical transverse movement)
  - Dollying (toward/away – horizontal lateral movement)
- Object motion
- Scene change
- Problem: if you don't know about camera motion ahead in time, then it is harder to distinguish between object motion and camera motion

Object tracking



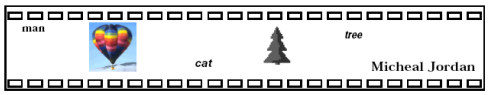
Object identity



Time

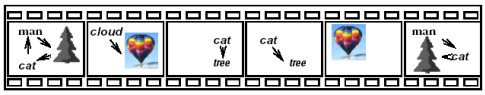
- What is time?
- How do we represent time?
- How do we represent actions and events?
- How do we ask queries about time?

Video queries



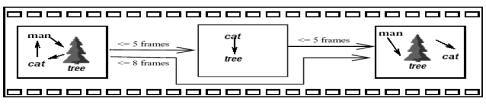
Object queries

Video queries



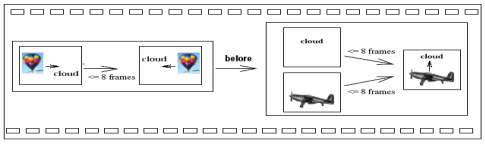
Frame queries

Video queries



Simple action queries

Video queries



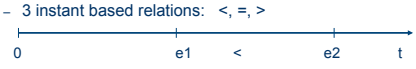
Composite action queries

Time Models

- Instants
  - 3 instant based relations: <, =, >

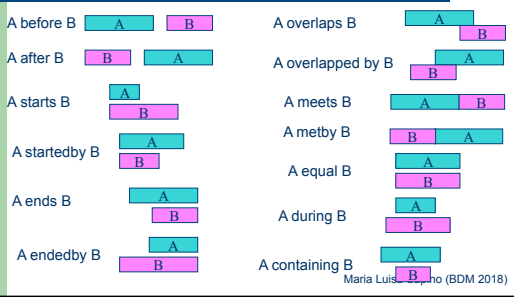
Time Models

- Instants
  - 3 instant based relations: <, =, >



- Intervals
  - 13 interval based relations

Intervals...



Temporal Logic; Axioms

- before(I1,I2) and before(I2,I3) -> before(I1,I3)
- meets(I1,I2) and during (I2,I3) ->
  overlaps(I1,I3) or during(I1,I3) or meets(I1,I3)

Definitions

- before(I1,I2) and before(I2,I3) -> before(I1,I3)
- meets(I1,I2) and during (I2,I3) ->
  overlaps(I1,I3) or during(I1,I3) or starts(I1,I3)
- in(I1,I2) <-> during(I1,I2) or starts(I1,I2) or finishes(I1,I2)

Rules

- before(I1,I2) and before(I2,I3) -> before(I1,I3)
- meets(I1,I2) and during (I2,I3) ->
  overlaps(I1,I3) or during(I1,I3) or starts(I1,I3)
- in(I1,I2) <-> during(I1,I2) or starts(I1,I2) or finishes(I1,I2)
- holds(p,I) <-> forall i (in(i,I) ->holds(p,i))
- holds(and(p,q),I) <-> holds(p,I) and holds(q,I)
- holds(not(p),I) <-> forall i (in(i,I) -> not holds (p,i))

Interval Algebra

- Predicates: 13 relationships between intervals
- Operators: and, or
- Result: all relationships that hold between every pair of intervals

Interval Algebra

- Predicates: 13 relationships between intervals
- Operators: and, or
- Result: all relationships that hold between every pair of intervals
- overlaps(a,b) and starts(a,b) and meets(b,c)
  - what else can you deduce from these facts?????

Minimal Labeling Problem

- NP-hard
- Special cases: if there are no "or"s, this can be solved in $O(n^3)$ time
  - constraint propagation algorithm

Point algebra

- Use "instants" instead of "intervals"
- Use "before", "equal", and "after" instead of the 13 interval relationships
- (similar to situation calculus)

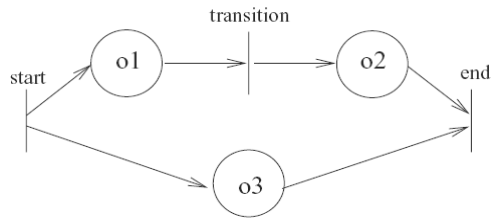
Instants vs intervals

- Each interval can be described by two instants st and et

Instants vs intervals

- Each interval can be described by two instants st and et
- If there are no disjunctions, interval algebra = instant algebra
- during(A,B) =
  st(A) > st(B) and et(A) < et(B) and
  st(A) < et(A) and st(B) <et(B)

Object Composition Petri Nets



How to compare OCPNs