INF 5300 Advanced Topic: Video Content Analysis

Tracking

Asbjørn Berge

Outline

- Introduction to the tracking problem
  - What is tracking?
  - Approaches & assumptions
  - Tracking applications
  - State of the art & challenges
- Tracking preliminaries
- Non-probabilistic methods
  - Mean shift tracking
  - Lucas-Kanade tracking
  - "Brute-force" - assignments

3

Outline

Introduction to the tracking problem

- What is tracking?
- Approaches & assumptions
- Tracking applications
- State of the art & challenges

Tracking preliminaries

- Object representations
- Tracking categories

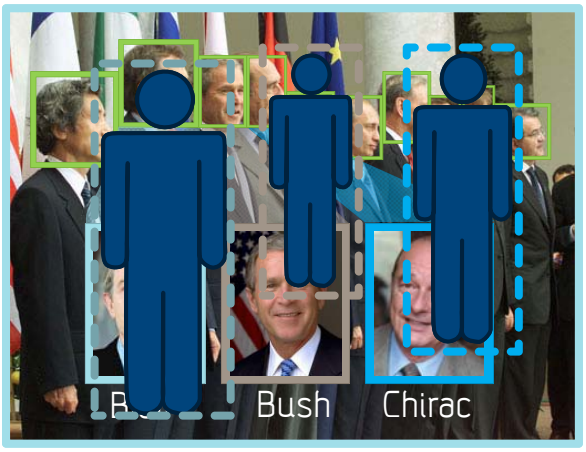
Non-probabilistic methods

- Mean shift tracking
- Lucas-Kanade tracking
- "Brute-force" - assignments

4

What is tracking?

- detection
- recognition
- tracking



5

What is tracking?

- Definition: using **image measurements** and a **predictive dynamic model** to consistently estimate the state(s) X_t of one or more object(s) over the discrete time steps corresponding to video frames.

SINTEF Technology for a better society

6

What is tracking?

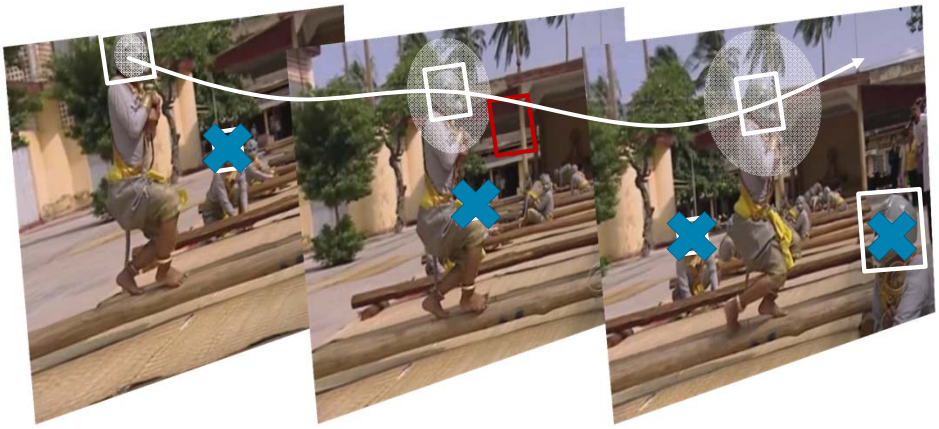
- Why not just do detection?
  - Estimate the state X at each time step
  - - inefficient
  - - data association problem

SINTEF Technology for a better society

7

What is tracking?

- It is better to do tracking
  - Maintain an estimate of X over time, predict the future location
  - + efficient, restricts search space
  - + smoothes noisy measurements
  - - requires knowledge about object behavior



8

Tracking assumptions

- Smooth camera
  - No instant transitions between viewpoints
  - Any camera pose/parameter changes are gradual
- Object motion can be modeled
  - Linear models
  - Non-linear mod
- Likelihood of object presence at a location in the image can be modeled
  - Typically uses local image information

Fundamentals

- Tracking task:
  - In the simplest form, tracking can be defined as the problem of estimating the trajectory of an object in the image plane as it moves around a scene. In other words, a tracker assigns consistent labels to the tracked objects in different frames of a video. Additionally, depending on the tracking domain, a tracker can also provide object-centric information, such as orientation, area, or shape of an object.
  - Two subtasks:
    - Build some model of what you want to track
    - Use what you know about where the object was in the previous frame(s) to make predictions about the current frame and restrict the search
- Repeat the two subtasks, possibly updating the model

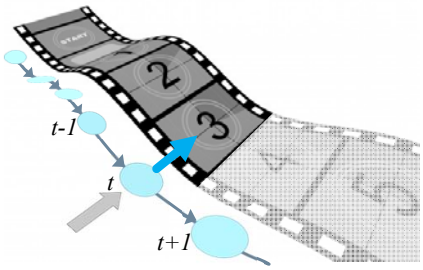
Fundamentals

- Tracking objects can be complex due to:
  - loss of information caused by projection of 3D world on 2D image
  - noise in images
  - complex object shapes / motion
  - nonrigid or articulated nature of objects
  - partial and full object occlusions
  - scene illumination changes
  - real-time processing requirements
- Simplify tracking by imposing constraints:
  - Almost all tracking algorithms assume that the object motion is smooth with no abrupt changes
  - The object motion is assumed to be of constant velocity
  - Prior knowledge about the number and the size of objects, or the object appearance and shape

11

Approaches to tracking

- Sequential
  - (recursive, online)
  - + Inexpensive ⟶ real-time
  - - no future information
  - - cannot revisit past errors



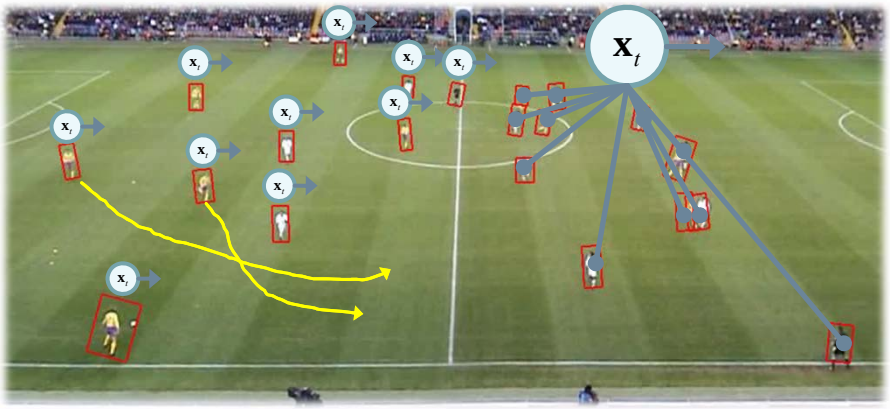
- Batch Processing
  - (offline)
  - - Expensive ⟶ not real-time*
  - + considers all information
  - + can correct past errors

$t=1,\ldots,T$

12

Approaches to tracking

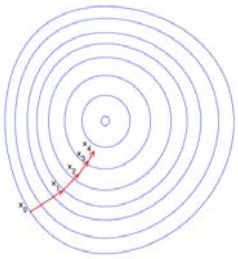
- Parallel trackers
  - several single-object trackers
  - computationally less expensive
  - how to handle interaction, cross-overs?
- Joint state
  - single multi-object representation
  - computationally expensive
  - principled interaction models

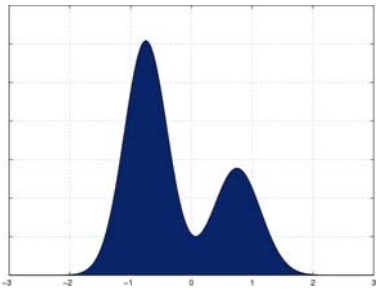


13

Approaches to tracking

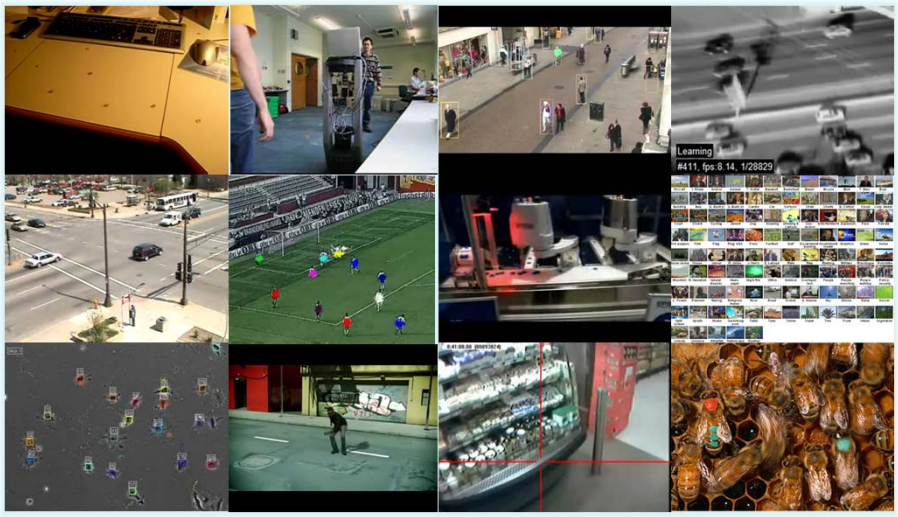
- Non-probabilistic
  - + quick convergence*
  - + efficient
  - - stuck in local minima
  - - does not model multiple objects
- Probabilistic
  - + flexible, principled
  - + multi-modal
  - - slower
  - - interpretation





14

Tracking applications



15

Tracking applications

- Tracking is an essential step in many computer vision based applications

Detection + Feature Extraction → Tracking → Activity Recognition + Event Recognition → Behavior Analysis + Social Models

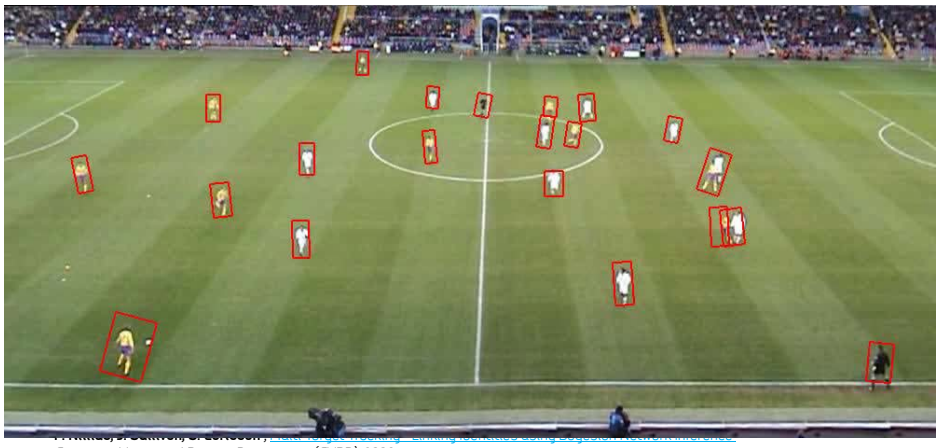
Prithwijit Guha, Amitabha Mukerjee and K. S. Venkatesh, Spatio-temporal Discovery: Appearance + Behavior = Agent Knowledge, Computer Vision, Graphics and Image Processing 4338 516-527, 2007

SINTEF Technology for a better society

16

Tracking Applications

- Sports



Computer Vision and Pattern Recognition (CVPR), 2006

SINTEF Technology for a better society

17

Tracking Applications

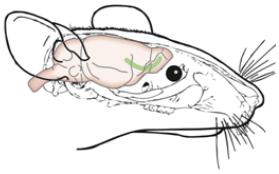
- Surveillance

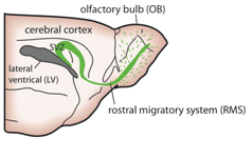
K. Smith, P. Quelhas, and D. Gatica-Perez, Detecting Abandoned Luggage Items in a Public Space, Performance Evaluation of Tracking and Surveillance (PETS) Workshop at CVPR, New York, NY, June 18 2006

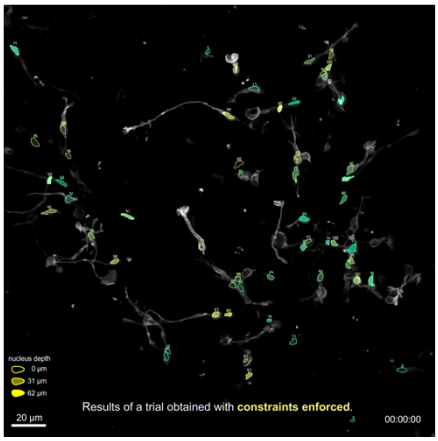
18

Tracking Applications

- Biomed & Microscopy





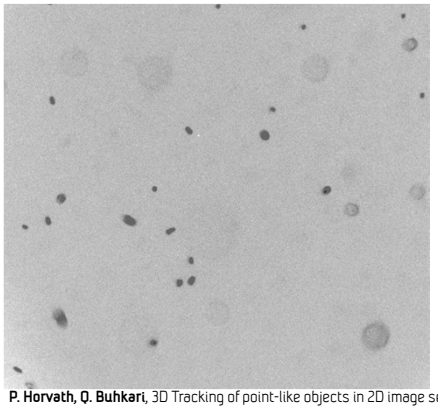


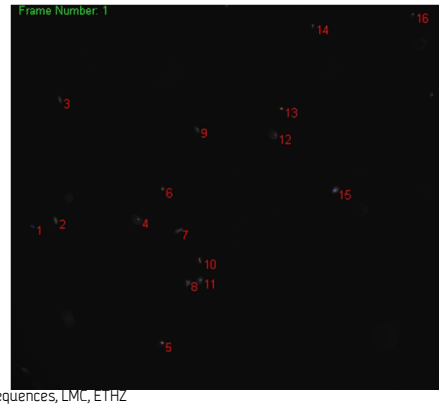
K. Smith, A. Carleton, and V. Lepetit, General Constraints for Batch Multiple-Target Tracking Applied to Large-Scale Videomicroscopy, Computer Vision and Pattern Recognition (CVPR), Anchorage, AK, June 2008

19

Tracking applications

- Biological Research
  - Goal: develop a method to "trap" Salmonella bacteria





P. Horvath, Q. Buhkari, 3D Tracking of point-like objects in 2D image sequences, LMC, ETHZ

20

What is the state of the art?

- Despite being classic computer vision problem, tracking is **largely unsolved**
  - Some limited successes
  - No general-purpose tracker
  - No standard data corpus for comparison
  - No standard evaluation methodology
  - Challenging problems remain

21

Tracking Challenges

- appearance change
- occlusion
- distraction
- illumination change
- difficult motion
- multiple objects
- scale change
- efficient solution

Ruei-Sung Lin, David Ross, Jongwoo Lim, Ming-Hsuan Yang, Adaptive discriminative generative model and its Applications Neural Information Processing Systems Conference (NIPS), 2004

22

Tracking Challenges

- appearance change
- occlusion
- distraction
- illumination change
- difficult motion
- multiple objects
- scale change
- efficient solution

Amit Adam, Ehud Rivlin and Ilan Shimshoni, Robust Fragments-based Tracking using the Integral Histogram (pdf). *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2006*

23

Tracking Challenges

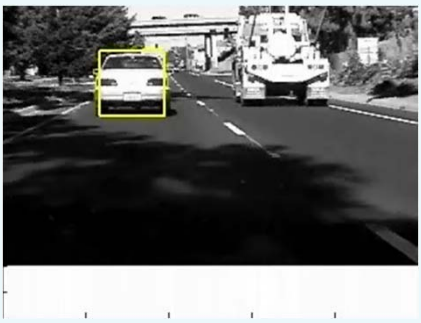
- appearance change
- occlusion
- distraction
- illumination change
- difficult motion
- multiple objects
- scale change
- efficient solution

Michael Isard and Andrew Blake CONDENSATION -- conditional density propagation for visual tracking International Journal of Computer Vision (IJCV), 29, 1, 5--28, (1998)

SINTEF Technology for a better society

24

Tracking Challenges



- appearance change
- occlusion
- distraction
- illumination change
- difficult motion
- multiple objects
- scale change
- efficient solution

Ruei-Sung Lin, David Ross, Jongwoo Lim, Ming-Hsuan Yang, Adaptive discriminative generative model and its Applications Neural Information Processing Systems Conference (NIPS), 2004

SINTEF Technology for a better society

25

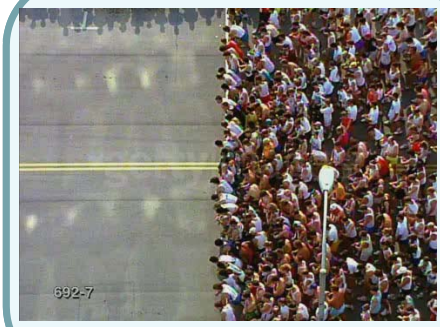
Tracking Challenges

- appearance change
- occlusion
- distraction
- illumination change
- difficult motion
- multiple objects
- scale change
- efficient solution

Michael Isard and Andrew Blake CONDENSATION -- conditional density propagation for visual tracking
International Journal of Computer Vision (IJCV), 29, 1, 5--28, (1998)

26

Tracking Challenges

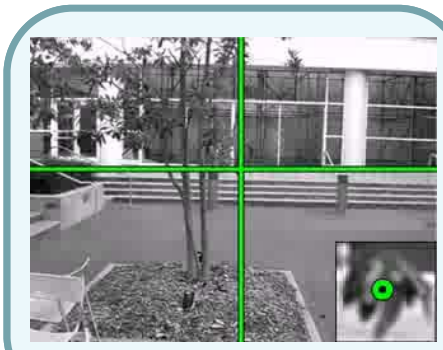


- appearance change
- occlusion
- distraction
- illumination change
- difficult motion
- multiple objects
- scale change
- efficient solution

Saad Ali and Mubarak Shah, Floor Fields for Tracking in High Density Crowd Scenes, European Conference on Computer Vision (ECCV), 2008.

27

Tracking Challenges

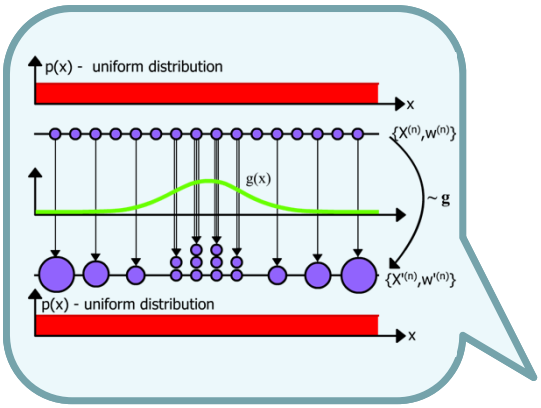


- appearance change
- occlusion
- distraction
- illumination change
- difficult motion
- multiple objects
- scale change
- efficient solution

Shawn Lankton, James Malcolm, Arie Nakhmani, and Allen Tannenbaum. Tracking Through Changes in Scale
Proceedings of International Conference on Image Processing (ICIP), 2008.

28

Tracking Challenges



- appearance change
- occlusion
- distraction
- illumination change
- difficult motion
- multiple objects
- scale change
- finding efficient solution

29

Outline

Introduction to the tracking problem
- What is tracking?
- Approaches & assumptions
- Tracking applications
- State of the art & challenges

Tracking preliminaries
- Object representations
- Tracking categories

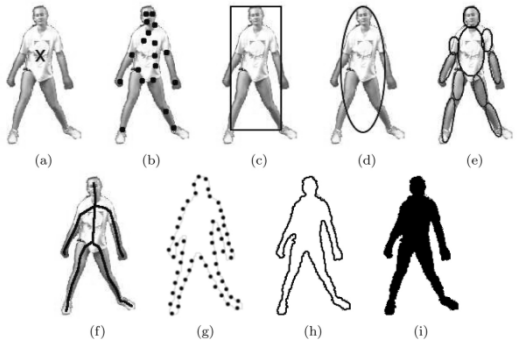
Non-probabilistic methods
- Mean shift tracking
- Lucas-Kanade tracking
- 'Brute-force' - assignments

30

Object Representation

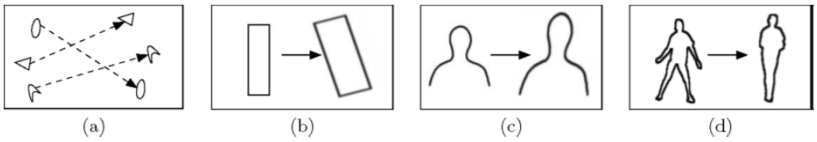
Object representation = Shape + Appearance

Shape representations:



- Object representations. (a) Centroid, (b) multiple points, (c) rectangular
- patch, (d) elliptical patch, (e) part-based multiple patches, (f) object skeleton, (g) control points on object contour, (h) complete object contour, (i) object silhouette

Object Tracking



- (a) Point Tracking. Objects detected in consecutive frames are represented by points, and a point matching is done. External mechanism *detect* the objects in every frame.
- (b) Kernel Tracking. Kernel = object shape and appearance. E.g. kernel = a rectangular template or an elliptical shape with an associated histogram. Objects are tracked by computing the motion (parametric transformation such as translation, rotation, and affine) of the kernel in consecutive frames.
- (c)+(d) Silhouette Tracking. Such methods use the information encoded inside the object region (appearance density and shape models). Given the object models, silhouettes are tracked by either shape matching (c) or contour evolution (d). The latter one can be considered as object segmentation applied in the temporal domain using the priors generated from the previous frames. (See lecture on active contours)

Object Tracking: Measuring difference of tracked objects

- Direct comparison: between template t(i,j) and candidate g(i,j)

$$d_1(t,g) = \sum_{i=1}^{m}\sum_{j=1}^{n}|g(i,j)-t(i,j)|$$

$$d_2(t,g) = \sum_{i=1}^{m}\sum_{j=1}^{n}(g(i,j)-t(i,j))^2$$

$$d_3(t,g) = \sum_{i=1}^{m}\sum_{j=1}^{n}g(i,j)\cdot t(i,j)$$

- Bhattacharyya coefficient between two distributions:

$$bd(H_1,H_2) = \sqrt{1-bc(H_1,H_2)}$$

$$bc(H_1,H_2) = \sum_{i=1}^{n}\sqrt{H_1(i)\cdot H_2(i)}$$

Outline

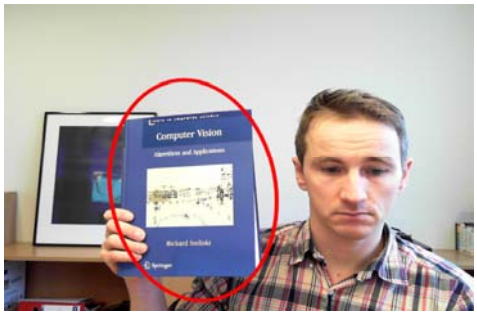
Introduction to the tracking problem
- What is tracking?
- Approaches & assumptions
- Tracking applications
- State of the art & challenges

Tracking preliminaries
- Object representations
- Tracking categories

Non-probabilistic methods
- Mean shift tracking
- Lucas-Kanade tracking
- "Brute-force" - assignments

Mean shift tracking

- The mean-shift algorithm is an efficient approach to tracking objects by appearance
- Ideally, we want an indicator function that retunrns *1* for pixels on the object we are tracking and *0* for all other pixels.
- Not possible - Instead we compute likelihood maps where the value at a pixel is proportional to the likelihood that the pixel comes from the object we are tracking.
- Not limited to only color:
  - edge orientations
  - Texture
  - motion
- Handles occlusions and camouflage poorly

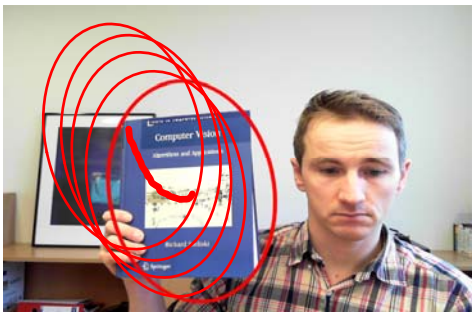


D. Comaniciu, V. Ramesh, and P. Meer, *Kernel-based object tracking*. IEEE Trans. Patt. Analy. Mach. Intell. 25, 564–575, 2003

Mean shift tracking

OpenCV's mean shift tracker implementation uses an algorithm called Camshift. Camshift consists of four steps:

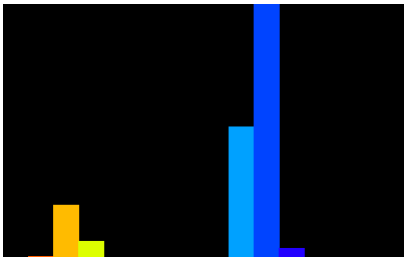
1. Create a color histogram to represent the object
2. Calculate a "object probability" for each pixel in the incoming video frames
3. Shift the location of the *kernel* in each video frame
4. Calculate the size and angle of the ellipse (adapt *kernel*)



Color histogram

Camshift represents the object it's tracking as a histogram of color values in the HSV color model.

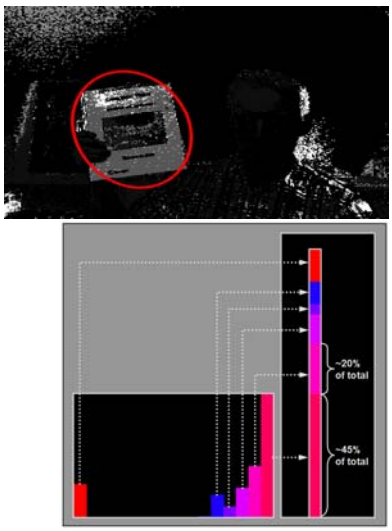
CV book



Face

Calculate object probability

- The histogram is in Camshift created only once, at the start of tracking.
- Afterwards, it's used to assign a " object probability" value to each image pixel in the video frames that follow.
- The probability that a pixel selected randomly from the initial region would fall into the rightmost bin is 45%, and so on.
- The hue value for each pixel is thus used to assign a estimated object probability to the pixel.



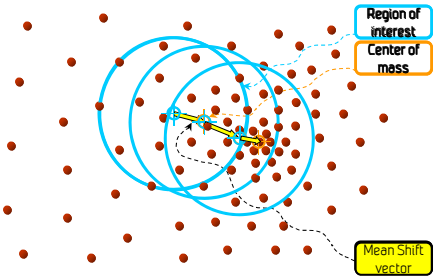
Shift to a new location

Let pixels form a uniform grid of data points, each with a weight (pixel value) proportional to the "likelihood" that the pixel is not on the object we want to track.

"Shifts" the location estimate centered over the area with the highest concentration of bright pixels in the object probability image.

Vector from previous location bycomputing the center of gravity of the probability values within a kernel.

Running mean-shift with kernel ***K*** on a weight image ***w*** is equivalent to performing gradient ascent in a (virtual) image formed by convolving ***w*** with some shadow kernel ***H***



$$\Delta x = \frac{\sum_a K(a-x)\, w(a)\, (a-x)}{\sum_a K(a-x)\, w(a)}$$

Camshift demo

- "Likelihood" based on color is noisy and is also quantized hard in the example implementation
- Two (three) parameters in Camshift for tweaking the noise sensitivity based on S and V in the HSV color space.
  - Vmin and Vmax : intensity thresholds for colors, low intensity colors are noisy in Hue, and high intensity colors are "too close" to white
  - Smin – saturation threshold. Colors that have low saturation could fit any Hue.

Math details without the formulae

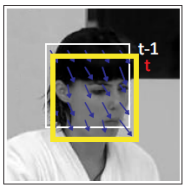
Spatial smoothing of similarity function by introducing a spatial kernel (Gaussian, box filter)

Take derivative of similarity with respect to colors. This tells what colors we need more/less of to make current hist more similar to reference hist.

Result is weighted mean shift we used before. However, the color weights are now computed "on-the-fly", and change from one iteration to the next.

Lucas Kanade Tracking

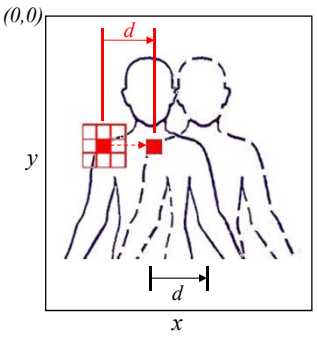
- Traditional Lucas-Kanade is typically run on small, corner-like features (e.g. 5x5) to compute *optical flow*.
- Observation: There's no reason we can't use the same approach on a larger window around the object being tracked.
  - Summarize tracked features
  - Match templates



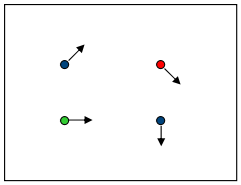
Sparse motion flow

Lucas-Kanade

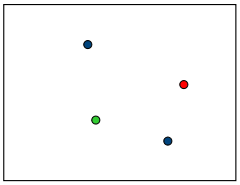
- Originally intended for fast image registration
- Selects features based on texture:
  - Coefficient matrix based on covariance of image gradients within a window around the proposed feature
  - Eigenvalues of coefficient matrix must be large and similarly valued
- Tracks features based on error:
  - Error between image intensities
  - L_2 Norm (Sum of Squares) used to define error
  - Small changes between frames assumed



Reminder: optical flow



$H(x, y)$



$I(x, y)$

- How to estimate pixel motion from image H to image I?
  - Solve pixel correspondence problem
    - given a pixel in H, look for nearby pixels of the same color in I

Key assumptions

- color constancy: a point in H looks the same in I
  - For grayscale images, this is brightness constancy
- small motion: points do not move very far

Tracking Point Features : Lucas-Kanade

Assume constant brightness:

$$f(u, v; I) = I_x u + I_y v + I_t = 0$$

image — pixel displacement — image spatial derivatives — image temporal derivative

(optic flow constraint equation)

Estimate the pixel displacement **u** = (u, v)^T by minimizing:

$$E_{LK}(u, v) = K_\rho * \left((f(u, v; I))^2 \right)$$

convolution kernel

Differentiating with respect to *u* and *v*, setting the derivatives to zero leads to a linear system:

$$\begin{bmatrix} K_\rho * (I_x^2) & K_\rho * (I_x I_y) \\ K_\rho * (I_x I_y) & K_\rho * (I_y^2) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} K_\rho * (I_x I_t) \\ K_\rho * (I_y I_t) \end{bmatrix}$$

Iterate using Newton-Raphson method

Gradient covariance matrix

LK: Review

- Brightness constancy
- • One equation two unknowns

$$0 = I_t(\mathbf{p_i}) + \nabla I(\mathbf{p_i}) \cdot [u\ v]$$

temporal gradient — spatial gradient — **unknown flow vector**

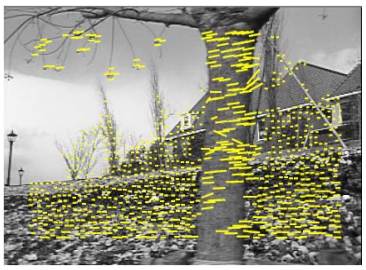
- • How to get more equations for a pixel?
  - Basic idea: impose additional constraints
  - one method: pretend the pixel's neighbors have the same (u,v)
  - If we use a 5x5 window, that gives us 25 equations per pixel!

$$\underbrace{\begin{bmatrix} I_x(\mathbf{p_1}) & I_y(\mathbf{p_1}) \\ I_x(\mathbf{p_2}) & I_y(\mathbf{p_2}) \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}}) & I_y(\mathbf{p_{25}}) \end{bmatrix}}_{A_{25\times 2}} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{d_{2\times 1}} = - \underbrace{\begin{bmatrix} I_t(\mathbf{p_1}) \\ I_t(\mathbf{p_2}) \\ \vdots \\ I_t(\mathbf{p_{25}}) \end{bmatrix}}_{b_{25\times 1}}$$

Utility of Point Features

Advantages:

- highly repeatable and extensible (work for a variety of images)
- efficient to compute (real time implementations available)
- local methods for processing (tracking through multiple frames)



tracking multiple point features = sparse optical flow
sparse point feature tracks yield the image motion

Detection of Point Features

Gradient covariance matrix:

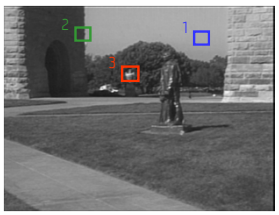
$$Z = \begin{bmatrix} K_\rho * (I_x^2) & K_\rho * (I_x I_y) \\ K_\rho * (I_x I_y) & K_\rho * (I_y^2) \end{bmatrix}$$

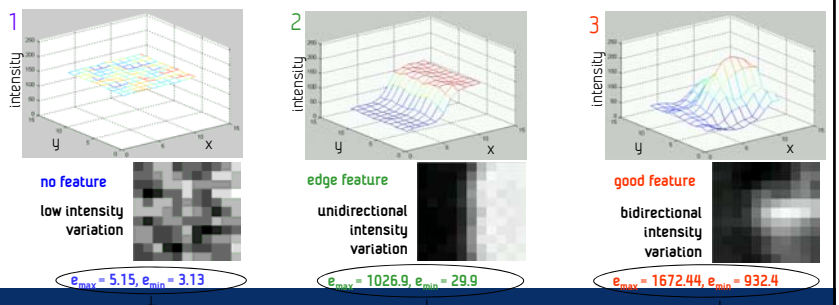
image gradients; convolution kernel

Good feature:

$$min(e_{\min}, e_{\max}) > \varepsilon_f$$

eigenvalues of Z; threshold



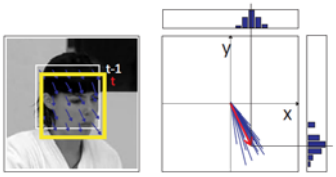


One LK-tracking example; TLD tracker

Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-Backward Error: Automatic Detection of Tracking Failures," *International Conference on Pattern Recognition*, 2010, pp. 23-26.

- Continously learns appearance by recalculating a classifier

TRACKER: Median Shift



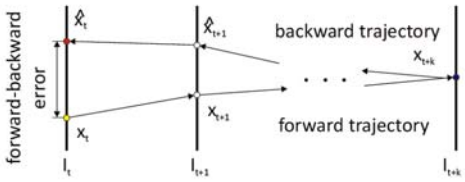
Sparse motion flow

Object displacement

DETECTOR: randomized forest, 2bitBP features

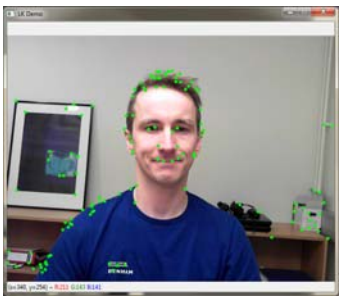
LK tracking: reliability estimates

Selection of reliable points: median is estimated based on 50% of the most reliable points, reliability estimated using combined forward-backward error.
Tracking failure: median residual > threshold.



Demo LK-based tracking

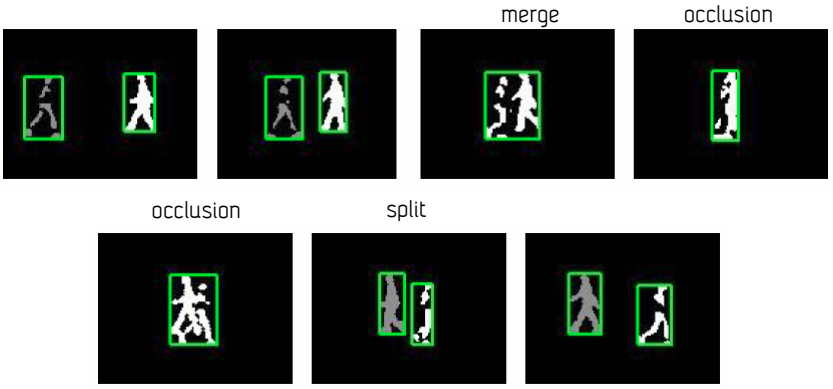
- Shi-Tomasi feature tracker



- Continously learning feature tracker

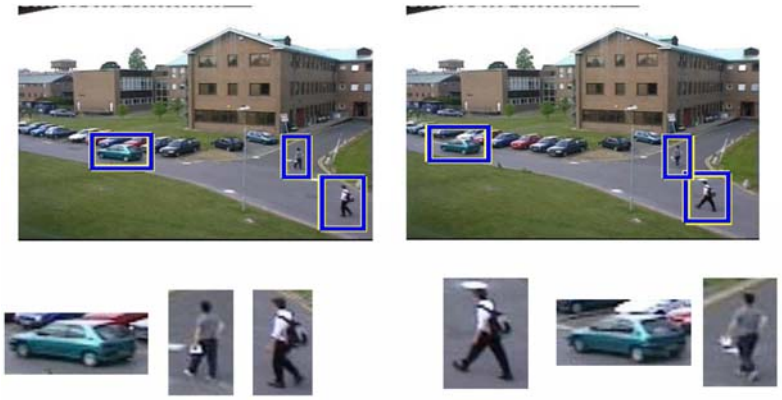
Recall: Blob merge/split

- When two objects pass close to each other, they are detected as a single blob.
- Often, one object will become occluded by the other one. One of the challenging problems is to maintain correct labeling of each object after they split again.



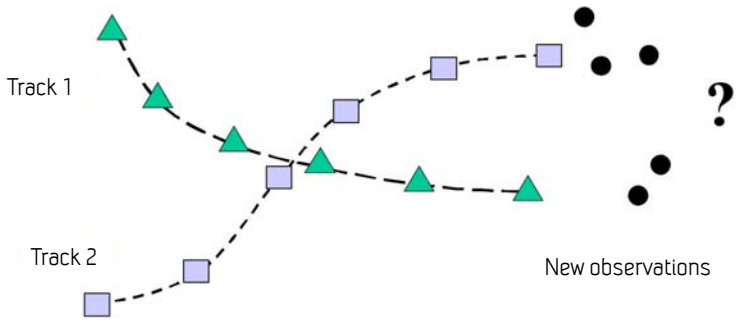
Data association

- More generally, we seek to match a set of blobs across frames, to maintain continuity of identity and generate trajectories.



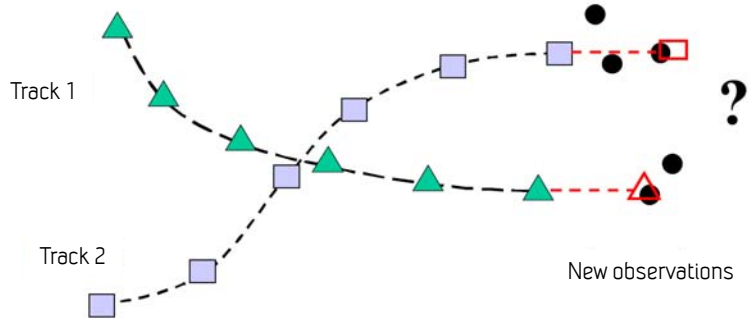
Data Association

- More generally, we seek to match a set of blobs across frames, to maintain continuity of identity and generate trajectories.



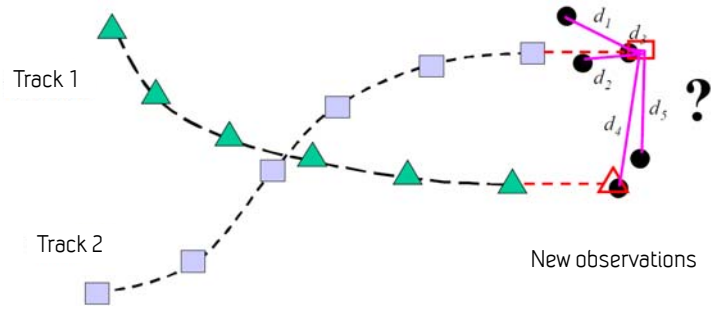
Data Association Scenarios

- Intuition: predict next position along each track



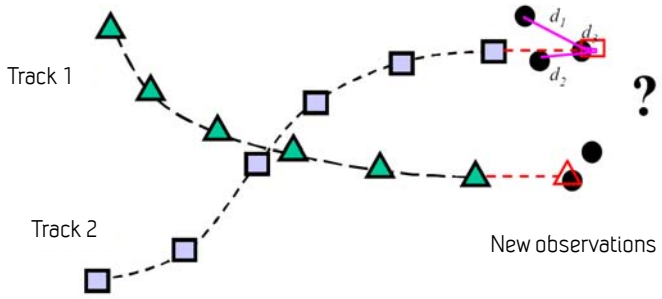
How to determine which observations to add to which track?

- Intuition: predict next position along each track + match should be close to predicted position



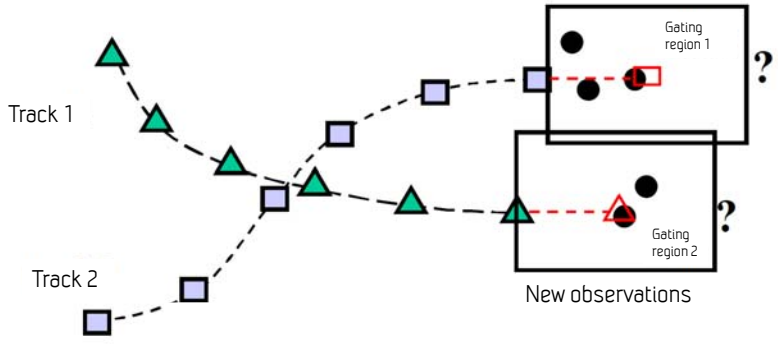
Track matching

- But some matches are fairly unlikely, still

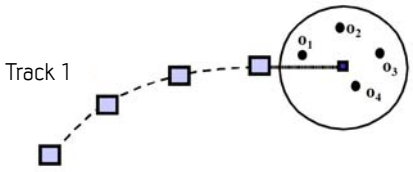


Gating

A method for pruning matches that are geometrically unlikely from the start.
Allows us to decompose matching into smaller subproblems.
Kalman filter – next lecture



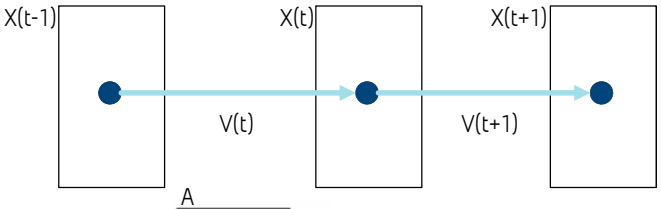
Global nearest neighbor



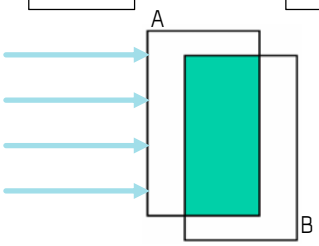
- Global Nearest Neighbor (GNN)
- Evaluate each observation in track gating region.
- Choose "best" one to incorporate into track.
- a_{1j} = score for matching observation j to track 1
- Could be based on
  - Euclidean or Mahalanobis distance to predicted location (e.g. $\exp\{-d^2\}$).
  - similarity of appearance (e.g. template correlation score, color histogram etc)

Data Association

- We have been talking as if our objects are points. (which they are if we are tracking corner features or radar blips). But our objects are blobs – they are an image region, and have an area.



- It is common to assume that objects move with constant velocity : V(t)=V(t+1)



$$score = \frac{2 * area(A \text{ and } B)}{area(A) + area(B)}$$

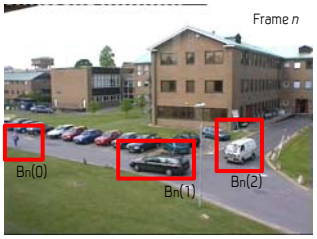
A = bounding box at time t, adjusted by velocity V(t)
B = bounding box at time t+1

Map the object region forward in time to predict a new region.

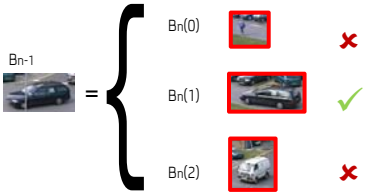
Appearance information

- Correlation of image templates between frames is an obvious choice (moving blobs)

Extract blobs:



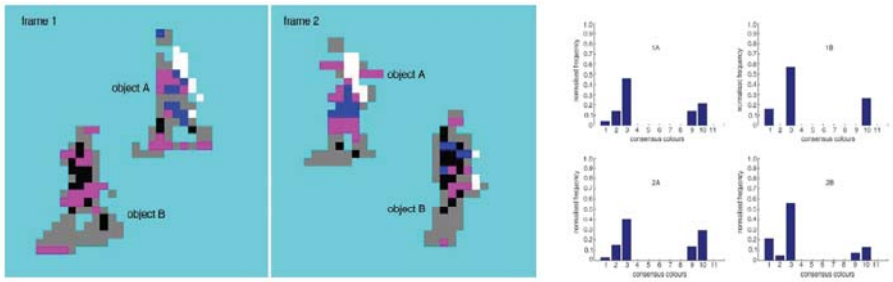
Data association:



Update appearance template:

Color Descriptor

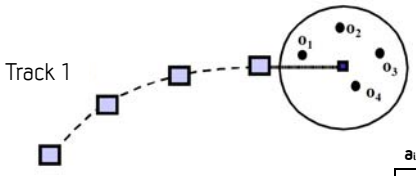
A. Gilbert, R. Bowden ; *Incremental, Scalable Tracking of Objects Inter Camera* -; In Computer Vision and Image Understanding CVIU, Vol 111 Pages 43-58, 2008



Describe colors using some kind of perceptive space. Munsell consensus colors seems like a reasonable choice.

Compare color histograms using some distance function, e.g., Bhattacharyya coefficient $\Delta(m,d) = \sqrt{1 - \sum_{i=1}^{n} \sqrt{m_i \times d_i}}$ or the sum of minimum buckets.

Global nearest neighbor

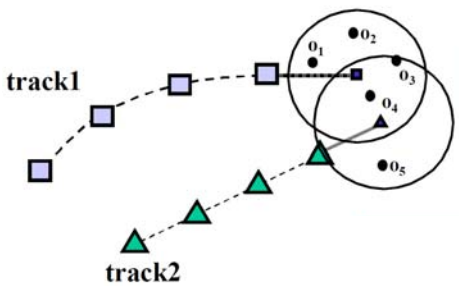


	a_{i1}
1	3.0
2	5.0
3	6.0
4	9.0

- Global Nearest Neighbor (GNN)
- Evaluate each observation in track gating region.
- Choose "best" one to incorporate into track.
- a_{1j} = score for matching observation j to track 1
- Could be based on
  - Euclidean or Mahalanobis distance to predicted location (e.g. $\exp\{-d^2\}$).
  - similarity of appearance (e.g. template correlation score, color histogram etc)
- Choose best match $a_{m1} = \max\{a_{11}, a_{21}, a_{31}, a_{41}\}$

Global nearest neighbor

Problem: several tracks may compete for the same observation, usually because of occlusion



	a_{i1}	a_{i1}
1	3.0	
2	5.0	
3	6.0	1.0
4	9.0	8.0
5		3.0

Linear Assignment Problem

We have N objects in previous frame and M objects in current frame. We can build a table of match scores $m(i,j)$ for $i = 1 \dots N$ and $j = 1 \dots M$. For now, assume $M = N$.

	1	2	3	4	5
1	0.95	0.76	0.62	0.41	0.06
2	0.23	0.46	0.79	0.94	0.35
3	0.61	0.02	0.92	0.92	0.81
4	0.49	0.82	0.74	0.41	0.01
5	0.89	0.44	0.18	0.89	0.14

Problem: choose a 1-1 correspondence that maximizes sum of match scores.

Assignment problem

Mathematical definition. Given an NxN array of benefits $\{X_{ai}\}$, determine an NxN permutation matrix M_{ai} that maximizes the total score:

maximize $$E = \sum_{a=1}^{N} \sum_{i=1}^{N} M_{ai} X_{ai}$$

subject to $$\forall \sum_{a=1}^{A} M_{ai} = 1$$

$$\forall \sum_{i=1}^{I} M_{ai} = 1$$

$$M_{ai} \in \{0,1\}$$

Constraints ensuring M is a permutation matrix

The permutation matrix ensures that we can only choose one number from each row and from each column.

Example

- 5x5 matrix of match scores

	1	2	3	4	5
1	0.95	0.76	0.62	0.41	0.06
2	0.23	0.46	0.79	0.94	0.35
3	0.61	0.02	0.92	0.92	0.81
4	0.49	0.82	0.74	0.41	0.01
5	0.89	0.44	0.18	0.89	0.14

- Working from left to right, choose one number from each column, making sure you don't choose a number from a row that already has a number chosen in it.
- How many ways can we do this? 5 x 4 x 3 x 2 x 1 = 120 (N factorial)

Couple random tries

	1	2	3	4	5
1	0.95	0.76	0.62	0.41	0.06
2	0.23	0.46	0.79	0.94	0.35
3	0.61	0.02	0.92	0.92	0.81
4	0.49	0.82	0.74	0.41	0.01
5	0.89	0.44	0.18	0.89	0.14

score: 2.88

	1	2	3	4	5
1	0.95	0.76	0.62	0.41	0.06
2	0.23	0.46	0.79	0.94	0.35
3	0.61	0.02	0.92	0.92	0.81
4	0.49	0.82	0.74	0.41	0.01
5	0.89	0.44	0.18	0.89	0.14

score: 2.52

	1	2	3	4	5
1	0.95	0.76	0.62	0.41	0.06
2	0.23	0.46	0.79	0.94	0.35
3	0.61	0.02	0.92	0.92	0.81
4	0.49	0.82	0.74	0.41	0.01
5	0.89	0.44	0.18	0.89	0.14

score: 4.14

Greedy algorithm

- Choose largest value and mark it
  1. For $i = 1$ to $N - 1$
  2. Choose next largest remaining value that isn't in a row/col already marked
  3. End

	1	2	3	4	5
1	0.95	0.76	0.62	0.41	0.06
2	0.23	0.46	0.79	0.94	0.35
3	0.61	0.02	0.92	0.92	0.81
4	0.49	0.82	0.74	0.41	0.01
5	0.89	0.44	0.18	0.89	0.14

score: 3.77 **Is this the best we can do?**

- not as good as our current best guess!

Linear programming

- AKA the linear assignment problem

$$\min_M E_{assign}(M|A) = \min_M \sum_{i,j} M_{ij} \| X_i - (A + I) Y_j \|^2 = \min_M \sum_{i,j} M_{ij} Q_{ij}$$

$$\text{s.t.} \quad \sum_i M_{ij} = 1, \sum_j M_{ij} = 1$$

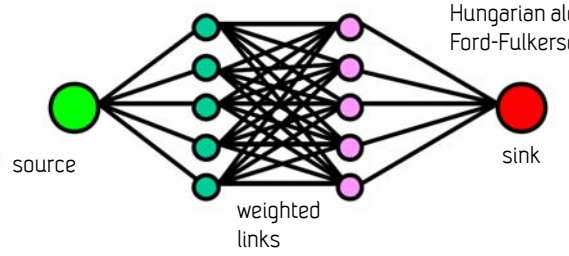
$$M_{ij} \in \{0,1\}$$

- This has the form of a 0-1 integer linear program. Could solve using the simplex method. However, bad (exponential) worst-case complexity (0-1 integer programming is NP-hard)

Some (possible) Solution Methods

- Can also be viewed as a maximal matching in a weighted bipartite graph, which in turn can be characterized as a max-flow problem.
- This has efficient numerical solution and an abundance of library code available
- Graph cuts is outside the scope of this lecture, see Dynamic Programming and Graph Algorithms in Computer Vision, Pedro Felzenszwalb and Ramin Zabih. To appear in IEEE Transactions on Pattern Analysis and Machine Intelligence

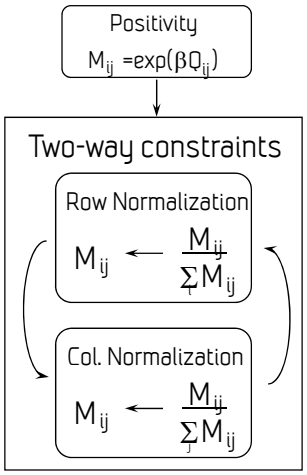
Possible solution methods:
Hungarian algorithm
Ford-Fulkerson algorithm



Softassign

- Main points:
  - relax *0,1* constraint to be *0 <= Mij <=1*
  - init with *Mai = exp(β*score)*. This ensures positivity and also spreads out scores as (βapproaches infinity)
  - perform repeat row and col normalizations to get a doubly stochastic matrix (rows and cols sum to 1)
  - Have an outer loop that adjusts *β*. This corresponds to so called *deterministic annealing*
- Steven Gold and Anand Rangarajan,*Softmax to Softassign: Neural Network Algorithms for Combinatorial Optimization*, Journal of Artificial Neural Networks, 1996

Softassign algorithm



- Step I: $M_{ij} = \exp(-Q_{ij}/T)$.
- Step II: Double Normalization
- Repeat until M converges, outer loop adjusts β

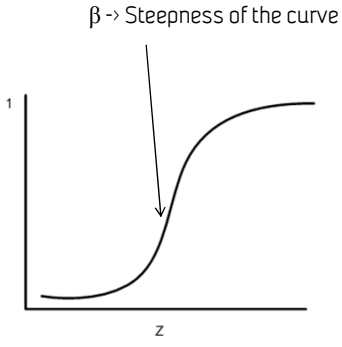
- Outlier rejection using slack variables. (Add rows + cols)
- In practice, should use an iterative version to avoid numerical issues with large B.

Intuition? Consider softmax

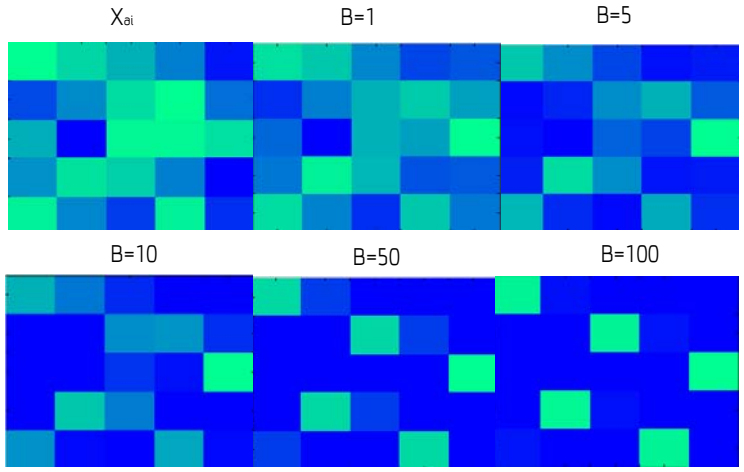
- Softmax is a similar algorithm, but just operates on a single vector of numbers.

$$m_j = \frac{\exp(\beta X_j)}{\sum_{i=1}^{I} \exp(\beta X_i)}$$

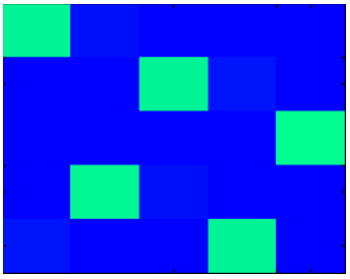
- The exp() function serves to ensure that all numbers are positive (even negative numbers map to positive values through exp)
- As B increases, the *m* associated with the max element approaches 1, while all other m values approach 0.



Softassign for different B



Softassign result



	1	2	3	4	5
1	0.95	0.76	0.62	0.41	0.06
2	0.23	0.46	0.79	0.94	0.35
3	0.61	0.02	0.92	0.92	0.81
4	0.49	0.82	0.74	0.41	0.01
5	0.89	0.44	0.18	0.89	0.14

score: 4.26

If you exhaustively search all 120 assignments, you would find the global maximum is indeed 4.26

Reading materials

- Good survey on tracking algorithms:
  - A. Yilmaz et al. *Object tracking: A survey*, ACM Comput. Surv., 38(4), 2006
- Detect and assign
  - C. Stauffer and W. E. L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 22(8):747-757, 2000.
  - I.D. Reid and K.R. Connor. Multiview segmentation and tracking of dynamic occluding layers. In BMVC 2005, 2005.
- Mean shift tracker
  - D. Comaniciu, V. Ramesh, and P. Meer, Kernel-based object tracking. *IEEE Trans. Patt. Analy. Mach. Intell. 25*, 564–575, 2003
  - G.R. Bradski et al. *Computer Vision Face Tracking For Use in a Perceptual User Interface*, *Interface* 2, pp 12–21, 1998 – in OpenCV http://opencv.itseez.com/modules/video/doc/motion_analysis_and_object_tracking.html#camshift
- LK tracker
  - Jianbo Shi and Carlo Tomasi *Good Features to Track*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94), 1994, pp. 593 - 600.
  - Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-Backward Error: Automatic Detection of Tracking Failures," *International Conference on Pattern Recognition*, 2010, pp. 23-26.
- Short credit: Some of the slides blatantly "stolen" from Kevin Smith, ETHZ