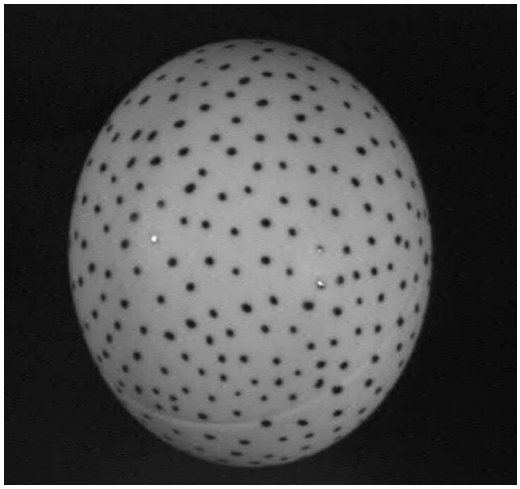
Motion Tracking



Motion tracking

Suppose we have more than two images

- How to track a point through all of the images?
  - In principle, we could estimate motion between each pair of consecutive frames
  - Given point in first frame, follow arrows to trace out it's path
  - Problem: DRIFT
    - » small errors will tend to grow and grow over time—the point will drift way off course

Feature Tracking

- Choose only the points ("features") that are easily tracked
- How to find these features?
  - windows where $\sum \nabla I (\nabla I)^T$ has two large eigenvalues
- Called the Harris Corner Detector

Feature Detection

Tracking features

Feature tracking

- Compute optical flow for that feature for each consecutive H, I

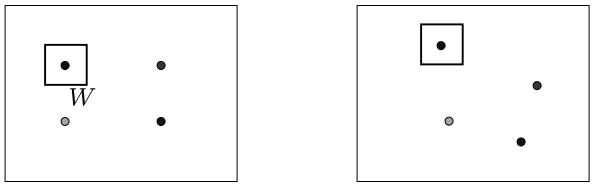
When will this go wrong?

- Occlusions—feature may disappear
  - need mechanism for deleting, adding new features
- Changes in shape, orientation
  - allow the feature to deform
- Changes in color
- Large motions
  - will pyramid techniques work for feature tracking?

Handling large motions

L-K requires small motion

- If the motion is much more than a pixel, use discrete **search** instead



- Given feature window W in H, find best matching window in I
- Minimize sum squared difference (SSD) of pixels in window

$$min_{(u,v)} \left\{ \sum_{(x,y) \in W} |I(x+u, y+v) - H(x,y)|^2 \right\}$$

- Solve by doing a search over a specified range of (u,v) values
  - this (u,v) range defines the **search window**

Tracking Over Many Frames

Feature tracking with m frames

1. Select features in first frame
2. Given feature in frame i, compute position in i+1
3. Select more features if needed
4. i = i + 1
5. If i < m, go to step 2

Issues

- Discrete search vs. Lucas Kanade?
  - depends on expected magnitude of motion
  - discrete search is more flexible
- Compare feature in frame i to i+1 or frame 1 to i+1?
  - affects tendency to drift..
- How big should search window be?
  - too small: lost features. Too large: slow

Incorporating Dynamics

Idea

- Can get better performance if we know something about the way points move
- Most approaches assume constant velocity

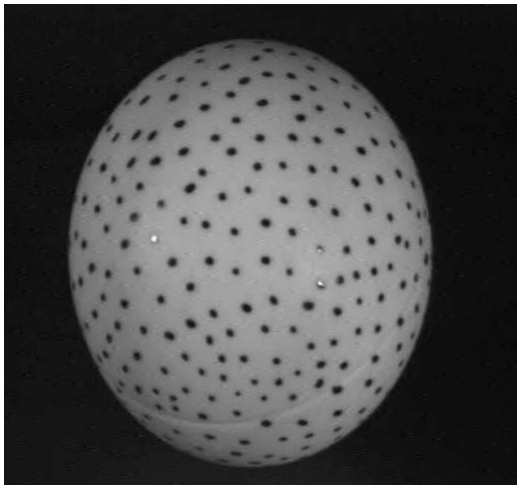
$$\dot{\mathbf{x}}_{i+1} = \dot{\mathbf{x}}_i$$
$$\mathbf{x}_{i+1} = 2\mathbf{x}_i - \mathbf{x}_{i-1}$$

or constant acceleration

$$\ddot{\mathbf{x}}_{i+1} = \ddot{\mathbf{x}}_i$$
$$\mathbf{x}_{i+1} = 3\mathbf{x}_i - 3\mathbf{x}_{i-1} + \mathbf{x}_{i-2}$$

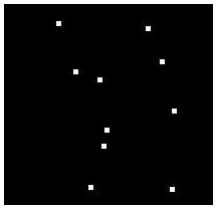
- Use above to predict position in next frame, initialize search

Point Tracking (Similar Features)



Point Tracking

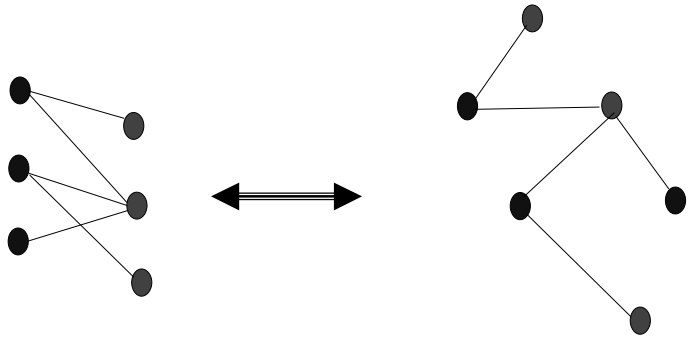
- All objects are similar
- Only Motion information is available



Graph Theory (Again)

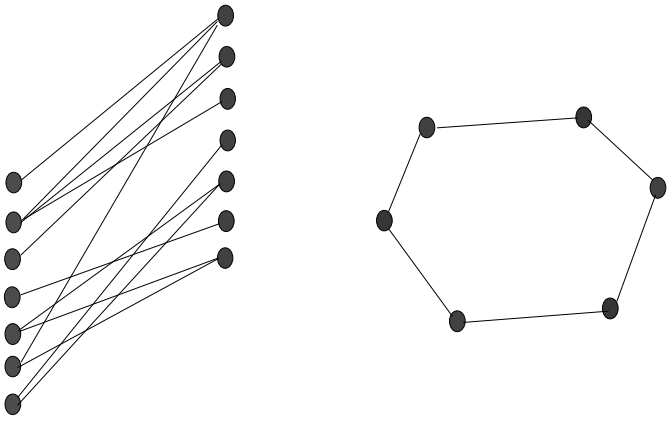
Graph

A graph G(V,E) is a triple consisting of a vertex set V(G) an edge set E(G) and a relation that associates with each edge two vertices called its end points.



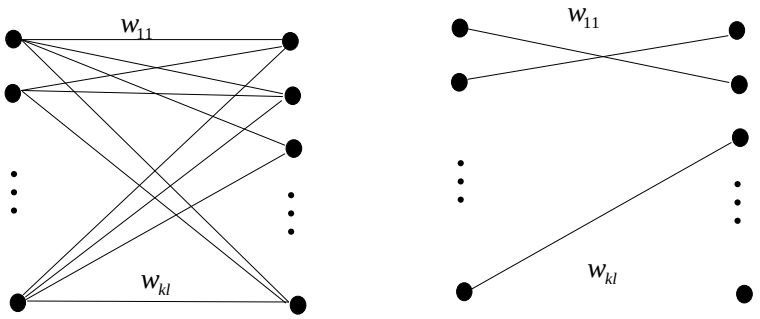
Bipartite Graph

A graph G is bipartite if its vertex set can be partitioned in two subsets in such a way that no two vertex in same set have a common edge.



Matching

Matching is a set of edges such that no two of them have a common vertex

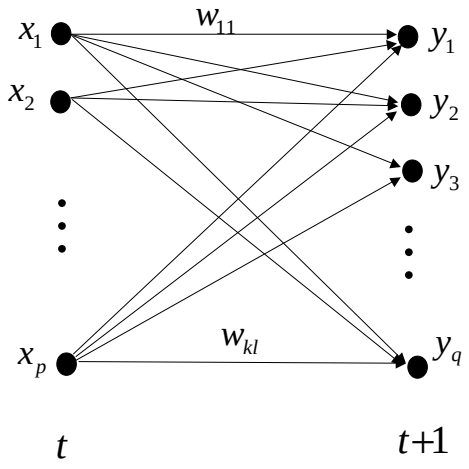


Point Tracking

- Points corresponds to vertices in the bipartite graph
- Points at time instants *t* and *t*+1 form partite sets of graph.
- The cost of corresponding a point at instant *t* to a point at instant *t*+1 is the weight of edge between the corresponding vertices.

S. Ullman, "The interpretation of visual motion," MIT Press, Cambridge, MA.

Point Tracking



Find Minimum Matching of Bipartite Graph

Maximum (Minimum)Matching is a set of edges such that no two of them have a common vertex and the sum of weights is maximum (minimum) among all such sets

Algorithm

- Compute costs w_{ij} for each pair of points
- Construct a bipartite graph based on computed costs
- Prune all edges having weights exceeding certain threshold
- Find the minimum matching of constructed graph. (Hungarian Algorithm)

Cost Computation

Ullman: $w_{ij} = \left\| y_j - x_i \right\|$

Sethi & Jain: $w_{ij} = c\left[1 - \frac{(y_j - x_i).v_{x_i}}{\left\| y_j - x_i \right\| \left\| v_{x_i} \right\|}\right] + (1-c)\left[1 - \frac{GM\left(\left\| y_j - x_i \right\|, \left\| v_{x_i} \right\|\right)}{AM\left(\left\| y_j - x_i \right\|, \left\| v_{x_i} \right\|\right)}\right]$

$GM(a,b) = \sqrt{ab}$

$AM(a,b) = \frac{a+b}{2}$

Rangarajan & Shah: $w_{ij} = \frac{\left\| v_{x_i} - (y_j - x_i) \right\|}{\sum_{k=1}^{p}\sum_{l=1}^{q}\left\| v_{x_k} - (y_l - x_k) \right\|} + \frac{\left\| y_j - x_i \right\|}{\sum_{k=1}^{p}\sum_{l=1}^{q}\left\| y_l - x_p \right\|}$

Can we increase the search space further in time?

- An N-Dimensional Matching is a set *M* of hyper-edges or s-tuples such that no two hyper-edges of *M* have common vertices.
- A minimum N-D Matching is an N-D Matching with minimum weight
- Finding a minimum N-D Matching is NP-Hard Problem

Change Detection

C. Stauffer and W.E.L. Grimson, “Learning patterns of activity using real time tracking,” IEEE Trans. On PAMI, 22(8):747-757, Aug 2000