The Uses of Correspondence, or elementary 3D, tracking, etc

D.A. Forsyth, University of Illinois at Urbana Champaign

SOME APPLICATION SLIDES MISSING

Main points

- Matching locations across images reveals
  - how you have moved; the shape of things; what else is moving
- Low compute matching for scattered points is very powerful
  - Structure from motion or SLAM
- You can fill in shape information
- High compute matching yields very dense information
  - but requires some thought about network structure

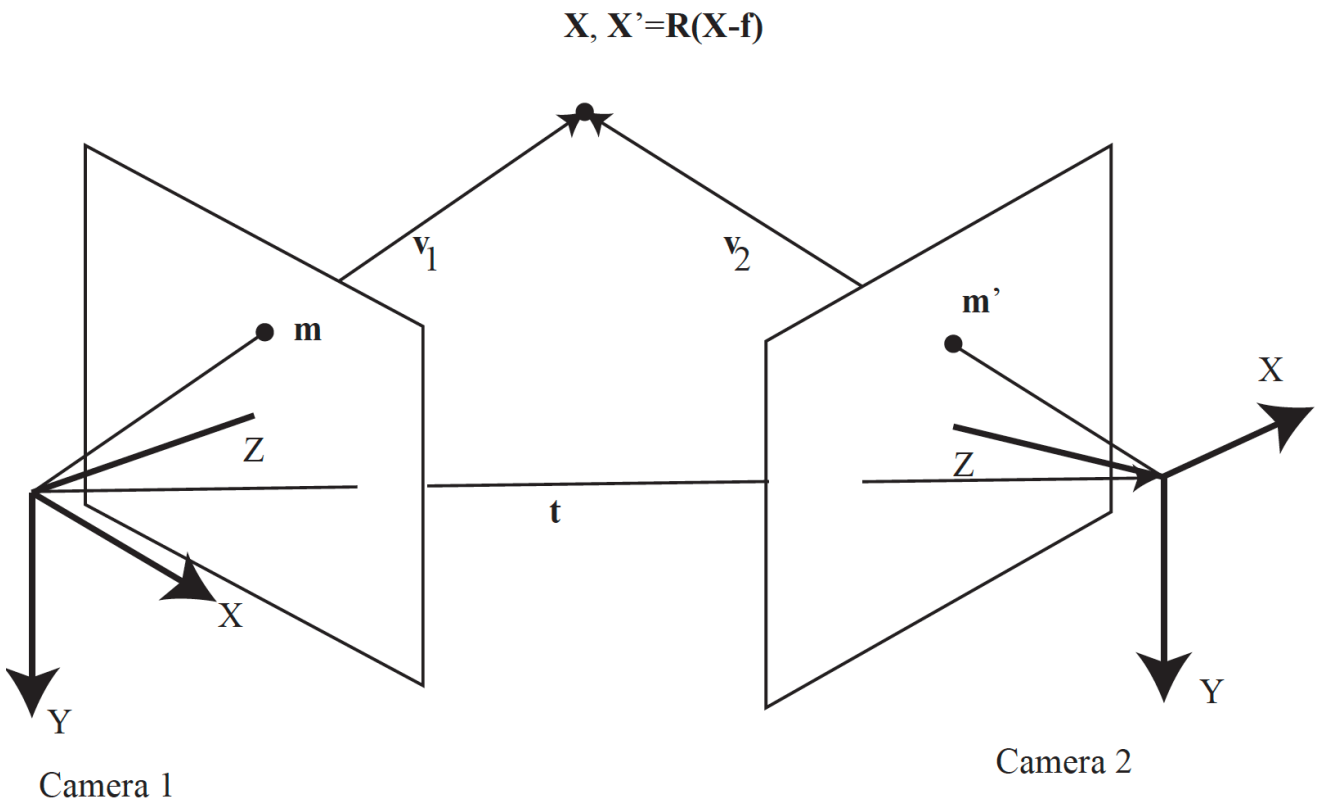
Calibration

- Procedure
  - View N known reference points in 3D
  - Obtain locations in image plane
  - Produce
    - Camera intrinsics and extrinsics
- Why
  - recovering intrinsics for future use
  - where is an object/camera in the camera/world frame?
- How
  - Optimization: minimize reprojection error

Strategies

- Chuck it into an optimizer and run
  - AAARGH!
- Find a good start point
  - all works out
- Variants
  - Different point locations; points on plane; etc., etc.

Reconstruct point from two views



Triangulation

- Have m, m', R, t
- Compute reprojection error
  - sum:
    - residual between X projected into 1 and m
    - residual between X projected into 2 and m'
- Choose X that minimizes

Now assume you know $\mathcal{R}$ and $\mathbf{t}$, the intrinsic calibration matrices $\mathcal{K}$ and $\mathcal{K}'$, and have estimated locations $\mathbf{m}$ and $\mathbf{m}'$ for a pair of points that correspond. These estimates may not be exact – for example, they might come from an interest point matcher – but any error is small. You must recover the point in 3D.

The first camera has camera matrix $\mathcal{K}\mathcal{C}_p$. The second camera has camera matrix

$$\mathcal{K}' \left[\mathcal{R}\right| - \mathcal{R}\mathbf{t}]$$

(recall notation from Section 21.3, and check that this camera has focal point at $\mathbf{t}$).

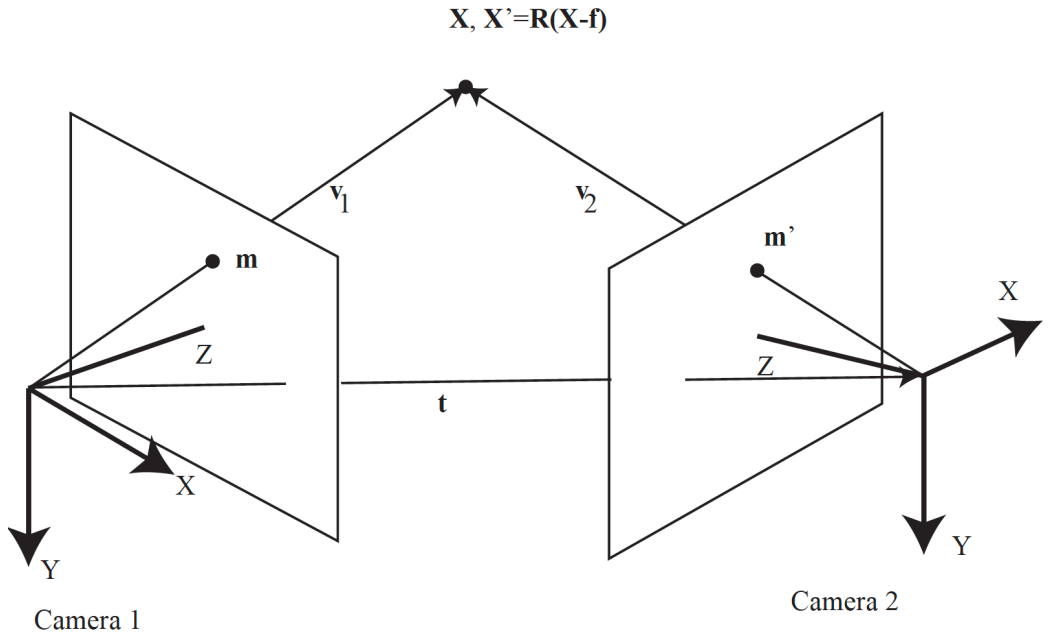
Stereo

- Given two images of a static scene
  - construct a depth map by:
    - correspondence
    - triangulation

- Photometric consistency
  - points that match will have same color (intensity)
  - largely, but not wholly, true
    - specular surfaces, glossy surfaces, etc.

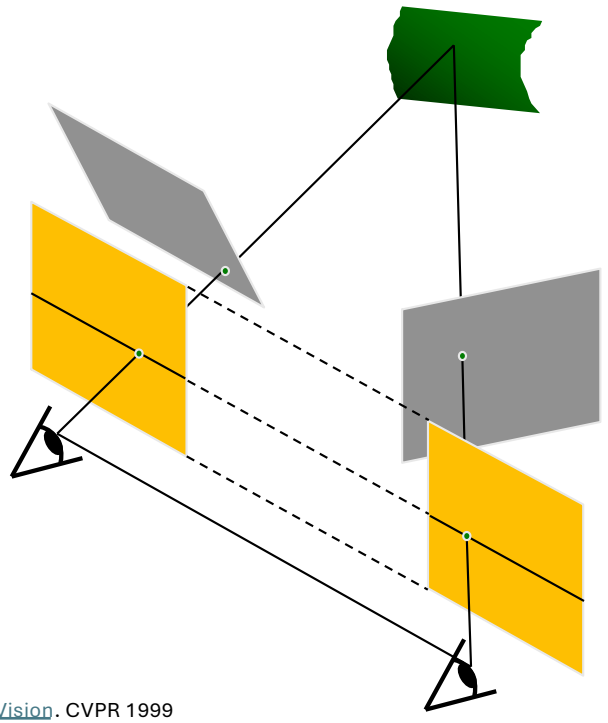
Recall triangulation

- IF you know m, m’ correspond
  - you can get X



Stereo image rectification

- If the image planes are not parallel, we can find homographies to project each view onto a common plane parallel to the baseline



•C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision. CVPR 1999

Stereo image rectification

- Before rectification:

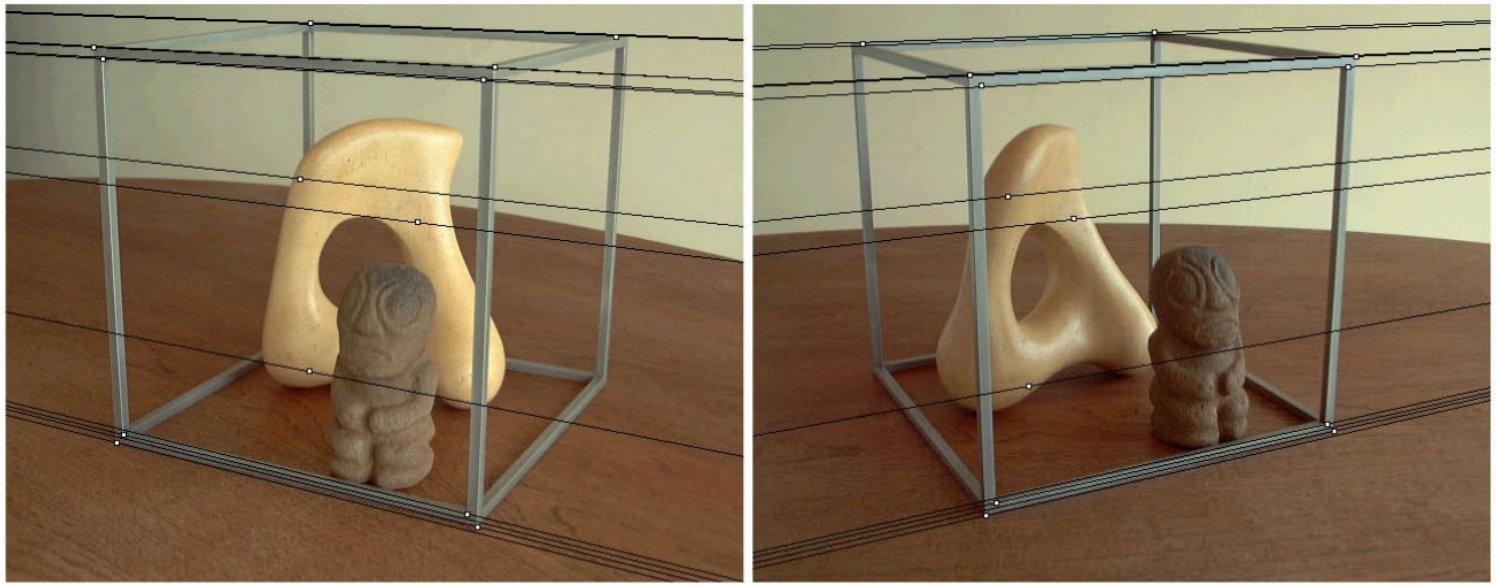


Image source

Stereo image rectification

- After rectification:

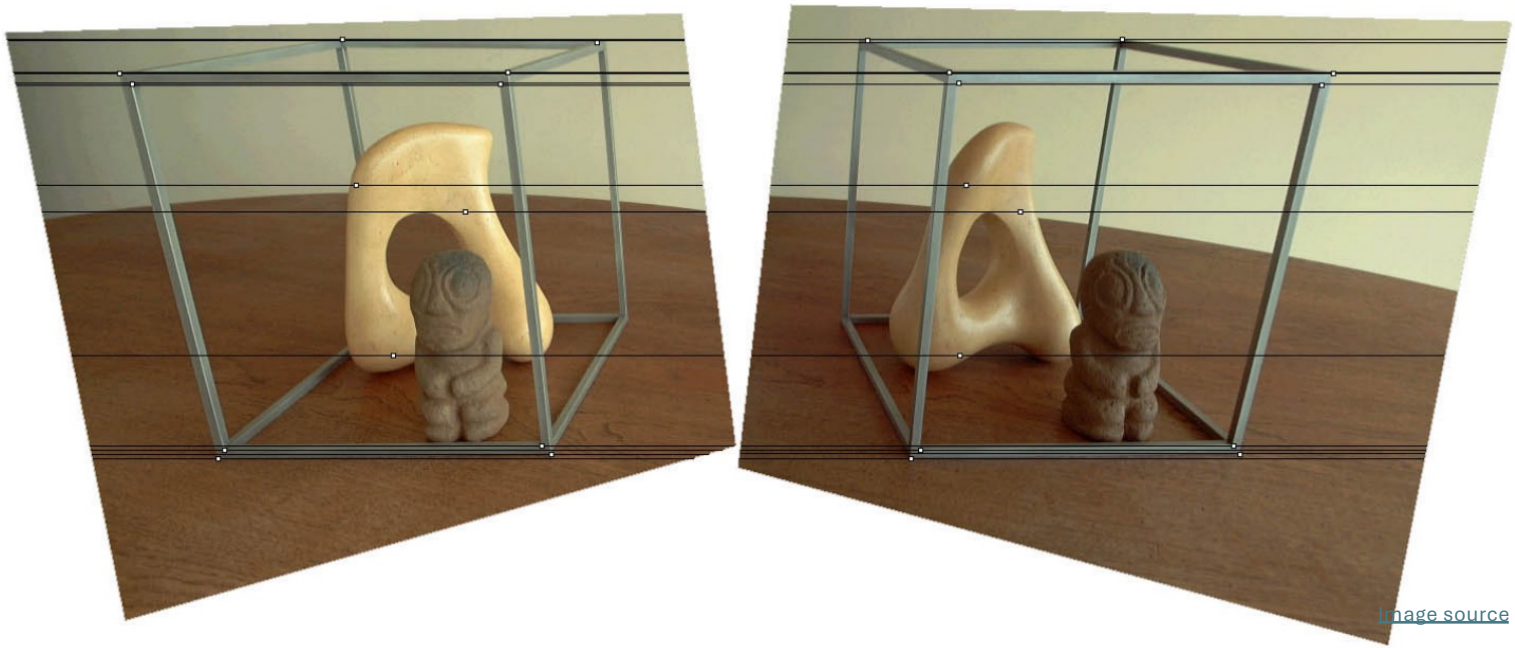
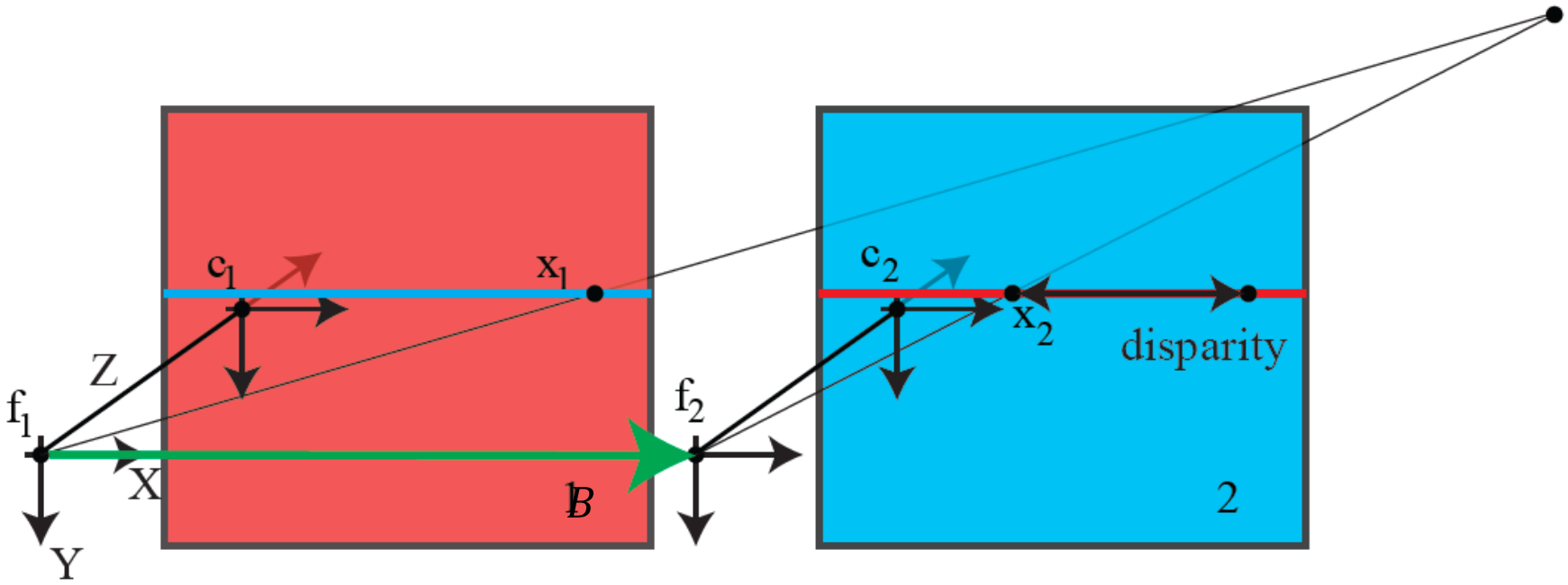
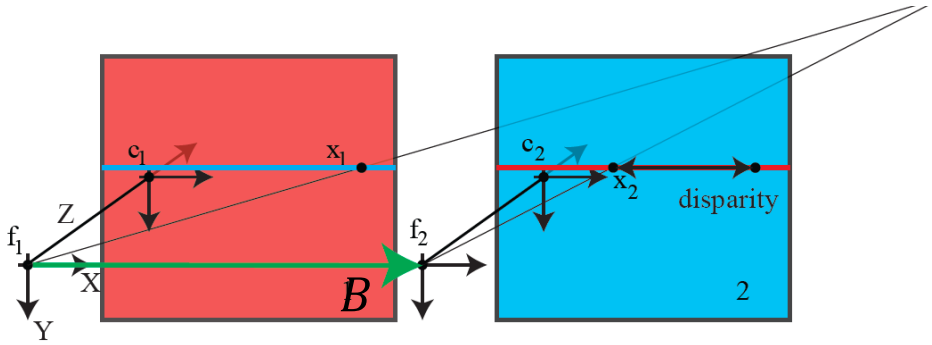


Image source

Disparity and baseline

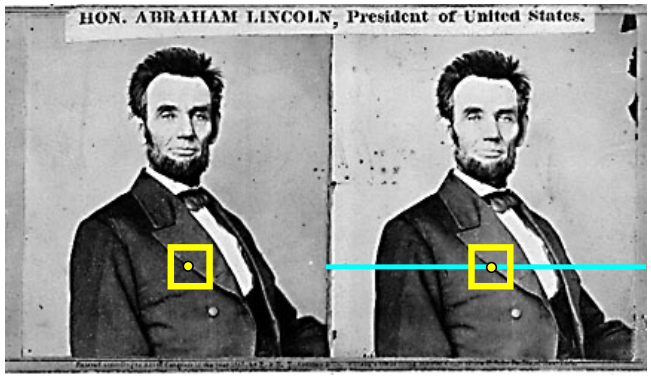


Disparity and baseline



The configuration has baseline B. The second camera maps (X, Y, Z) *in the world coordinate system* to $(X/Z - B/Z, Y/Z)$. This configuration has the extremely useful property that the epipolar lines are the lines $y =$ constant. You will occasionally see this configuration referred to as an *epipolar configuration*. The point being viewed appears at different locations in the two images. The difference in position is known as the *disparity*.

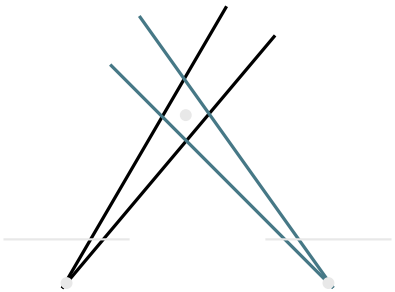
Basic stereo recipe



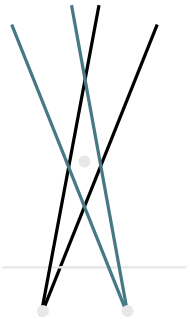
- If necessary, rectify the two stereo images to transform epipolar lines into scanlines
- For each pixel x in the first image
  - Find corresponding epipolar scanline in the right image
  - Examine all pixels on the scanline and pick the best match x'
  - Triangulate the matches to get depth information

Hard to pick the best match
--might be far away
– some points don't have matches

Effect of baseline on stereo results

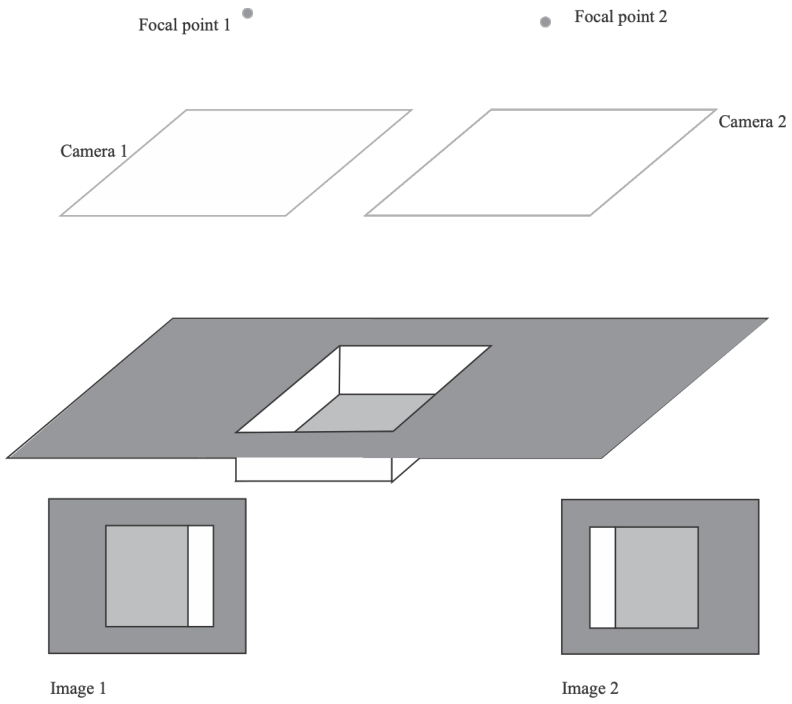


- Larger baseline
  - \+ Smaller triangulation error
  - – Matching is more difficult



- Smaller baseline
  - – Higher triangulation error
  - \+ Matching is easier

Some points don’t have matches



This is a depth cue, though not much used directly

Effect sometimes known as
Da Vinci stereopsis

Stereo as optimization

- Depth => Disparity => Matches

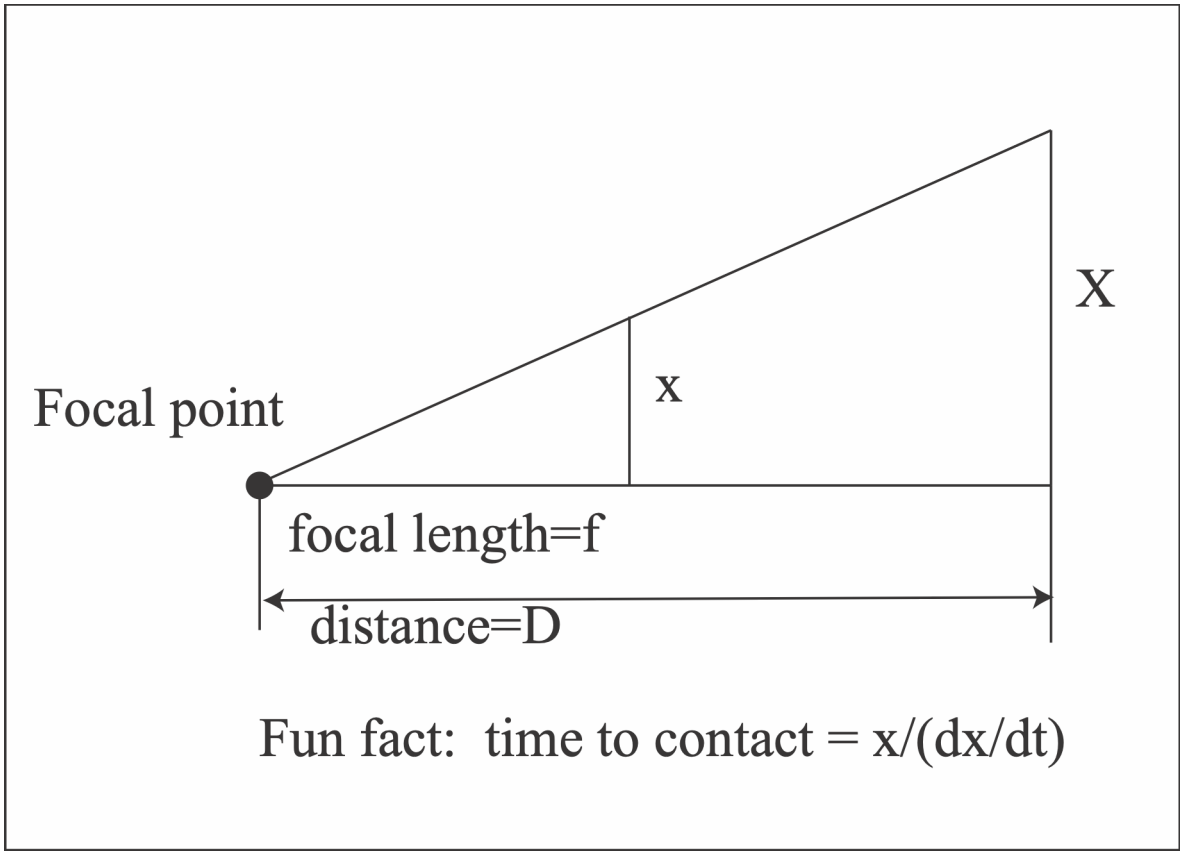
- Search for disparities such that
  - Matches are good
  - Disparities meet various constraints
  - Depth map is “like” real depth map

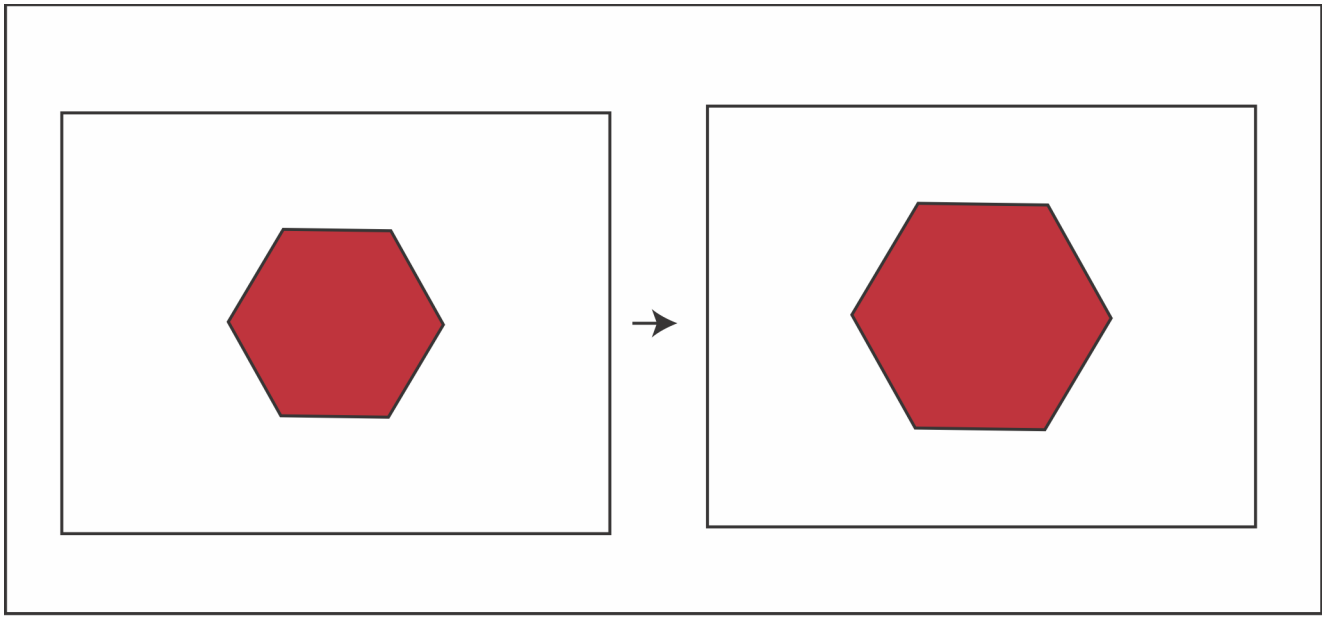
- Highly developed, effective theory

Optical flow

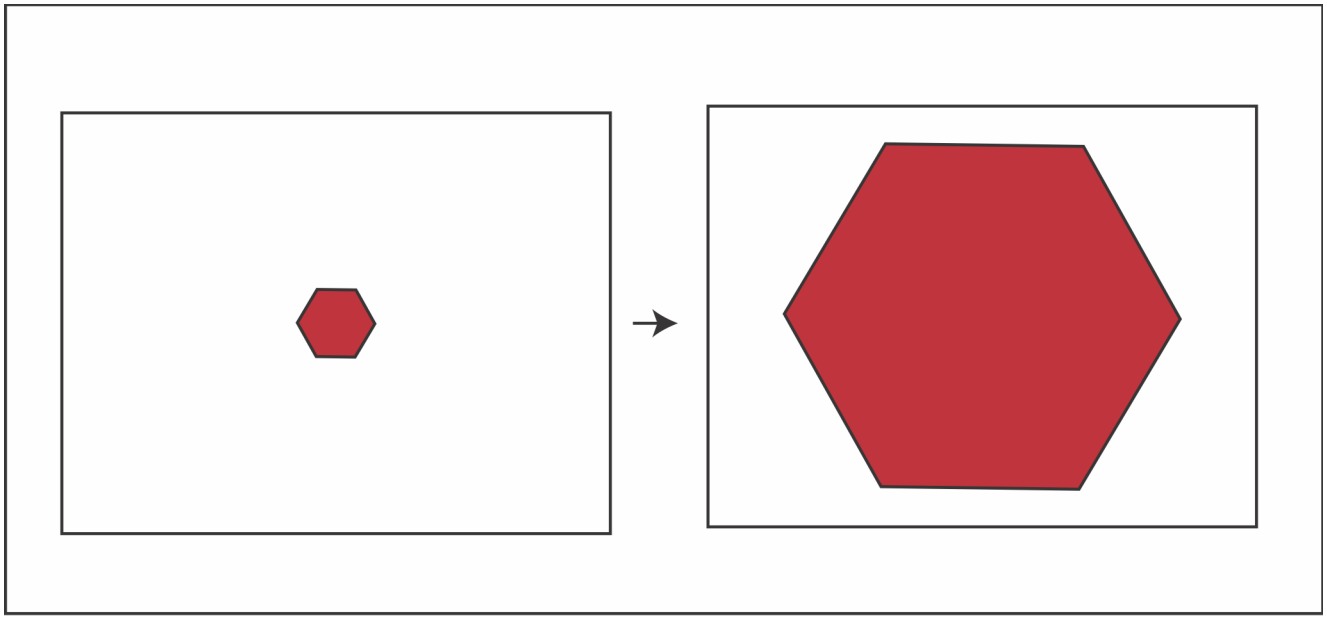
- Given two images of any scene
  - where stuff could move
- Build a field of arrows
  - tail at pixel
  - head at new location of surface behind pixel
    - in next frame
- Uses
  - inference about your motion (egomotion)
  - inference about motion in the world
  - inference about depth
  - segmentation
  - compression

Time to contact example

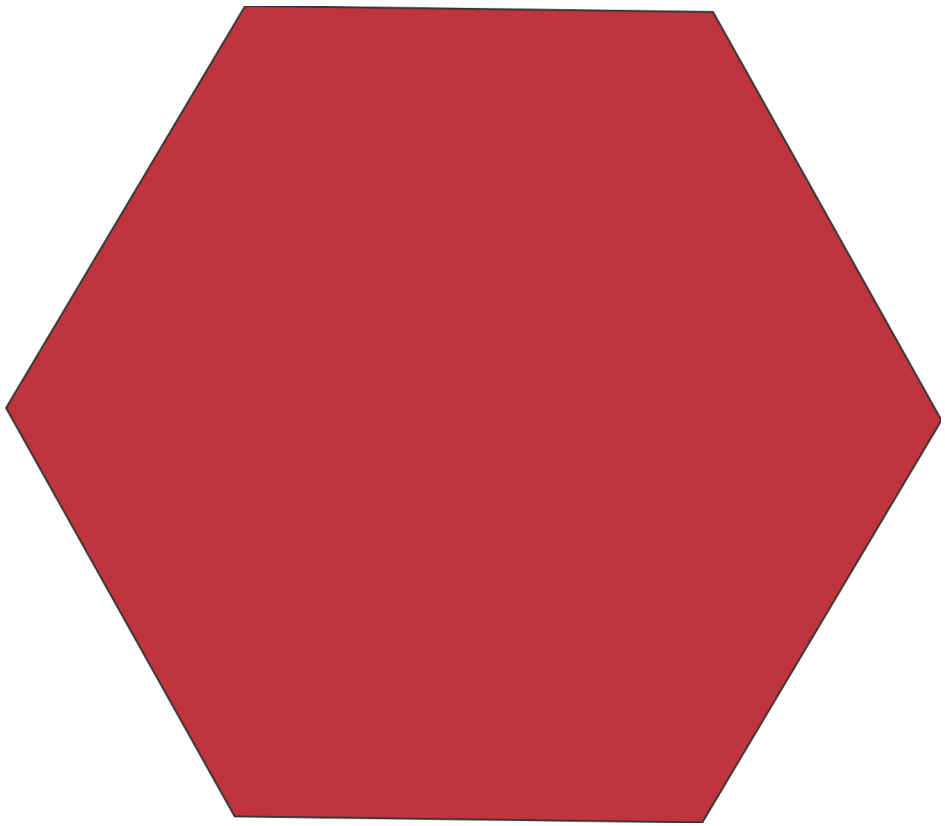


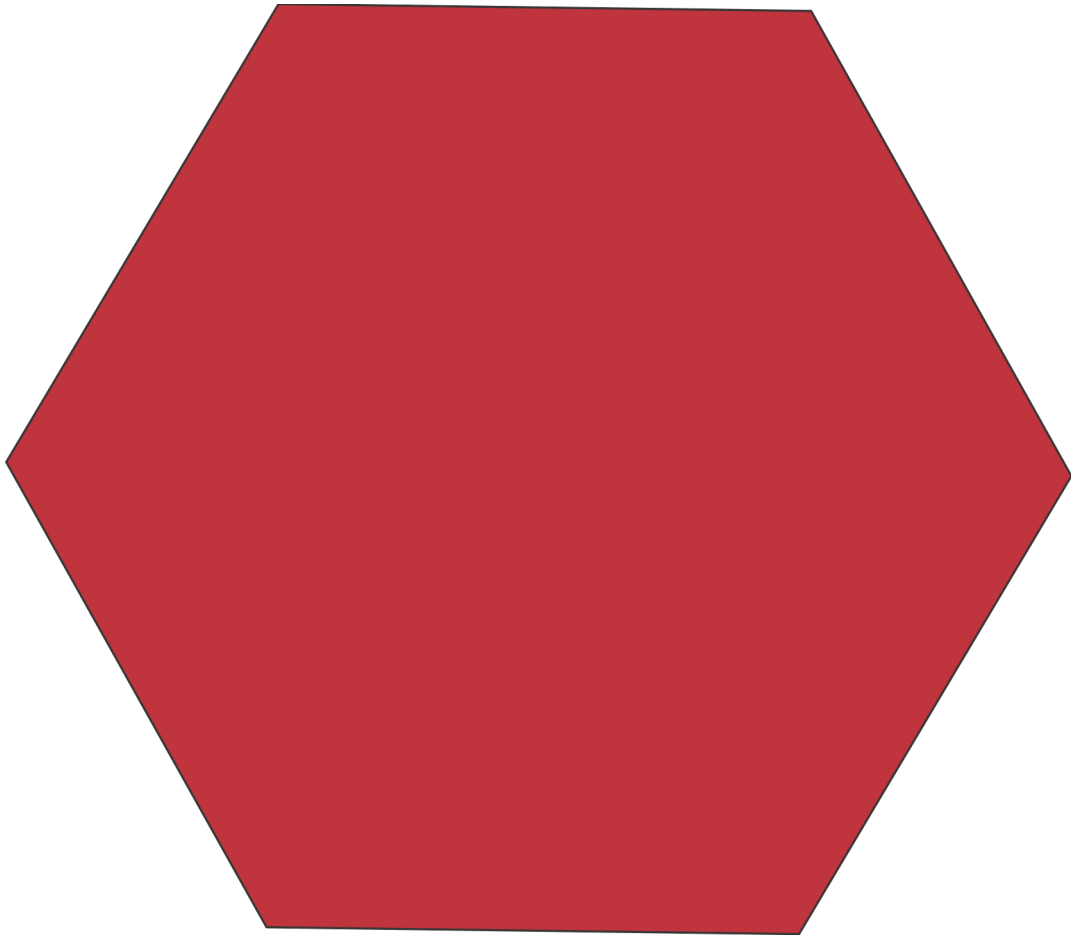


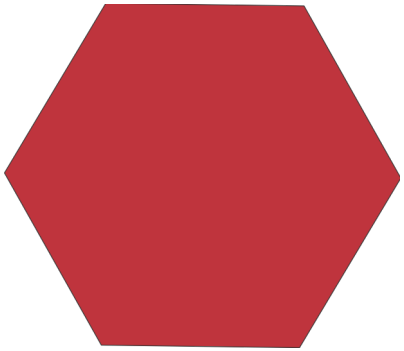
TTC - Long

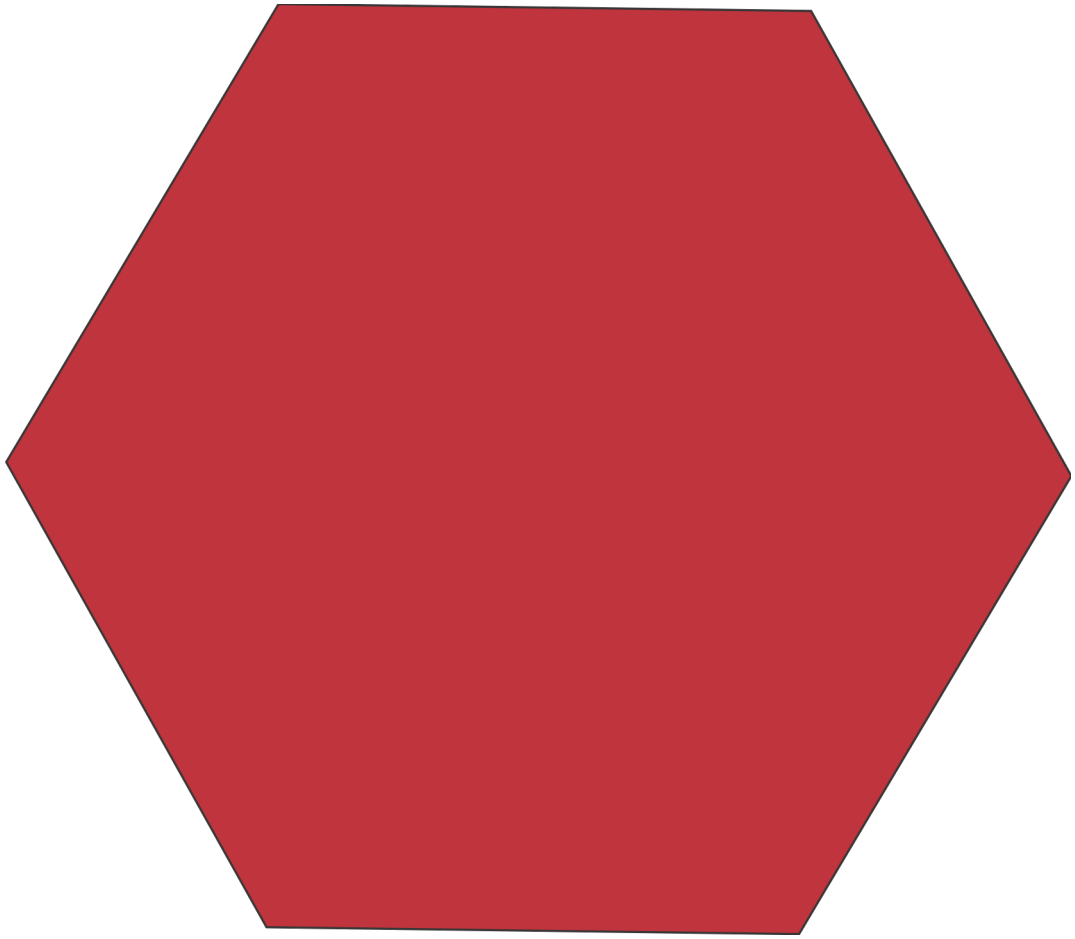


TTC - AAARGH!

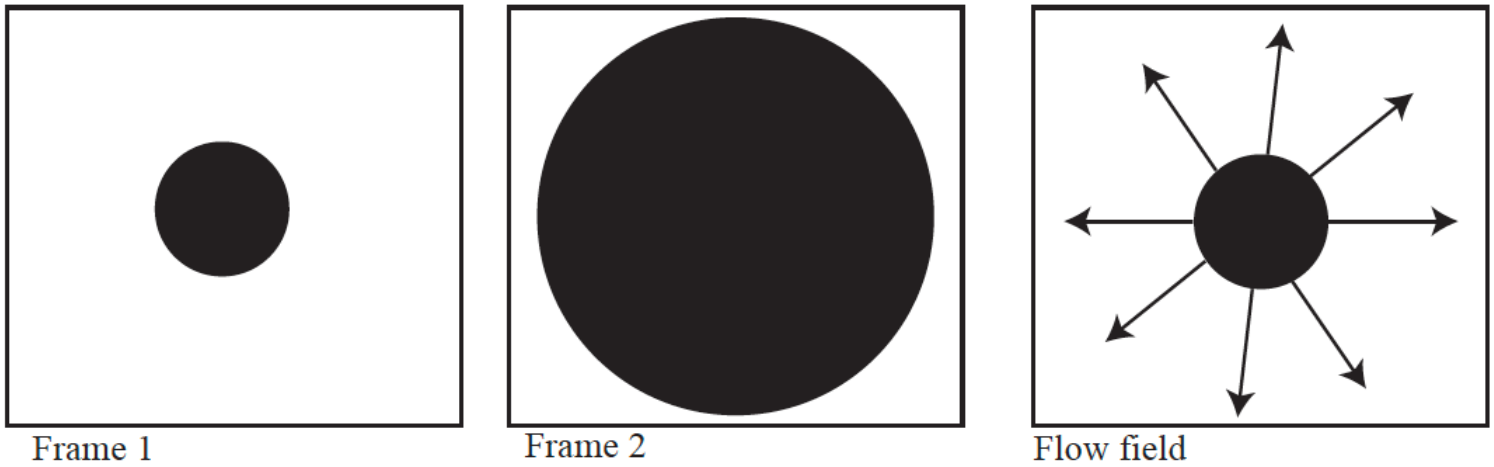






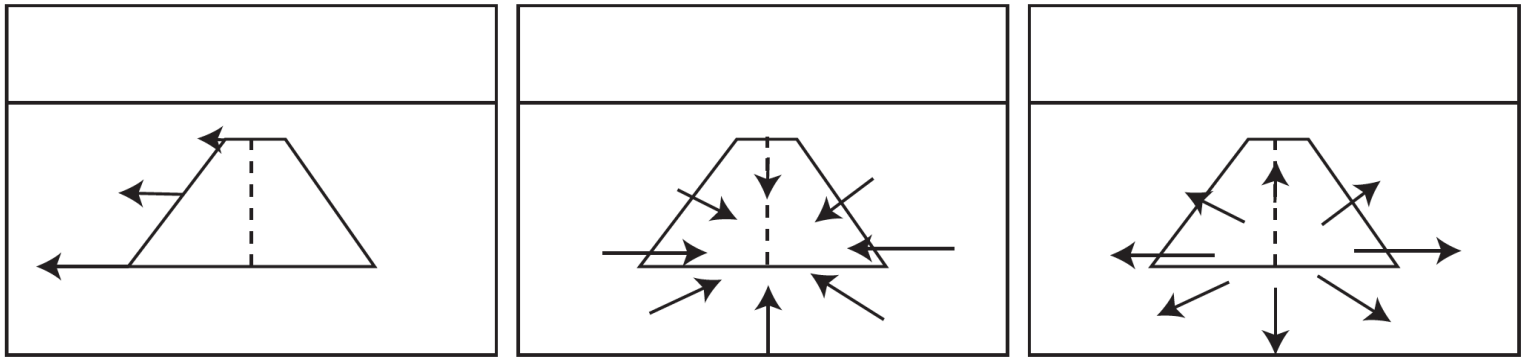


Flow fields reveal your motion

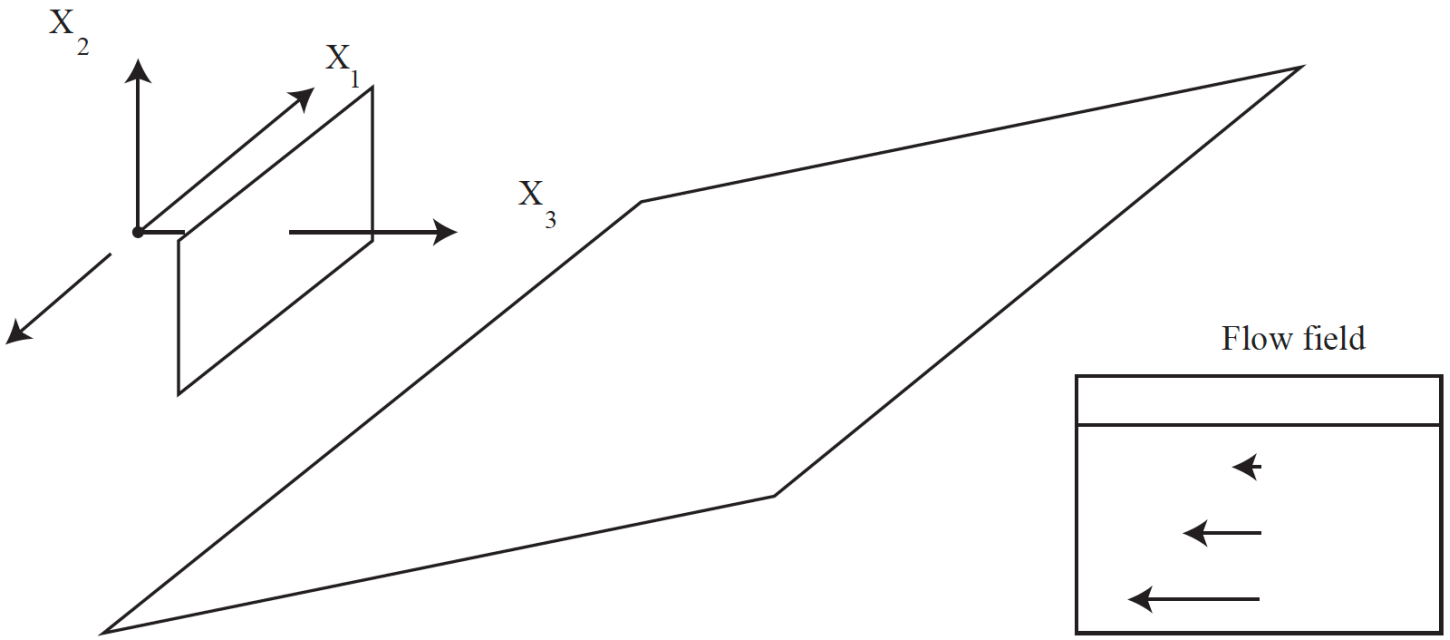


Frame 1 Frame 2 Flow field

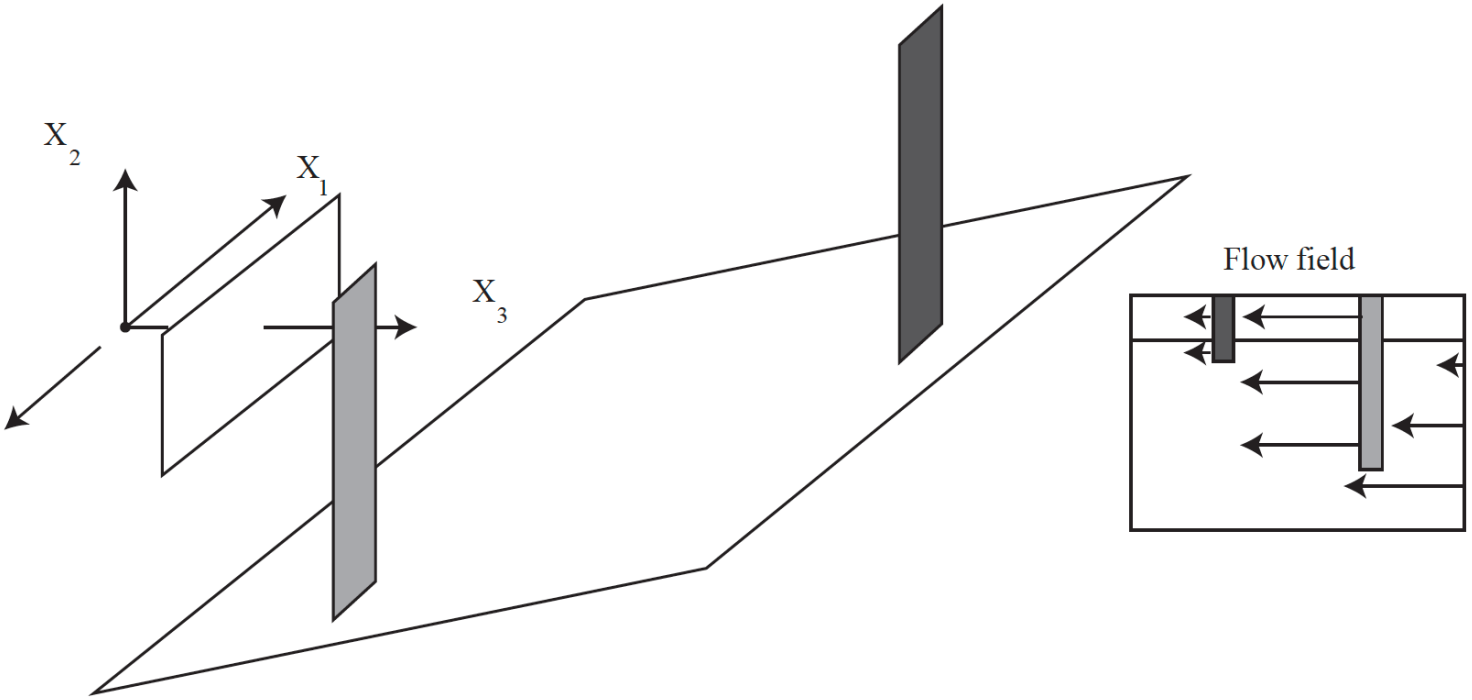
Flow fields reveal your motion



Flow fields reveal shape



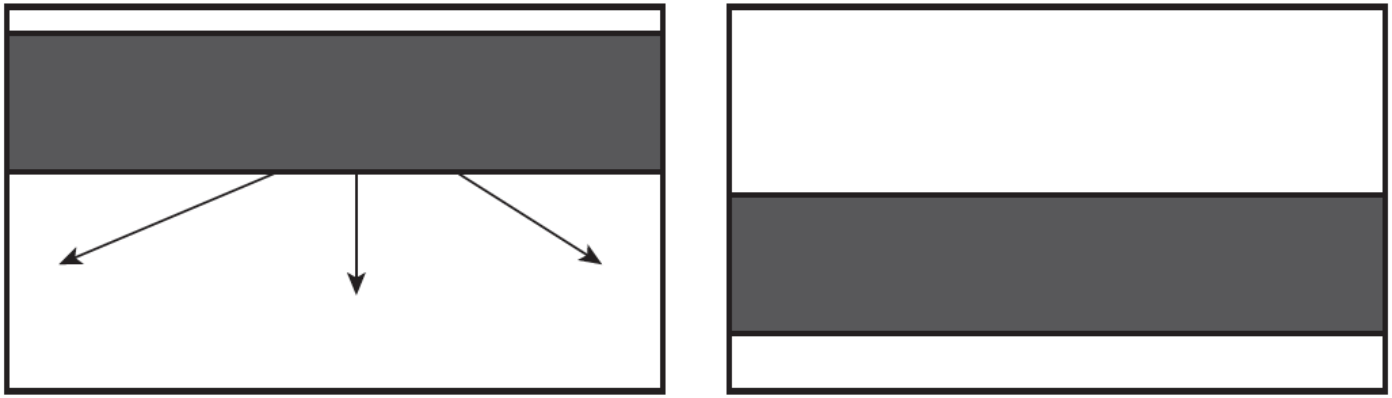
Flow fields reveal segmentation



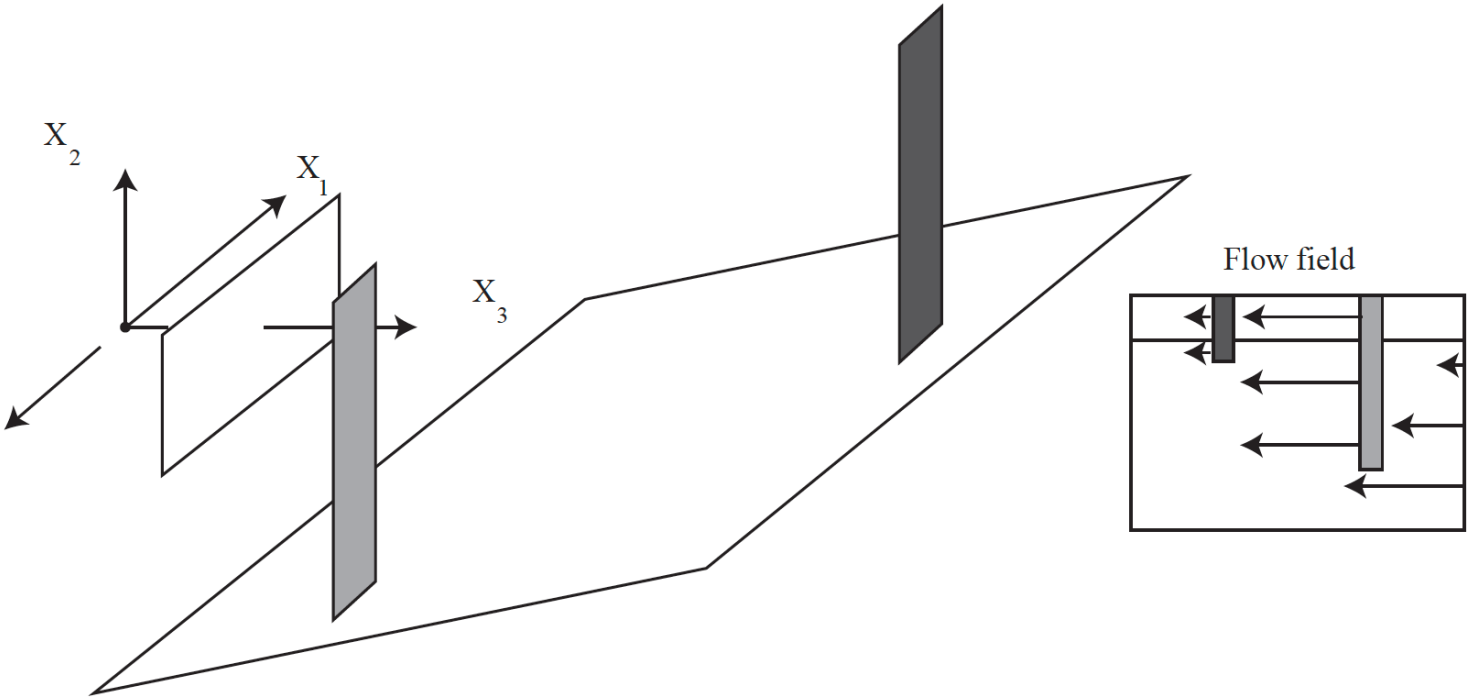
Why estimating flow is hard

- Flow is ambiguous at most points
  - the aperture problem
- You must interpolate
  - but there are discontinuities
- Flow is not defined everywhere
  - some things are visible only in one frame
- Some objects move by large amounts
- Motion blurring complicates estimation

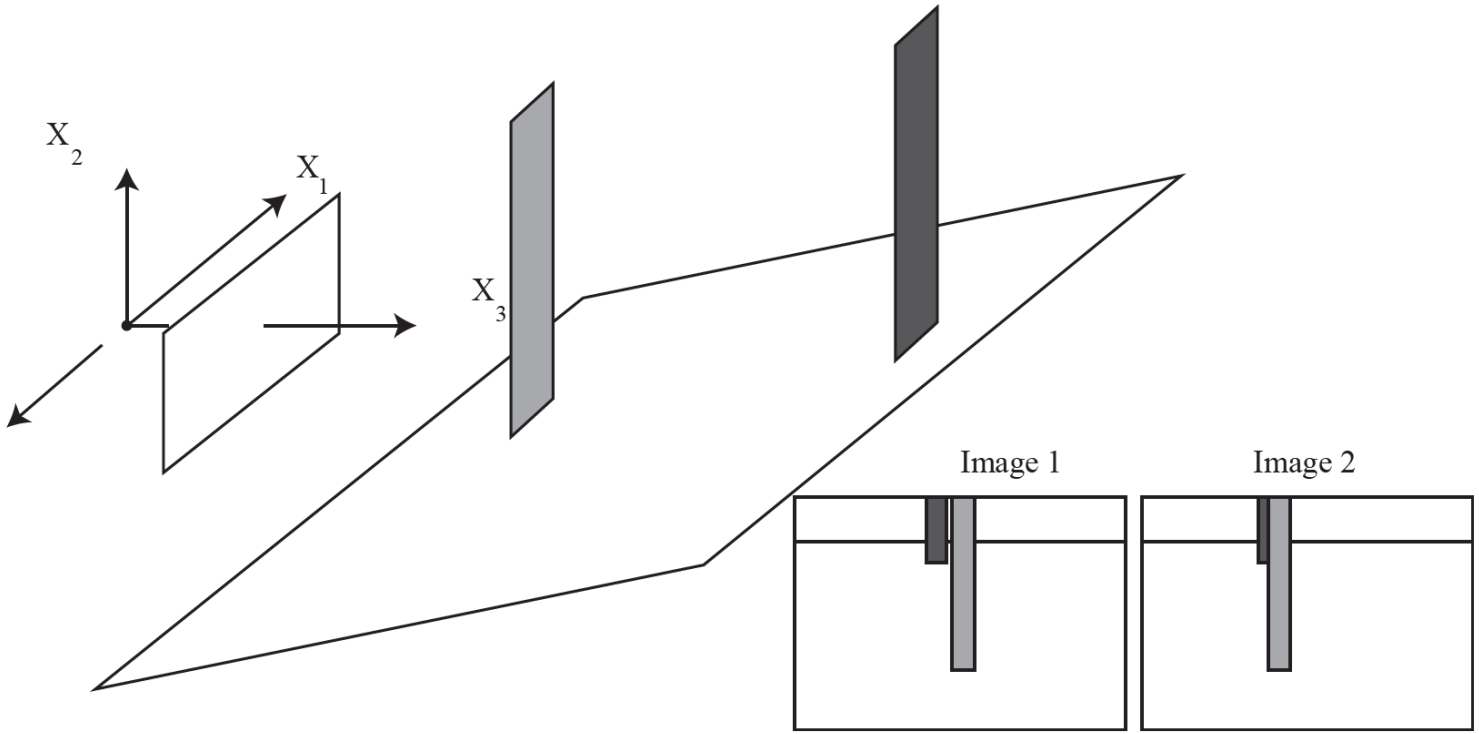
The aperture problem



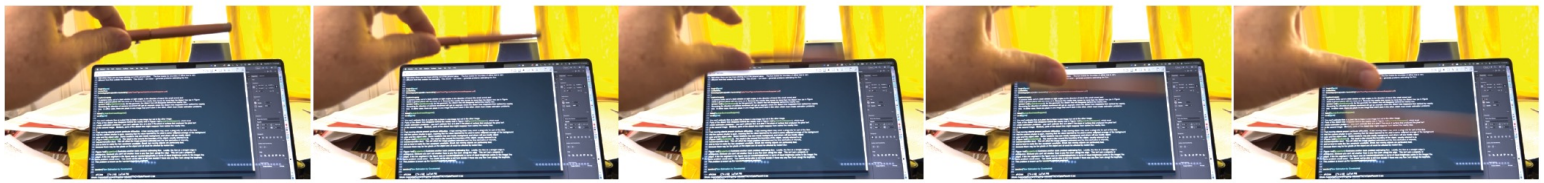
Flow is often discontinuous



Flow is not defined everywhere



Large movements and motion blur



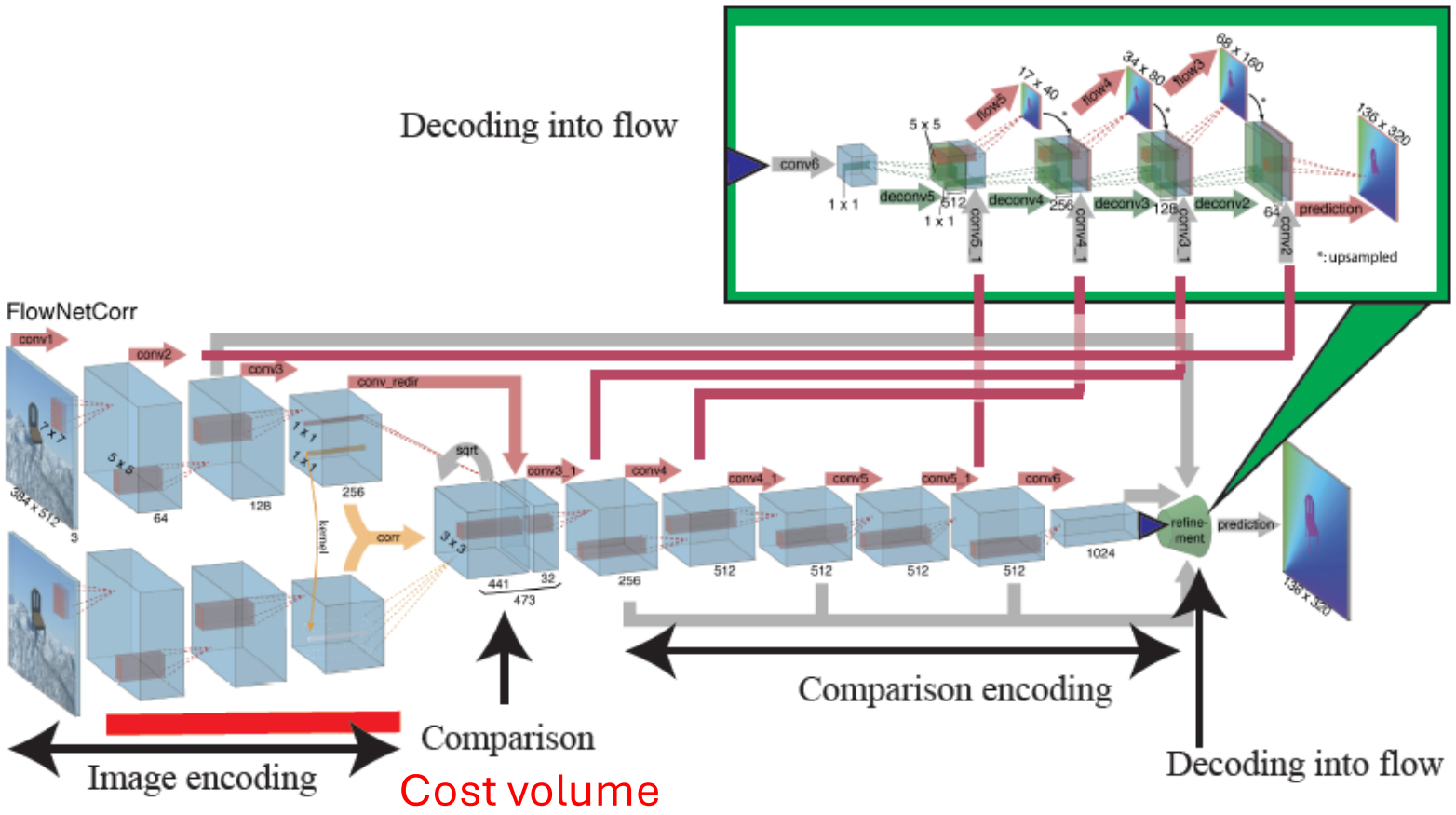
Learned flow and stereo estimates

- You might
  - pass two images into encoder/decoder network
  - hope to get stereo/flow at the output
  - This doesn’t work, because
    - you need to compare possibly distant locations
      - to identify correspondences
    - and impose structure
      - to avoid implausible flow/disparity fields
- Cost volume
  - a tensor that compares locations
- You can fuse single image depth information with stereo/flow

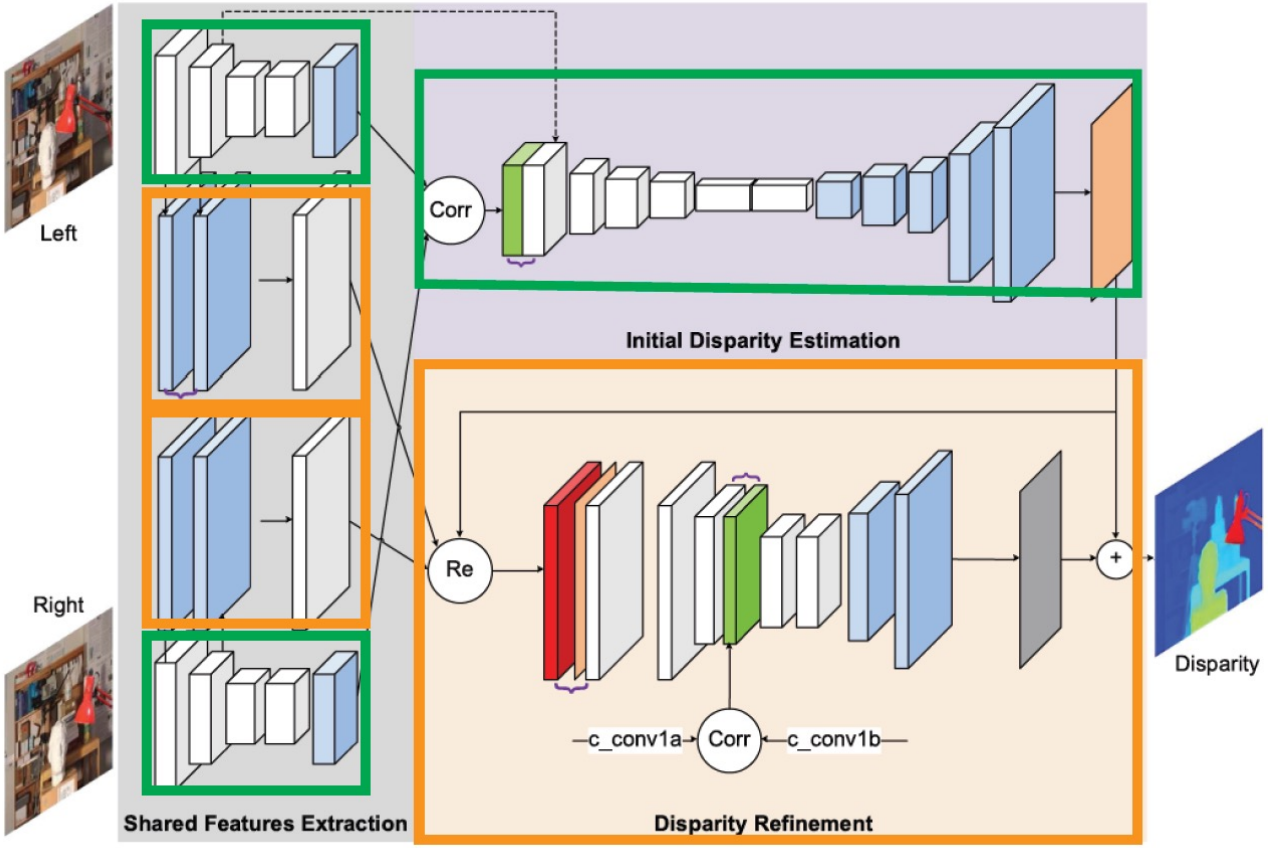
Flow and stereo are related, but different

- In each
  - you must find what matches between images
  - matches will have the same color (mostly)
  - The core computation is matching from image to image
- Stereo
  - two views of a static scene
    - epipolar geometry applies
    - strong constraints on what can match
- Optical flow
  - things could move
    - flow structure isn't obliged to meet epipolar constraints

Flownet


Decoding into flow
conv6
1 x 1
deconv5
512
1 x 1
deconv4
256
deconv3
128
deconv2
64
prediction
5 x 5
17 x 40
34 x 80
68 x 160
136 x 320
flow5
flow4
flow3
conv5_1
conv4_1
conv3_1
conv2
*: upsampled
FlowNetCorr
conv1
conv2
conv3
conv_redir
7 x 7
5 x 5
1 x 1
1 x 1
384 x 512
3
64
128
256
sqrt
kernel
corr
3 x 3
441
32
473
conv3_1
conv4
conv4_1
conv5
conv5_1
conv6
256
512
512
512
512
1024
refine-ment
prediction
136 x 320
Comparison encoding
Comparison
Cost volume
Image encoding
Decoding into flow

Stereo with cost volumes and residuals



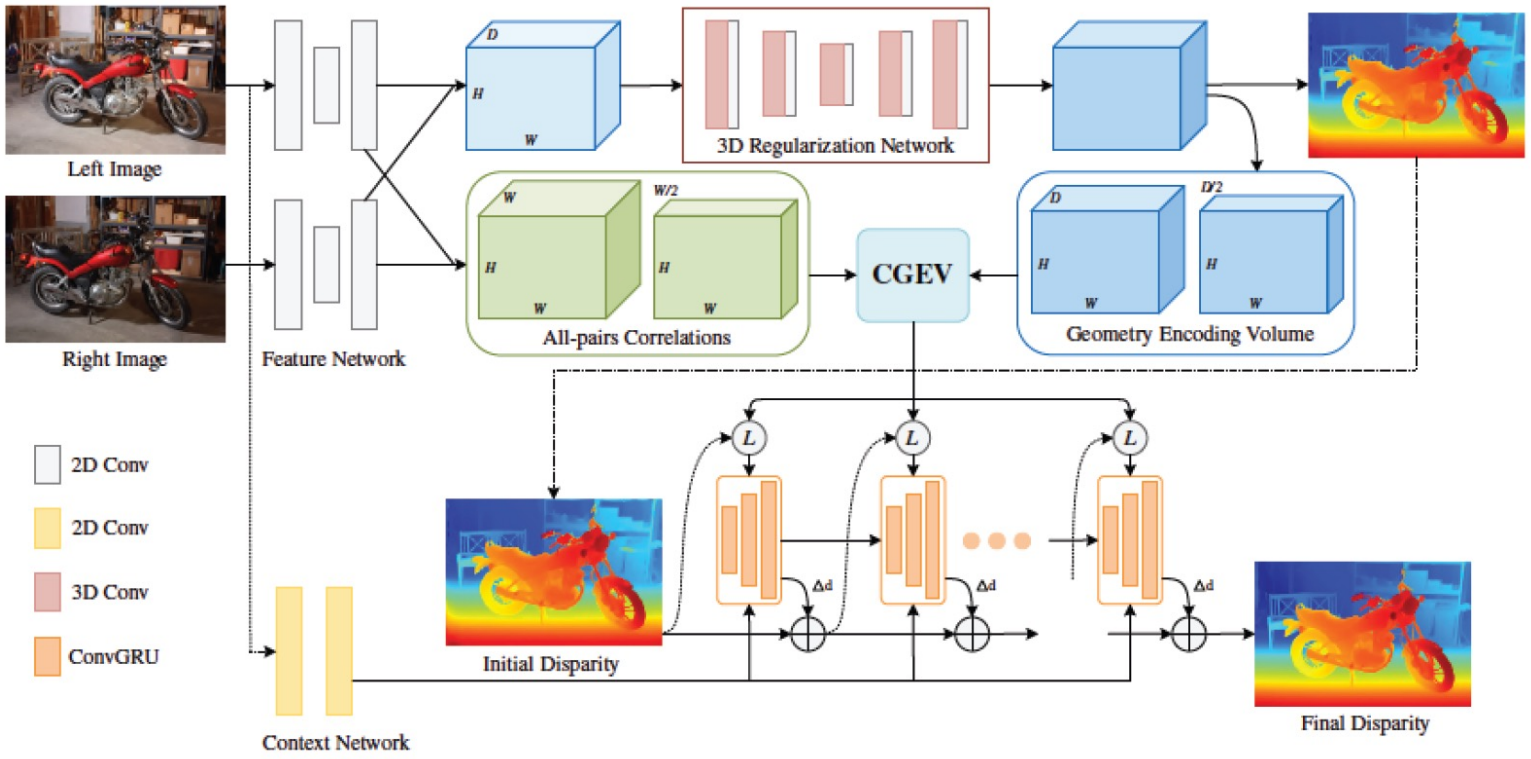


FIGURE 5.11: *The architecture of IGEV stereo (compare with Figure 5.10). IGEV improves over RAFT-Stereo by computing an estimate of 3D geometry (***top left** *– compare single image depth maps), encoding that, and using it during the iterative reestimation procedure.* Image credit: *Figure 3 of* Iterative Geometry Encoding Volume for Stereo Matching, *by G. Xu, X. Wang, X. Ding and X. Yang*

Estimating a fundamental matrix

- If you know 8 (7) correspondences
  - you can get fundamental matrix
  - but you don’t know exactly
- Strategy (RANSAC):
  - use interest point descriptors to find matches (most, but not all, are right)
  - iterate:
    - choose 8 (7) at random
      - fit FM using optimization problem
      - check what others agree
  - take the best
- Care in formulating the optimization problem is rewarded!

Fundamental vs essential matrix

- Fundamental matrix is a geometric concept
  - true whatever the coordinate system
- Essential matrix
  - what happens in coordinates

The essential matrix

Remember this: *Given two cameras related by a rotation $\mathcal{R}$ and a translation $\mathbf{t}$, the essential matrix is*

$$\mathcal{E} = [\mathbf{t}]_X{}^T \mathcal{R}.$$

A point $\mathbf{X}$ in 3D projects to $\mathbf{x}_1$ on camera 1's image plane in camera 1's coordinate system. The same point projects to $\mathbf{x}_2'$ on camera 2's image plane in camera 2's coordinate system. For any $\mathbf{X}$ not on the baseline,

$$\mathbf{x}_1^T \mathcal{E} \mathbf{x}_2' = 0$$

- it follows from fundamental matrix that something like this should be there; but here we have an expression relating it to camera transformation

Estimating the essential matrix

Procedure: 32.5 *Estimating the essential matrix*

Use procedure 32.3 to estimate the fundamental matrix, then use the camera intrinsics to compute

$$\mathcal{M} = \mathcal{K}_1^T \mathcal{F} \mathcal{K}_2.$$

From $\mathcal{M}$ compute the SVD to obtain $\mathcal{U}$, Σ and $\mathcal{V}$. Write $\Sigma_e = \text{diag}(1, 1, 0)$. Then the estimate of the essential matrix is

$$\hat{\mathcal{E}} = \mathcal{U} \Sigma_e \mathcal{V}^T.$$

Monocular visual odometry - I

- A calibrated camera
  - views a static scene,
  - moves,
  - views again

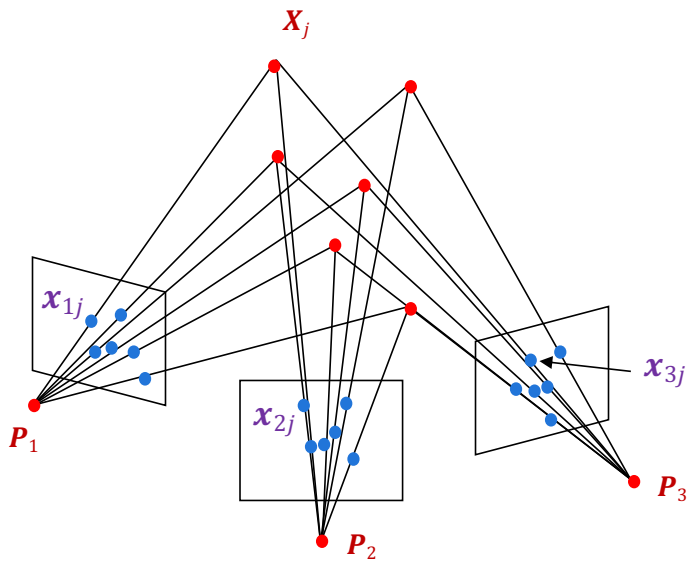
- Q: how did it move?
  - We care: this allows us to recover movement from single cameras
  - We should be able to tell
    - Recall epipoles etc are quite informative about movement

Monocular visual odometry - II

- Recovering how it moved:
  - Fit a fundamental matrix (above)
  - From it, recover essential matrix (above; calibrated)
  - From that, recover rotation, translation
    - out of the form of the matrix
      - details are delicate
    - Fact:
      - you can get direction but not scale of translation

Core problem

- Given:
  - m images of n fixed 3D points
    - $\boldsymbol{x}_{ij} \cong \boldsymbol{P}_i \boldsymbol{X}_j, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n$
  - with known correspondences
- Estimate:
  - m projection matrices $\boldsymbol{P}_i$
  - n 3D points $\boldsymbol{X}_j$



Cases:

- Offline
  - work with all images
  - structure from motion
    - underlying Q:
      - how to make best use of all information
    - roughly familiar structure
      - set up huge optimization problem by constructing start point
      - efficiency is now a serious issue
- Online
  - build best estimate available up to time t
  - SLAM
    - simultaneous localization and mapping
    - build a map at the same time as determining where you are
    - new Q:
      - how do you make best use of all information up to now
      - efficiency often crucial
- Hybrids are quite usual, and important

Applications

- Enable inspection in hard to reach areas with drone photos and 3D reconstruction
- Create 3D model from images
- Provide tools to inspect on images and map interactions to 3D

Source: D. Hoiem

Applications: where is my drone?

- Highly relevant

- Taste prevents the use of news pictures…

Structure from motion ambiguity

- Transform the scene using a transformation $\boldsymbol{Q}$
- Apply inverse to the camera matrices
- The image observations do not change:

$$\boldsymbol{x} \cong \boldsymbol{PX} = (\boldsymbol{PQ}^{-1})(\boldsymbol{QX})$$

Structure from motion ambiguity

- Cases:
  - nothing is known about cameras – Q is projective tx
  - camera calibrations known – Q is scaled Euclidean

Recall: Two calibrated cameras

- Obtain interest points and some matches
- Obtain fundamental matrix
  - as before, using matches
- Obtain essential matrix
  - as before, using calibration
- Obtain rotation, translation up to scale
  - as before, using odometry
- Triangulate

Three calibrated cameras

- Do two cameras for c1, c2
  - reconstruct all points in both c1 and c2
- Now figure out c3
  - using reconstructions - calibration
  - There is only one missing scale!
- You can now insert more points into 3D!
  - anything in c1 and c3 but not c2
  - anything in c2 and c3 but not c1
  - just more triangulation

Three calibrated cameras

- You can now insert more points into 3D!
  - anything in c1 and c3 but not c2
  - anything in c2 and c3 but not c1
  - just more triangulation

3 or more calibrated cameras

- Issue:
  - in what order should you insert cameras?

- Online:
  - no issue
    - in the order you get them, BUT
      - how do you get best estimate of points, cameras from data?
- Offline
  - major issue

Representative SFM pipeline

N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. SIGGRAPH 2006
http://phototour.cs.washington.edu/

SFM software

- Bundler
- OpenSfM
- OpenMVG
- VisualSFM
- COLMAP
- See also Wikipedia's list of toolboxes

Core idea

- Observe the world from a moving platform
- Build:
  - a map of the world
  - an estimate of pose in that map
- Online:
  - notice that new observations update:
    - estimate of pose
    - estimate of map
- Filtering problem

Some more details

- With two cameras and some calibration:
  - we can recover the position of 3D points
    - in the vehicle's coordinate system
      - see triangulation movie, etc.

- Together with an EKF, we can use this to recover
  - points in world coordinates (a map)
  - vehicle location

- In fact, we can do all this with one camera
  - with some minor care

Variants

- More complex vehicle dynamics
- Different measurement process
  - one camera
  - LIDAR
  - SONAR (v. hard)
  - photo-consistency constraints (direct methods)
- Multiple processes at different timescales
  - fast:
    - estimate configuration, landmarks
  - slow:
    - refine estimates (something like bundle adjustment)
    - take more local image data into account

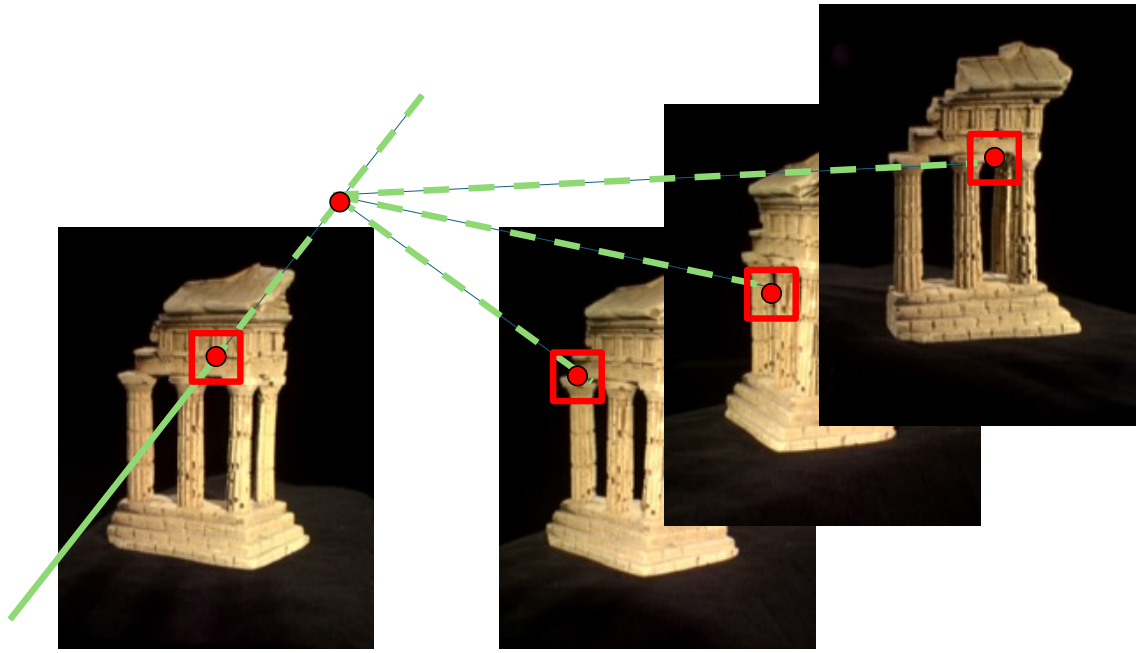
SFM is about scattered points

- Reconstruct a denser representation
  - from many pix
    - (or other stuff)
- Q:
  - in what form?
- Here:
  - Multiple depth maps, Point clouds
- Notes:
  - Voxel grids, Implicit functions, Density functions, Primitives
- Others
  - CSG rep’ns
  - NURBS
  - etc.

Options

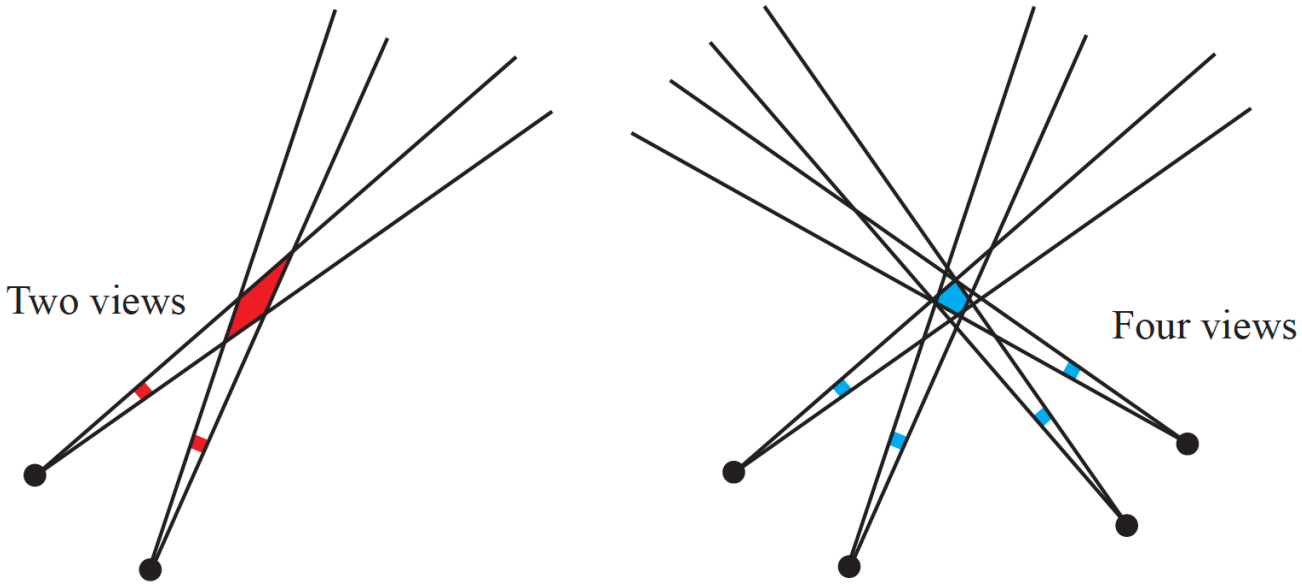
- Here:
  - Multiple depth maps
  - Point clouds
- Notes:
  - Voxel grids
  - Implicit functions
  - Density functions
  - Primitives
- Others
  - CSG rep’ns
  - NURBS
  - etc.

Multi-view stereo: Basic idea



Source: Y. Furukawa

Improved error estimates

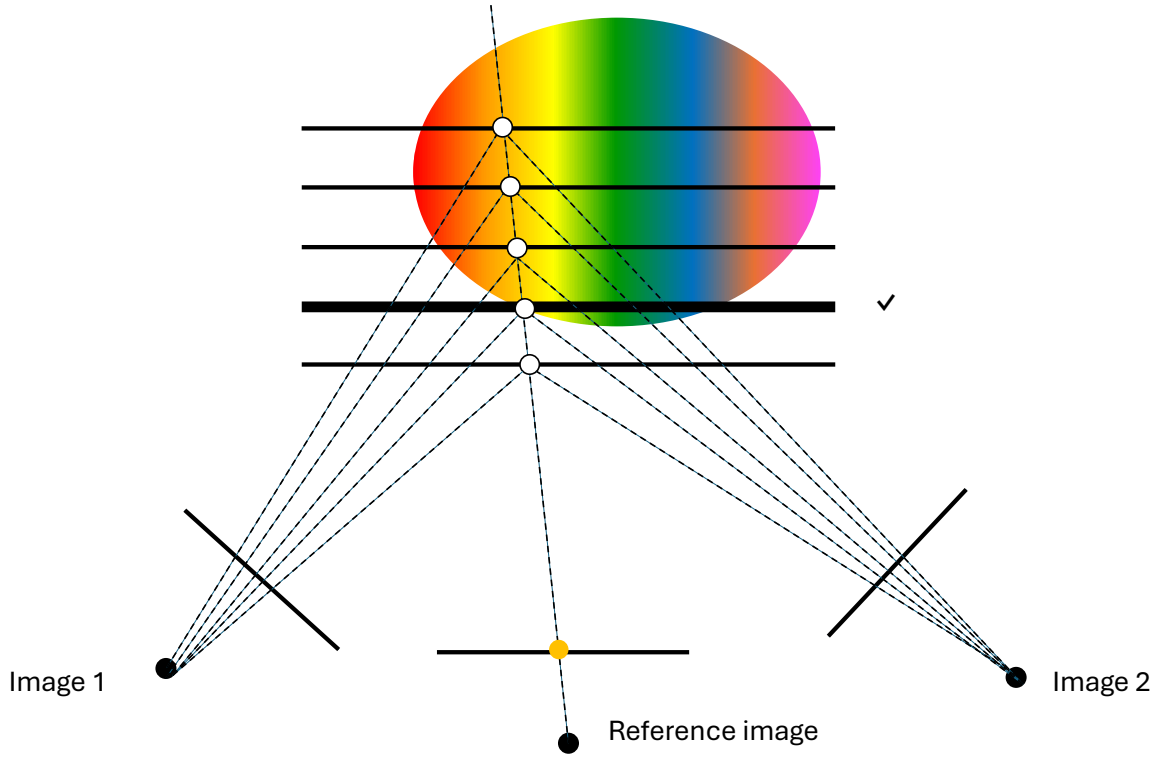


Rep'n: Multiple depth maps

- One per image
  - or per cluster of images
  - consistent

- Advantages
  - straightforward to get
  - known how to pass to a triangle mesh (if dense enough)
- Disadvantages
  - resolution issues (eg zooming camera)
  - can be tricky to render cleanly
  - some geometric calculations are hard
    - volume

Plane sweep stereo: Key idea

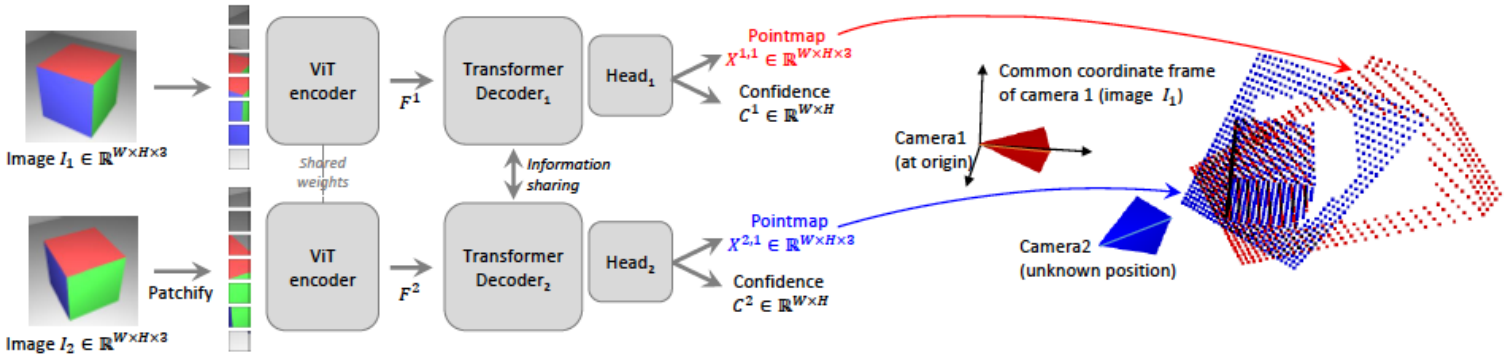


DepthAnything V3 depth maps

https://github.com/ByteDance-Seed/Depth-Anything-3

DUST3R – I

- Accept two images, make a point map for each



DUSt3R: Geometric 3D Vision Made Easy, Wang et al, 2024

Point maps

- one (3D) point per pixel
  - so WxHx3
- I1 is in canonical frame, both calibrated
  - point map at (i, j) with depth z(i, j) contains

$$\mathcal{K}^{-1} \begin{bmatrix} iz(i,j) \\ jz(i,j) \\ z(i,j) \end{bmatrix}$$

- point map for I2 is in I1 frame

Now train

- Simplest
  - accept image pair
  - produce pointmap
- Train w/ ground truth
  - recalling I2 pointmap is in I1’s frame
  - lots of data of form (image pair, depthmap pair)
- Loss:
  - scaled L2 regression loss between prediction, g.t.
  - weighted by confidence term

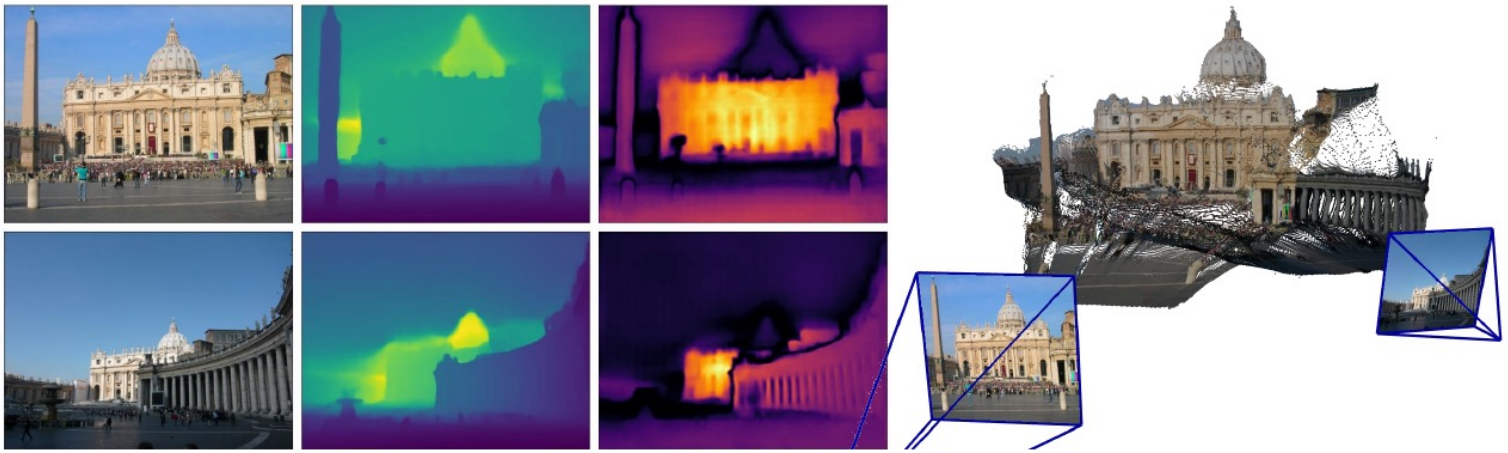


Image Depth Confidence Point cloud

DUSt3R: Geometric 3D Vision Made Easy, Wang et al, 2024

Downstream

- I1, I2 pixel correspondences
  - find nearest neighbors in pointmaps
- Camera intrinsics
  - pointmap is in I1's frame
    - easy optimization
- Relative camera pose
  - predict twice using I1, I2 then I2, I1
  - register the two predictions
    - first in I1's frame, second in I2's frame
- Absolute pose
  - register

More than two frames

- Like SFM pipeline:
  - find pairs with many overlapping points
  - reconstruct for every such pair
  - register reconstructions

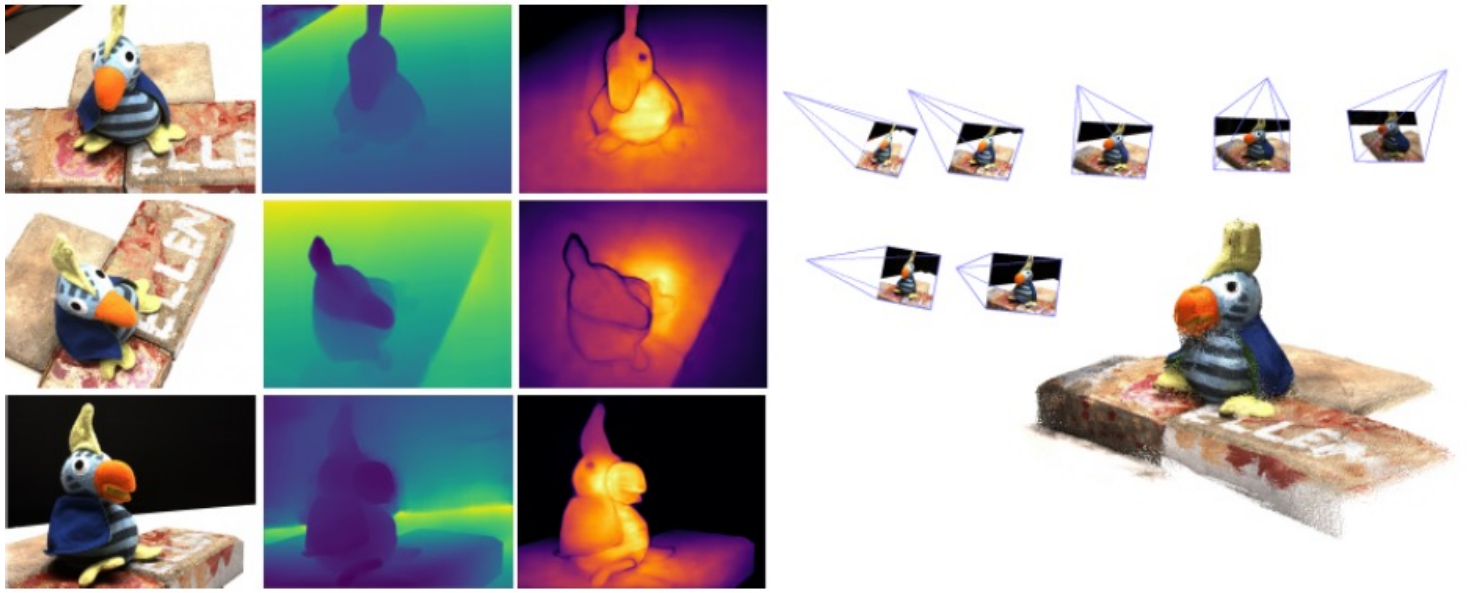


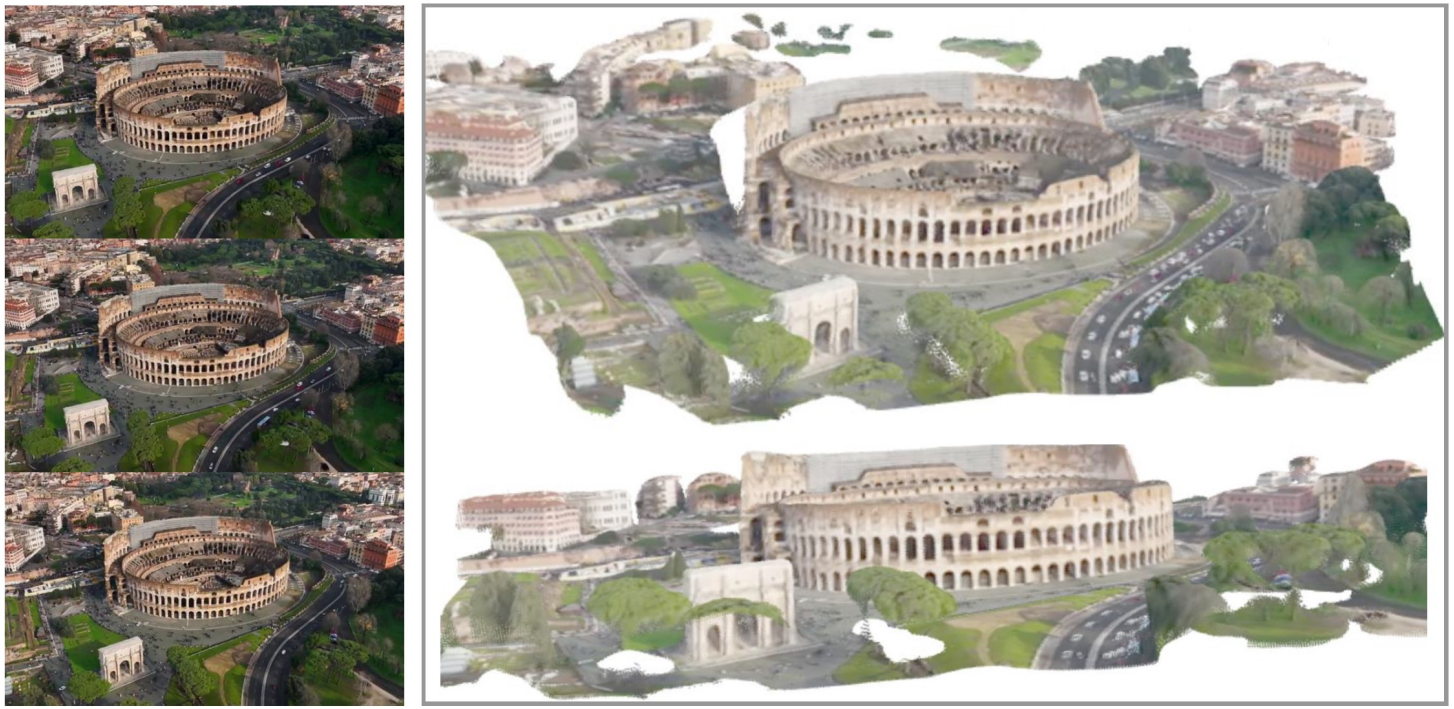
Image Depth Confidence Point cloud

DUSt3R: Geometric 3D Vision Made Easy, Wang et al, 2024

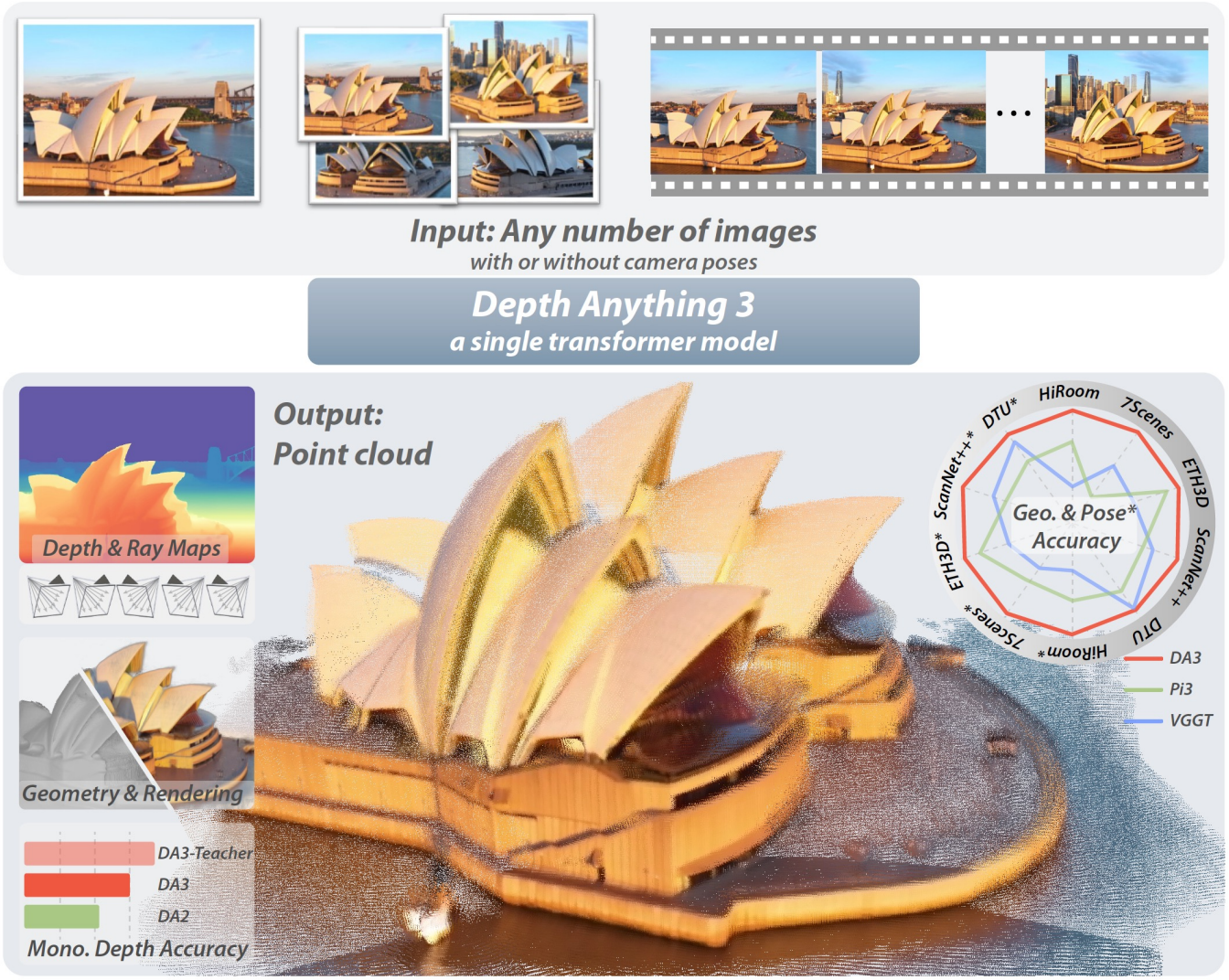
VGGT

https://vgg-t.github.io

VGGT



https://vgg-t.github.io

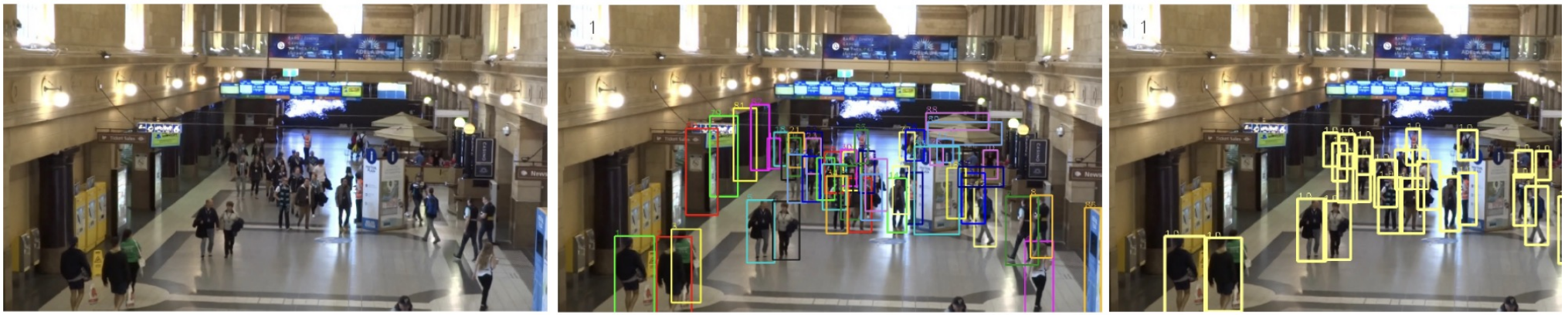


https://github.com/ByteDance-Seed/Depth-Anything-3

Tracking

- Join up instances of objects over time for long sequences
  - Despite various inconveniences
    - objects getting occluded and reappearing
    - objects deforming as they move
    - etc
- Applications:
  - obvious security stuff
  - what do people do?
    - improved architectural design
  - increasingly useful in grasp planning, etc.

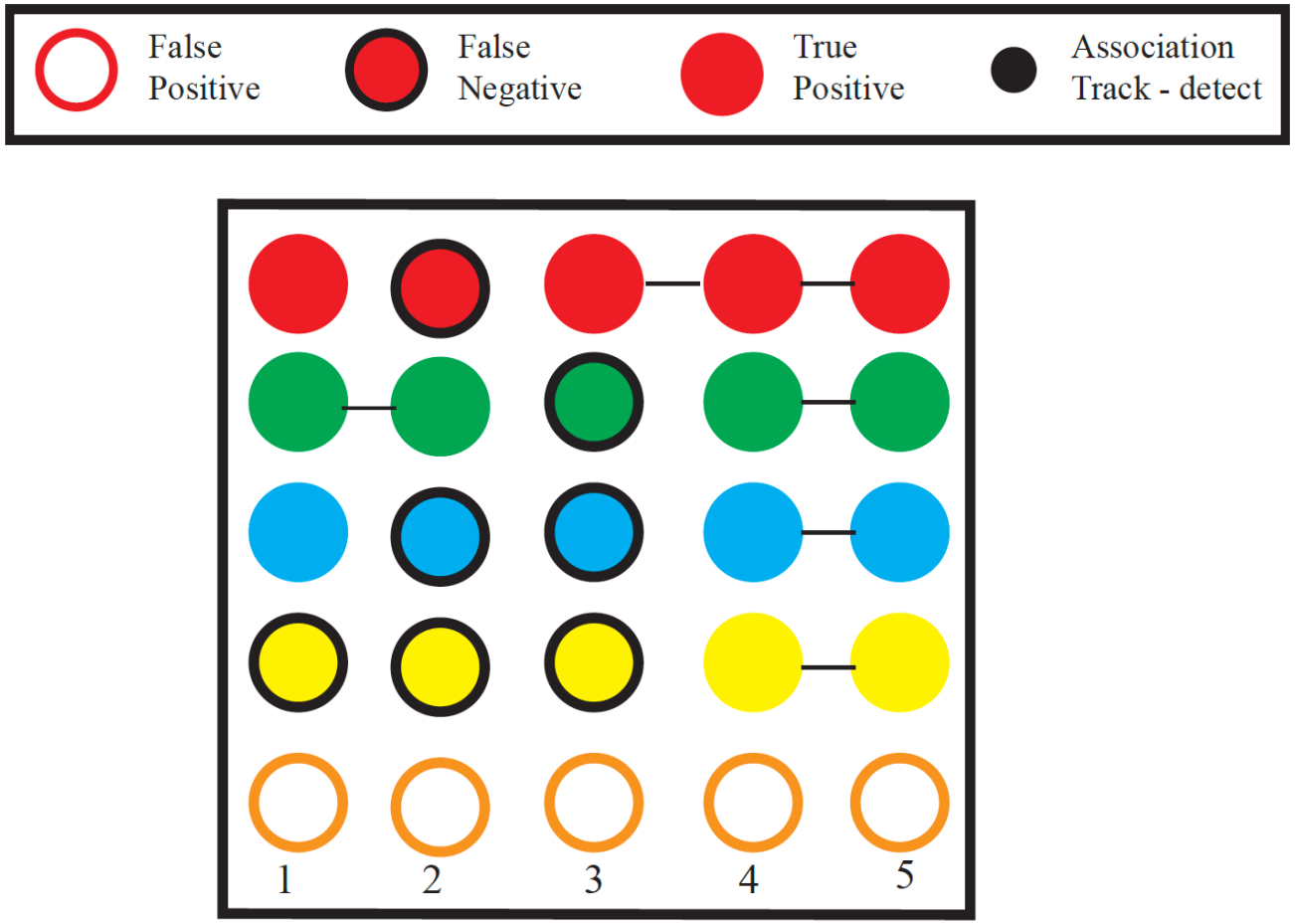
MOT-20 example



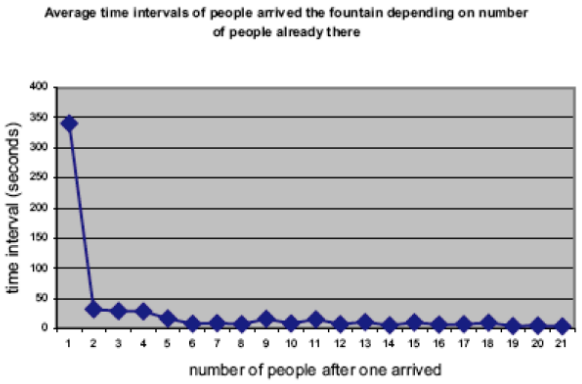
Tracking by detection

- Assume
  - a reliable detector
  - Link detects across time

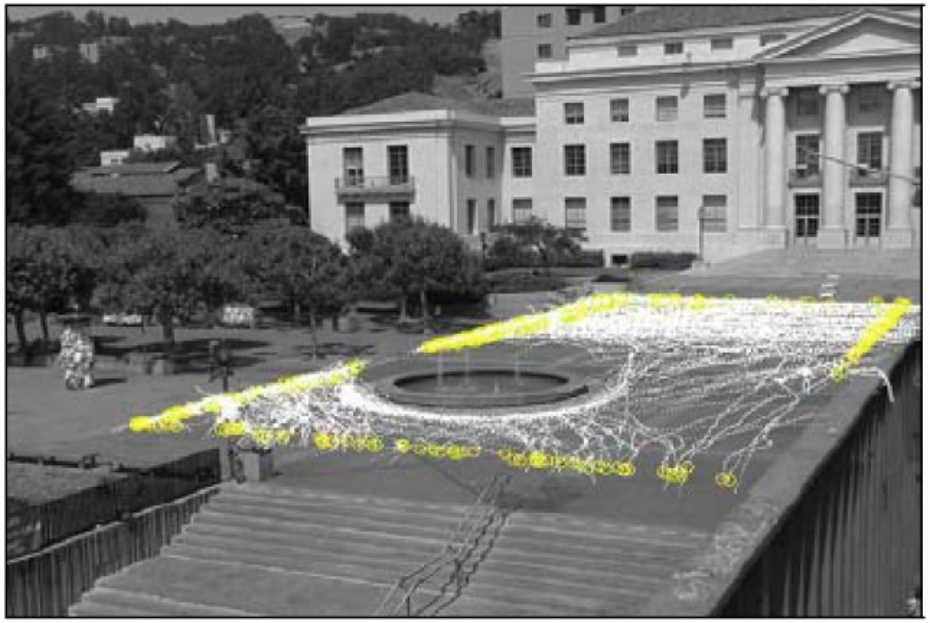
Simplest -> broken tracks



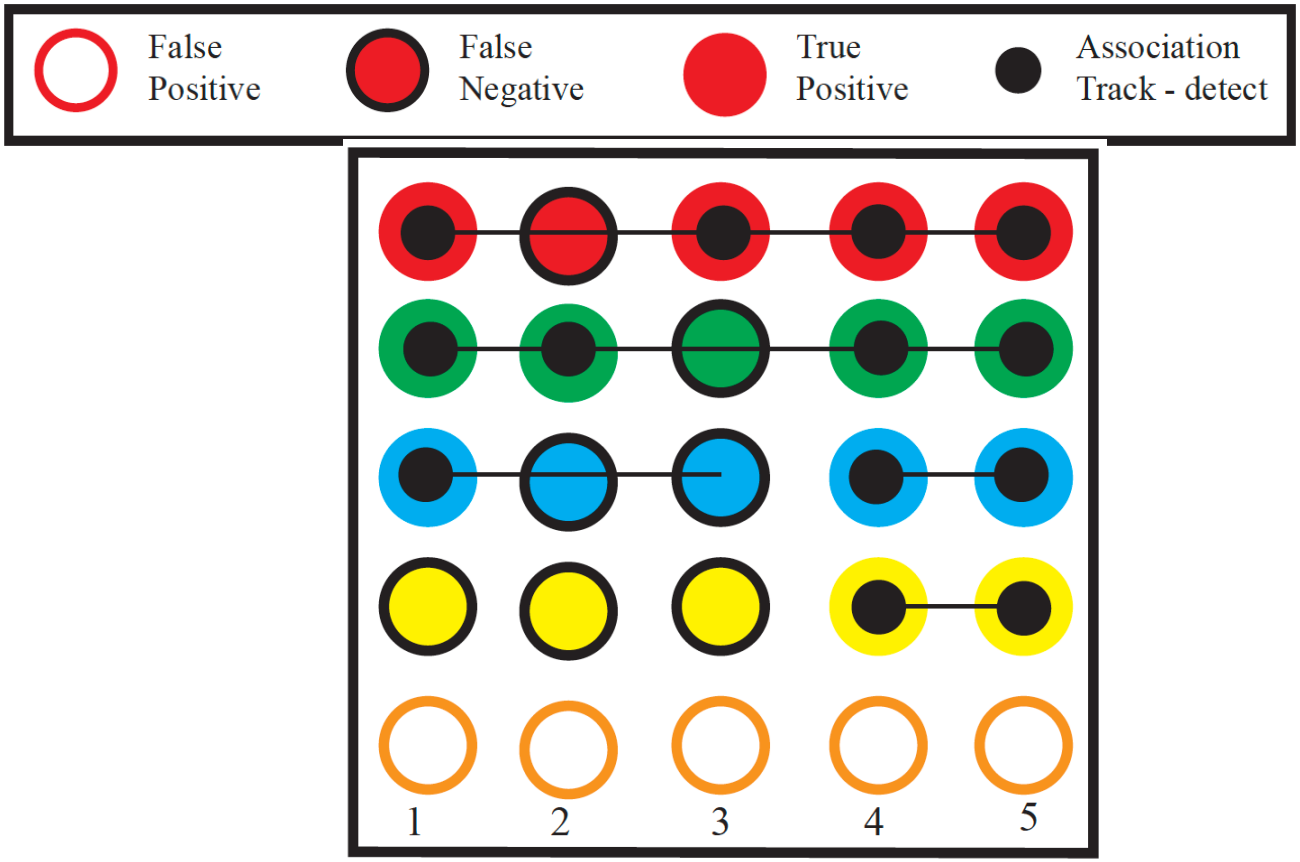
Point tracks reveal public behaviour



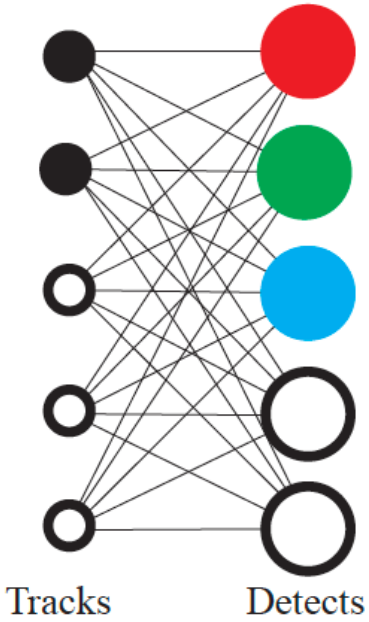
Yan+Forsyth, 04



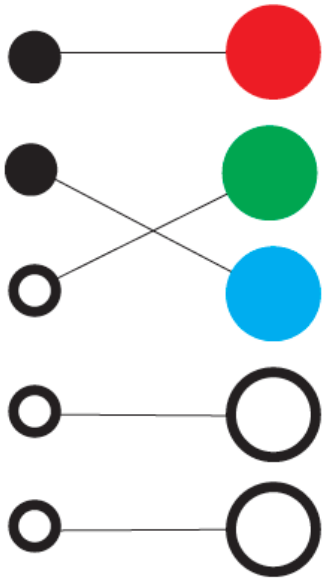
Abstract tracks



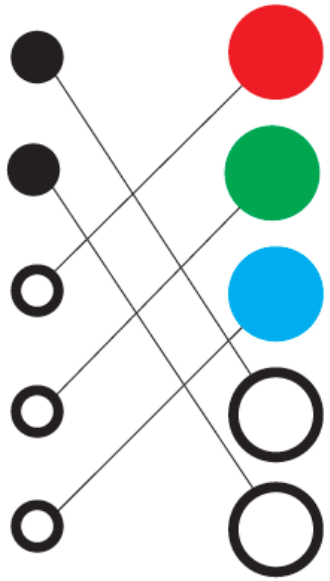
Correspondence



Bipartite graph



A



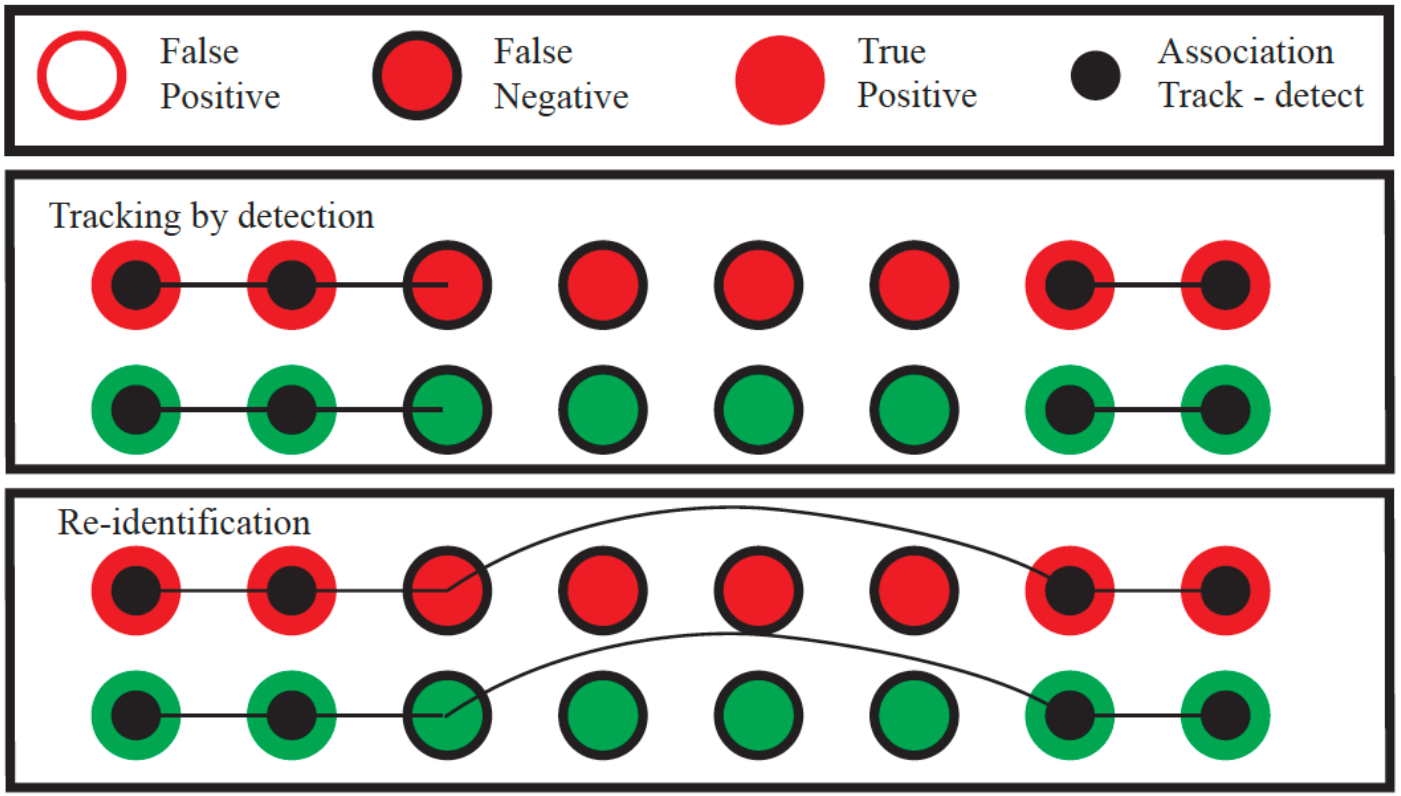
B

Use the Hungarian algorithm
Weights from appearance, dynamics

Appearance weights

- Distance between feature vectors
  - tracks could have multiple
    - memory
- Some test of distance
  - between predicted pos’n and true pos’n
    - 1-IOU has a good record

Reidentification



TbD variants- Exploit FasterRCNN

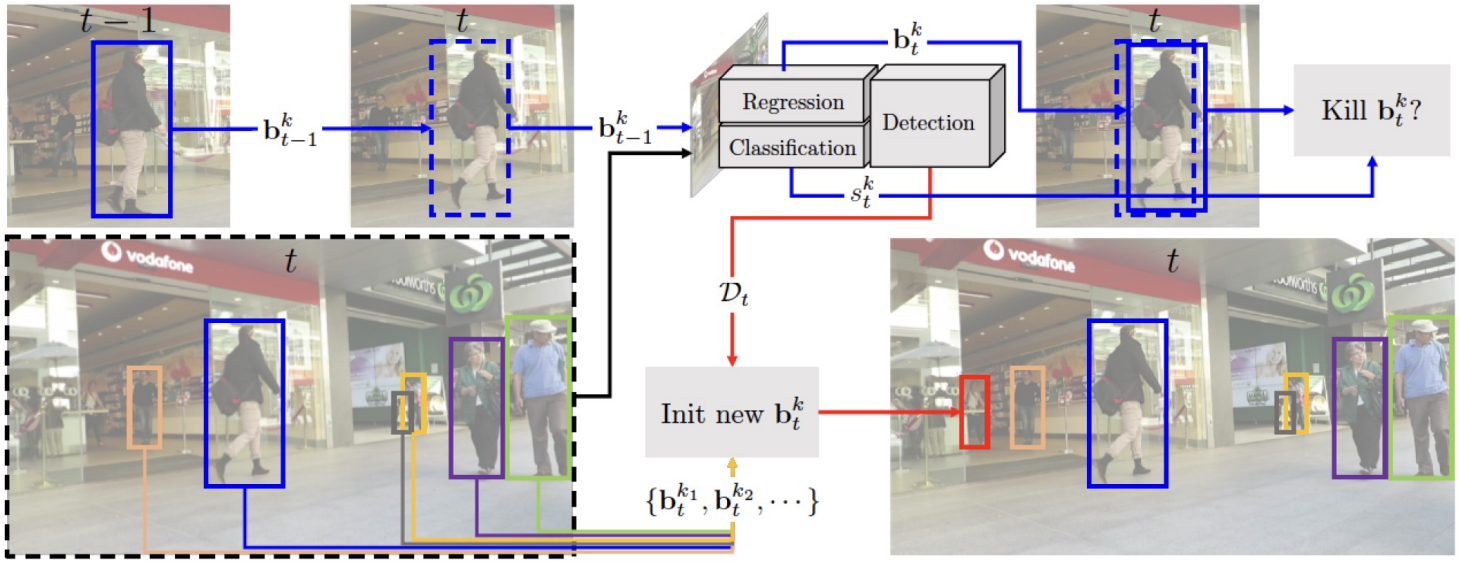
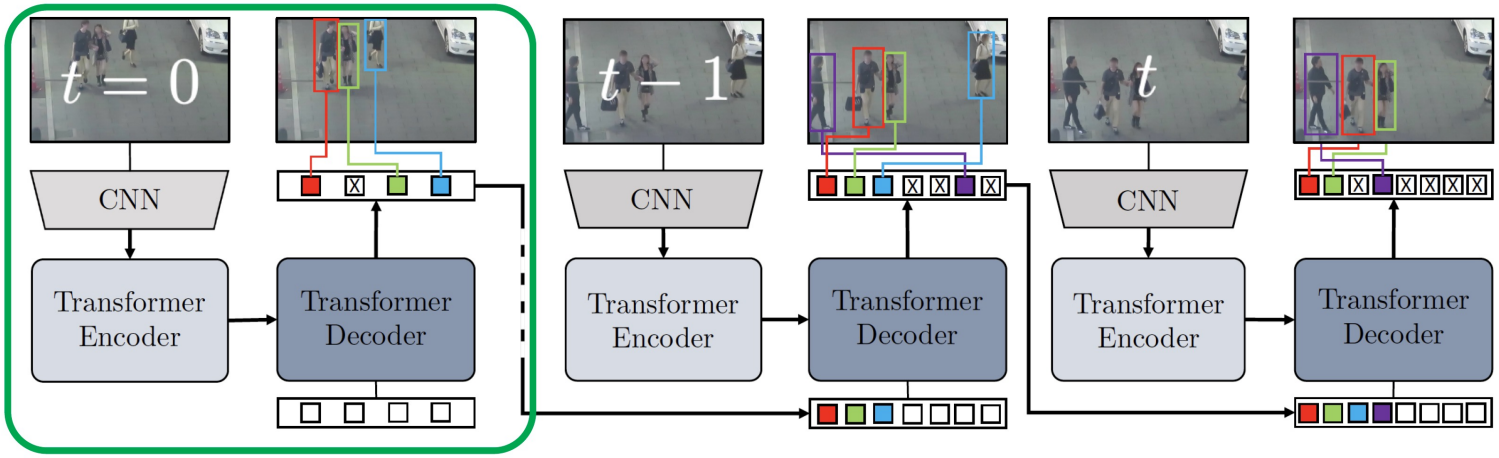


Figure 1 of "Tracking without bells and whistles", Bergmann et al, 2019.

TbD variants- Exploit DetR



of "Tracking without bells and whistles'TrackFormer: Multi-Object Tracking with Transformers", Meinhardt et al, 2021

TbD variants – open world

- Use RPN to detect objects
  - adjust RPN score
- Use appearance match for correspondence

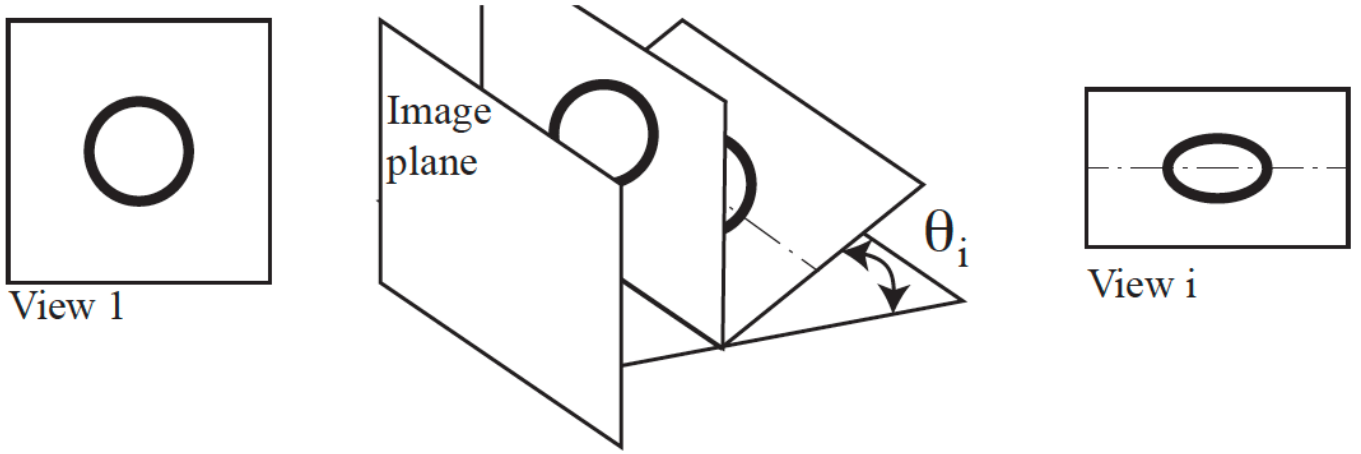
across time is straightforward, details in text. Image credit: *Figure 1 of "Opening up OpenWorld Tracking", Liu et al, 2022.*

TbD variants – tracking segments

shows the correspondence from frame to frame. Image credit: *Figure 1 of "Tracking Anything with Decoupled Video Segmentation", Liu et al, 2022.*

Why not points?

- Idea
  - apply TbD recipe to interest points
- Obstacle
  - appearance match using SIFT is a problem
  - Foreshortening!



Particle video

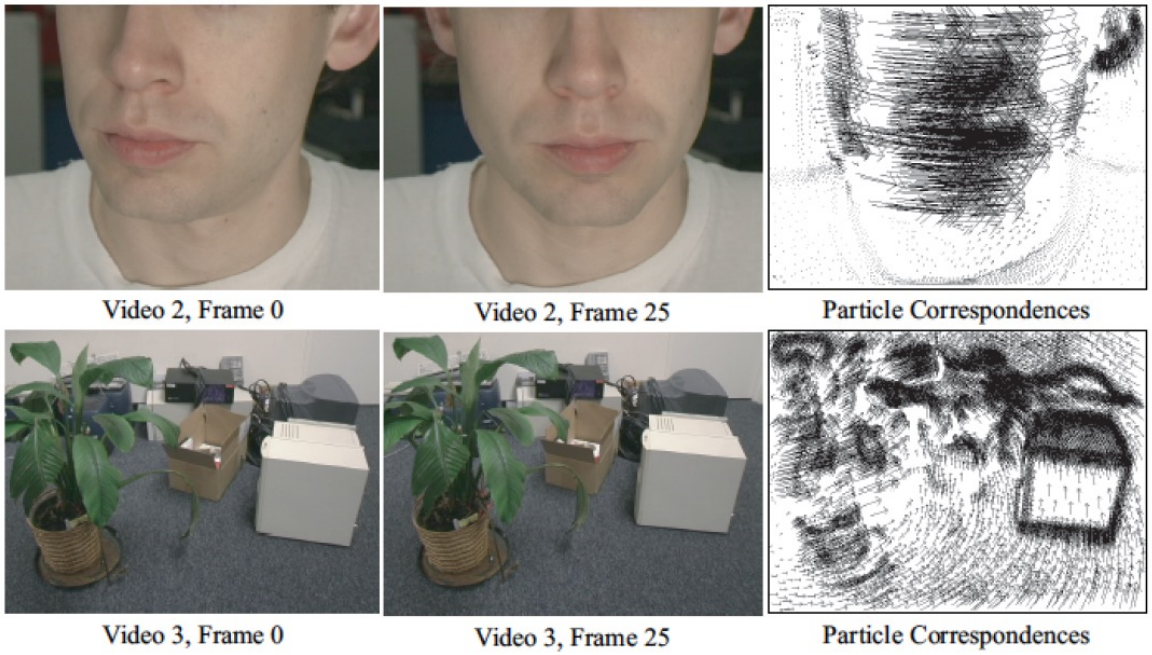


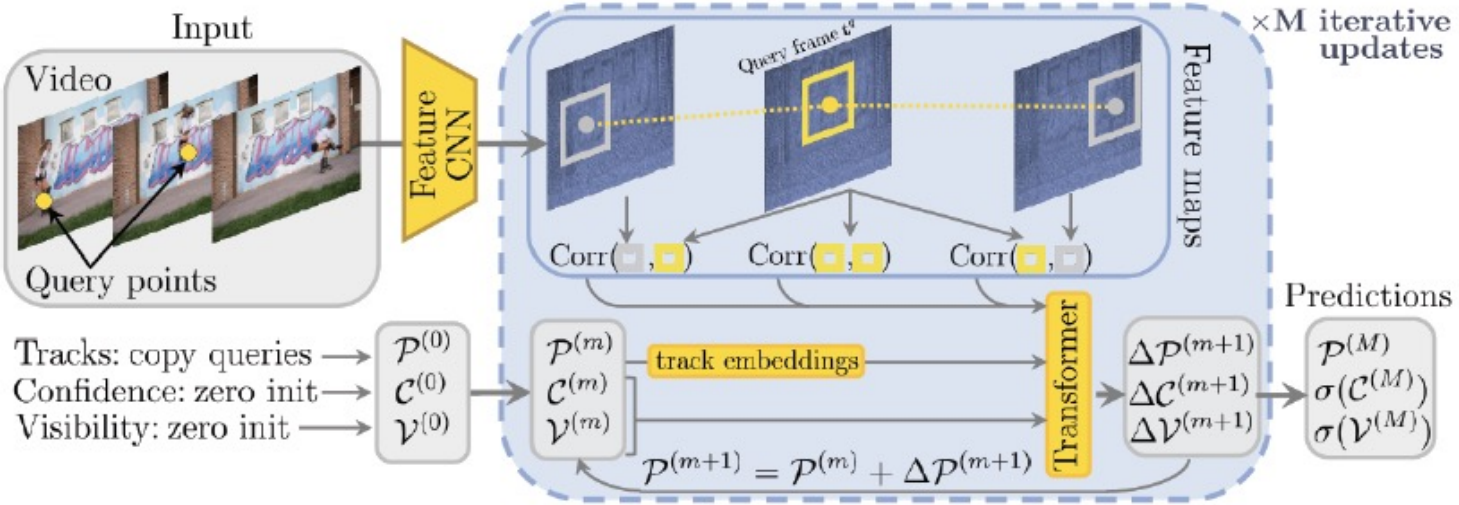
Figure constructed using parts of Figure 6 of "Particle Video: Long-Range Motion Estimation using Point Trajectories", Sand and Teller, 06.

TAPIR

- Make coarse point tracks
  - refine across space/time
- Predict visibility and confidence at each pt

frame. Figure constructed using parts of Figure 4 of "TAPIR: Tracking Any Point with per-frame Initialization and temporal Refinement", Doersch et al, 2023

CoTracker3



parts of Figure 2 of "CoTracker3: Simpler and Better Point Tracking by Pseudo-Labelling Real Videos", Karaev et al, 2024

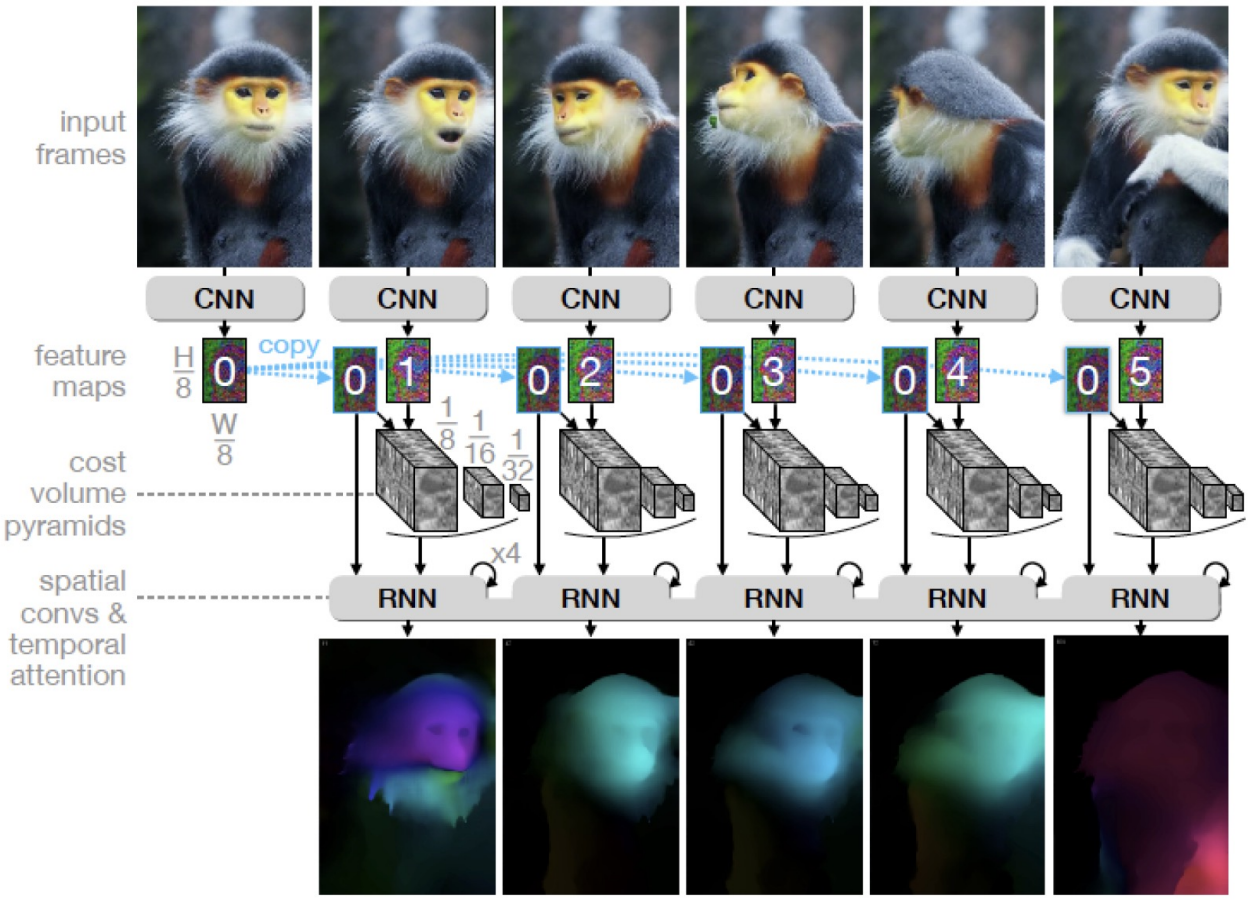


FIGURE 1.16: *Figure constructed using parts of Figure 2 of "AllTracker: Efficient Dense Point Tracking at High Resolution", Harley et al, 202X*