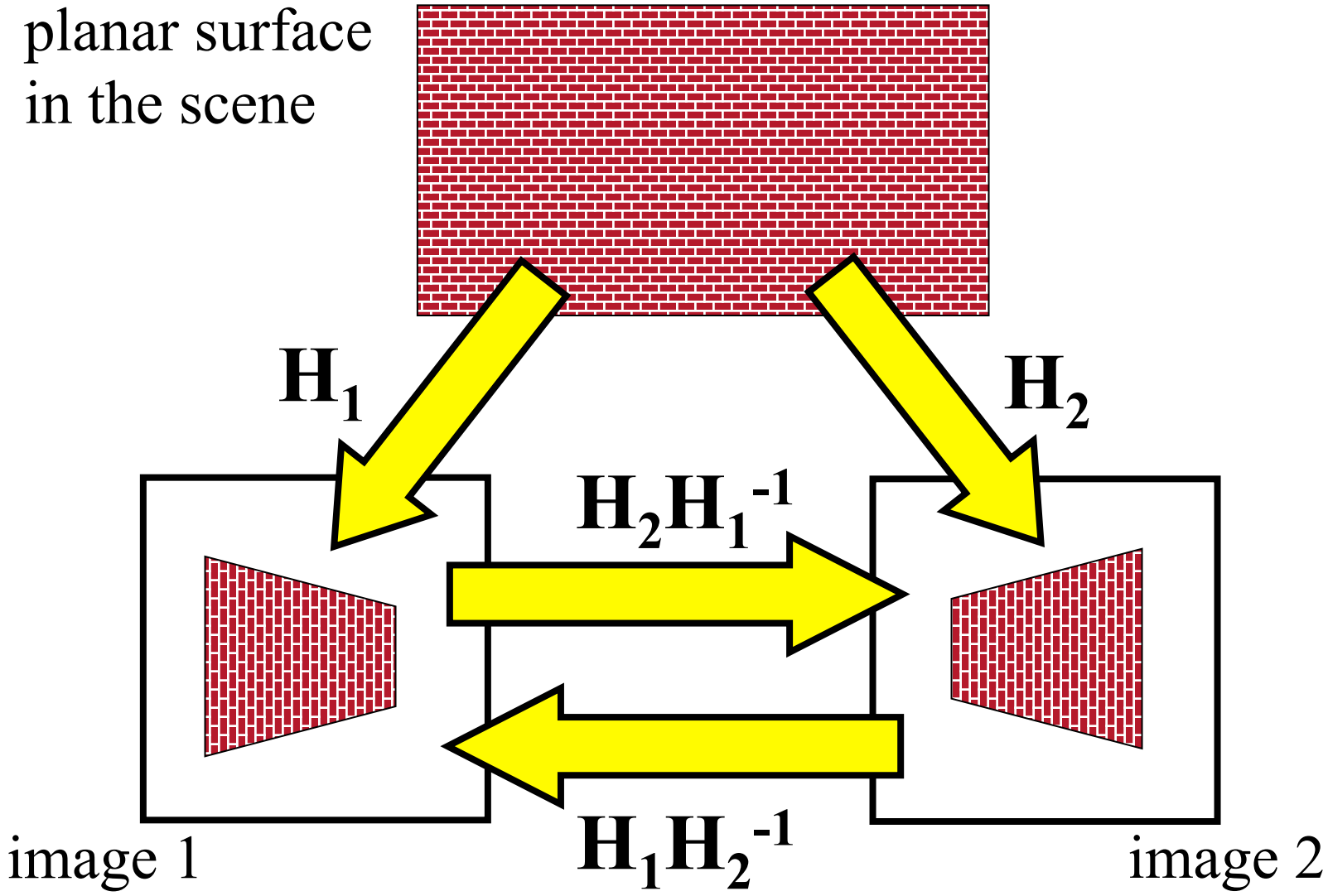
Robert Collins
CMPEN454

Planar Stabilization

Robert Collins
CMPEN454

Images of Planar Surfaces

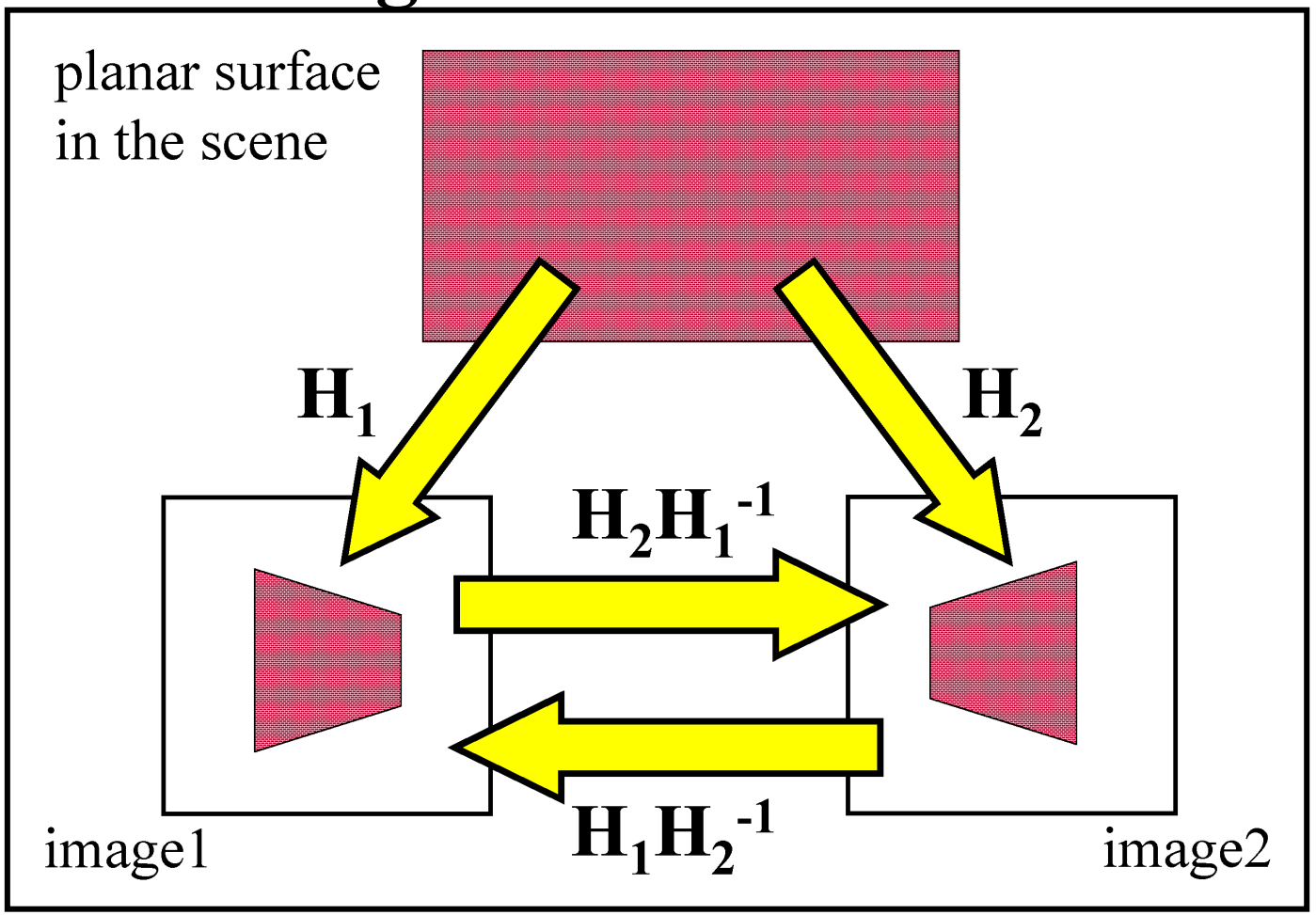


Robert Collins
CMPEN454

Ghosting Example

Mosaic

Images of Planar Surfaces



Note that mapping from one image of a plane to another is also just a 3x3 homography H.

Application: Two-frame Stabilization

- Stabilizing aerial imagery
  - find corners in two images
  - hypothesize matches using NCC
  - do RANSAC to find matches consistent with an affine transformation
  - take the inlier set found and estimate a full projective transformation (homography) using least squares.

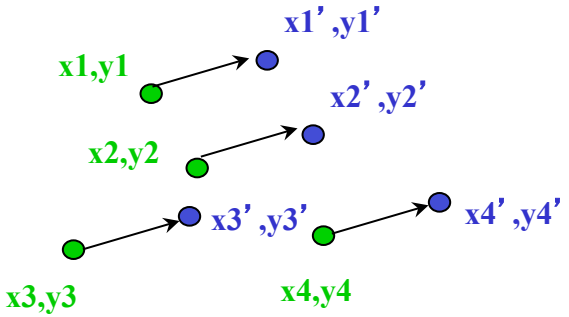
Before we continue, we need to take a side trip to explain least squares and RANSAC.

Estimating a Transformation from Point Correspondences

Let's say we have found point matches between two images, and we think they are related by some parametric transformation (e.g. translation; similarity; affine). How do we estimate the parameters of that transformation?

General Strategy

- Least-Squares estimation from point correspondences



Some important (related) questions:

- How to parameterize the transformation
- How many degrees of freedom does it have
- How many point correspondences are needed

Example: Translation Estimation

equations

$$x' = x + t_x$$
$$y' = y + t_y$$

matrix form

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

How many degrees of freedom?

How many independent variables are there? **Two**

How many point correspondences are needed?

Each correspondence (x,y)=>(x',y') provides two equations

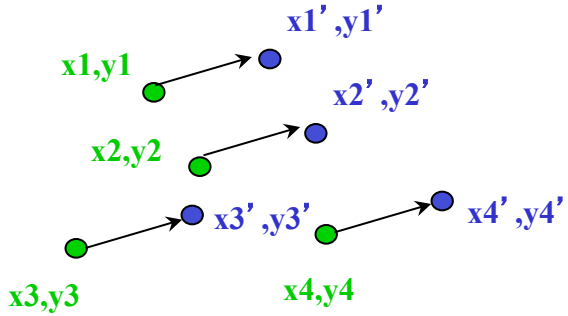
$$\frac{\text{DoF}}{2} = 2/2 = 1$$

Note: If this is not an integer, take the ceil() of it [i.e. round up]

Example: Translation Estimation

equations

$$x' = x + t_x$$
$$y' = y + t_y$$



Least Squares Estimation:

Minimize $E = \sum_{i=1}^{n} \left((x_i + t_x - x_i')^2 + (y_i + t_y - y_i')^2 \right)$ **wrt t_x, t_y**

$$\frac{\partial E}{\partial t_x} = \sum_{i=1}^{n} 2(x_i + t_x - x_i') = 0$$
$$\frac{\partial E}{\partial t_y} = \sum_{i=1}^{n} 2(y_i + t_y - y_i') = 0$$

$$\Rightarrow$$

$$t_x = \sum_{i=1}^{n} (x_i' - x_i)/n$$
$$t_y = \sum_{i=1}^{n} (y_i' - y_i)/n$$

Another example

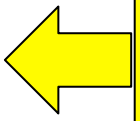
to try on your own

Similarity transformation (rotation, translation, scale)

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & -b & c \\ b & a & d \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

where: $a = s\cos\theta$

$b = s\sin\theta$

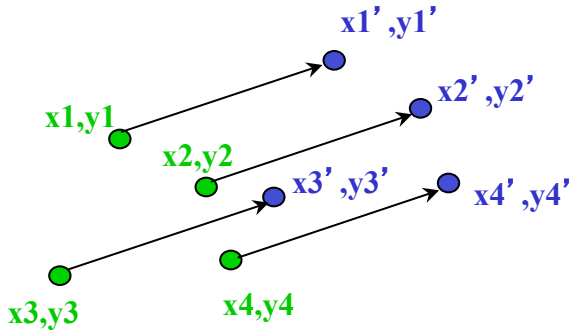


Note the choice of parameterization! We don't want to be taking derivatives of sin/cos, for example.

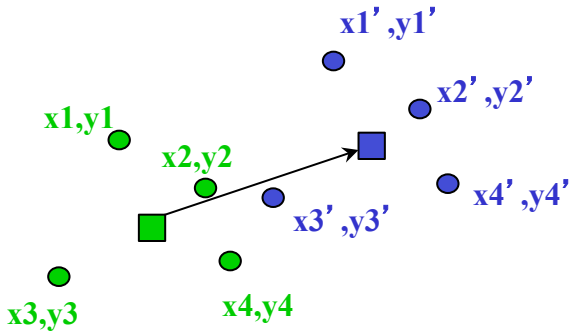
Motivation for Robust Estimation

$$t_x = \sum_{i=1}^{n} (x_i' - x_i)/n$$

$$t_y = \sum_{i=1}^{n} (y_i' - y_i)/n$$

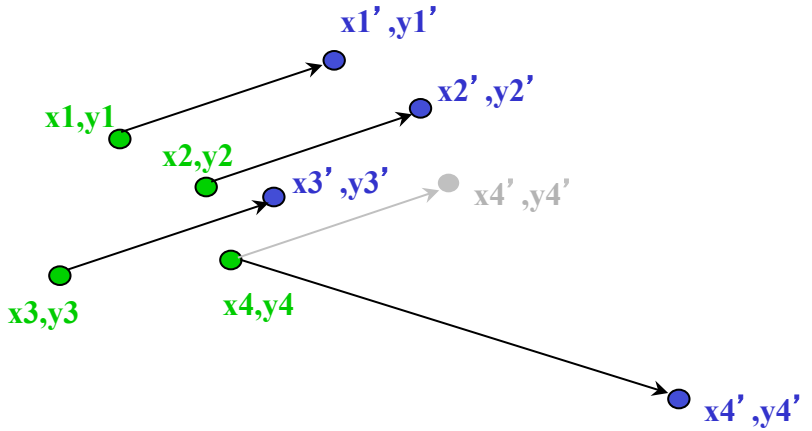


We see that the least squares estimate can be written as an offset between centers of mass (means).

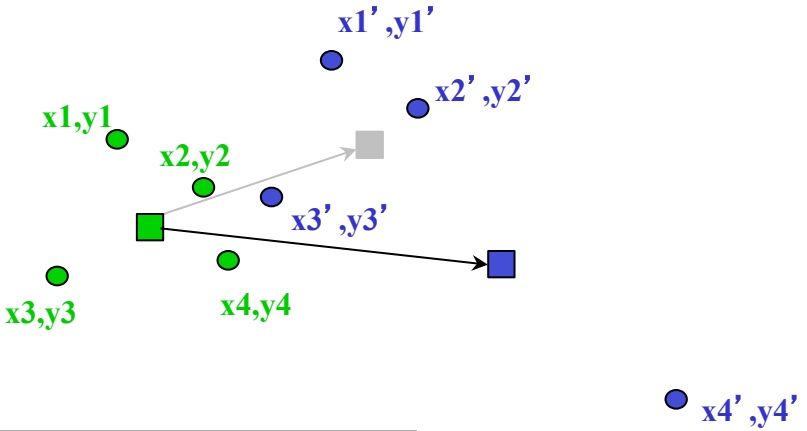


Motivation for Robust Estimation

If one or more matches is grossly incorrect…



That pulls the whole transformation out of alignment.



Least-squares is very sensitive to even a single gross error in the matching data.

Robust Estimation

- View estimation as a two-stage process:
  - Classify data points as outliers or inliers
  - Fit model to inliers while ignoring outliers

- Example technique: RANSAC
  (RANdom **SA**mple **C**onsensus)

M. A. Fischler and R. C. Bolles (June 1981). "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". *Comm. of the ACM* **24**: 381--395.

Estimating a Transformation via RANSAC

outline for estimating a geometric transformation:

given a set of point correspondences (xi,yi)→(xi',yi') for i=1,2,...,M

set global inlier count = 0

Loop N times

select minimal set of S point correspondences at random

compute transformation from that set using least squares

map all other points (xj,yj) by that transformation to get (x_{pred},y_{pred})

for each (xi',yi') that is "close enough" to (x_{pred},y_{pred})

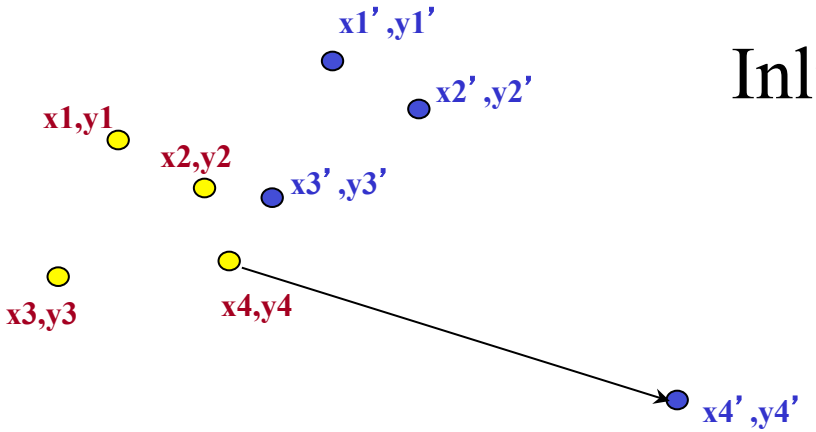
increment inlier count by one

if inlier count is greater than the global one, update the global inlier count

End loop

we now have the largest set of inliers found. Do something with it.
(like do least squares estimation of the transformation using largest inlier set)

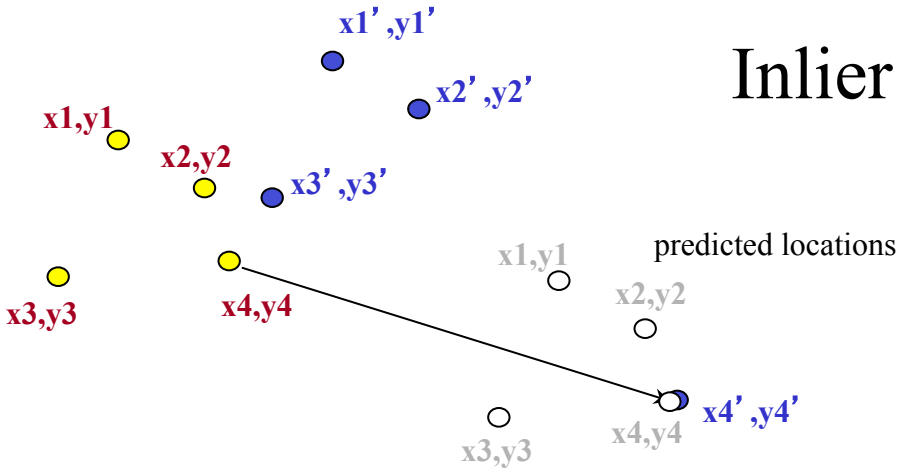
RANSAC Estimate of Translation



Inlier Count: 0

Choose one (xi,yi)→(xi',yi') pair at random and estimate the translation between them

RANSAC Estimate of Translation

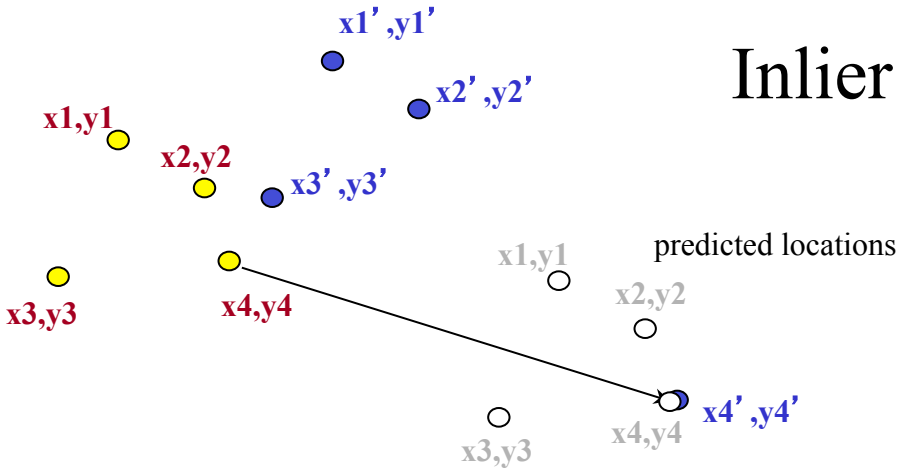


Inlier Count: 0

Choose one (xi,yi)→(xi',yi') pair at random and estimate the translation between them

Predict where other (xi,yi) points should go based on that translation estimate.

RANSAC Estimate of Translation



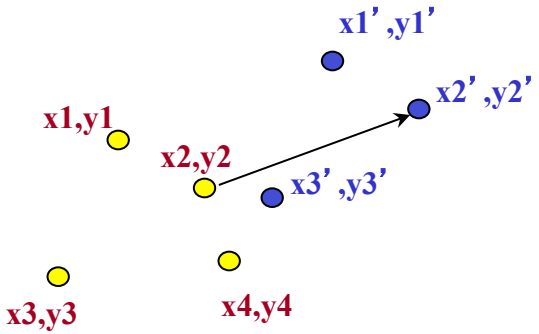
Inlier Count: 1

Choose one (xi,yi)→(xi',yi') pair at random and estimate the translation between them

Predict where other (xi,yi) points should go based on that translation estimate.

Count number of (xi',yi') points that lie near their predicted positions.

RANSAC Estimate of Translation



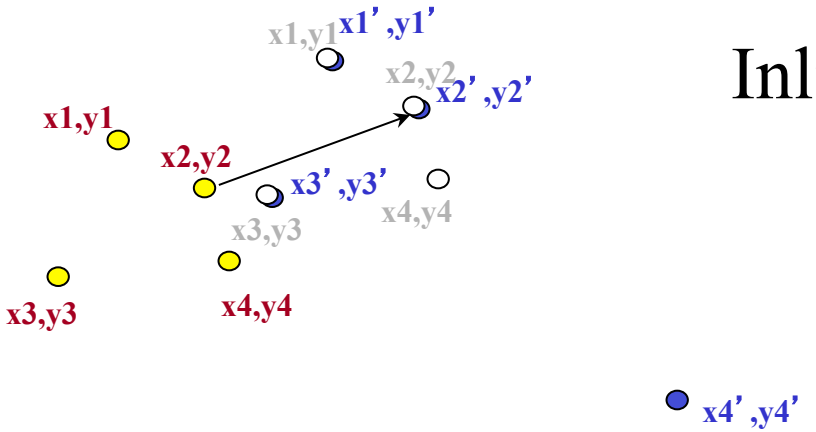
Inlier Count: 1

Choose another (xi,yi)→(xi',yi') pair at random and estimate the translation between them

Predict where other (xi,yi) points should go based on that translation estimate.

Count number of (xi',yi') points that lie near their predicted positions.

RANSAC Estimate of Translation



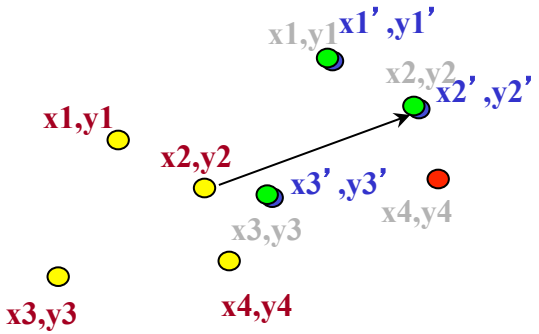
Inlier Count: 1

Choose another (xi,yi)→(xi',yi') pair at random and estimate the translation between them

Predict where other (xi,yi) points should go based on that translation estimate.

Count number of (xi',yi') points that lie near their predicted positions.

RANSAC Estimate of Translation



Inlier Count: 3

Choose another (xi,yi)→(xi',yi') pair at random and estimate the translation between them

Predict where other (xi,yi) points should go based on that translation estimate.

Count number of (xi',yi') points that lie near their predicted positions.

Application: Two-frame Stabilization

- Stabilizing aerial imagery
  - find corners in two images
  - hypothesize matches using NCC
  - do RANSAC to find matches consistent with an affine transformation
  - take the inlier set found and estimate a full projective transformation (homography)

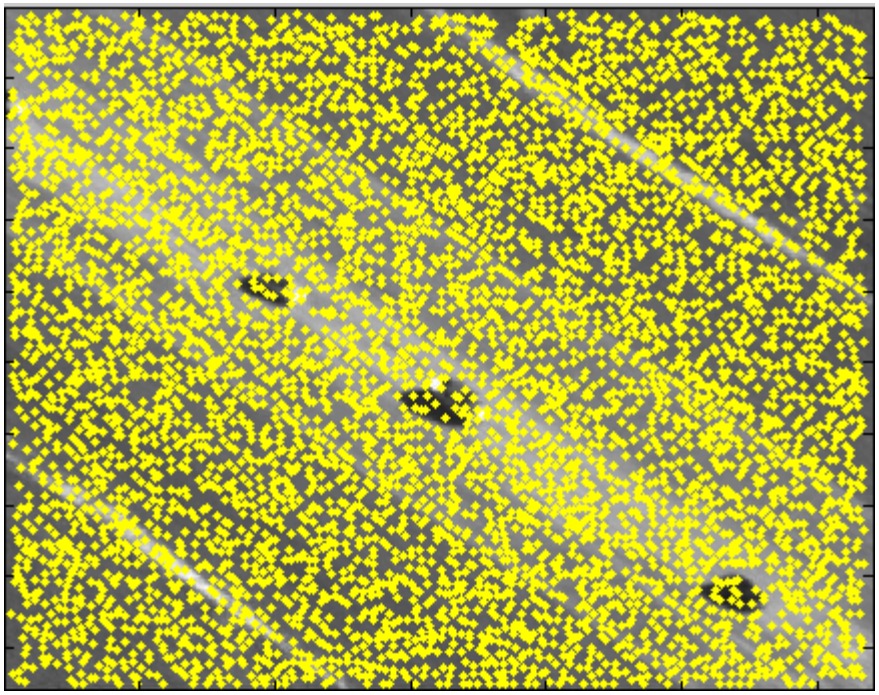
Stabilization Application

Input: two images from an aerial video sequence.

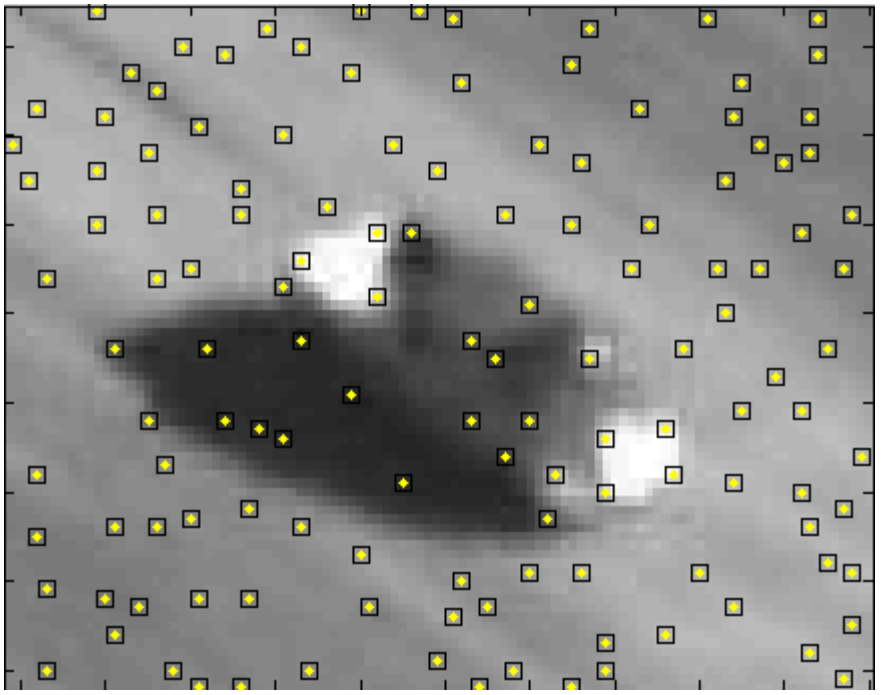
Note that the motion of the camera is "disturbing"

Stabilization Application

Step1: extract Harris corners from both frames. We use a small threshold for R because we want LOTS of corners (fodder for our next step, which is matching).



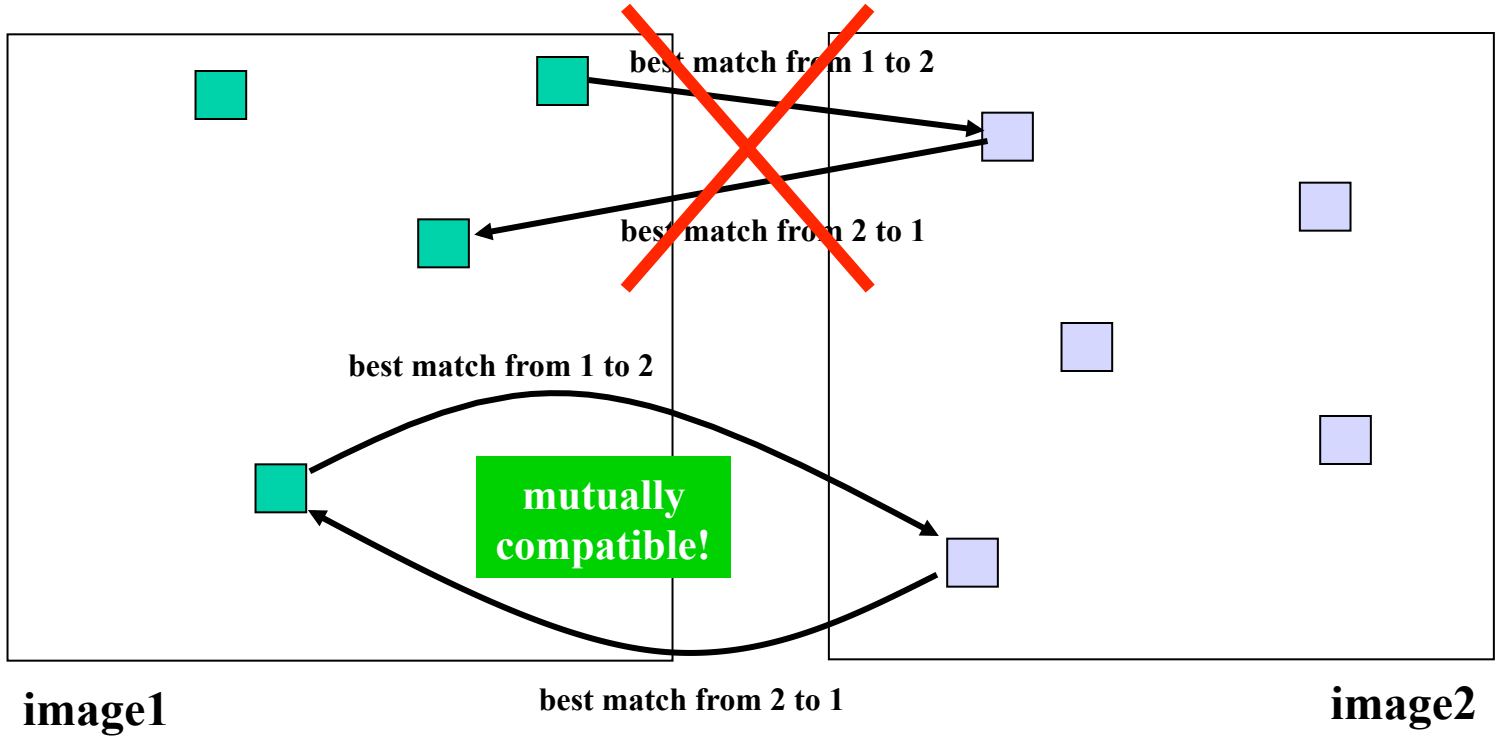
Harris corners for first frame



Detailed view

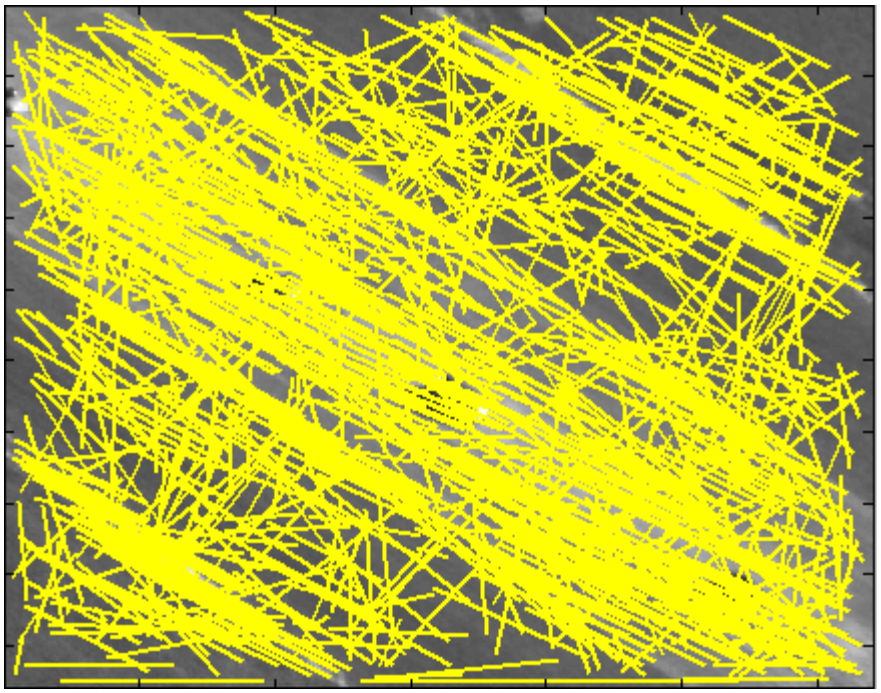
Stabilization Application

Step2: hypothesize matches. For each corner in image 1, look for matching intensity patch in image2 using NCC. Make sure matching pairs have highest NCC match scores in BOTH directions.



Stabilization Application

Step2: hypothesize matches.

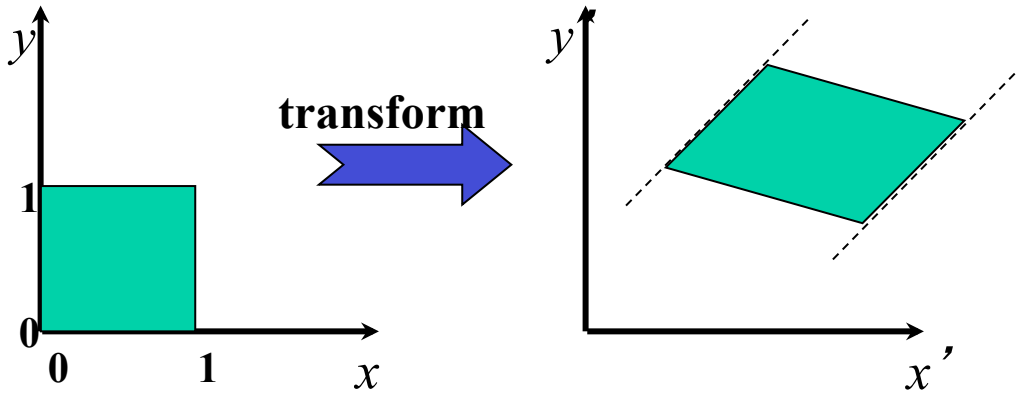


yikes!

As you can see, a lot of false matches get hypothesized. The job of RANSAC will be to clean this mess up.

Stabilization Application

Step3: Use RANSAC to robustly fit best affine transformation to the set of point matches.



$$x_i' = a\,x_i + b\,y_i + c$$
$$y_i' = d\,x_i + e\,y_i + f$$

$$\begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} \sim \begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}$$

How many unknowns?

How many point matches are needed?

Stabilization Application

Step3: Use RANSAC to robustly fit best affine transformation to the set of point matches.

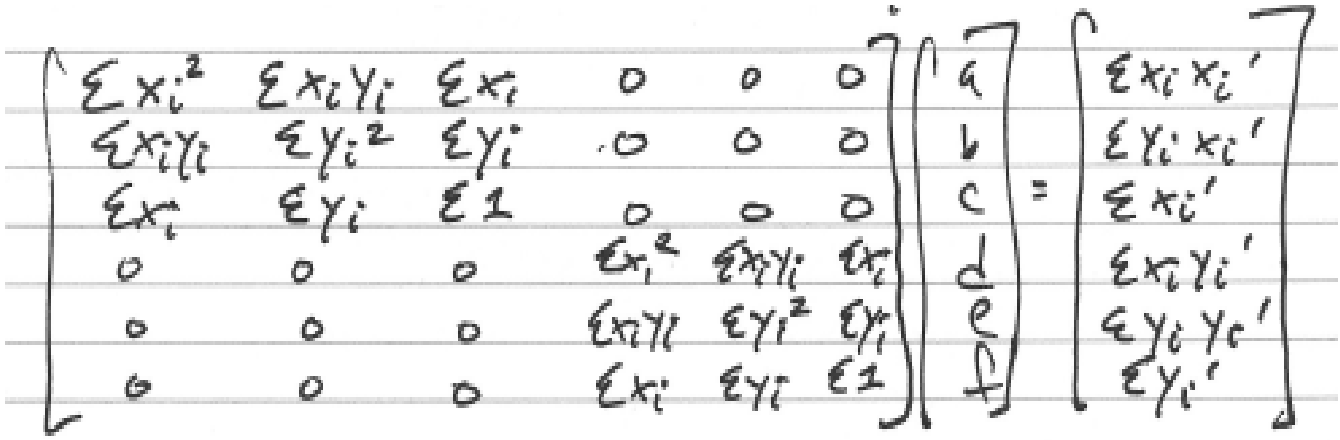
Affine transformation has 6 degrees of freedom.
We therefore need 3 point matches [each gives 2 equations]

Randomly sample sets of 3 point matches. For each, compute the unique affine transformation they define. How?

Robert Collins
CMPEN454

Stabilization Application

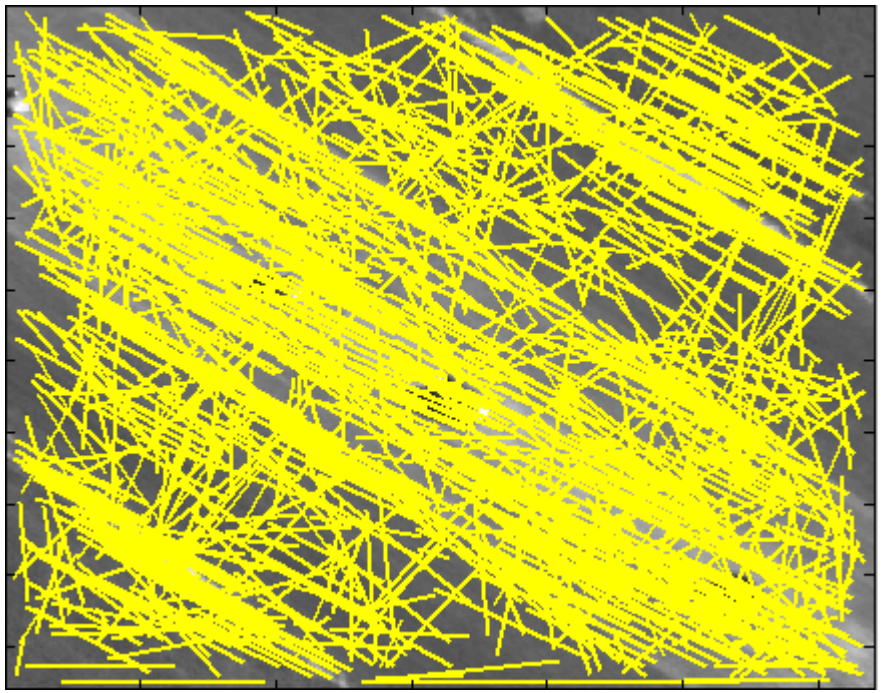
How to compute affine transformation from 3 point matches?
Use Least Squares! (renewed life for a nonrobust approach)



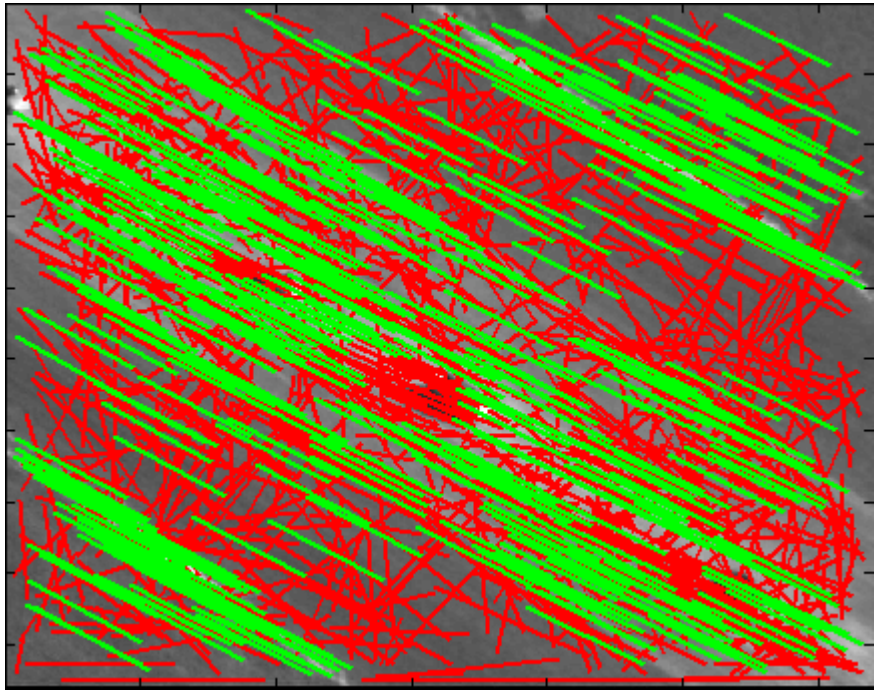
Then transform all points from image1 to image2 using that computed transformation, and see how many other matches confirm the hypothesis.

Repeat N times.

Stabilization Application



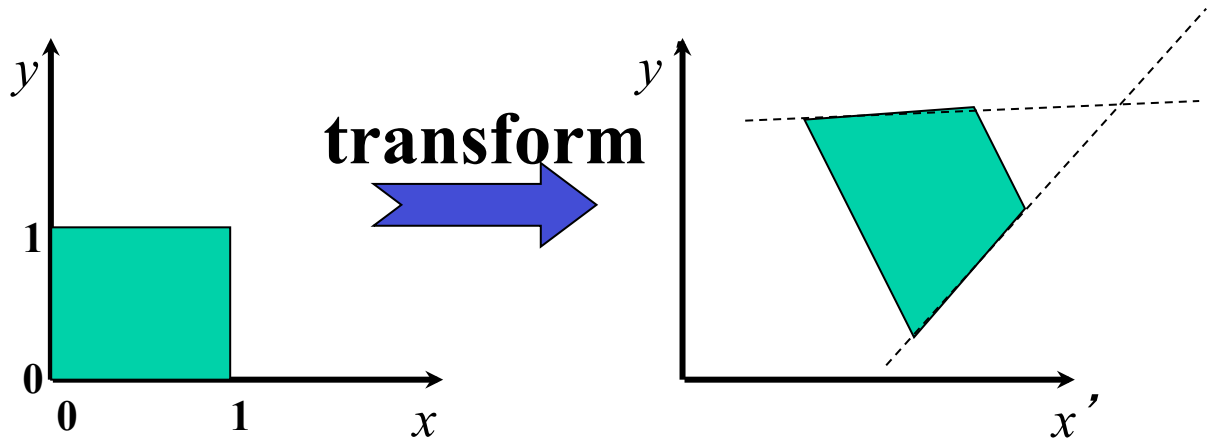
original point matches



labels from RANSAC
green: inliers
red: outliers

Stabilization Example

Step4: Take inlier set labeled by RANSAC, and now use least squares to estimate a projective transformation that aligns the images. (we discussed estimating a homography previously).



Projective Transformation

Stabilization Example

Step4: estimate projective transformation that aligns the images.

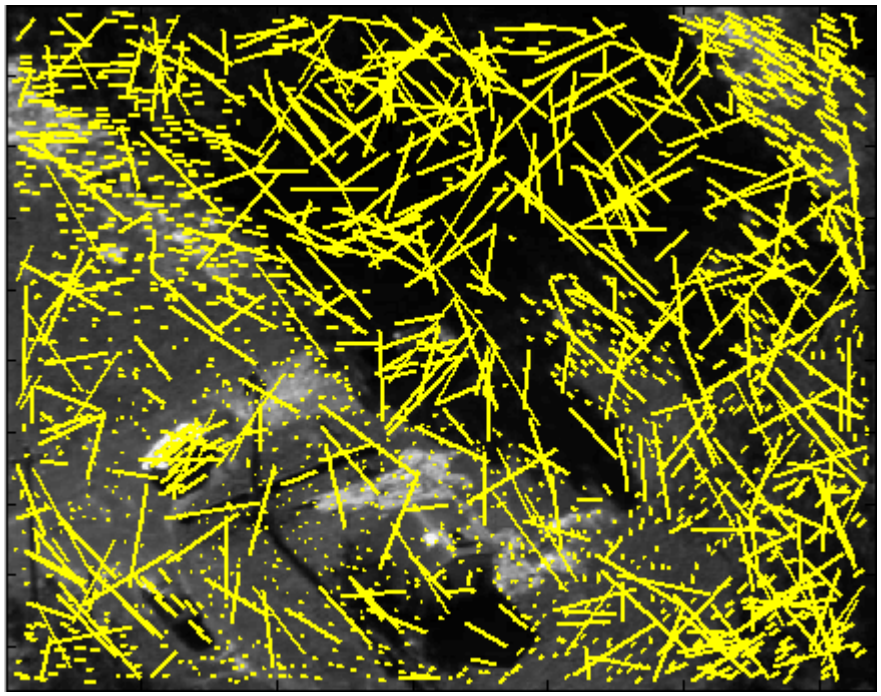
Now it is easier for people (and computers) to see the moving objects.

Robert Collins
CMPEN454

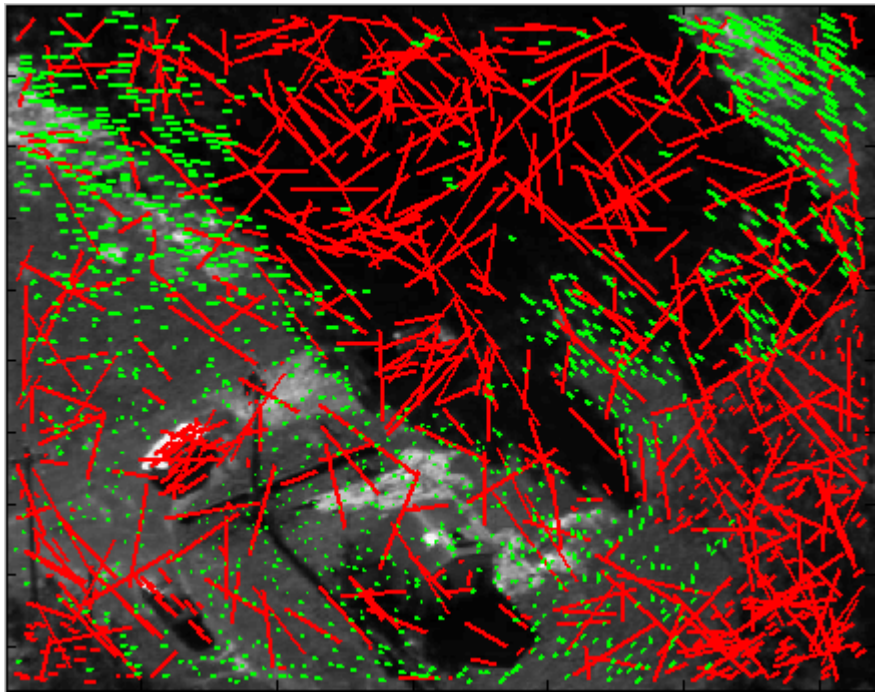
Stabilization Examples

Robert Collins
CMPEN454

Stabilization Examples



original point matches



labels from RANSAC
green: inliers
red: outliers

Stabilization Examples

Recall: Ghosting

Points not on plane won't align correctly.

Residual Parallax Motion

Analogous to the concept of ghosting we mentioned earlier:

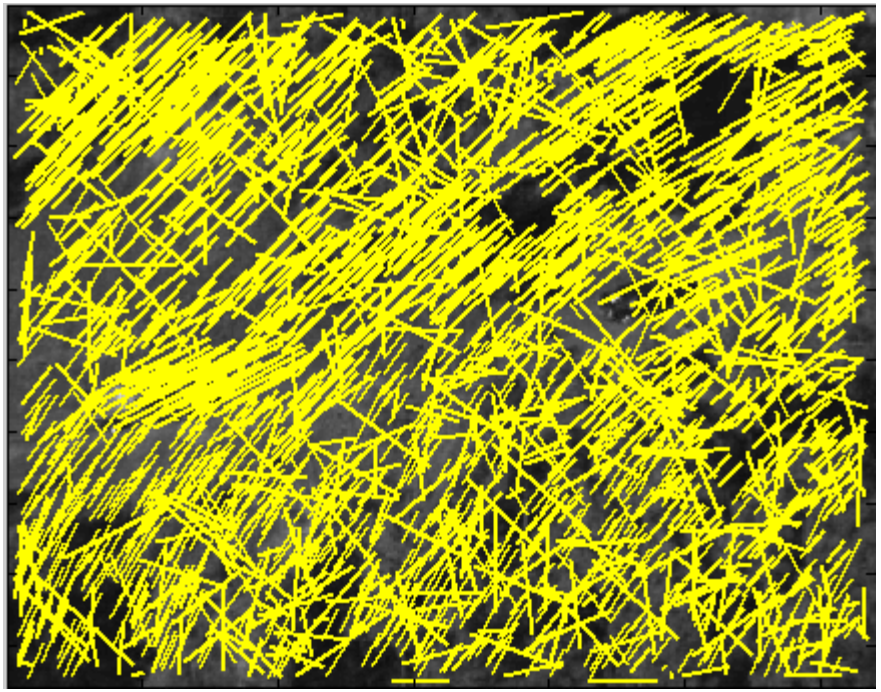
- Only points on the ground plane are truly stable
- Points off the plane still appear to be moving
- The further a point is from the ground, the more it appears to be moving.

Plane + parallax : cues to 3D scene geometry!

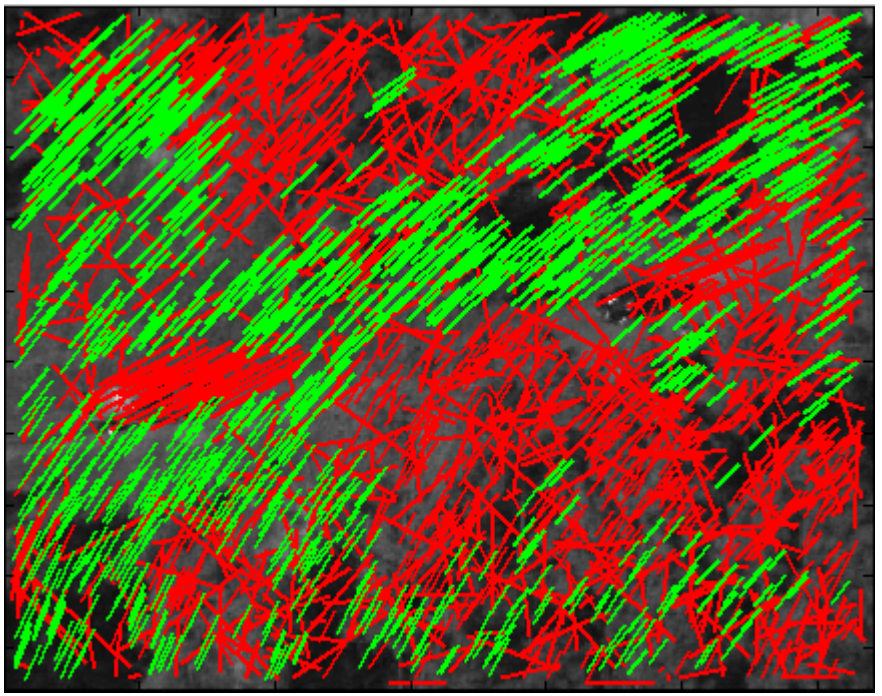
Robert Collins
CMPEN454

Stabilization Examples

Stabilization Examples



original point matches



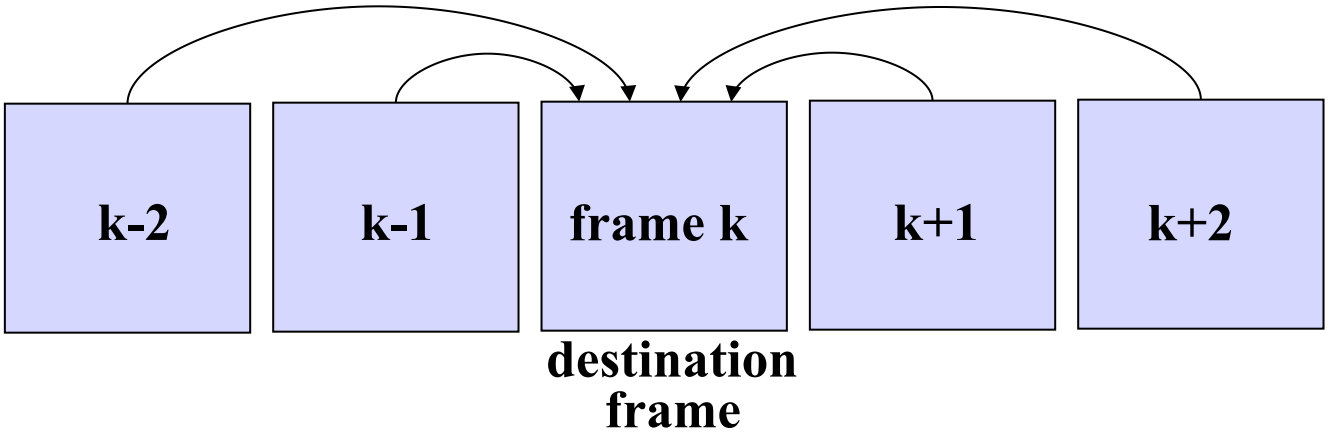
labels from RANSAC
green: inliers
red: outliers

Stabilization Examples

Robert Collins
CMPEN454

Video Stabilization

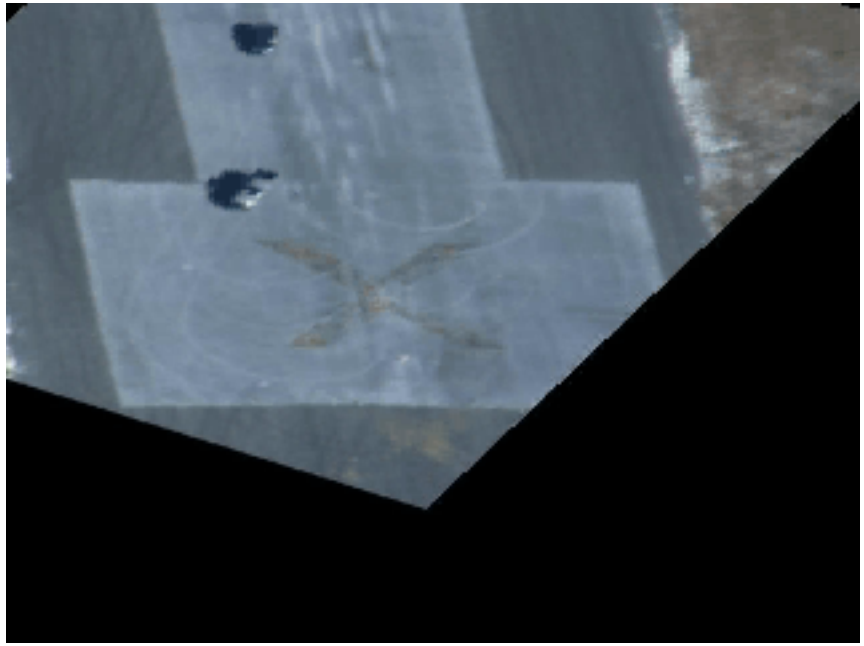
Given a sequence of video frames, warp them into a common image coordinate system.



This "stabilizes" the video to appear as if the camera is not moving.

Video Stabilization Example

original video



stabilized video

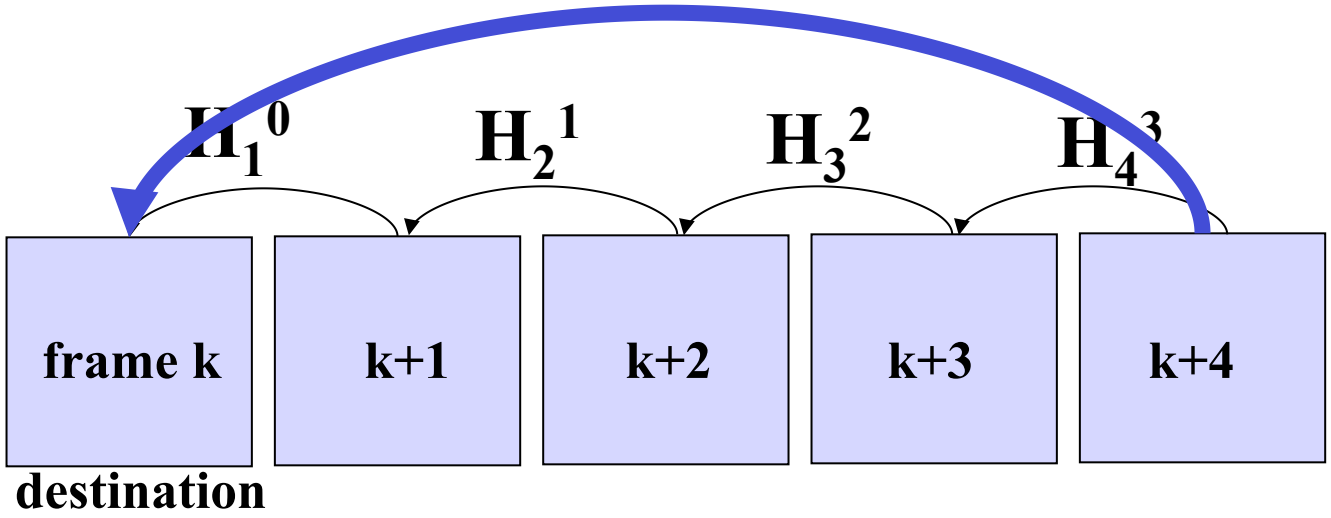
See also real-time video stabilization videos by Sarnoff:

https://www.youtube.com/watch?v=BqUXpaScGUQ

Stabilization by Chaining

What if the reference image does not overlap with all the source images? As long as there are pairwise overlaps, we can chain (compose) pairwise homographies.

$$H_4^0 = H_1^0 * H_2^1 * H_3^2 * H_4^3$$



Not recommended for long sequences, as alignment errors accumulate over time.