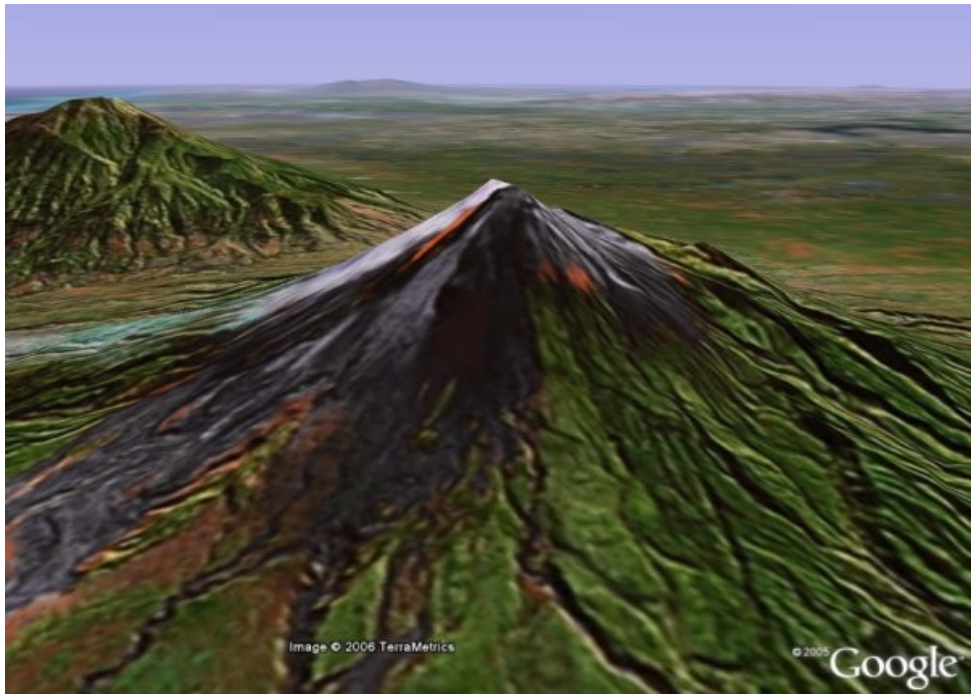
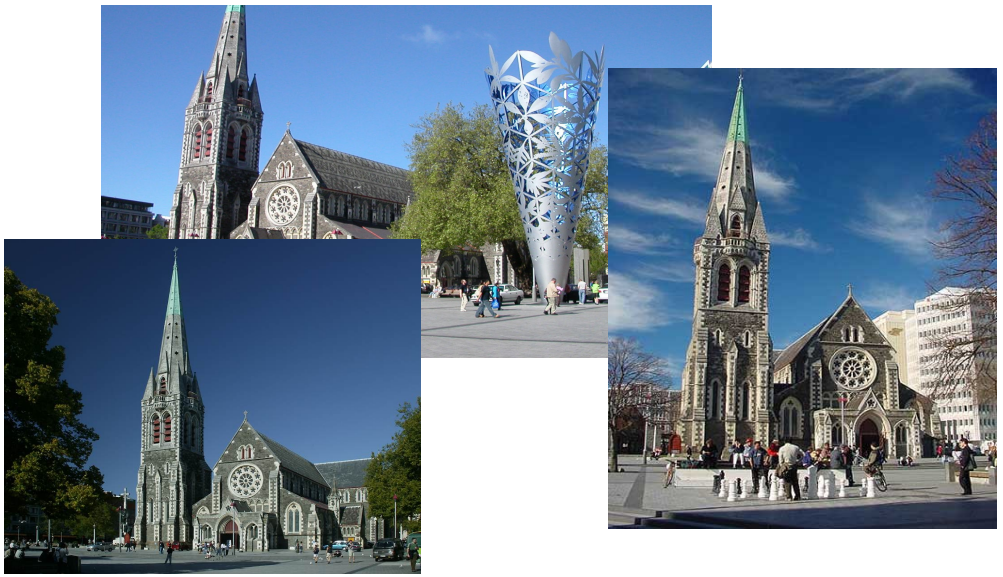
3D reconstruction using computer vision



3D Reconstruction

- One of the most important parts of CV

2D images 3D model



- Applications: CV for robot control, CV for self-driving cars, CV for measuring, medical imaging, photorealistic graphics, reverse engineering, AR, many more...

Overview

- Most important algorithms for recovering 3D structure from images: RANSAC and Bundle Adjustment
- Inputs: images
- Outputs: 3D structure and camera positions

Also known as:

- Structure from Motion (SfM)
- Photogrammetry

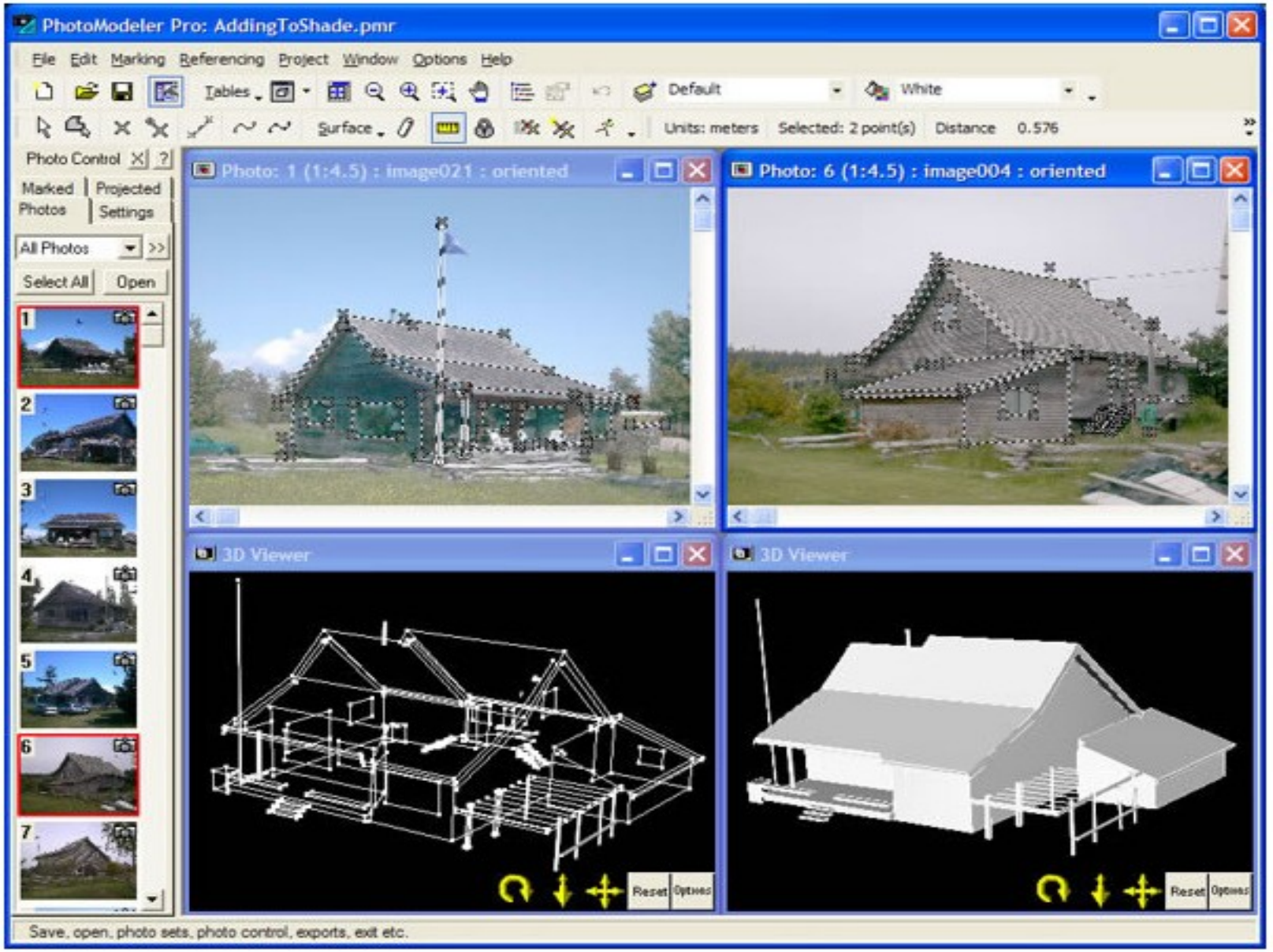
Summary

- Homography H: relates relative pose of 2 cameras viewing a planar scene. Estimate from feature correspondences using RANSAC.
- Essential matrix E: relates relative pose of 2 cameras viewing a 3D scene. Estimate from feature correspondences using RANSAC.
- Bundle adjustment (BA): initialise using RANSAC (for E), estimate a set of 3D points and camera poses which minimises reprojection error.
- Useful applications, exciting research opportunities!

Example application: aerial mapping

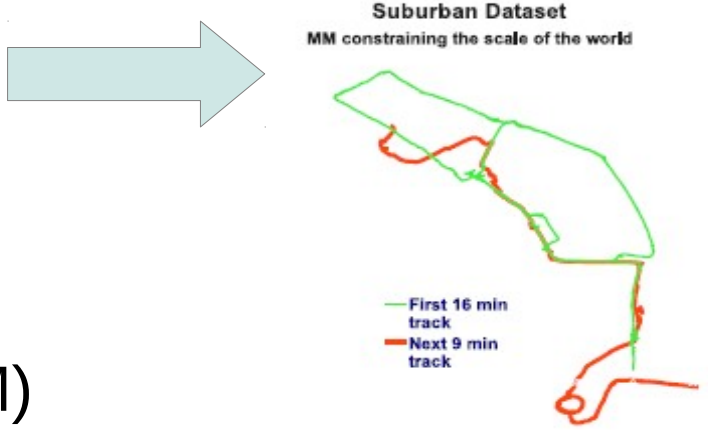
- Plane with camera captures some aerial images
  - Software produces a terrain map

Example application: 3D models from images (e.g. Photomodeller)



Example application: robot positioning

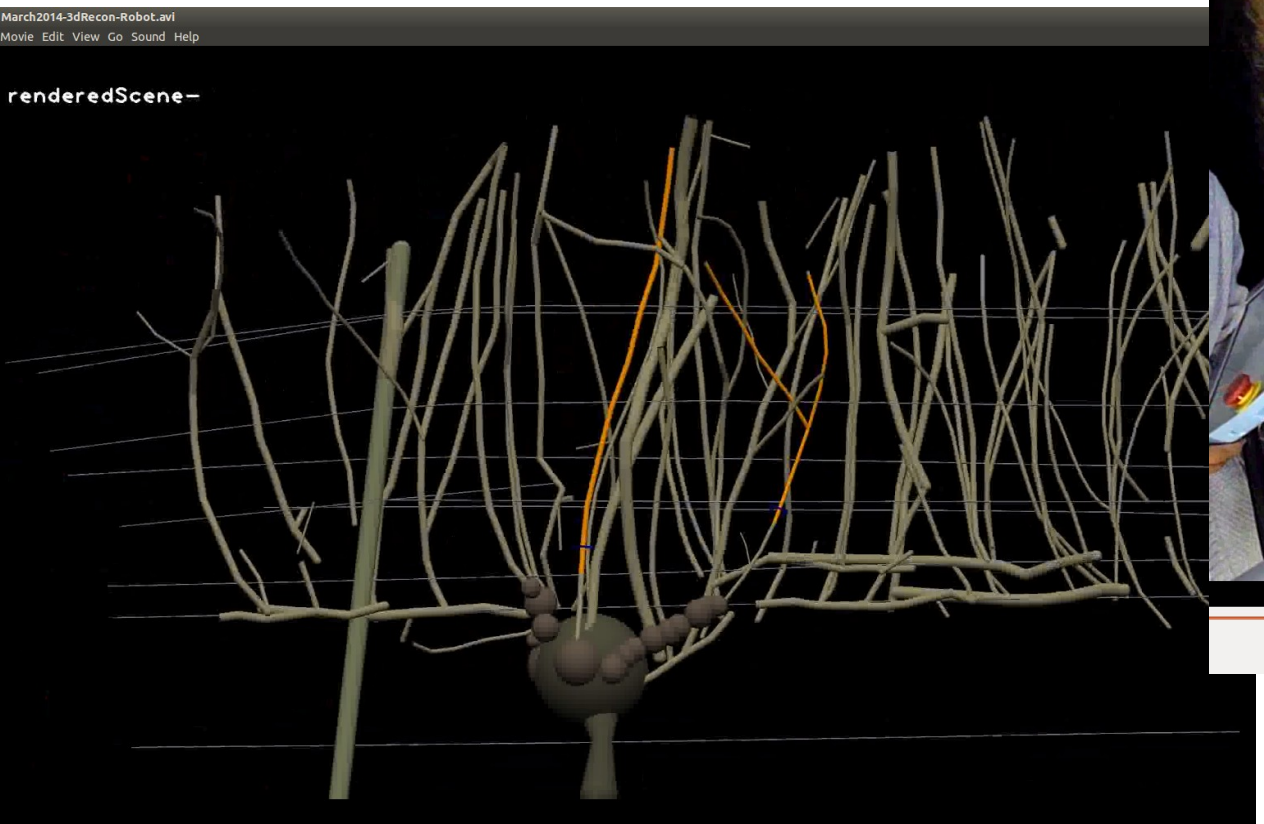
- Attach camera to robot
- Reconstruct camera positions and 3D structure
- Camera positions = robot positions

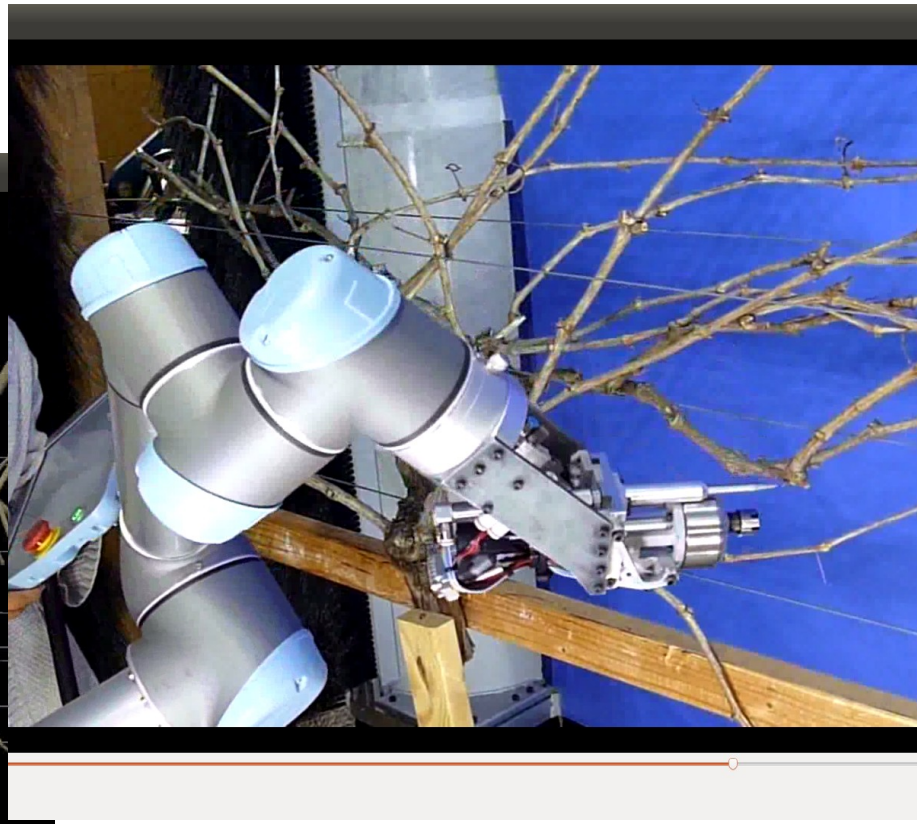


(one way of doing Simultaneous Localisation and Mapping, or SLAM)

Example application: robot control

- Make 3D model of grape vines
- Control robot arm to prune vines





Very closely related:

- Even '3D' cameras (Kinect) need photogrammetry
  - Similar techniques used for:
    - Fuse multiple 3D images into a larger model
    - Robot navigation using laser scanners
    - CT scanners (3D x-ray)
    - MRI scanners (brain imaging)

All of these applications have the form:

Inputs: images

Outputs: camera positions and 3D structure

Lecture outline:

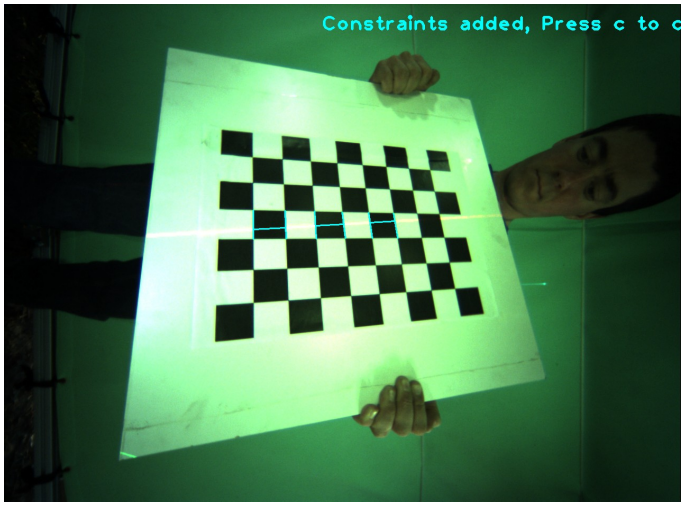
- Recap of camera calibration and feature matching
- Describe RANSAC algorithm for estimating camera positions from 2 images
- Describe a sequence of algorithms for structure from motion/3D reconstruction from N images
- Discuss some of the challenges, and variations to the 'standard' approach

Background 1: camera calibration

- Calibration gives a mapping from pixel coordinates to normalised image coordinates

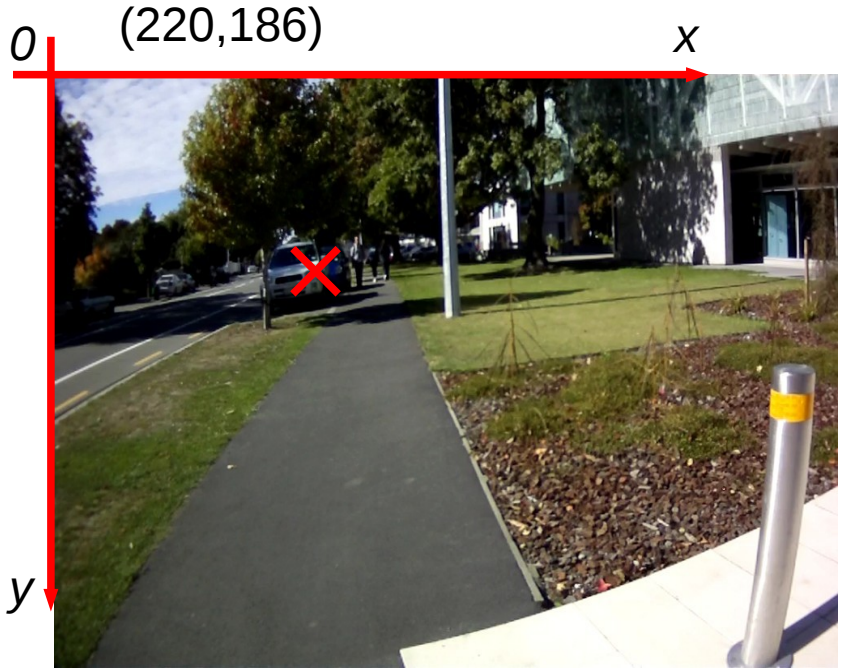
- Estimate and correct lens distortion, focal length, image centre

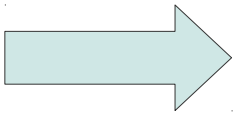
- Functions and example programs in OpenCV

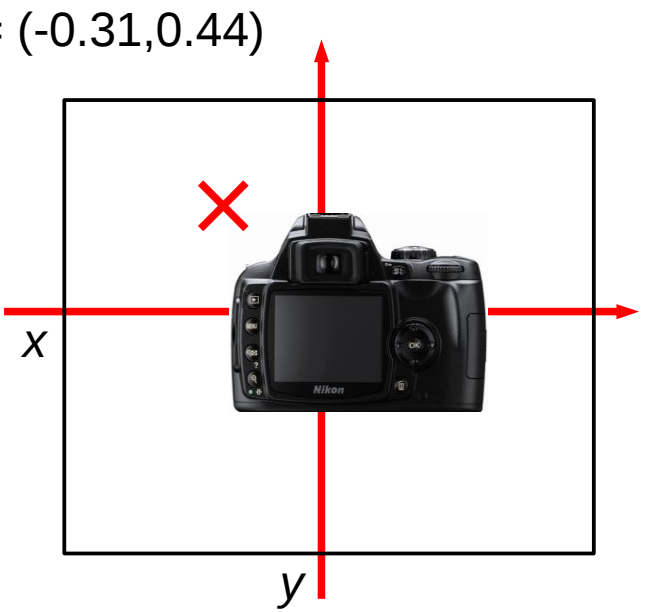


Background 1: camera calibration

- Calibration maps the location of each pixel to the corresponding position of that point on a plane at a distance of 1 unit, with z-axis in direction of camera's principal point.







Background 1: camera calibration

In this lecture, we assume the cameras are calibrated, and we use normalised image feature locations

Background 2: feature matching

- Fundamental first step of photometric reconstruction: find matching positions of objects appearing in 2 images

Background 2: feature matching

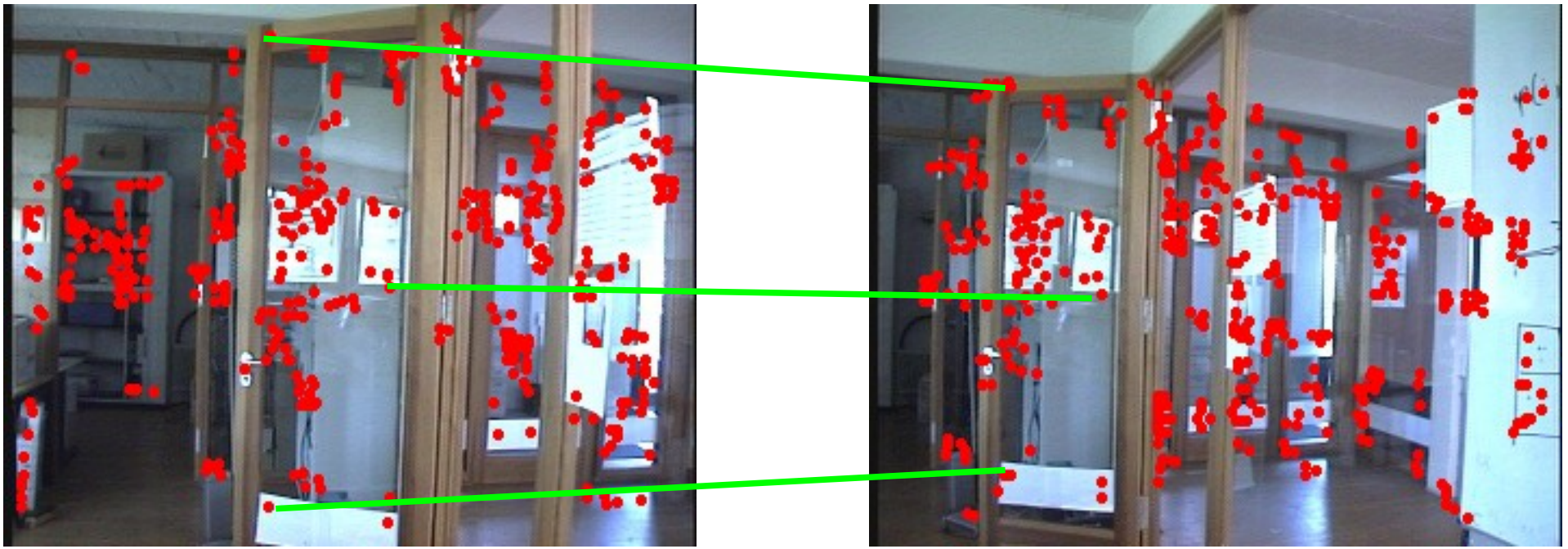
- First choose some features to match: point features are convenient.
- Corner detector finds all of the corner points visible in 2 images. We need to match corner points in one image to corner points in the other.

Background 2: feature matching

- “Corner points” – found in many scenes, a corner on a 3D object is usually detected in every image of the object
- Desirable properties for a corner detector:
  - **Repeatable** – same corners found in every image of an object
  - **Localisable** – detected locations match actual location of 3D point which causes corner
- cv::goodFeaturesToTrack finds Harris corners

Background 2: feature matching

- Find pairwise matches by looking at appearance of image around each corner.
- Representations of appearance are known as “feature descriptors”. Examples: image patches, SIFT, SURF, etc.



Background 2: feature matching

- Problem with all feature matching (or tracking) algorithms: incorrect matches

Objects which look the same but are different (self-similar environments)

Corner features on occlusion boundaries which which don't actually correspond to 3D points

Background 2: feature matching

- Feature matching algorithms give a mixture of correctly matched features (inliers) and incorrectly matched features (outliers/gross outliers)
- “Gross outliers” means location errors are much higher than what which would be expected from localisation (e.g. 100 pixels rather than 0.3 pixels)

Background 3: homogeneous coordinates

- Add an extra dimension, so 3D points are represented by 4D homogeneous points, 2D points are represented by 3D homogeneous points.
- In 2D (coordinates of a point in an image):

  $\mathbf{x} = (x,y)^T \rightarrow (x,y,1)^T$ in homogeneous coordinates

- In 3D (coordinates of an object in space):

  $\mathbf{X} = (X,Y,Z)^T \rightarrow (X,Y,Z,1)^T$ in homogeneous coordinates

Background 3: homogeneous coordinates

- Reason for homogeneous coordinates: can represent both rotation and translation using matrix multiplication, and can represent projection by normalisation.
- Used in 3D computer graphics (simulate 3D world, project into 2D image)

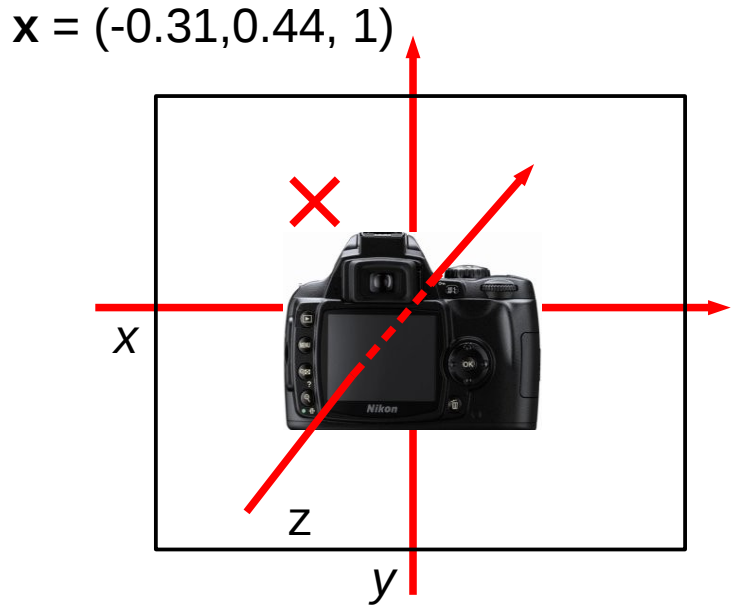
Background 3: homogeneous coordinates

- To transform a 3D point **X** using the rotation matrix R, and translation **t** → make a 4x4 transformation matrix T:

$$T = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \qquad T(\mathbf{X}) = \begin{pmatrix} R\mathbf{X} + \mathbf{t} \\ 1 \end{pmatrix}$$

Background 3: homogeneous coordinates

- Homogeneous coordinates:
  - Camera is at (0,0,0)
  - Image points lie on the plane at z=1
  - Points s(x,y,1) appear in the same place in the image for any s (equivalent in homogeneous coordinates)



Background

(1) Camera calibration – map pixel coordinates to "normalised image coordinates", which corrects for lens distortion, focal length (zoom), etc.

(2) Feature matching: find positions of objects visible in each of a pair of images

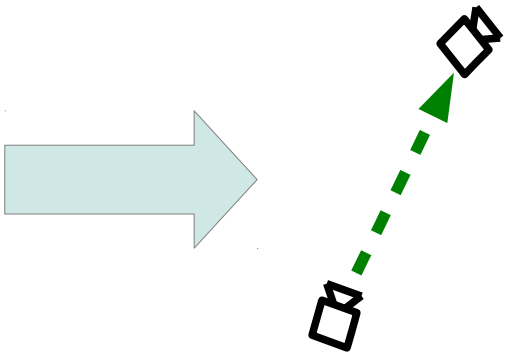
- Automatically match corner features between images
- Also finds many incorrect matches (outliers)

(3) Homogeneous coordinates: simple formula $P\mathbf{X}=\mathbf{x}$ for projecting 3D homogeneous point $\mathbf{X}$ to image position of 2D homogeneous point $\mathbf{x}$

2-view reconstruction

Aim: recover rotation, translation, 3D structure given 2 images.

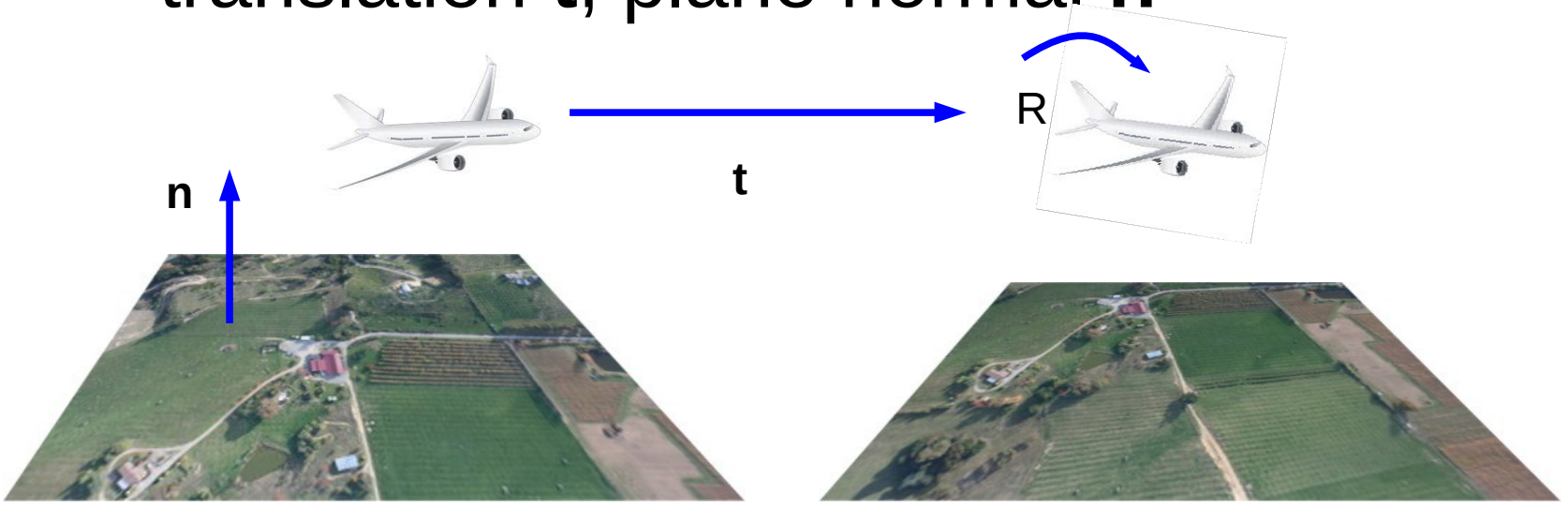
Input: Features matched between 2 frames
Output: Relative camera position, 3D structure



Firstly: planar scenes
Secondly: arbitrary static 3D scenes

Planar scenes: homography estimation

- Common situation, e.g.:
  - Aerial images
  - Many Augmented Reality (AR) applications
  - Mobile robot looking at ground
- 2 views of a planar scene → camera rotation R, translation **t**, plane normal **n**



Planar scenes: homography estimation

- Homography, H: a 3x3 matrix which is a convenient representation of the relative pose of the camera, and plane normal
- 2 important properties:

  (1) For an inlier feature match ($\mathbf{x} \rightarrow \mathbf{x}'$), in normalised homogeneous image coordinates, $H\mathbf{x} = \mathbf{x}'$

  (2) $H = R + (\mathbf{t}\,\mathbf{n}^T / d)$

  - R=relative orientation, **t**=translation unit vector, **n**=plane normal (unit vector), d is the ratio of the distance to the plane to the translation
  - Easy to convert between H and R, **t**, **n**, d

Planar scenes: homography estimation

- Note **Scale ambiguity** in $H = R + (\mathbf{t}\,\mathbf{n}^T / d)$ — without additional information we do not know the actual length of **t** (only the ratio, d).
  - Slow plane flying close to the ground, vs. fast plane flying high above the ground.
- Use property $H\mathbf{x} = \mathbf{x}'$ to estimate H from 4 or more inlier feature matches – algorithm in cv::findHomography
  - Doesn't work at all for outliers

Planar scenes: homography estimation

- Problem: don't know which features are matched correctly (which are inliers), and don't know H.
  - Solution: use the RANSAC framework to simultaneously estimate H while finding inliers

RANSAC framework

- RANSAC=RANdom SAmple Consensus
- **General-purpose** framework for fitting a model to data which is contaminated with gross outliers

Repeat:

(1) Randomly choose the minimum number of datapoints needed to generate a model (a hypothesis set)

(2) **Hypothesise**: compute a model from the hypothesis set

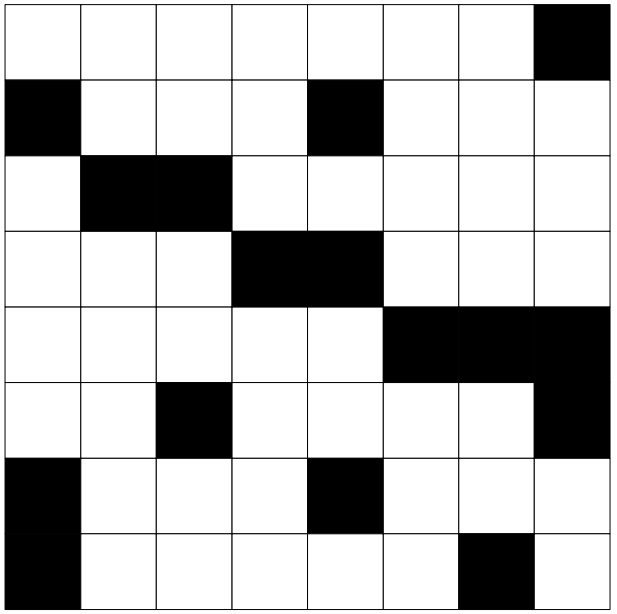
- If the hypothesis set contains only inliers, then the model will be approx. correct

(3) **Test**: Count how many datapoints would be inliers if the model was correct

until a model compatible with a large number of datapoints is found. These datapoints are inliers.

Classic example: RANSAC to find a straight line

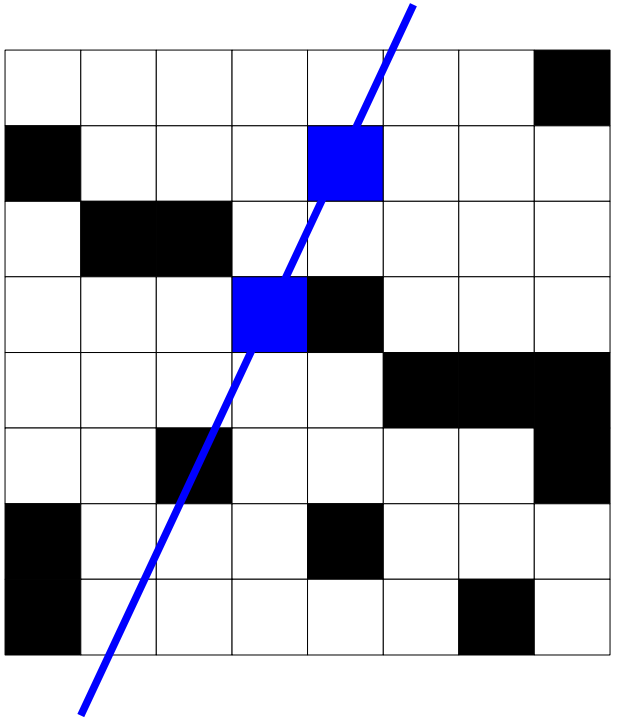
- Want to find a straight line in an image. Possible line pixels in black.

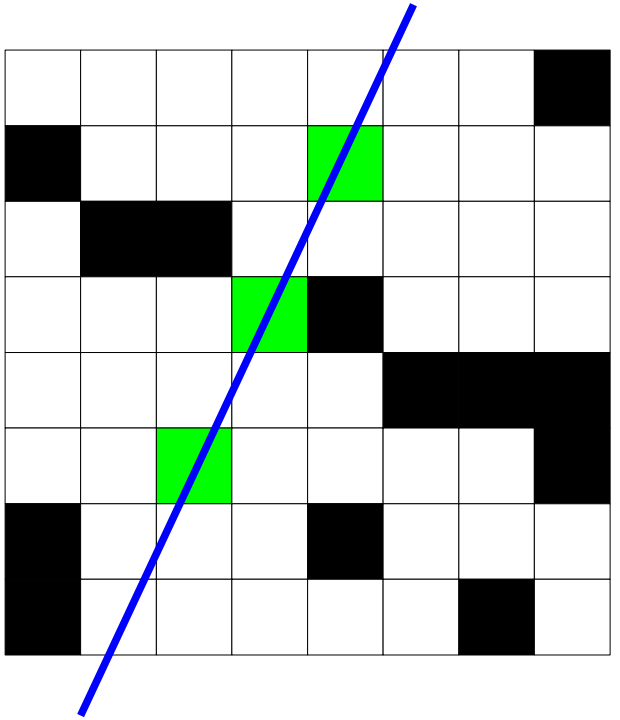


Classic example: RANSAC to find a straight line

- **Hypothesise**: choose 2 line pixels at random (blue)

  Fit a model (straight line)



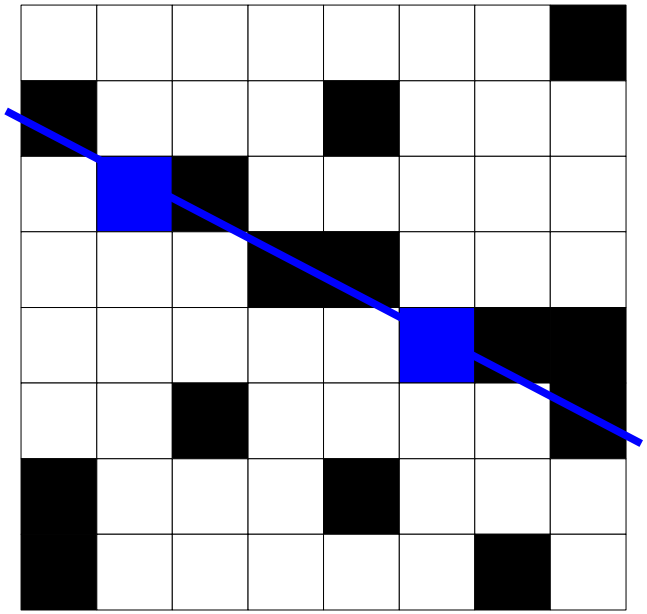


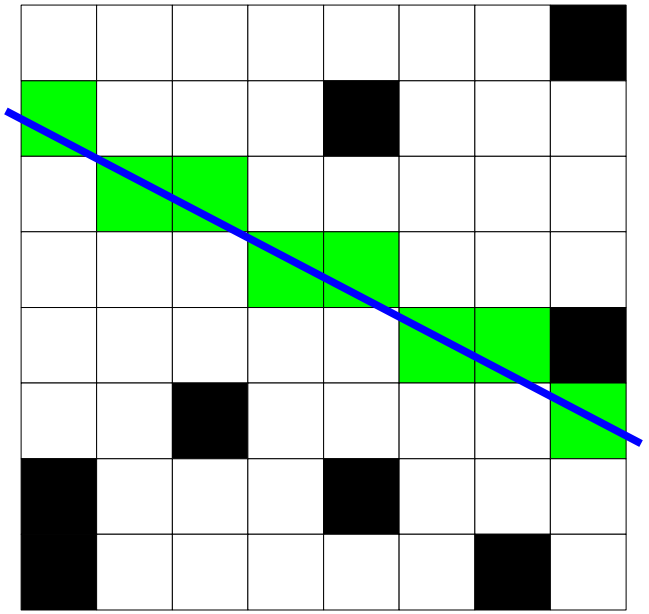
Test: Count number of points compatible with hypothesis (3)

Classic example: RANSAC to find a straight line

- **Hypothesise**: choose 2 line pixels at random (blue)

  Fit a model (straight line)





Test: Count number of points compatible with hypothesis (8)

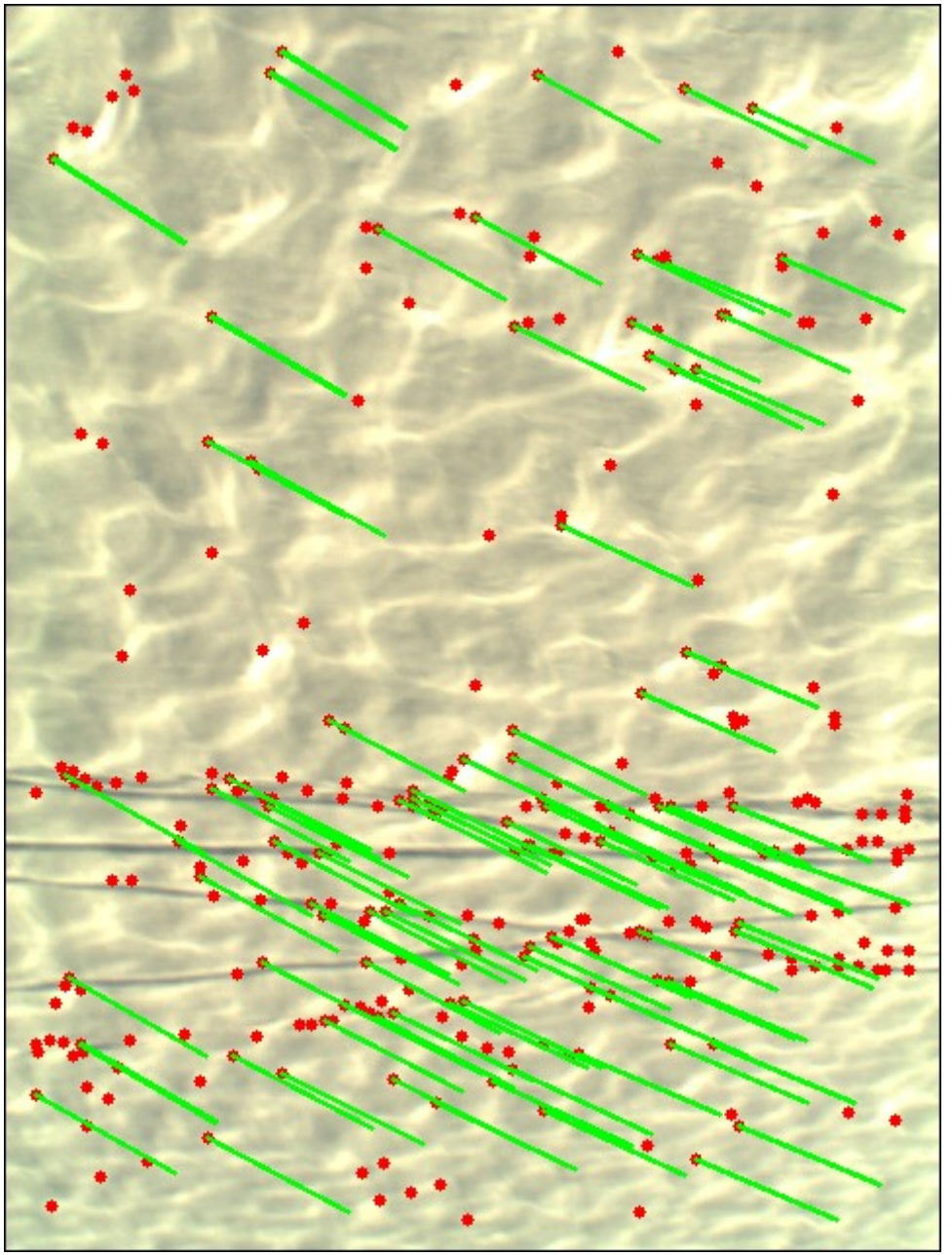
→ Accept

RANSAC for H

- **Hypothesise**: Choose random set of 4 feature matches
  - Compute H from these matches
- **Test**: Count number of feature matches where H**x** – **x**' < thresh
- **Repeat** until H compatible with large number of feature matches is found

- H → R, **t**, **n**, d

RANSAC for H: Results



3D scenes: essential matrix estimation

- Essential matrix, E: another 3x3 matrix. A convenient representation of the relative pose (R, **t**) of 2 cameras.
- $E = [\mathbf{t}]_x R$ where $[\mathbf{t}]_x$ is the matrix $\begin{pmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{pmatrix}$

  Property: $\mathbf{x}'^T E \mathbf{x} = 0$ for inlier feature match ($\mathbf{x} \rightarrow \mathbf{x}'$)
- Compute E from >= 5 inlier feature matches using $\mathbf{x}'^T E \mathbf{x} = 0$

RANSAC for E

- **Hypothesise**: Choose random set of 5 feature matches
  - Compute E from these matches
- **Test**: Count number of feature matches where $\mathbf{x}'^{T} E \mathbf{x}$ / (normalising constant) < thresh
- **Repeat** until E compatible with large number of feature matches is found

- Decompose E → R, **t**

RANSAC for E: results

E → 3D structure

- Choose cameras P=(I **0**), P'=(R **t**)
- For each inlier feature match (**x**, **x**'):
  - Solve equations P**X**=**x**, P'**X**=**x**' to find 3D point **X**

  → a sparse set of 3D points in the scene.

- Not in OpenCV, source code for E estimation at http://hilandtom.com/tombotterill/code

N-view reconstruction by Bundle Adjustment

- For 2 views, we can use RANSAC to find E, relative camera positions, inlier correspondences, 3D structure (but errors remain)
- Want accurate 3D reconstruction from N views
  - Aerial mapping: N>1000, 3D modelling N=10 to 100
- Solution: Bundle Adjustment algorithm (BA)

N-view reconstruction by Bundle Adjustment

Reprojection error: distance between where a 3D point, $\mathbf{X}$, is projected into an image, and where it is measured to lie, $\mathbf{x}$.

Reprojection error for 1 point = $|\mathbf{x} - P\mathbf{X}|$

BA finds the set of camera positions and 3D points which minimises the total reprojection error:

As an equation, BA aims to find 3D points $\{\mathbf{X}_j : j=1...M\}$, and cameras $\{P_i = (R_i\ \mathbf{t}_i) : i = 1...N\}$ so that:

$$C(\{\mathbf{X}_j\},\{P_i\}) = \Sigma_i \Sigma_{\{j \text{ appearing in view } i\}} |x_{ij} - P_i X_j|^2$$

is minimised.

N-view reconstruction by Bundle Adjustment

Find $\{\mathbf{X}_j\},\{P_i\}$ which minimise the total reprojection error $C(\{\mathbf{X}_j\},\{P_i\})$ by nonlinear gradient descent optimisation:

- Initialise $\{\mathbf{X}_j\},\{P_i\}$ by RANSAC on pairs of images
- Repeat:
  - Compute the gradient of C with respect to $\{\mathbf{X}_j\}$, $\{P_i\}$ (by differentiating C)
  - Adjust $\{\mathbf{X}_j\},\{P_i\}$ in the downhill direction to reduce $C(\{\mathbf{X}_j\},\{P_i\})$

N-view reconstruction by Bundle Adjustment

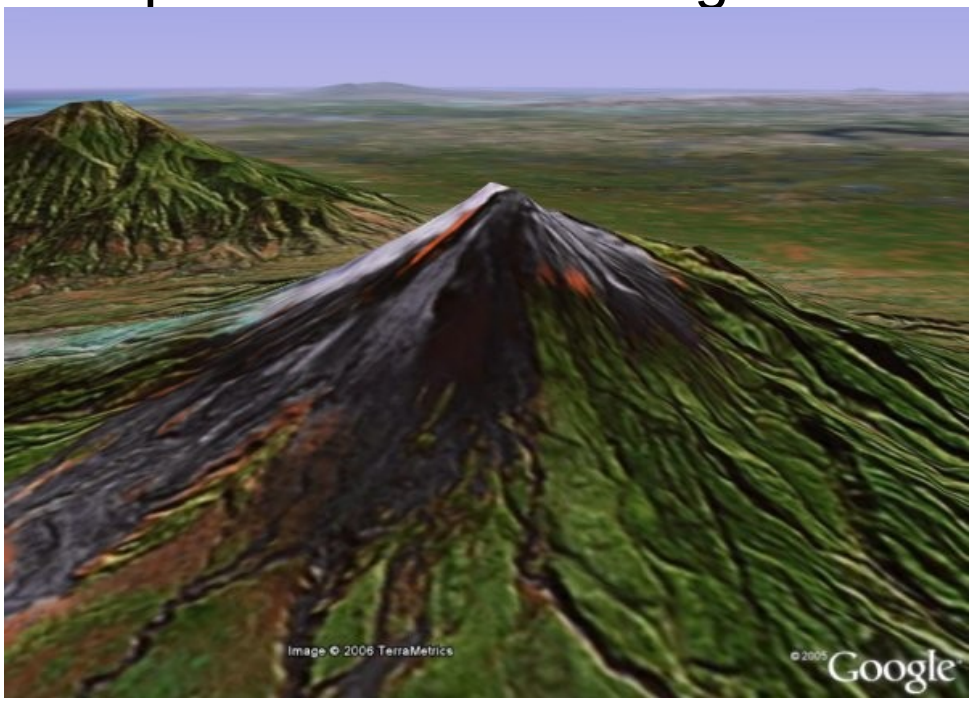
- BA in words:
  - Initialise the 3D structure and camera positions by RANSAC
    - Adjust the 3D structure and camera positions to reduce the total reprojection error
    - Repeat

N-view reconstruction by Bundle Adjustment

- Have described simplest form of BA: point features, calibrated cameras, assume all feature matches are inliers
- Errors in reconstruction remain, caused by:
  - Remaining outliers
  - Point localisation errors
  - Poorly-conditioned geometry
- In practice, we can make the solutions found more accurate by using additional information.

Extensions to BA: Use additional information

- Navigation data: e.g. use GPS positions to help initialise camera positions.
- Knowledge of the objects being imaged, e.g. buildings have planar walls at right-angles to each other, rivers flow downhill only, roads are flat.
- Incorporate this knowledge as extra constraints in the optimisation

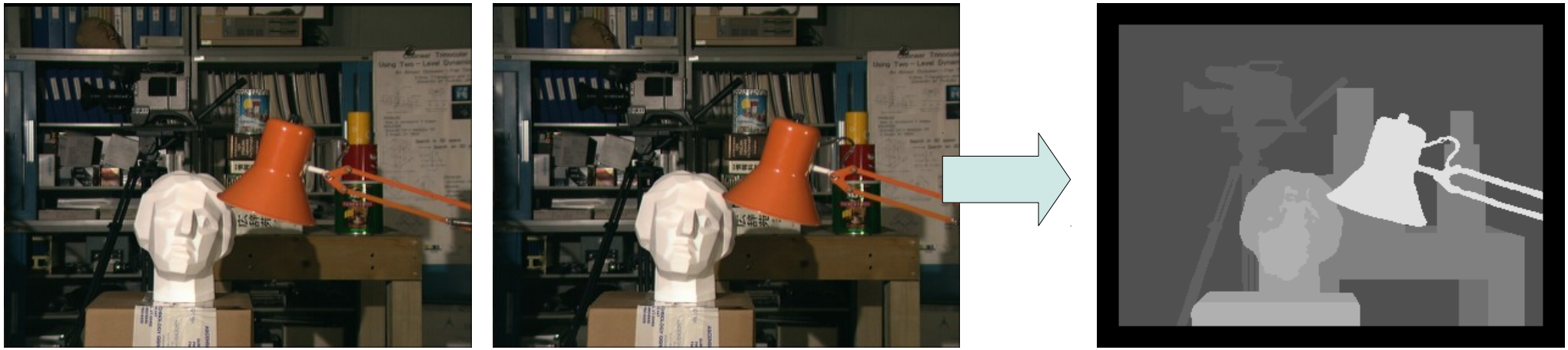


Extensions to BA

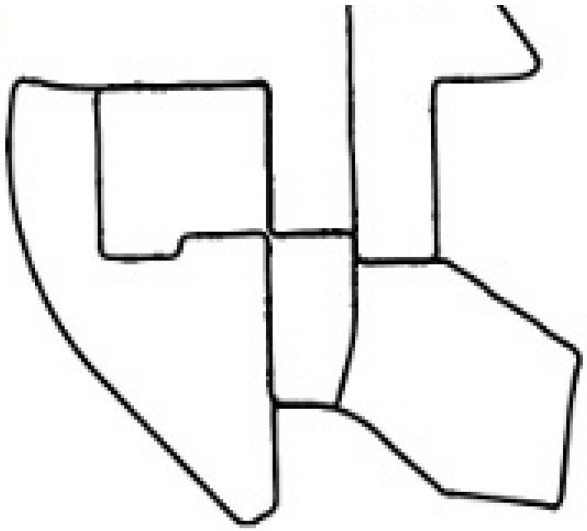
- Sometimes we don't know the camera calibration (e.g. photos or video off the web)
  - Estimate this in the optimisation as well – minimise $C(\{\mathbf{X}_j\},\{P_i\},\{K_i\})$
- Not restricted to point features—line segments or contours work well as well. Also use other 3D measurements from a laser scanner or structured light
- Sometimes tens of millions of features to reconstruct. Design optimisation algorithm carefully to reduce complexity

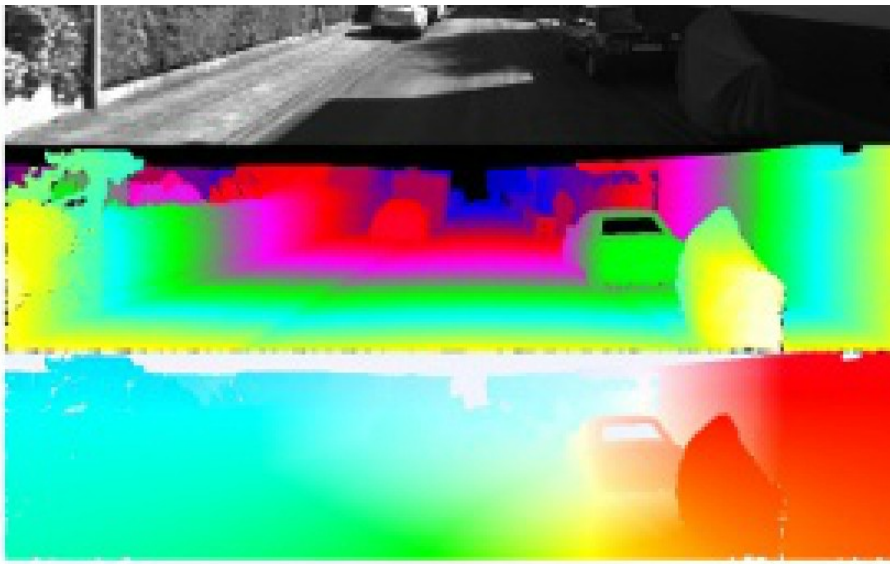
Relationship to dense stereo

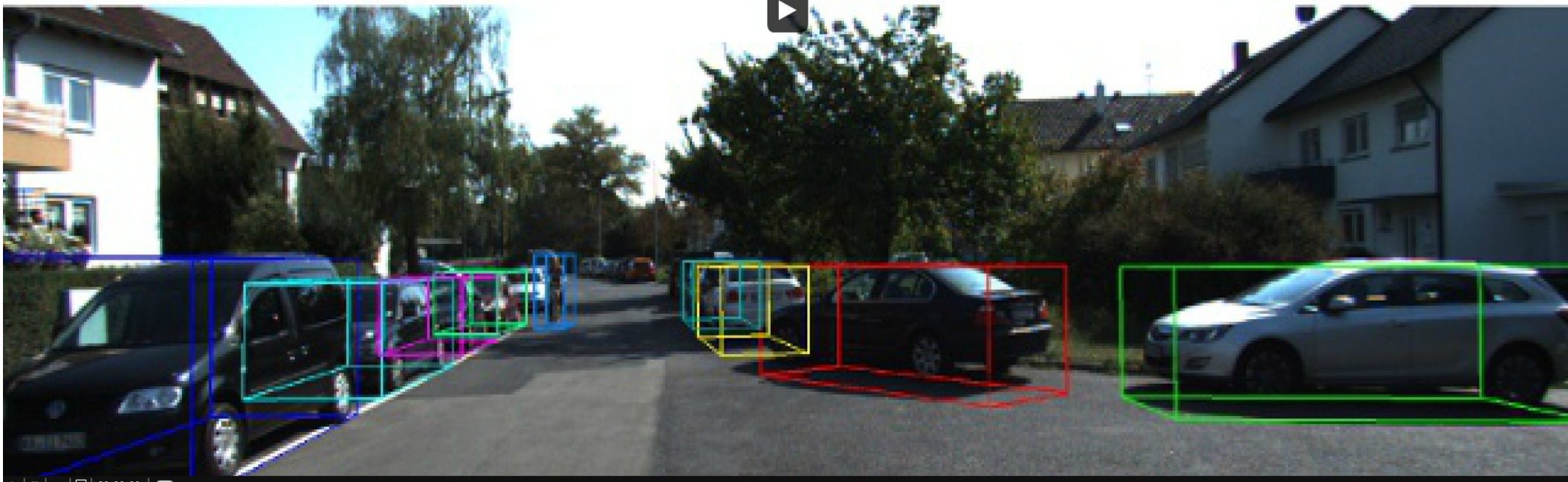
- BA (in its simplest form) finds a sparse structure
  - but can fit a mesh to give a dense reconstruction (assume objects are convex)
- An alternative for 3D reconstruction is to use a dense stereo algorithm, e.g. block-matching on the Bumblebee
  - Input: 2 views + calibration.
    Output: depth map—each pixel has a depth



Dense stereo → Self driving cars







Summary

- Homography H: relates relative pose of 2 cameras viewing a planar scene. Estimate from feature correspondences using RANSAC.
- Essential matrix E: relates relative pose of 2 cameras viewing a 3D scene. Estimate from feature correspondences using RANSAC.
- Bundle adjustment (BA): initialise using RANSAC (for E), estimate a set of 3D points and camera poses which minimises reprojection error.
- Useful applications, exciting research opportunities!