Comparison of Projection, Homography and Fundamental Matrix

COMP4102A

Dr. Gerhard Roth

Winter 2014

Version 1

Projection, Homography, Fund Matrix

- All these three are similar but also different
- Projection – 3d point to 2d pixel
  - Have only one image
- Homography – 2d pixel to 2d pixel
  - Have two images
- Fund matrix – 2d pixel to epipolar line (pixels)
  - Have two images
- Assume we are given intrinsic and extrinsic camera parameters of all the cameras
  - Then we can compute projection, homography and fund. matrix
  - This is done by simple algebraic substitution
- Then we can use these matrices appropriately

Projection, Homography, Fund Matrix

- We can use these matrices for other points
- Projection – project any 3d point to 2d
  - Many to one transformation since any 3d point on the same line from camera center projects to same 2d point
- Homography – transfer any 2d pixel from one image to the other (either way)
  - A one to one transformation between two images
- Fund matrix – transfer any 2d pixel to an epipolar line (pixels) in other image
- So we can apply our matrices to other pixels (or 3d points) than was used to create them!

Projection, Homography, Fund Matrix

- Assume we do not know either the intrinsic or the extrinsic camera parameters
- We are only given correspondences
- Projection matrix – 3d points to 2d pixels
- Homography – 2d pixels to 2d pixels
- Fund Matrix – 2d pixels to 2d pixels
- Then we can compute these matrices
  - Assuming that we are given enough correspondences
  - Stack all correspondences in an A matrix
  - Find eigenvector associated with smallest eigenvalue of the matrix A transpose A (this is the solution to Am = 0)
  - For correct fund matrix must do some post processing

Projection, Homography, Fund Matrix

- Computed a projection, homography or fund matrix from a set of correspondences
- We know what is in these matrices if we are given intrinsic and extrinsic parameters
  - Just some variables of these parameters (your assignment)
- We can decompose our matrix (all three)
  - Take numerical values and compute the intrinsic/extrinsic parameters that must exist to create these numbers!
  - For projection matrix we can compute intrinsic parameters
  - For homography and fund matrix assume intrinsic given
  - Projection matrix => Intrinsic and extrinsic
  - Homography => Plane orientation and location or rotation
  - Fund matrix – Extrinsic – correct R, but T up to a scale

Projection, Homography, Fund Matrix

- Each of these four matrices (P, H, F and E) are part of a homogeneous system
- So we can multiply each element in these matrices by the same value and still have exactly the same matrix!
- Therefore all scaled versions of the same matrix are equivalent (the same matrix)
- Important to understand degrees of freedom
- For example, homography matrix H has nine elements but only 8 degrees of freedom
- Need 4 x 2d correspondences = 8 in total

Homography – to camera pose

- Assume we have a stored 2d picture
- Want to find this picture in the image
  - Basically find_obj.cpp in the opencv examples
- Find surf features in both images
  - Then correspond surf features by using surf descriptor
- Find homography (consensus algorithm)
  - From this homography we can compute the camera pose (the rotation and translation relative to the plane in image)
  - Use very similar approach to what is used for camera calibration, but assume that K matrix is known
  - This is reasonable because we usually calibrate the camera in applications such as augmented reality

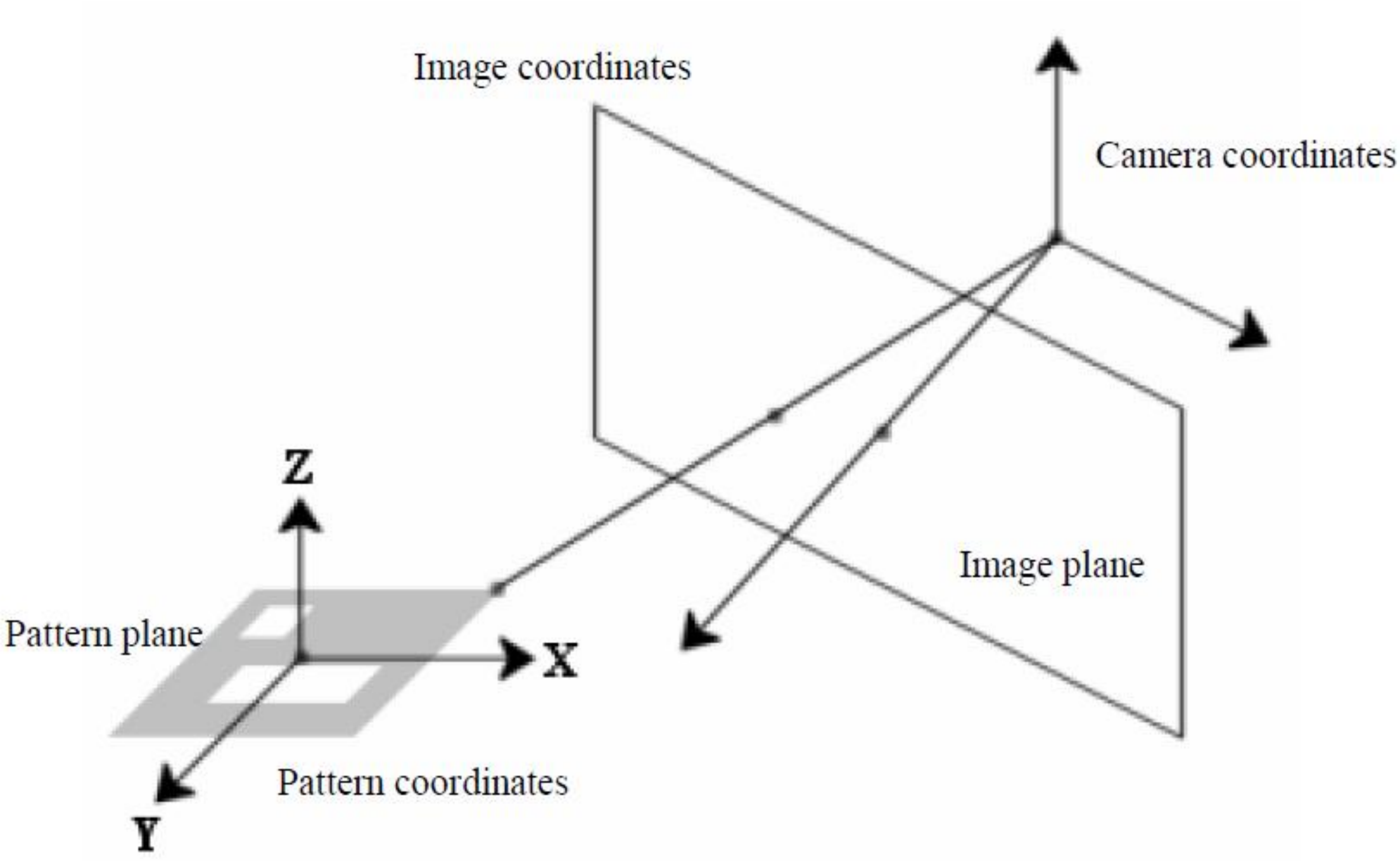
Homography – to camera pose

- Assume Z = 0 is on the plane in the world

$$\mathbf{M} = \begin{bmatrix} -f_x r_{11} + o_x r_{31} & -f_x r_{12} + o_x r_{32} & -f_x T_x + o_x T_z \\ -f_y r_{21} + o_y r_{31} & -f_y r_{22} + o_y r_{32} & -f_y T_y + o_y T_z \\ r_{31} & r_{32} & T_z \end{bmatrix}$$

- We know K, so know fx, fy, ox, and oy
- Then we can compute R, and T using the characteristics of Rotation matrices
- Similar (but not identical) to what was done to compute R, T for camera calibration

Homography – Assume proper Z = 0



• Need camera R, and T to augment

- With R and T we can augment (need this to properly draw a 3d virtual object)
- Homography (but no R,T) is enough for drawing a 2d augmentation (planar pattern)

Fundamental matrix – to camera pose

- Computed fund matrix between two images using a set of correspondences
  - Find surf features in both the left and right images
  - Correspond the surf features using surf descriptors
  - Find the fundament matrix (consensus algorithm)
- Assume we also know K for both images
  - Then since $\mathbf{F} = \mathrm{K}_r^{-\mathbf{T}} \mathbf{E} \mathrm{K}_l^{-1}$ we can compute E
  - But E = R S where S encodes translation between images
  - Can use SVD to decompose R into R and S (therefore T)
  - But we only know T up to a scale factor, why?
  - Because if we double T but also double distance to all the corresponding features we have the exact same images
  - Also scaling T produces same E because of homogenity!

Two View Reconstruction

- We have a set of correspondences that we used to compute the fundamental matrix
- Below are the correspondences that we used

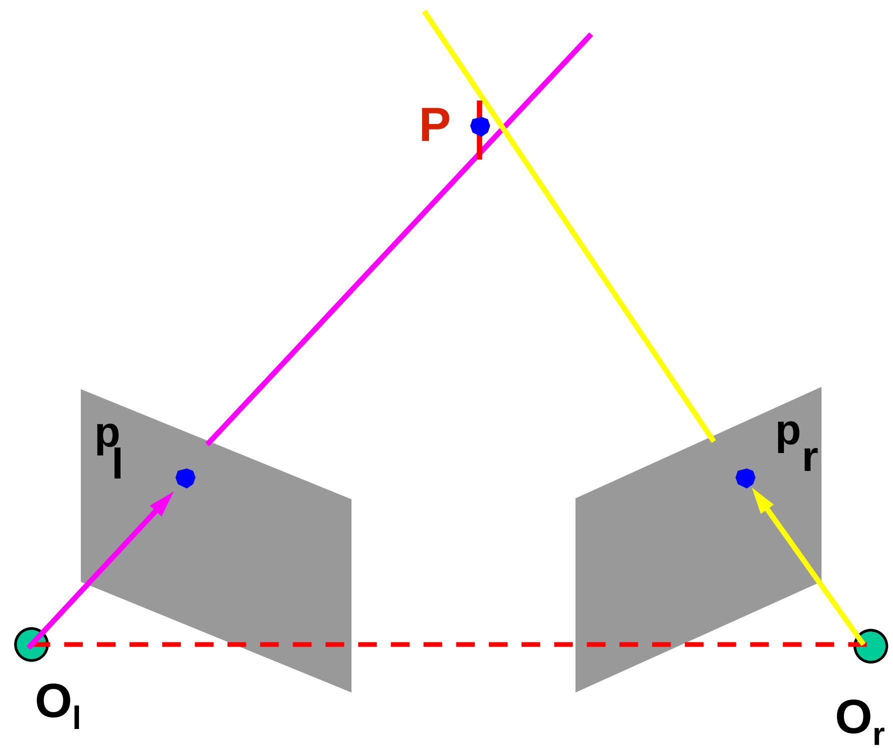
Two view reconstruction

- Using the intrinsic and extrinsic camera calibration and the pixel co-ordinates of the matching feature points we need to compute the 3D location of that feature point
- Can be done using simple geometric triangulation (p. 162, Ch. 7 of book)
  - Set up a vector equation by inspection
  - Then solve the associated linear system to get the 3d co-ordinates of the point P in space
  - Actually find the point that is closest to both of the rays from the origin of the camera through the projection of the points
  - Very simple and efficient process to solve for P

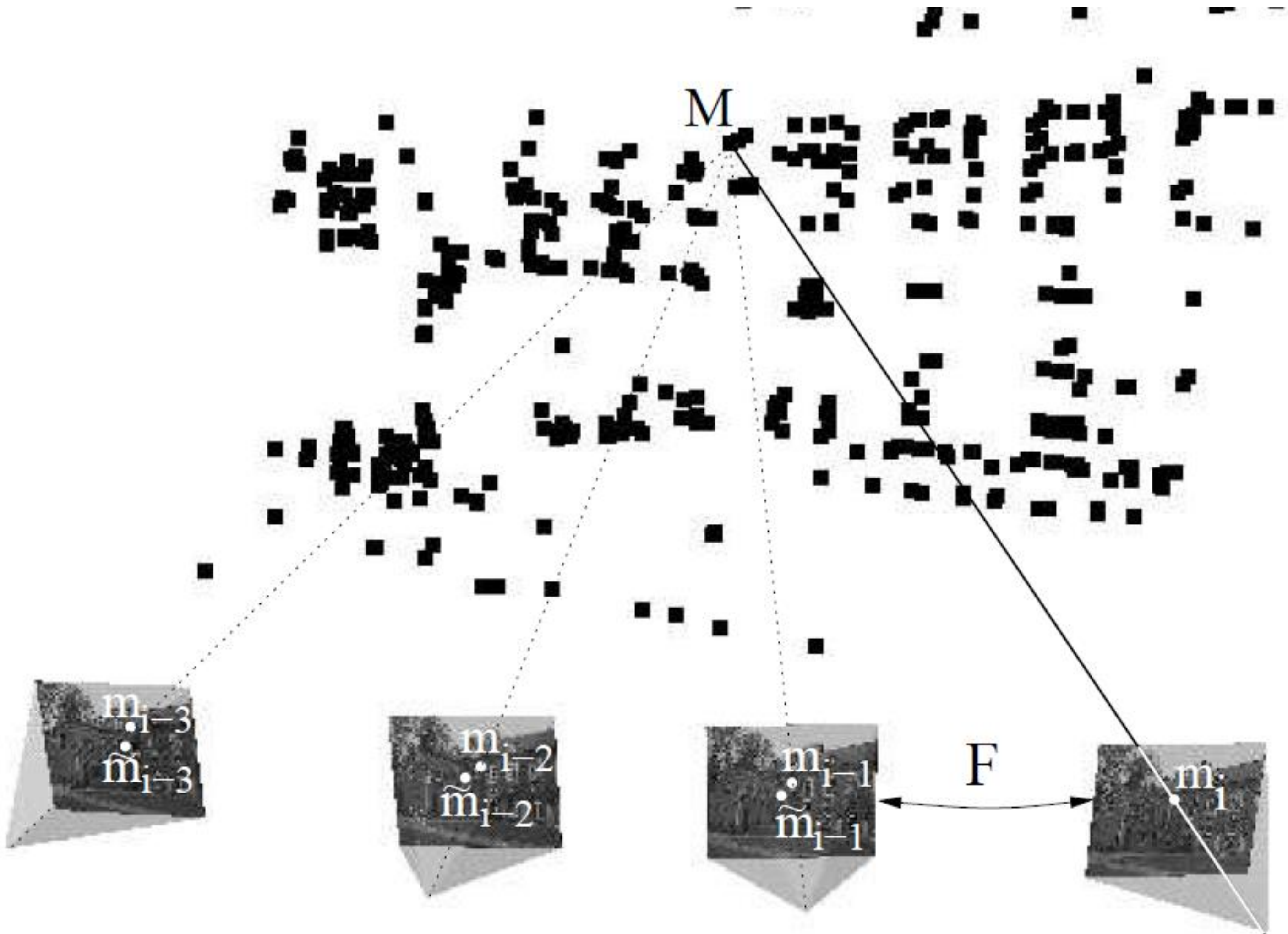
Known intrinsic and extrinsic

Solution

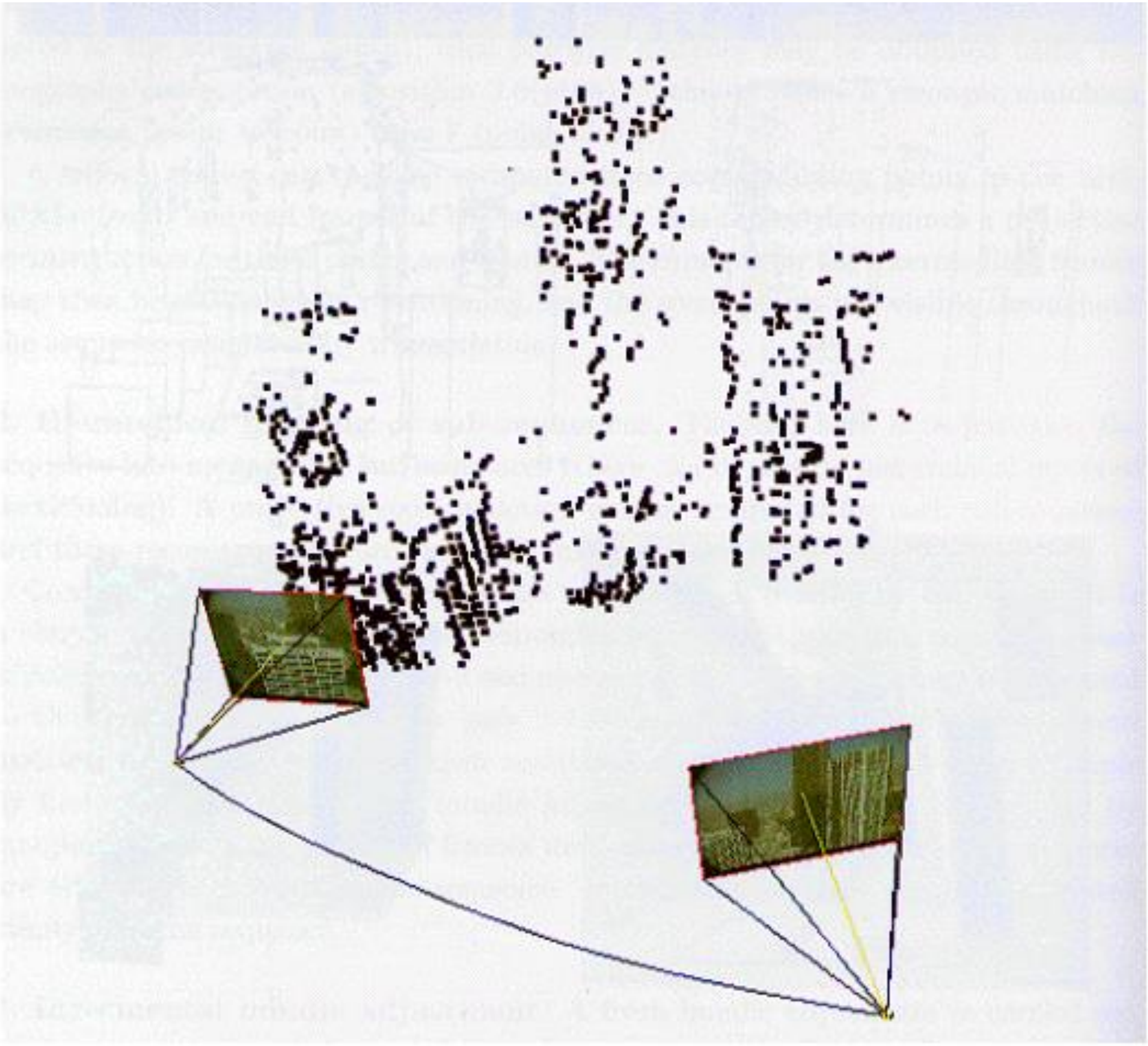
- Triangulation: Two rays are known and the intersection can be computed
- Problem: Two rays will not actually intersect in space due to errors in calibration and correspondences, and pixelization
- Solution: find a point in space with minimum distance from both rays
- We repeat this triangulation process for all corresponding points



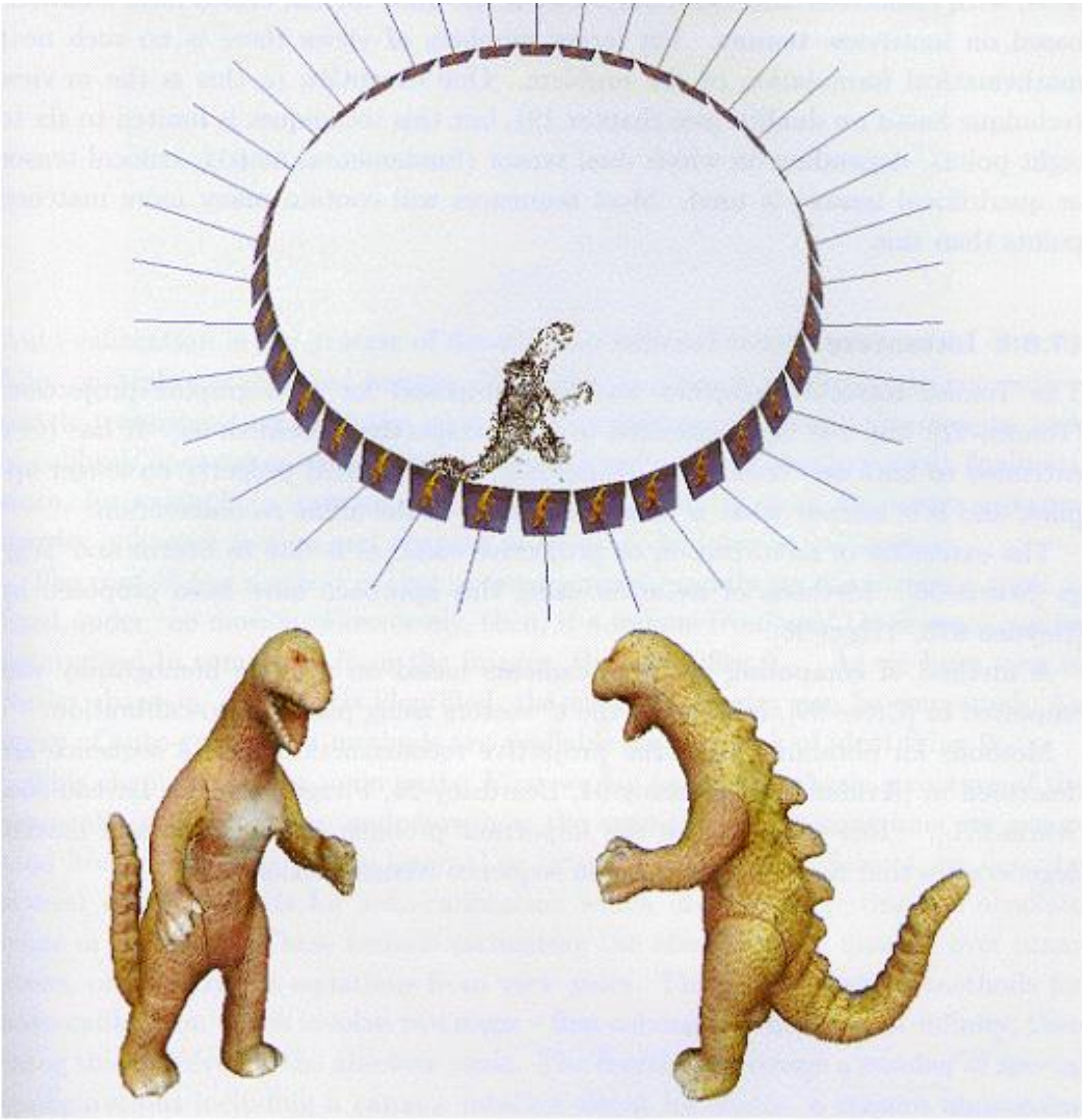
3d Reconstruction example



3d Reconstruction example



3d Reconstruction example



• Homography – relation to fund matrix

- Fund matrix requires camera translation
- If we take two cameras and slowly reduce translation to zero we have homography
- The only remaining motion is the translation
- In this situation, with rotation only we have a standard homography
- How can we tell from just correspondences which situation has occurred??
- Not so easy, best to use quality of matches
    - Look at how many feature points support homography and fund matrix, largest support is correct answer

All together – from images to models

- Input is a set of images, output a model!
- Basic steps
  - Find surf/sift features in all images
  - For all image pairs match the features
  - Compute fundamental matrix using these features
  - Find R, T (up to scale) for every image pair
  - Now rectify every image pair
  - Find dense depth (simple stereo) for each image pair
  - Make points into triangles (not need to know this step)
  - Texture the triangles (not need to know this step)
- Some of these steps are many years of work

Images to Models – Camera Hardware

- Two main possibilities
  - Single camera that you move around
  - Stereo head or active sensor
- To obtain depth from single camera we must have some translation between images
  - Otherwise this is a homography situation (rotation only)
- Depth depends on the camera motion
  - This technology is called “structure from motion”
- If you have stereo camera or active sensor always obtain depth with a certain accuracy
  - So you can rotate the stereo/active sensor and still get depth because depth comes from sensor, not the motion
  - No need to worry about rotations, any motion will do!