AMME4710: COMPUTER VISION AND IMAGE PROCESSING WEEK 10

Dr. Mitch Bryson

School of Aerospace, Mechanical and Mechatronic Engineering, University of Sydney

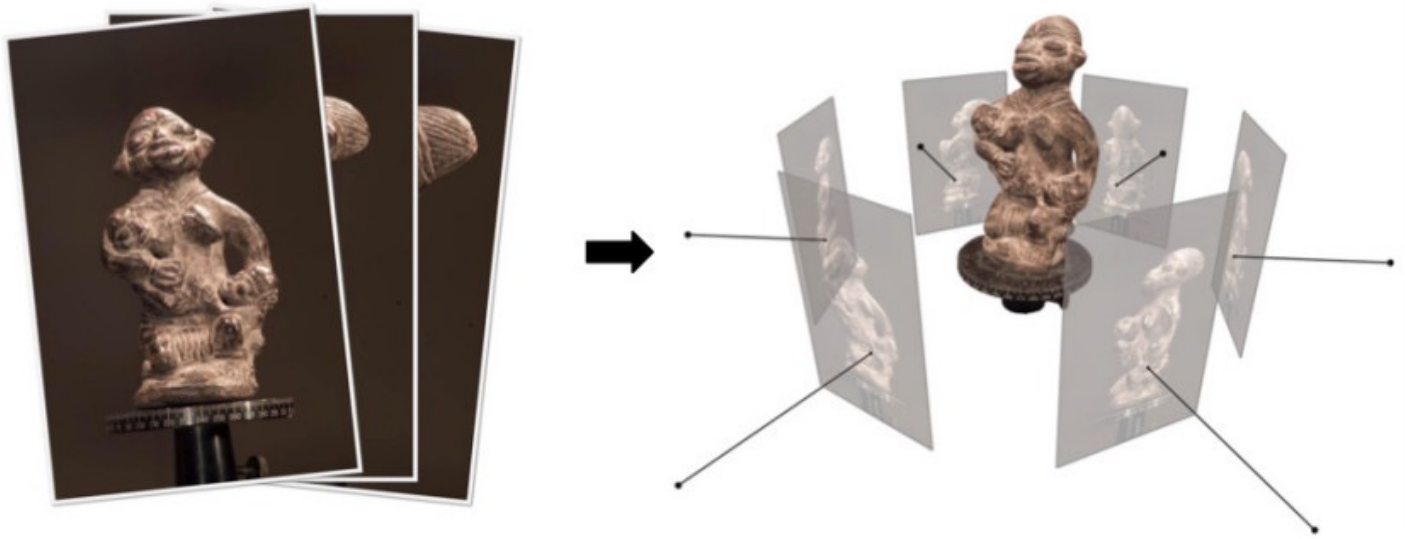
3D Structure from Motion

- In week 5 we examined algorithms for stereo-vision for which we could triangulate 3D world points using corresponding image points in each camera in a stereo pair
- It's also possible to estimate 3D points from two or more monocular images (with appropriate parallax) without knowing the relative pose between camera a-priori:
  - Algorithms must estimate both the 3D world points and relative camera poses simultaneously in a process called **Structure-from-motion**

N. Snavely, S.M. Seitz, R. Szeliski, "Modeling the World from Internet Photo Collections", International Journal of Computer Vision, 2007

3D Structure from Motion

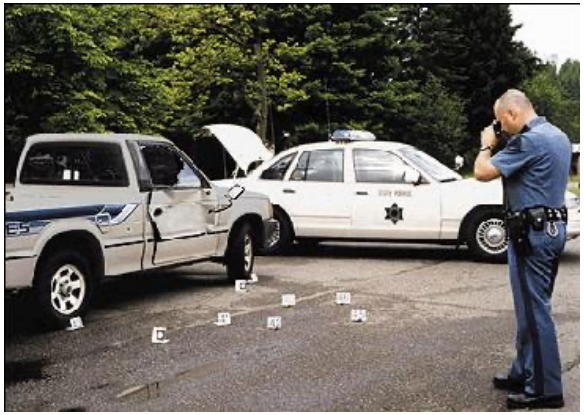
- Structure-from-motion algorithms use a collection of images from various perspectives to estimate the 3D locations of points and camera poses:
  - Feature points are extracted in multiple images (i.e. SIFT/SURF) and matched across images
  - These correspondences are used to estimate the 3D points of these features and camera positions and rotations
  - Poses and features are then used to create 3D models which can be “photo-textured” or coloured using the image data

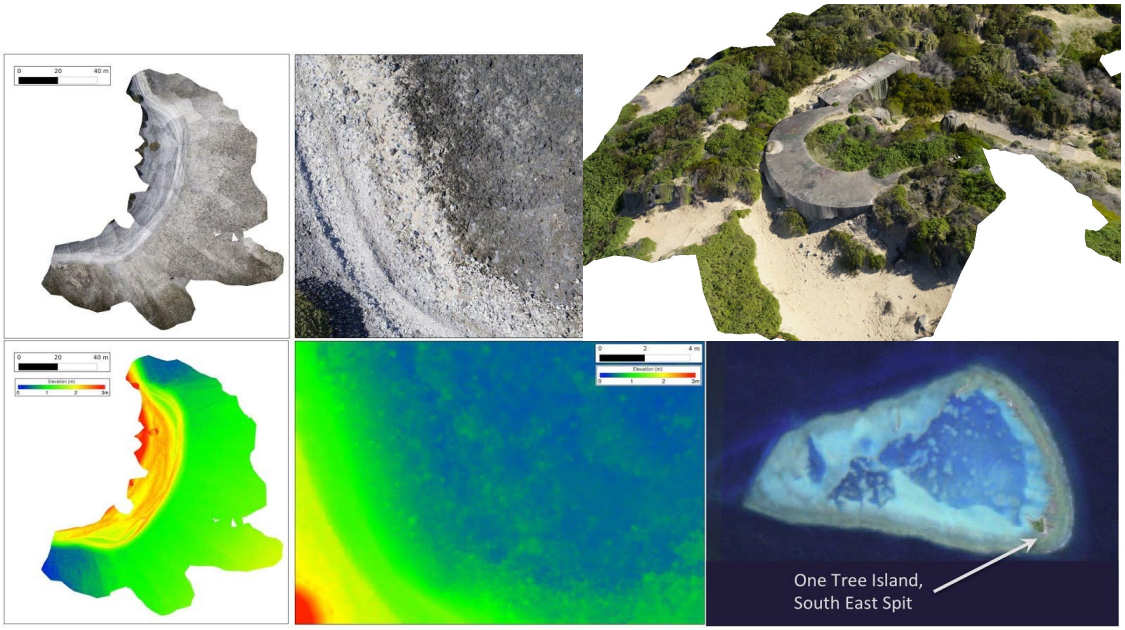


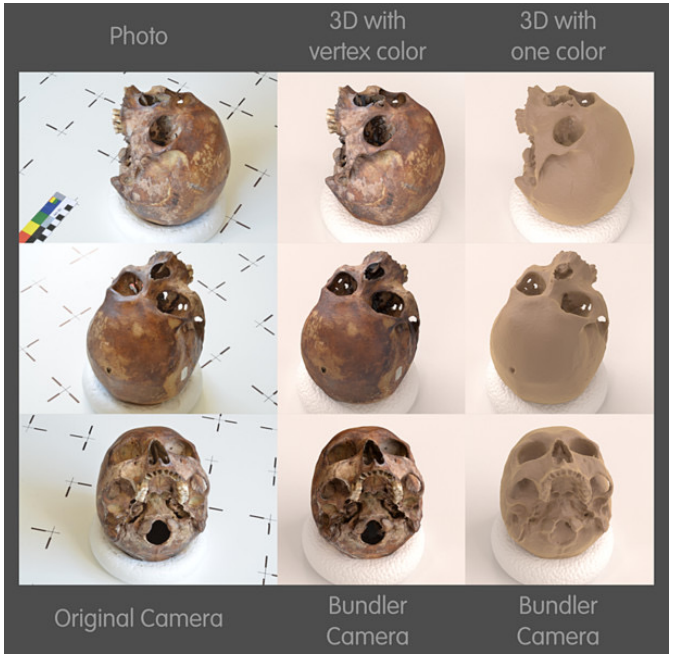
Y. Furukawa, C. Hernandez, ”Multi-view stereo: A Tutorial”, Found. and Trends in Computer Graphics and Vision, 9:1-2, 2013

3D Structure from Motion

- Applications:
  - Aerial Photogrammetry: Mapping, Geoscience,
  - Archaeology and Forensics
  - Manufacturing and Quality Control
  - Film, Games and Computer-generated Imagery
  - Robotics and Image-based Navigation

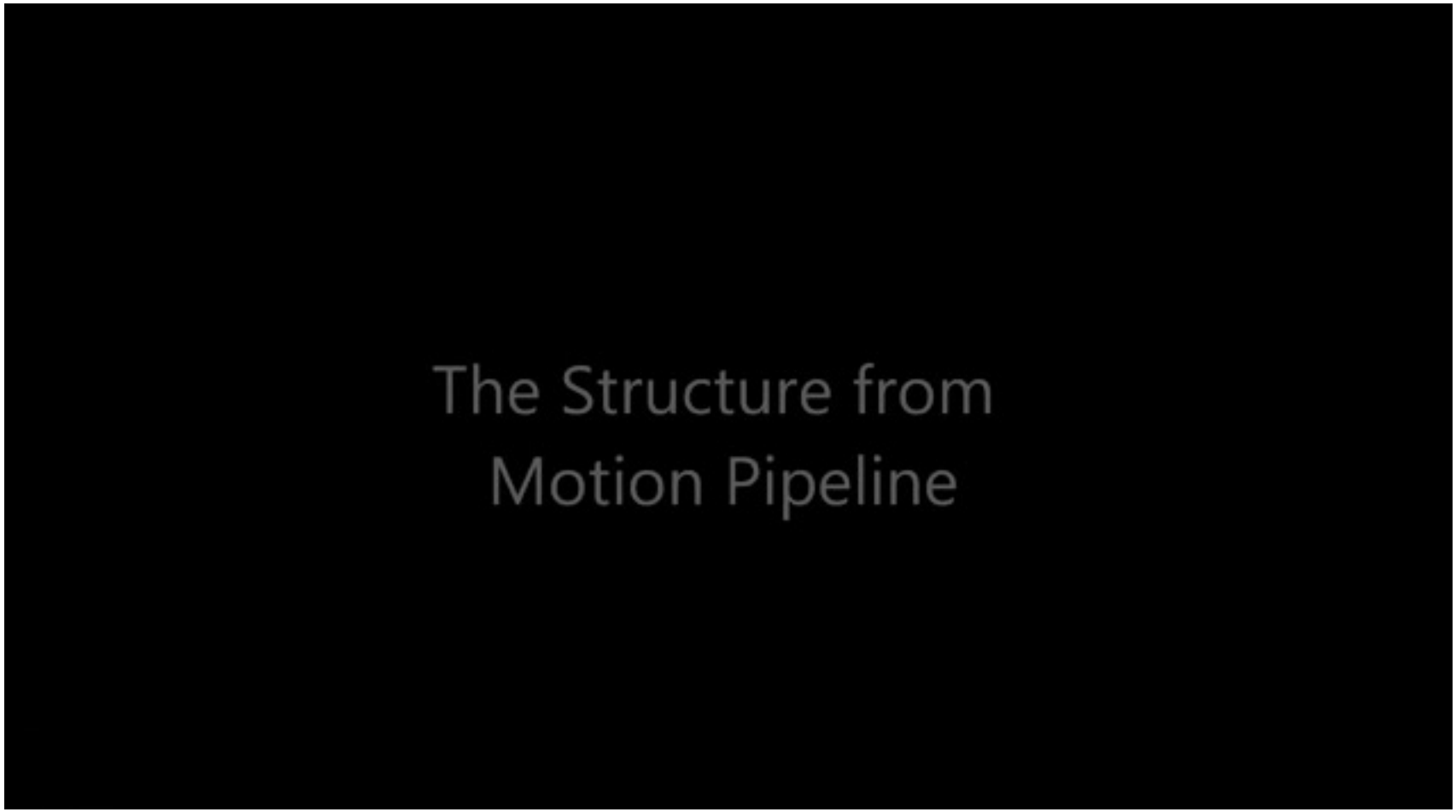




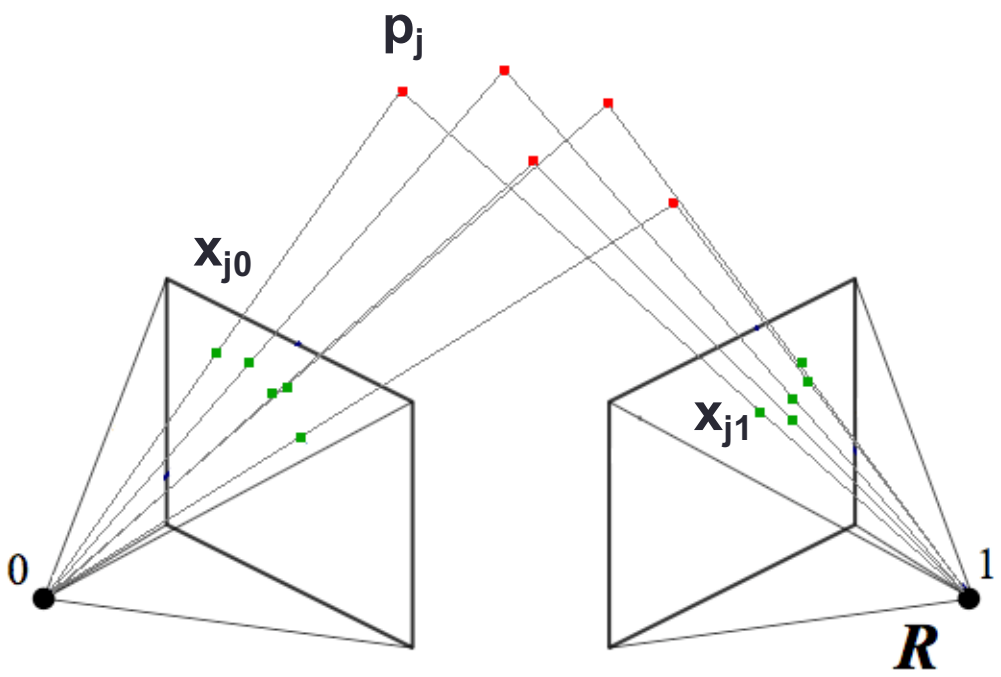


Courtesy of Cicero Moraes (CC BY-SA 3.0)

3D Structure from Motion

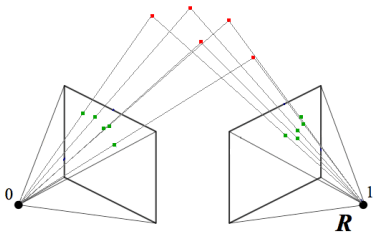


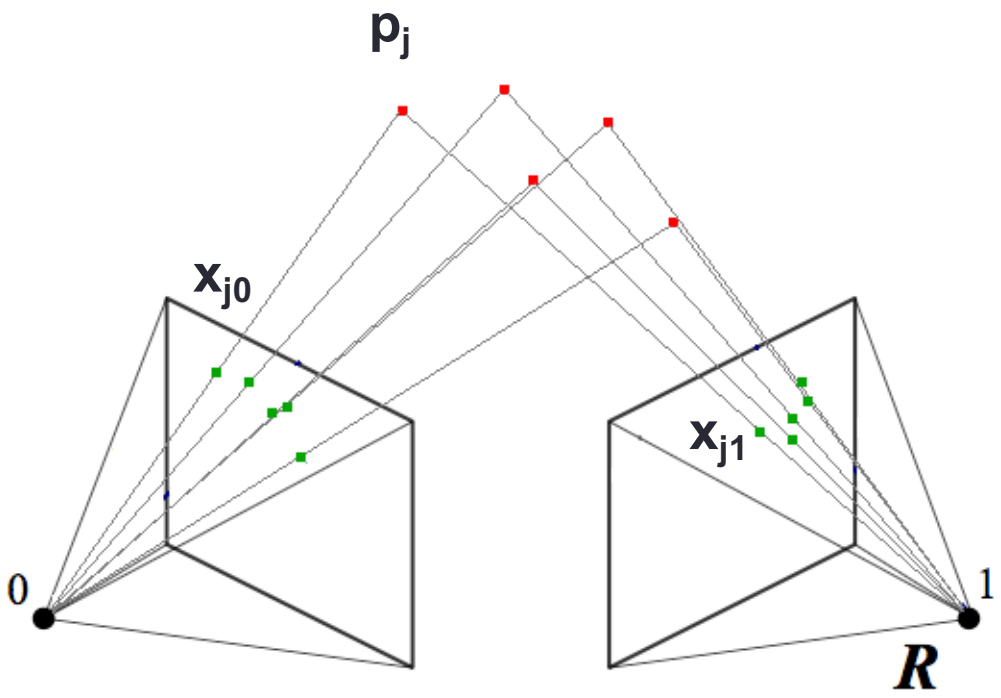
Two-view structure from motion



- Two-view structure-from-motion uses corresponding pairs of image points in two different images to estimate the relative rotation and translation (R,t) between the cameras:
  - 3D point locations can then be solved by triangulation

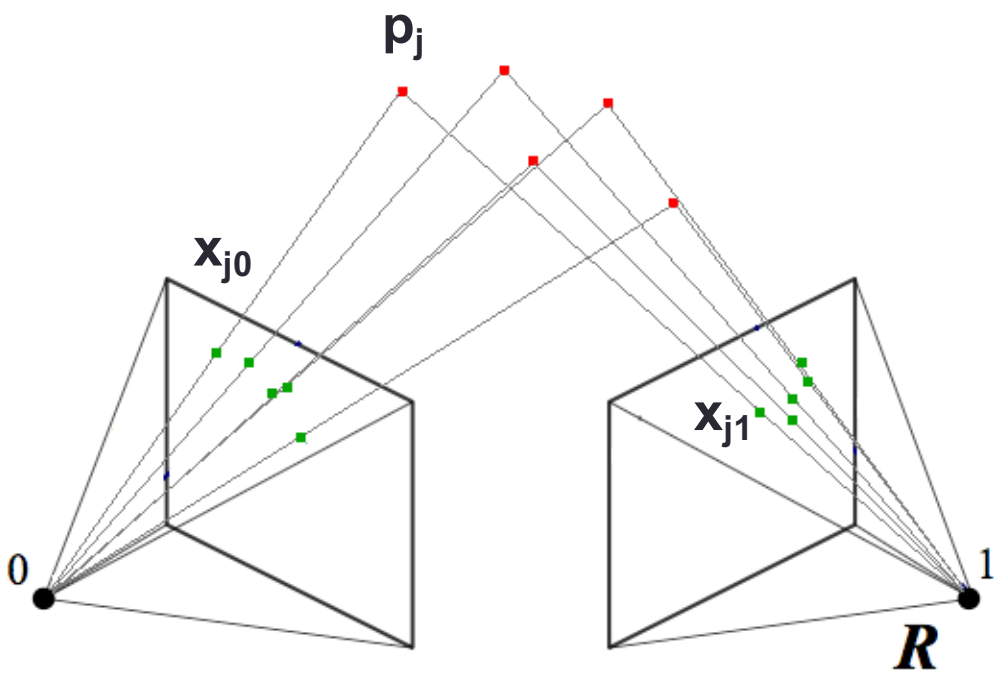
Two-view structure from motion





- The rotation R between cameras can be solved exactly, however the translation vector between cameras can only be solved up-to-scale:
  - Scale the vector t and 3D positions P_j by a constant scale factor s and the image points don't change, hence scale is not observable

Two-view structure from motion



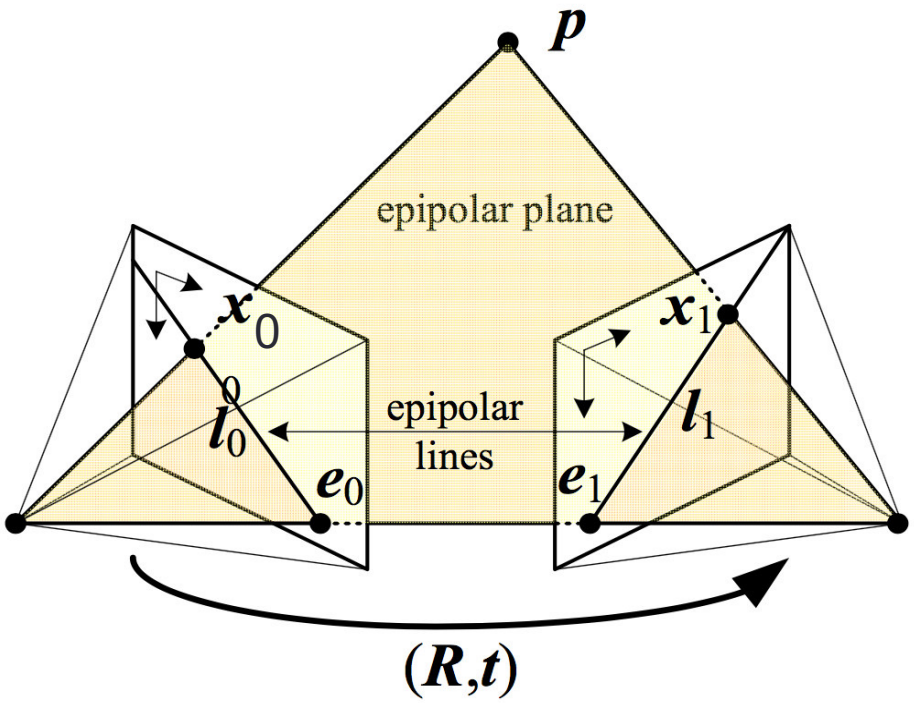
- Hence (up-to-scale) there are:
  - 5 unknowns for the pose (3 DoF in rotation and 2 DoF in translation)
  - 3N unknowns for the features
- For N features there 4N observations (pixel coordinates)
- Hence we need a minimum of 5 matching pairs to solve the problem
  - Solutions commonly use more (for redundancy and solution stability)

Review: Epipolar Geometry

- The epipolar constraint relates the possible locations of an image point from one image to the next
- The essential matrix **E** relates all pairs of normalised image coordinates in each image
- We can also formulate a similar relationship by using un-normalised (pixel coordinates) which holds even in cases for which we do not know the intrinsic parameters of each camera:

$$\mathbf{x}_1^T \mathbf{E} \mathbf{x}_0 = 0$$

$$\mathbf{E} = [\mathbf{t}\times]\mathbf{R}$$



$$[u, v, 1]_1 \mathbf{F} [u, v, 1]_0^T = 0$$

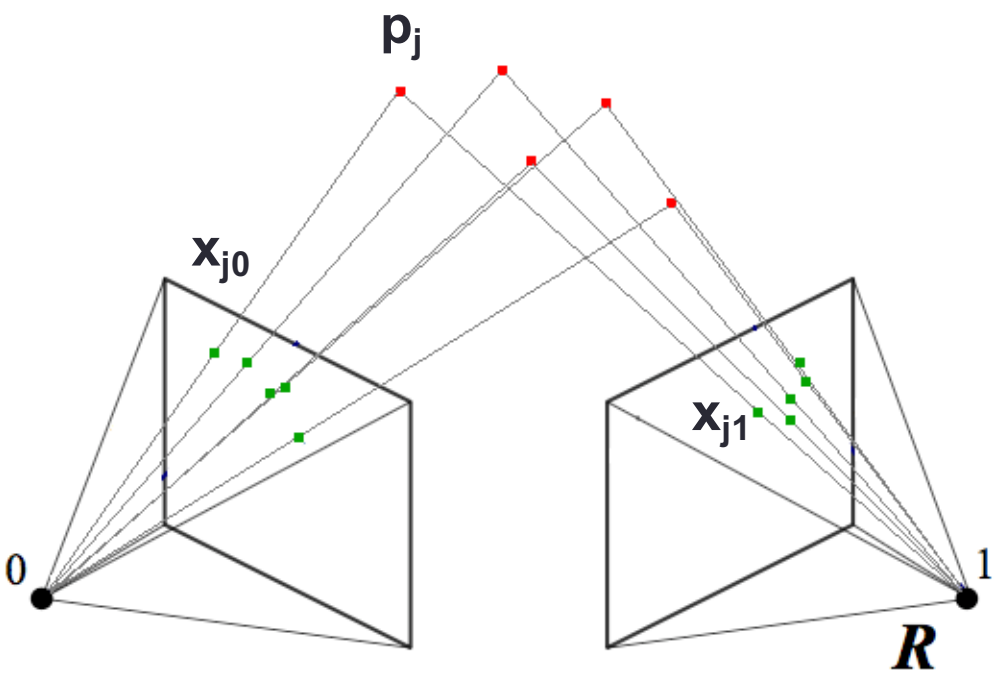
$$\mathbf{F} = \mathbf{K}_1^{-T} \mathbf{E} \mathbf{K}_0^{-1}$$

Where **F** is known as the **fundamental matrix**

Two-view structure from motion

$$\mathbf{E} = [\mathbf{t}\times]\mathbf{R}$$

$$\mathbf{E} = \mathbf{K}_1^T \mathbf{F} \mathbf{K}_0$$



- If we can compute the essential matrix **E**, the relative translation and rotation can be solved from this by matrix decomposition
- If the camera intrinsic matrix **K** is known, then we can compute **E** from **K** and the fundamental matrix **F**
- As discussed in week 5, we can compute **F** from sets of image point correspondences

Computing F: The eight-point algorithm

- Recall that the fundamental matrix F encodes the epipolar constraint between two matching feature points:

$$[u, v, 1]_1 \mathbf{F} [u, v, 1]_0^T = 0$$

- For the j[th] correspondence, we can rearrange this equation into the form:

$$\begin{bmatrix} u_{j0}u_{j1} & v_{j0}u_{j1} & u_{j1} & u_{j0}v_{j1} & v_{j0}v_{j1} & v_{j1} & u_{j0} & v_{j0} & 1 \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{bmatrix} = 0$$

- F has nine terms (3x3 matrix), but we can compute its terms up to an unknown scale factor given at least eight correspondences (eight separate constraint equations)

Computing F: The eight-point algorithm

- Given we have N > 8 correspondences we can form an over-determined system of equations:

$$\begin{bmatrix} u_{00}u_{01} & v_{00}u_{01} & u_{01} & u_{00}v_{01} & v_{00}v_{01} & v_{01} & u_{00} & v_{00} & 1 \\ u_{10}u_{11} & v_{10}u_{11} & u_{11} & u_{10}v_{11} & v_{10}v_{11} & v_{11} & u_{10} & v_{10} & 1 \\ & & & \vdots & & & & & \\ u_{N0}u_{N1} & v_{N0}u_{N1} & u_{N1} & u_{N0}v_{N1} & v_{N0}v_{N1} & v_{N1} & u_{N0} & v_{N0} & 1 \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\mathbf{W}\mathbf{f} = \mathbf{0}$$

- W is a Nx9 matrix and is rank-deficient: we can compute a solution using Singular Value Decomposition (SVD)
  - Any column of V whose corresponding singular value is zero is a valid solution for F

$$\mathbf{W} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T \qquad \hat{\mathbf{f}} = V(:, 9)$$

Assuming the 9th column of V corresponds to the singular value

Computing F: The eight-point algorithm

- Because of errors in pixel locations, the estimate of the fundamental matrix $\hat{\mathbf{F}}$ (derived from $\hat{\mathbf{f}}$) will potentially have full rank: the rank of **F** should be 2 (when constructed from the exact relative pose and intrinsic matrices), hence a better approximation can be achieved by computing a rank-2 approximation from $\hat{\mathbf{F}}$, using an SVD of $\hat{\mathbf{F}}$:

$$\hat{\mathbf{F}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

$$\mathbf{F} = U \begin{bmatrix} \Sigma_1 & 0 & 0 \\ 0 & \Sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T$$

- In practice, the eight-point algorithm can result in ill-conditioning, depending on the locations of image pixels:
  - A variation on the approach (referred to as the normalised eight-point algorithm) is typically used in practice, which preconditions the pixel locations by transformations that normalise the image offset and scaling (later accounted for in the computation of F)
- MATLAB provides an implementation of a normalised eight-point algorithm for computing F in the function `estimateFundamentalMatrix`

Two-view structure from motion

- Once F is estimated, we can compute the essential matrix E using the intrinsic matrix of each camera (for example acquired using single-camera calibration):

$$\mathbf{E} = \mathbf{K}_1^T \mathbf{F} \mathbf{K}_0$$

- The rotation matrix and translation vector can be recovered from a decomposition of E using an SVD: $\mathbf{E} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$

$$\mathbf{R} = \mathbf{U}\mathbf{W}\mathbf{V}^T \text{ or } \mathbf{U}\mathbf{W}^T\mathbf{V}^T$$

$$\mathbf{t} = \pm\mathbf{U}(:, \mathbf{3})$$

where $\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

- Assuming the two non-zeros singular values of ∑ are equal to 1: this can be achieved by scaling E by the inverse of the average of the non-singular values prior to re-computing the SVD

Two-view structure from motion

$$\mathbf{R} = \mathbf{U}\mathbf{W}\mathbf{V}^T or \mathbf{U}\mathbf{W}^T\mathbf{V}^T$$

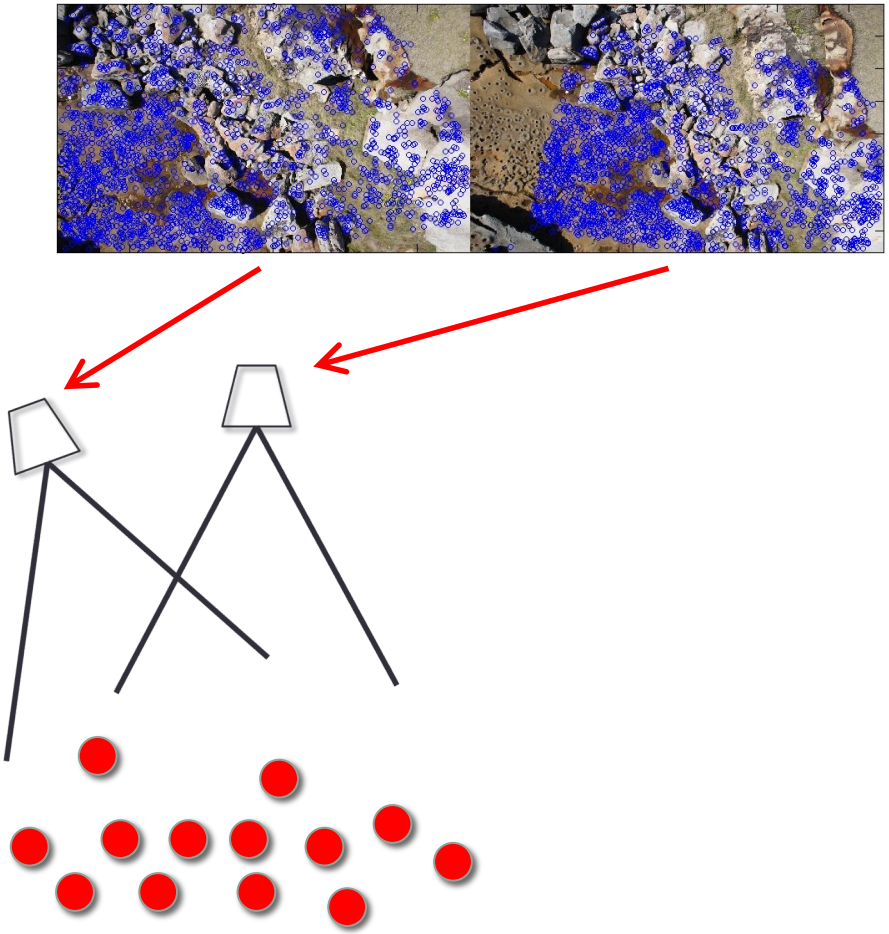
$$\mathbf{t} = \pm\mathbf{U}(:, \mathbf{3})$$

- There are two values for R and two values for t that satisfy the original matrix $\mathbf{E} = [\mathbf{t}\times]\mathbf{R}$

- This results in four possible combinations of (R,t), however only one of these combinations will result in feature triangulations that are consistently in front of both cameras
- The correct (R,t) is recovered by finding the pair that triangulates the most number of features in-front of both cameras (i.e. positive z-coordinate)

Extending to multiple views

- The process can be extended to multiple views of a scene by “chaining” relatives poses together
- Starting from two views, an initial set of 3D points and two poses are estimated
  - By default, the fixed world reference frame is defined as being aligned to the first camera

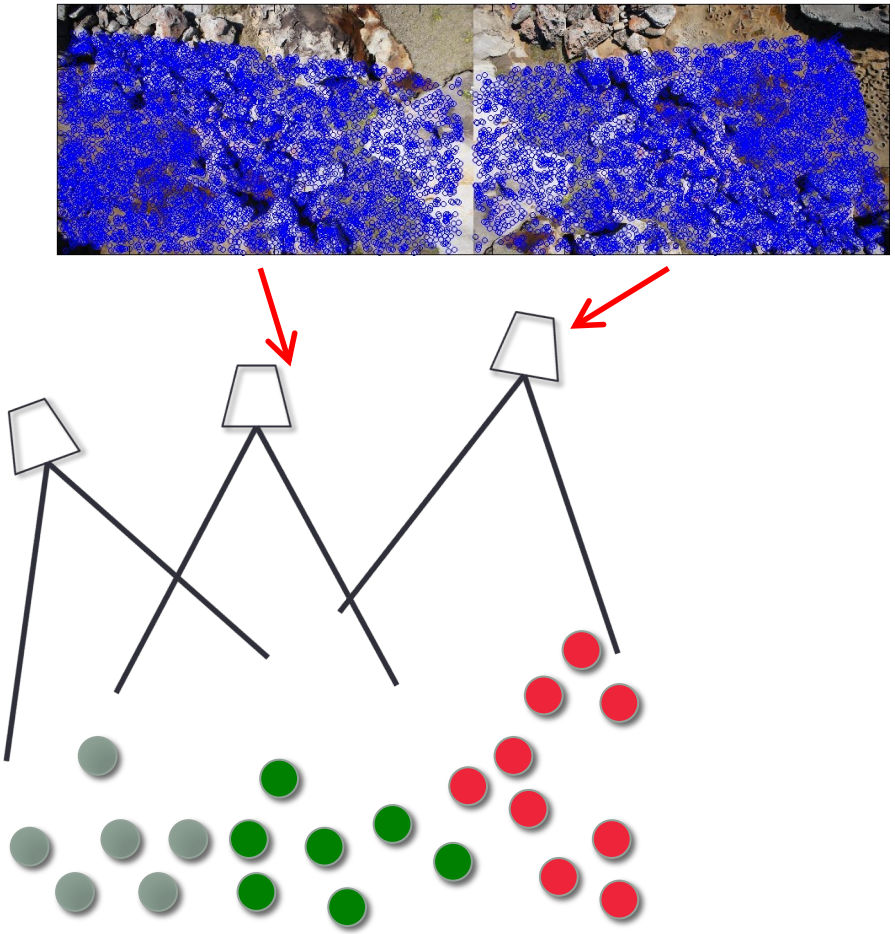


Extending to multiple views

- A third view is added and compared to the second view: the relative pose of the third view is estimated with respect to the second view $(\mathbf{R}_2^3, \mathbf{t}_{23})$
- The world-referenced pose of the second view is then:

$$\mathbf{R}_3 = \mathbf{R}_2^3 \mathbf{R}_2$$

$$\mathbf{t}_3 = \mathbf{R}_2^3 \mathbf{t}_2 + \mathbf{t}_{23}$$



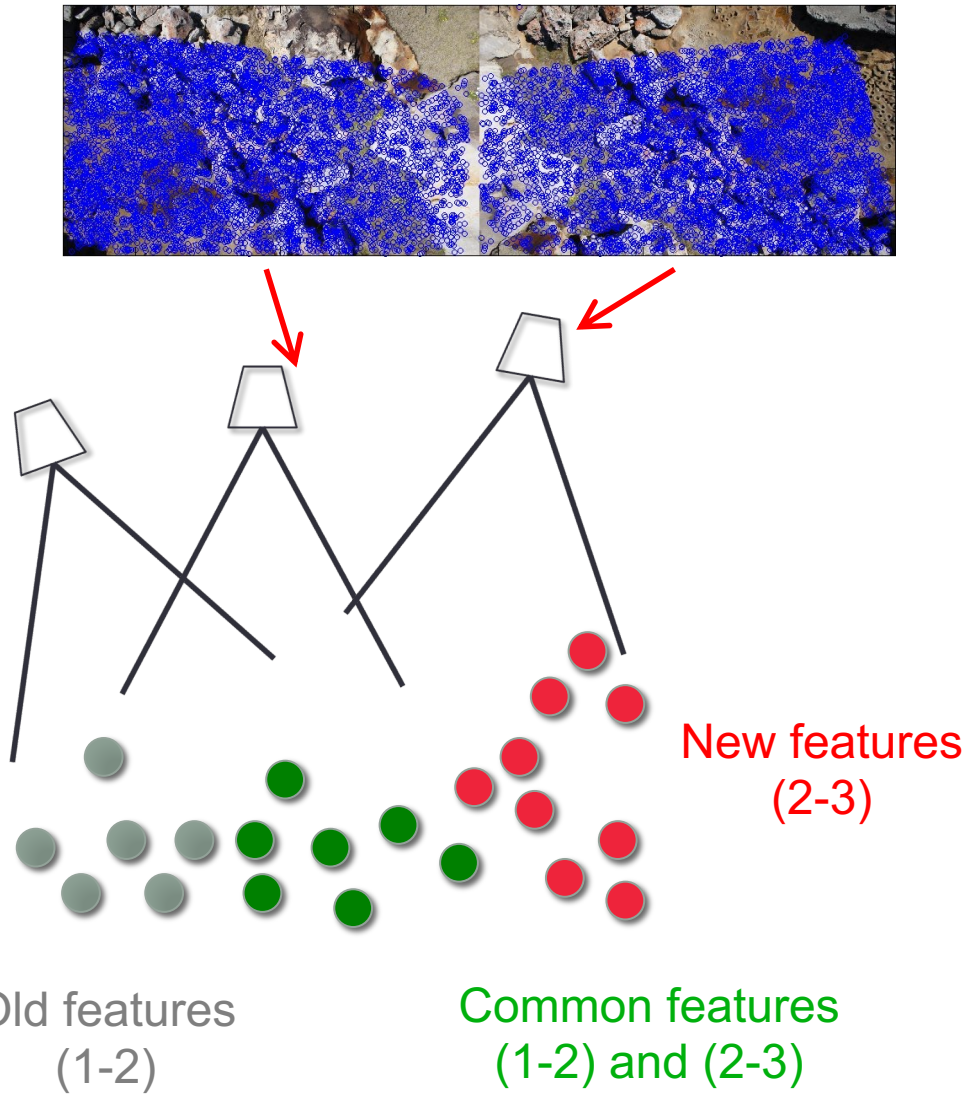
Extending to multiple views

- A third view is added and compared to the second view: the relative pose of the third view is estimated with respect to the second view $(\mathbf{R}_2^3, \mathbf{t}_{23})$
- The world-referenced pose of the second view is then:

$$\mathbf{R}_3 = \mathbf{R}_2^3 \mathbf{R}_2$$

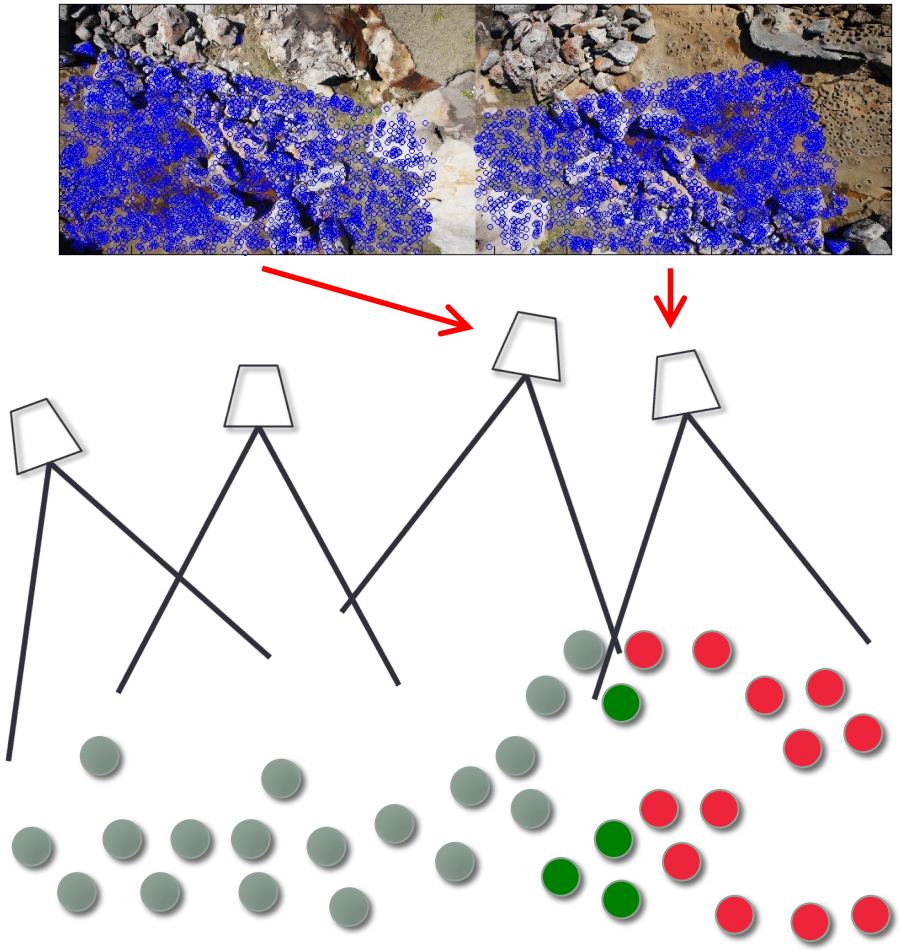
$$\mathbf{t}_3 = \mathbf{R}_2^3 \mathbf{t}_2 + \mathbf{t}_{23}$$

- Some features between poses 1 and 2 are in common with those between 2 and 3: these are matched and used to ensure the scale factors between all poses are consistent



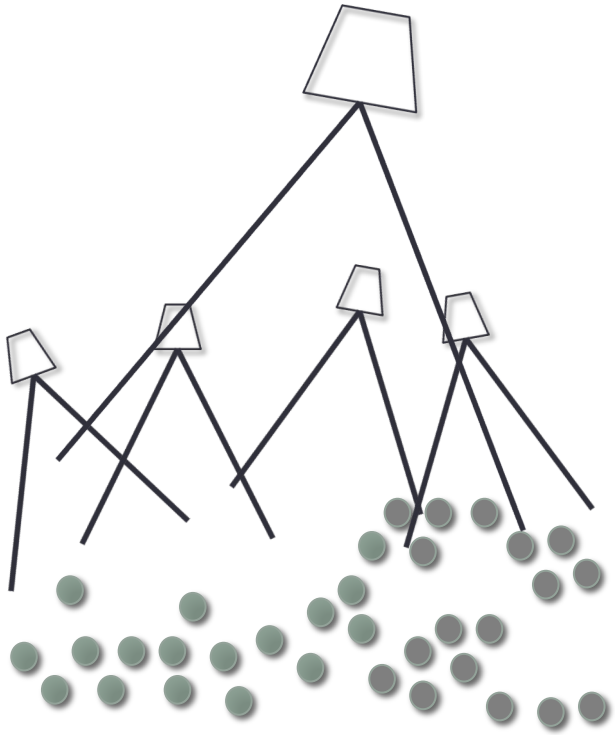
Extending to multiple views

- The process is repeated for a fourth camera: common features that have already been triangulated are used to maintain scale consistency and the relative pose is chained to solve for the pose relative to world coordinates of each new camera



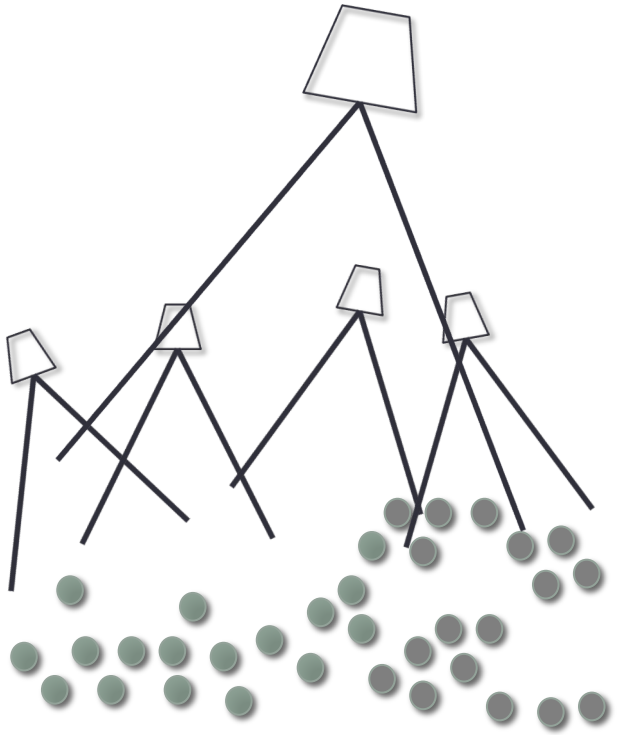
Bundle Adjustment

- The multi-view structure-from-motion method described has several disadvantages:
  - Relative pose estimates can be inaccurate due to ill-conditioning (small numbers of feature matches of poor parallax motion between frames)
  - Only accounts for feature matches between subsequent frames, i.e. ignores that fact that the 1st camera and the 5th or 20th camera may have many features in common)



Bundle Adjustment

- The multi-view structure-from-motion method described has several disadvantages:
  - Relative pose estimates can be inaccurate due to ill-conditioning (small numbers of feature matches of poor parallax motion between frames)
  - Only accounts for feature matches between subsequent frames, i.e. ignores that fact that the 1st camera and the 5th or 20th camera may have many features in common)



- **Bundle Adjustment** is an optimisation procedure for providing better estimate of global camera poses and features that uses all pixel correspondences between all frames
- The basic idea is to produce an estimate for the camera poses and 3D feature points that minimises a cost function associated with the re-projection errors of the features points

Bundle Adjustment

- The camera poses and feature points are composed into a single vector:

$$x = [\boldsymbol{\Psi}_1, \mathbf{t}_1, \boldsymbol{\Psi}_2, \mathbf{t}_2, \ldots, \boldsymbol{\Psi}_N, \mathbf{t}_N, \mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_M]^T$$

For i = 1 to N cameras and j = 1 to M feature points, where $\boldsymbol{\Psi}$ is vector representation of R, i.e. Euler angles or quaternions

Bundle Adjustment

- The camera poses and feature points are composed into a single vector:

$$x = [\mathbf{\Psi}_1, \mathbf{t}_1, \mathbf{\Psi}_2, \mathbf{t}_2, \ldots, \mathbf{\Psi}_N, \mathbf{t}_N, \mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_M]^T$$

For i = 1 to N cameras and j = 1 to M feature points, where $\mathbf{\Psi}$ is vector representation of R, i.e. Euler angles or quaternions

- Bundle adjustment performs an optimisation procedure to minimise the cost function:

$$f(x) = \sum_{N}^{i=1} \sum_{M}^{j=1} ([u, v]_{i,j} - Q(\mathbf{\Psi}_i, \mathbf{t}_i, \mathbf{p}_j))^2$$

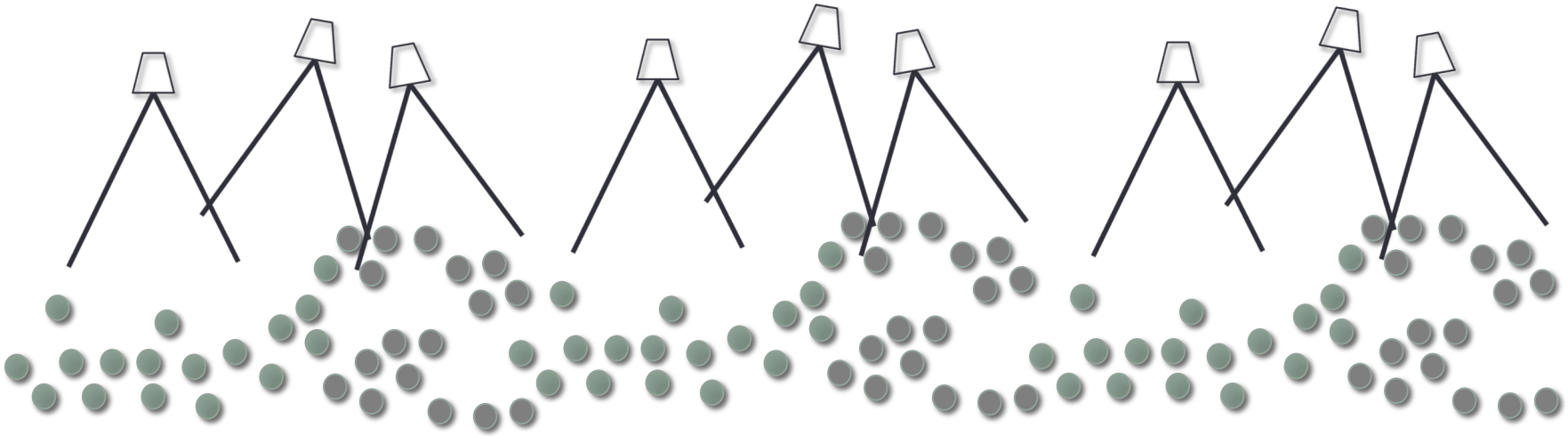
Where $[u, v]_{i,j}$ is the observed pixel location of feature j in camera i and

$$Q(\mathbf{\Psi}_i, \mathbf{t}_i, \mathbf{p}_j) = \mathbf{K}(\mathbf{R}(\Psi_i)_i \mathbf{p}_j + \mathbf{t}_i)$$

is the predicted or re-projected pixel location of feature j in camera i, based on the estimated camera poses/feature point positions

Bundle Adjustment

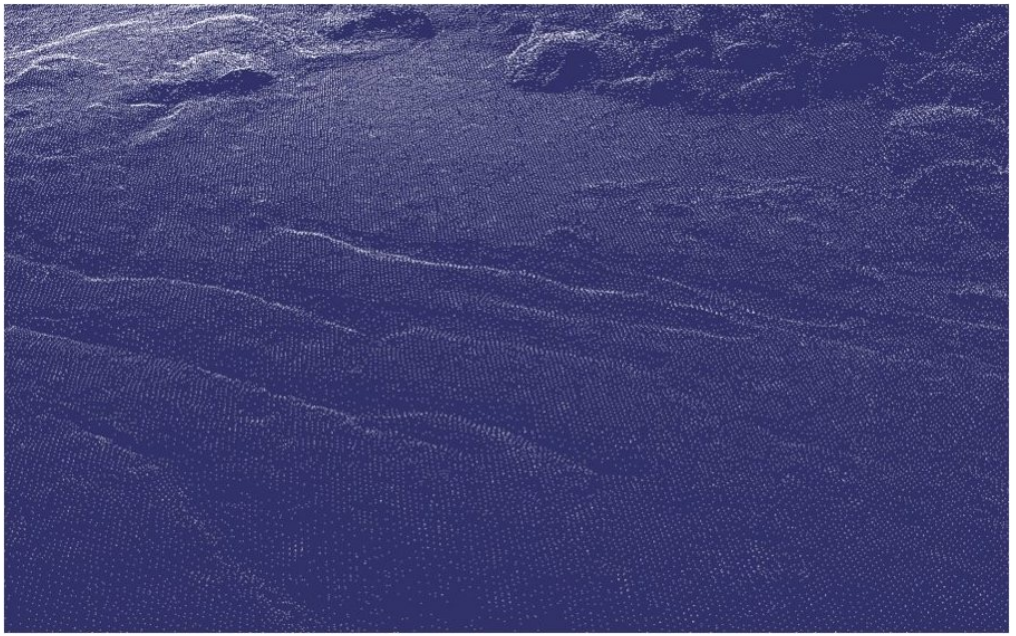
- The cost function f(x) is non-linear and can be minimised using non-linear least squares:
  - Common minimisation algorithms include Gauss-Newton or Levenberg-Marquardt* optimisation
  - These algorithms typically exploit “sparsity” arising in the matrix structure of the cost function in that most features are only observed in a few poses



* See for example: M.I. Lourakis, “A Brief Description of the Levenberg-Marquardt algorithm implemented by levmar”, http://users.ics.forth.gr/~lourakis/levmar/levmar.pdf

Meshing and Texturing

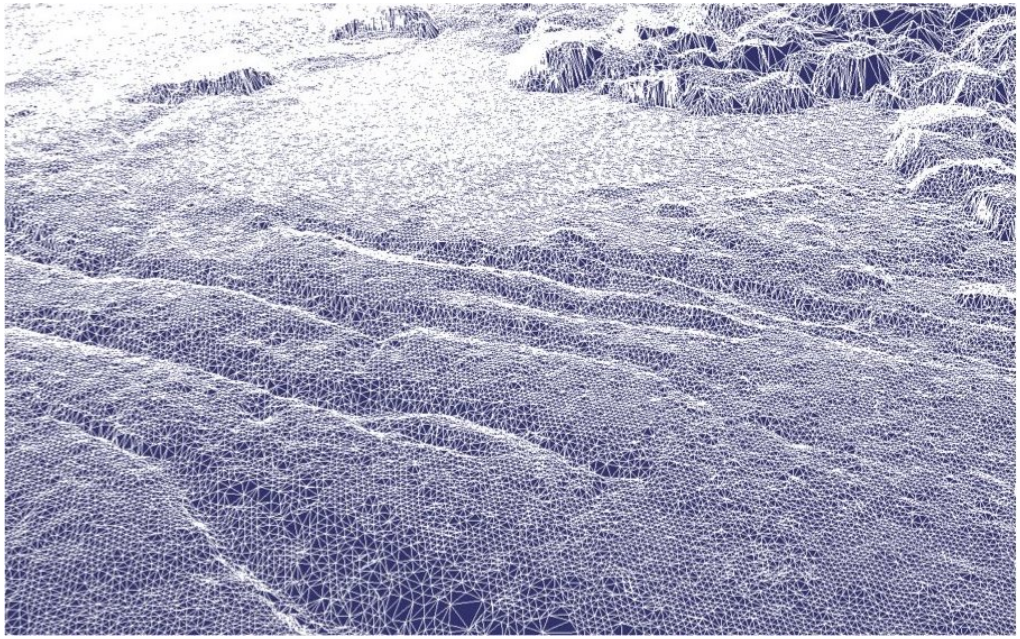
- Imaged feature points typically correspond to a surface which can be represented by a surface mesh (a collection of polygons)



3D Pointcloud

Meshing and Texturing

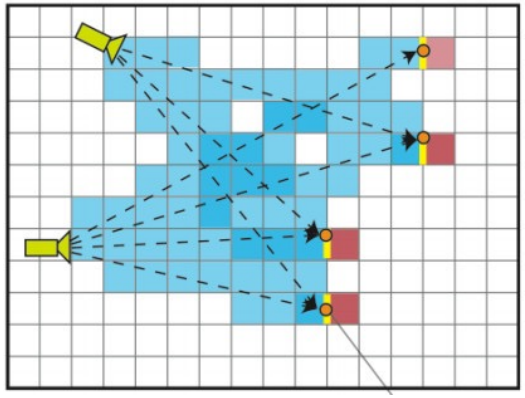
- Imaged feature points typically correspond to a surface which can be represented by a surface mesh (a collection of polygons)
- Starting from the 3D point map, a surface may derived using a variety of surface-reconstruction methods i.e.:
  - Simple triangulation (i.e Delaunay) (2.5D models)
  - Volumetric integration: a voxel occupancy map is constructed, and the surface is composed of the faces of each exposed voxel



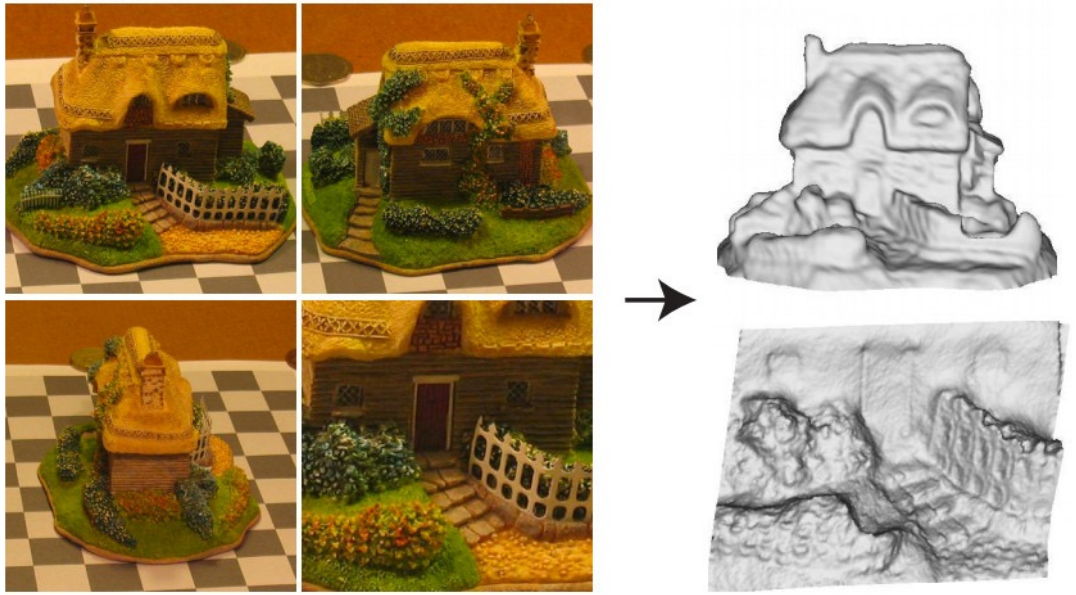
Triangulated Surface Model

Meshing and Texturing

- Imaged feature points typically correspond to a surface which can be represented by a surface mesh (a collection of polygons)
- Starting from the 3D point map, a surface may derived using a variety of surface-reconstruction methods i.e.:
  - Simple triangulation (i.e Delaunay) (2.5D models)
  - Volumetric integration: a voxel occupancy map is constructed, and the surface is composed of the faces of each exposed voxel



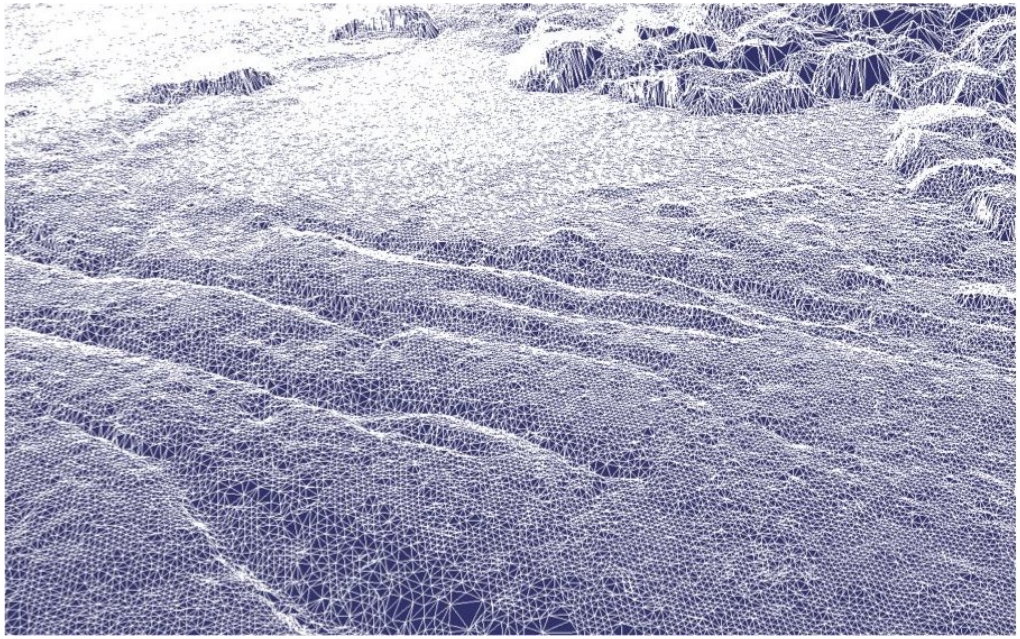
George Vogiatzis, P.H.S. Torr, and Roberto Cipolla. Multi-view stereo via volumetric graph-cuts. In IEEE Conference on Computer Vision and Pattern Recognition, 2005.



Triangulated Surface Model using volumetric integration

Meshing and Texturing

- Since extracted feature points are not typically of the same density/ resolution as the individual pixels in an image, a common approach to labelling a 3D mesh with image colour or brightness data is via texturing
- A common approach is to interpolate the image data in image coordinates to the spatial coordinate of the mesh face-by-face using the known projective relationships between cameras/3D points

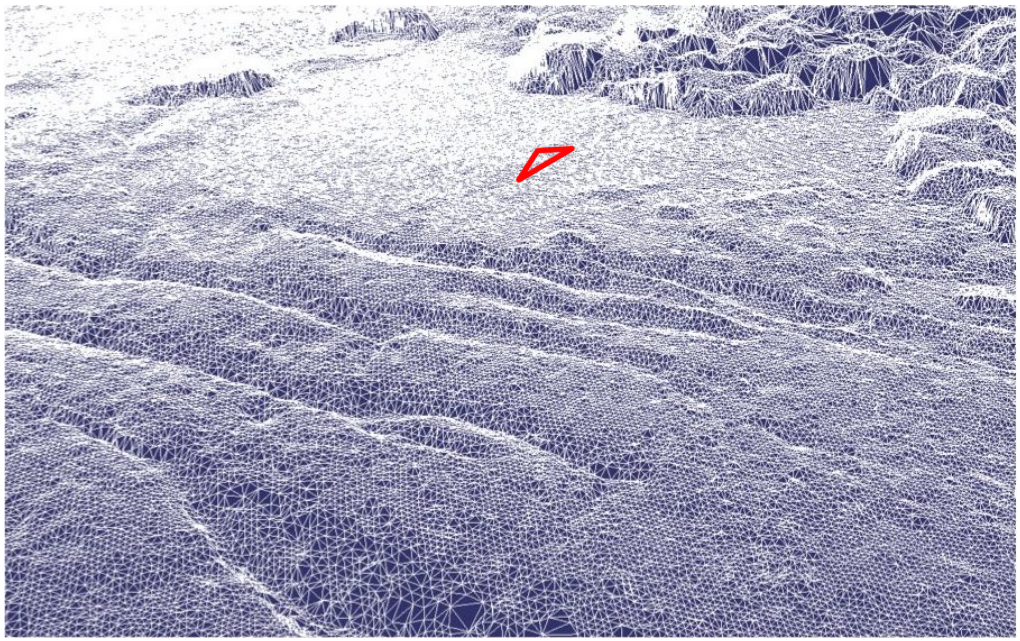


Triangulated Surface Model

Meshing and Texturing

- For each mesh polygon (i.e. triangle), one or several images are identified for which this scene element is visible (i.e. each vertex of the polygon re-projects into the image space of the corresponding camera, based on it's known pose)
- The best image is selected based on criteria associated with distance or angle to the camera, and the corresponding image data within the project vertices is interpolated onto the mesh face

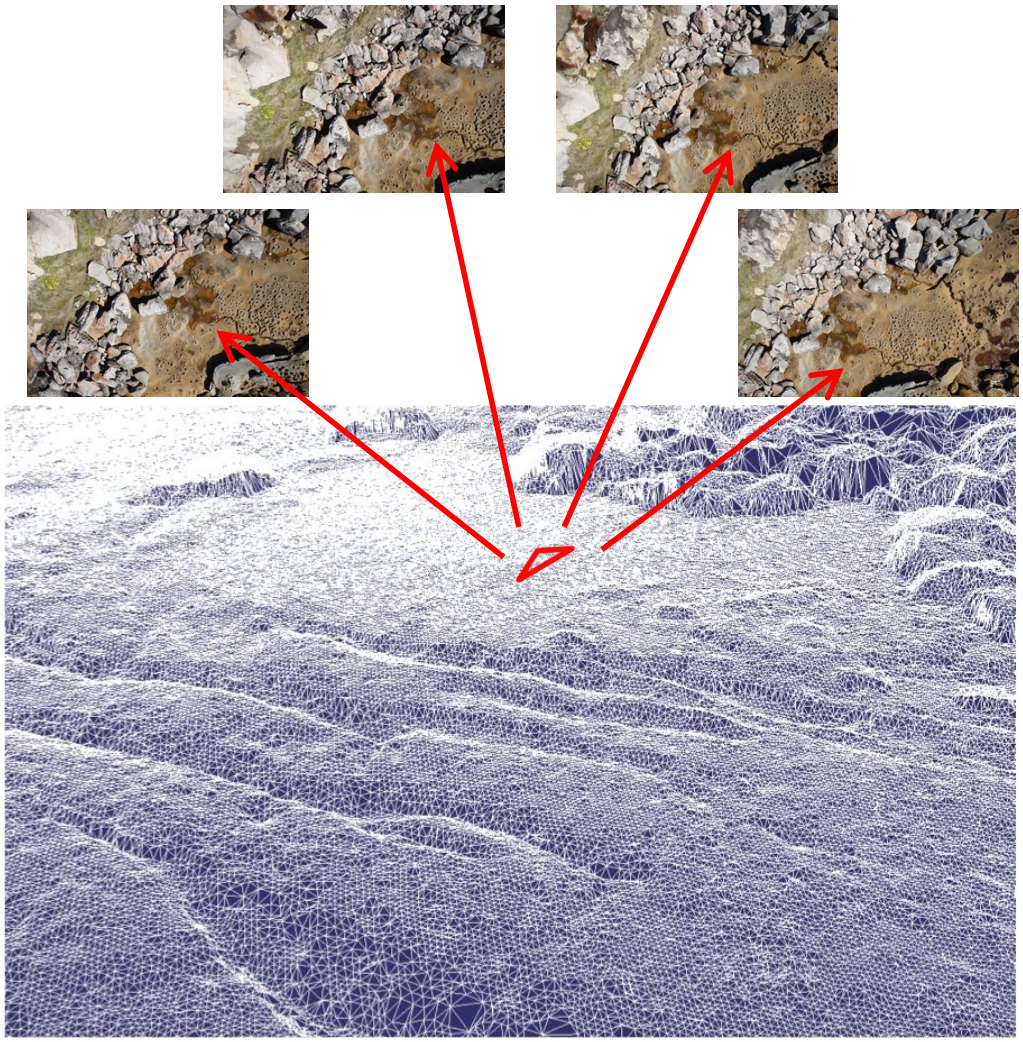
Individual triangle



Triangulated Surface Model

Meshing and Texturing

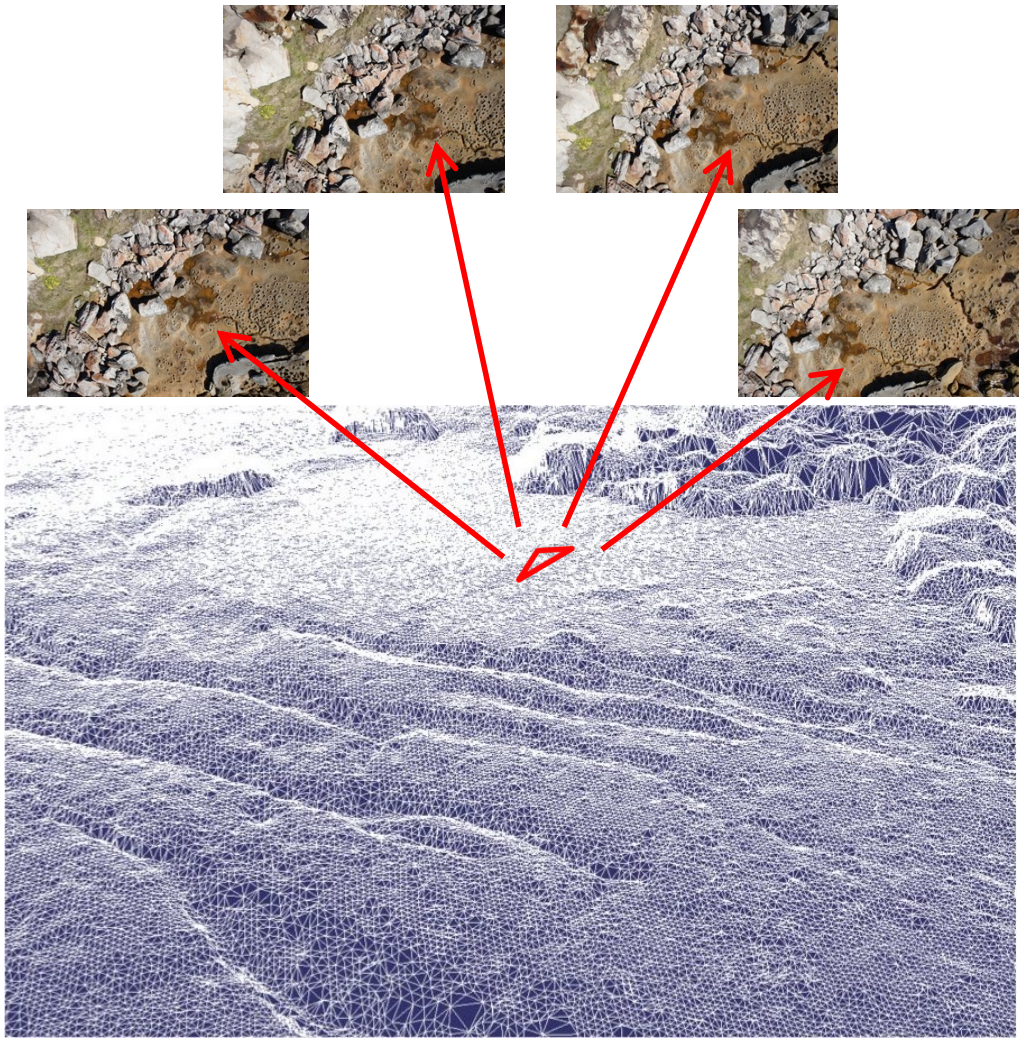
- For each mesh polygon (i.e. triangle), one or several images are identified for which this scene element is visible (i.e. each vertex of the polygon re-projects into the image space of the corresponding camera, based on it's known pose)
- The best image is selected based on criteria associated with distance or angle to the camera, and the corresponding image data within the project vertices is interpolated onto the mesh face



Triangulated Surface Model

Meshing and Texturing

- For each mesh polygon (i.e. triangle), one or several images are identified for which this scene element is visible (i.e. each vertex of the polygon re-projects into the image space of the corresponding camera, based on it's known pose)
- The best image is selected based on criteria associated with distance or angle to the camera, and the corresponding image data within the project vertices is interpolated onto the mesh face



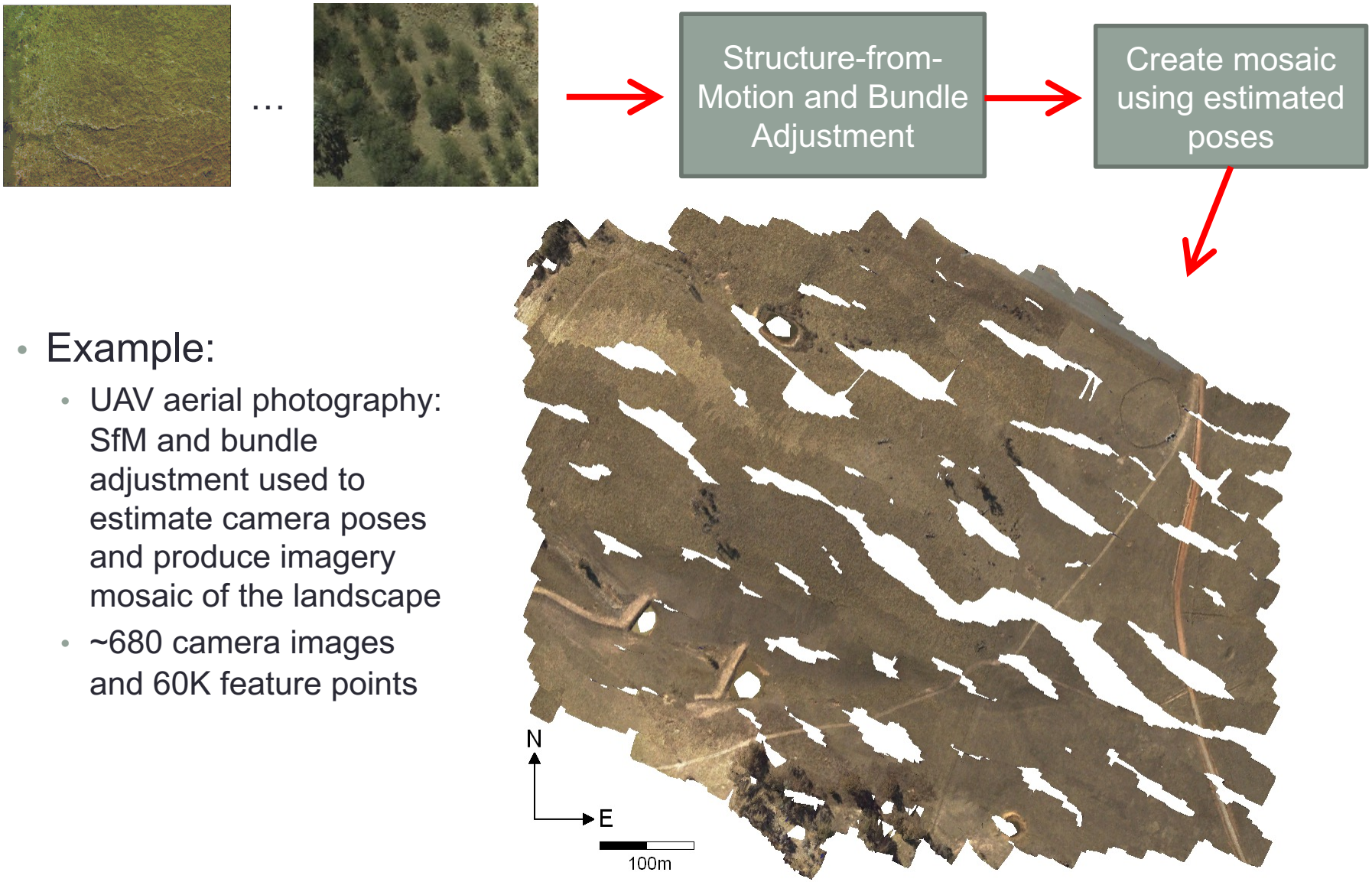
Triangulated Surface Model

Meshing and Texturing

- For each mesh polygon (i.e. triangle), one or several images are identified for which this scene element is visible (i.e. each vertex of the polygon re-projects into the image space of the corresponding camera, based on it's known pose)
- The best image is selected based on criteria associated with distance or angle to the camera, and the corresponding image data within the project vertices is interpolated onto the mesh face

Photo-textured Surface Model

Effect of Bundle Adjustment

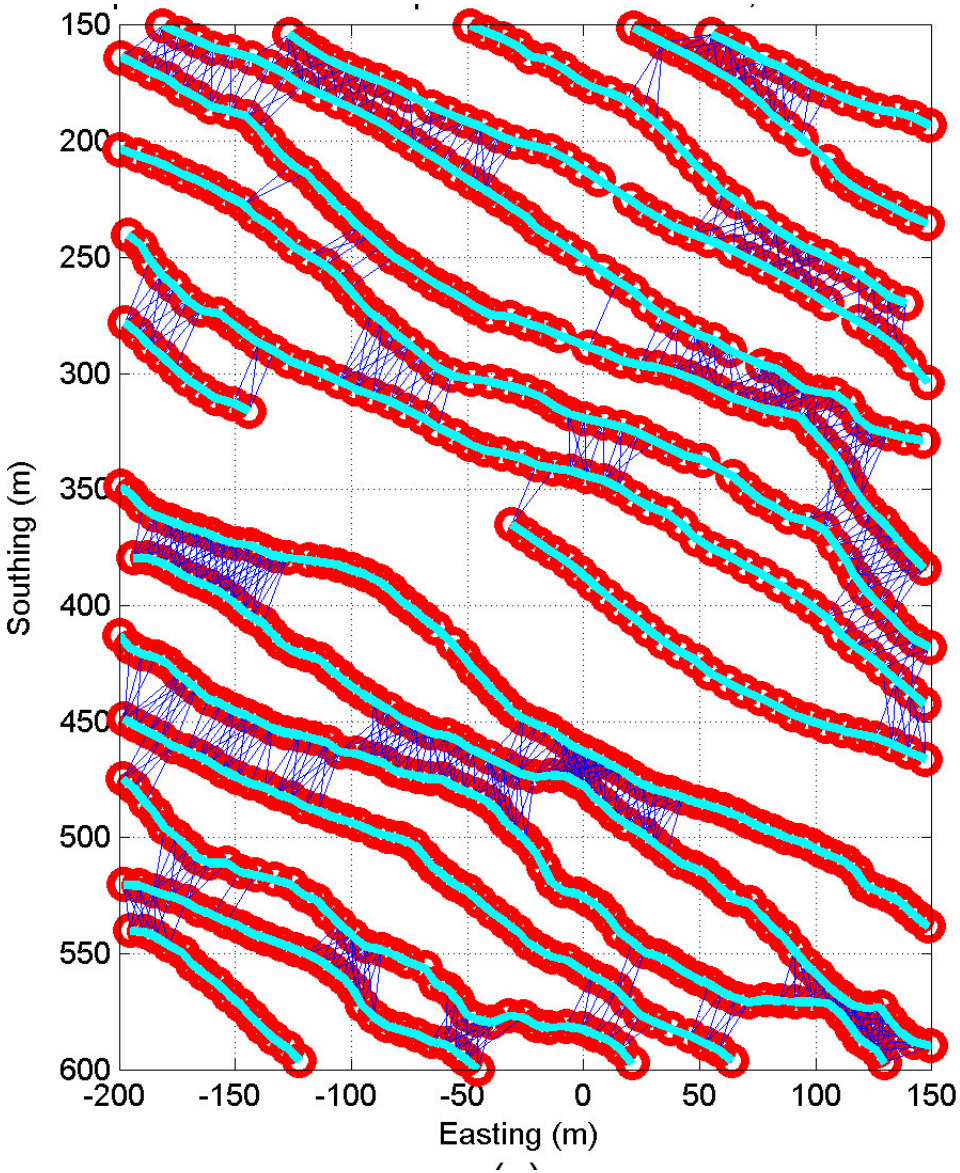


- Example:
  - UAV aerial photography: SfM and bundle adjustment used to estimate camera poses and produce imagery mosaic of the landscape
  - ~680 camera images and 60K feature points

Effect of Bundle Adjustment

- Example:
  - UAV aerial photography: SfM and bundle adjustment used to estimate camera poses and produce imagery mosaic of the landscape
  - ~680 camera images and 60K feature points

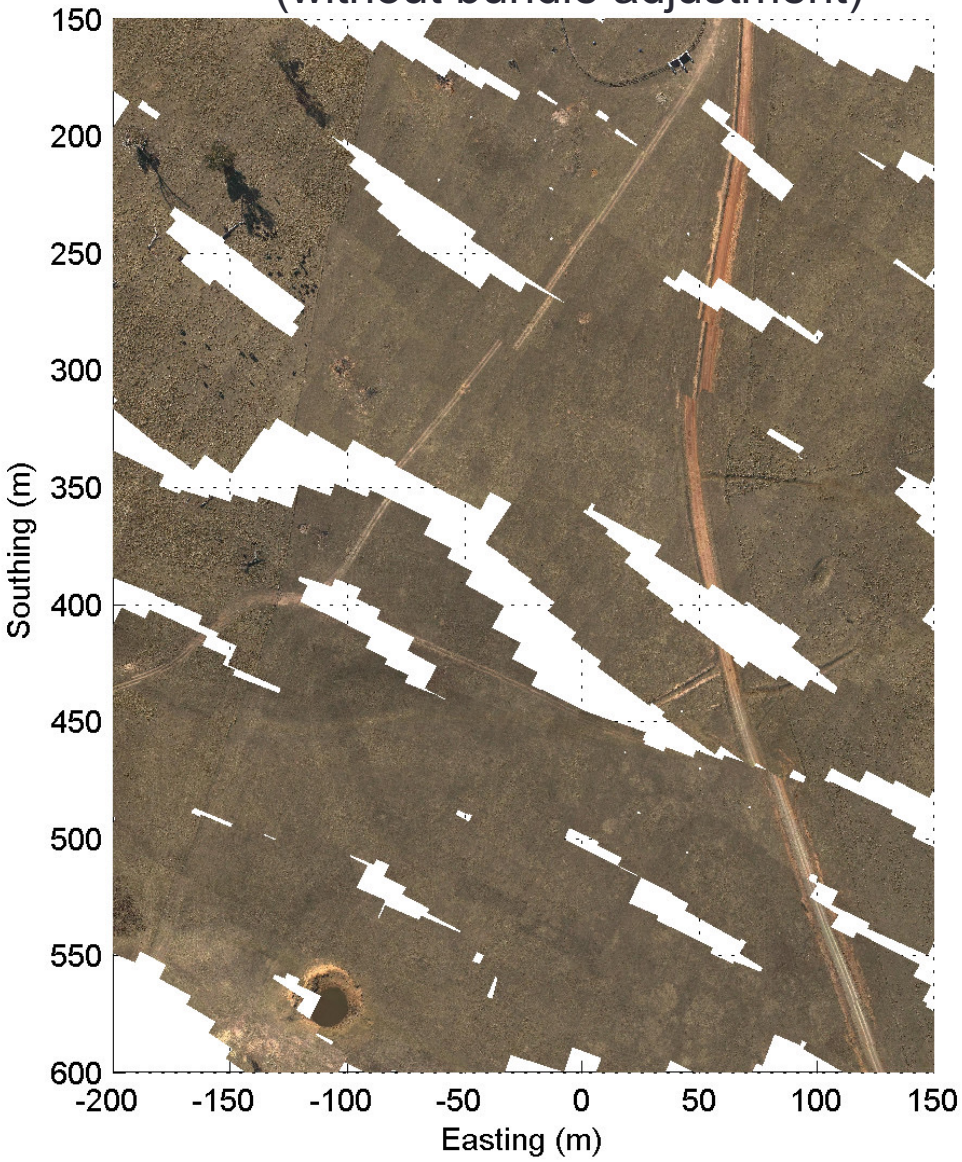
Estimated Camera positions (Flight Lines)



Effect of Bundle Adjustment

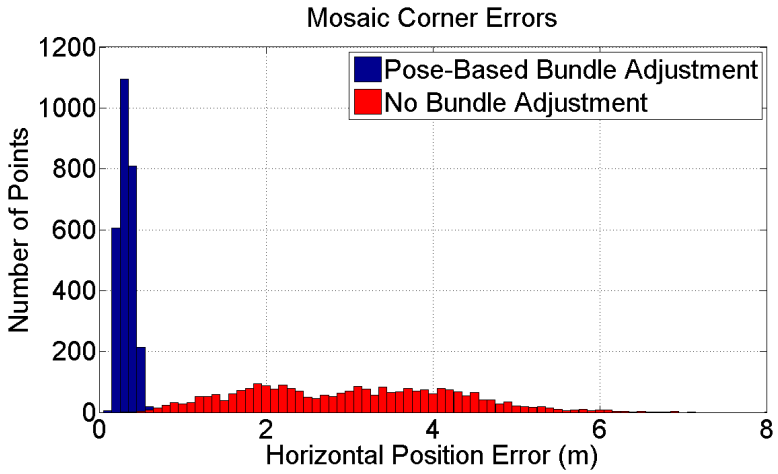
- Example:
  - UAV aerial photography: SfM and bundle adjustment used to estimate camera poses and produce imagery mosaic of the landscape
  - ~680 camera images and 60K feature points

Mosaic using poses estimated from structure-from-motion (without bundle adjustment)

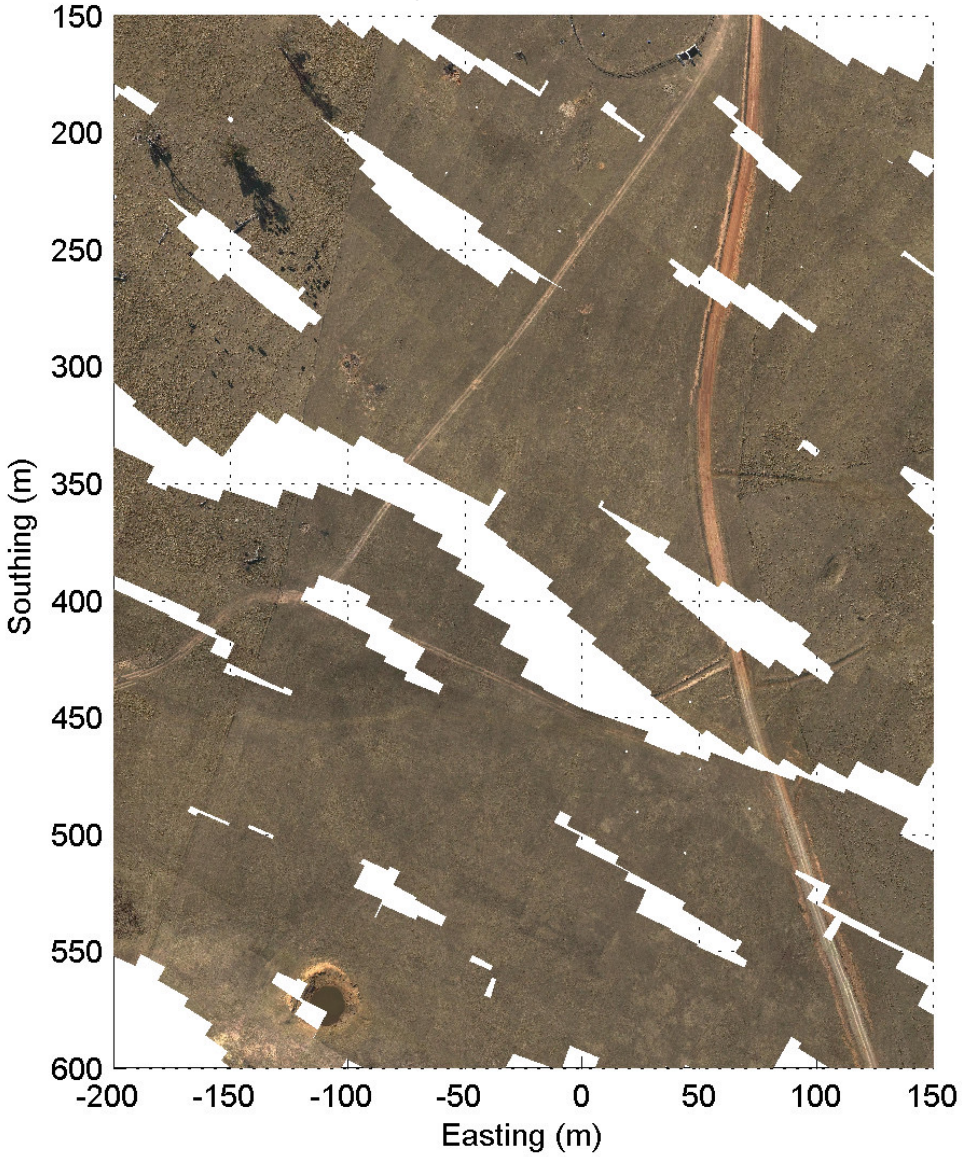


Effect of Bundle Adjustment

- Example:
  - UAV aerial photography: SfM and bundle adjustment used to estimate camera poses and produce imagery mosaic of the landscape
  - ~680 camera images and 60K feature points



Mosaic using poses estimated after bundle adjustment

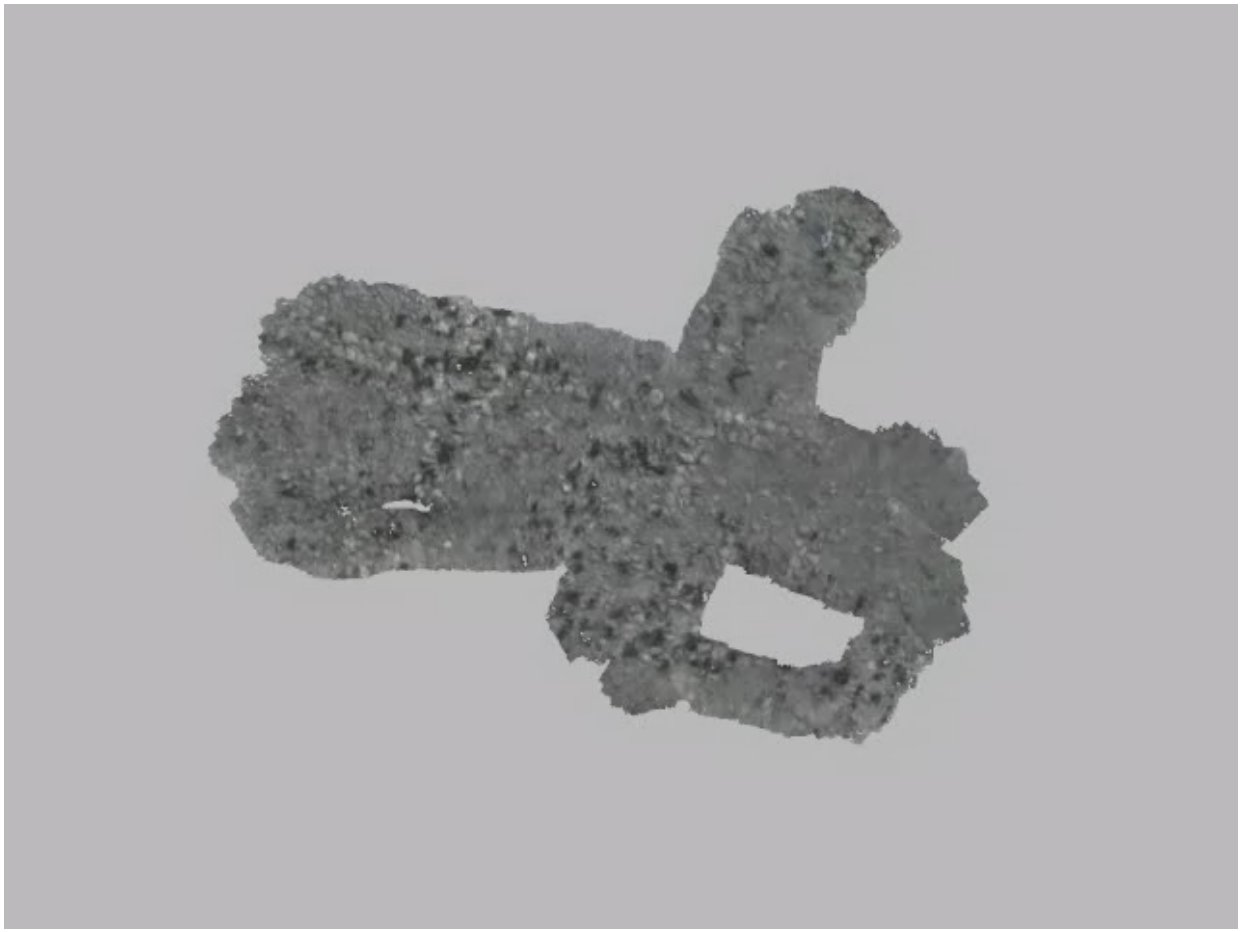


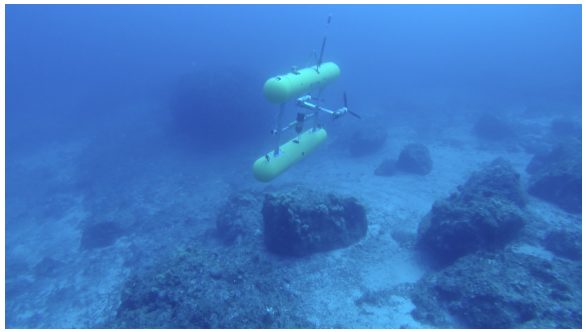
3D Models

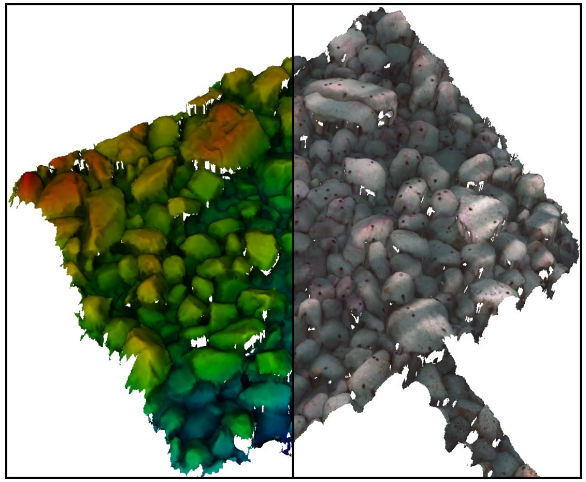
3D models generated from aerial photography

3D Models

3D models generated from underwater imagery







Real-world software for SfM and 3D model building

- There exist a number of software packages for implementing structure-from-motion, meshing and texturing, both commercial and free/open-source:
  - Bundler (http://www.cs.cornell.edu/~snavely/bundler/):
  - VisualSFM (http://ccwu.me/vsfm/):
  - COLMAP (https://colmap.github.io/)
  - Agisoft Photoscan/Metashape (http://www.agisoft.com/):

Further Reading and Next Week

- References:
  - R. Szeliski, “Computer Vision: Algorithms and Applications”, Springer, 2010 (Chapter 7, 12)
  - B. Triggs, P. McLauchlan, R. Hartley, A. Fitzgibbon, “Bundle Adjustment: a modern sythnthesis”, ICCV '99 Proc. of the International Workshop on Vision Algorithms: Theory and Practice, http://lear.inrialpes.fr/pubs/2000/TMHF00/Triggs-va99.pdf

- Next Week:
  - Face detection and recognition