Structure from motion

Structure from motion (SfM) is a photogrammetric technique for estimating three-dimensional (3D) structures from two-dimensional overlapping images or video sequences. Usual cases are static scene and moving camera and moving scene and static camera. SfM can be understood as a process of inversion of image formation. It is widely used in many applications, such as robot navigation, autonomous driving, and augmented reality. SfM computes 3D scene structure (tie points) and camera motion which are coordinates of projection centers and camera rotation in space.

SfM algorithm consists of several algorithms and steps to reconstruct 3D structure. First it is necessary to detect features in images, match features, compute orientation of images and finally it is important to optimize model, see figure 1.

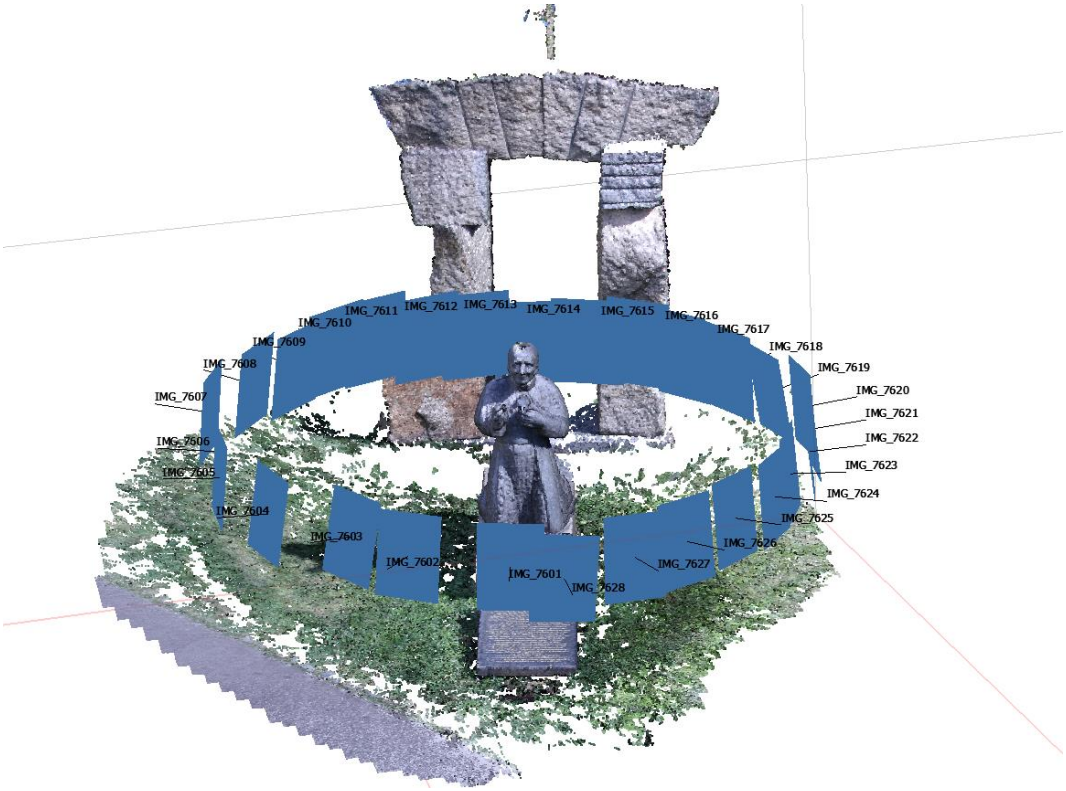

Figure 1: 3D reconstruction of cardinal Beran statue, dense cloud with cameras.

1. Find and track features

Features

It is not easy to explain what feature exactly is. But it is natural ability for human brain to find features in image and across set of images. Feature must be well identifiable in set of images. Feature is represented by coordinates of image point, but it is not only single point in image. Feature surroundings is also very important feature property. Let's see figure 2 for more explanation. Image patches *A* and *B* is impossible to determine location of patches by coordinates of single point. We can determine area where patch *A* can be placed. It is possible to find several locations for patch *B*. Patches *C* and *D* are edges. We can find several possible locations but find the right one is still difficult. Patches *E* and *F* are corners, and it is simple to find their exact location in the image [1].

Figure 2: Image features [1]

Scale-Invariant Feature Transform (SIFT)

There are several corner detectors like Haris Corner Detector and Shi-Tomasi Corner Detector which were able to detect and match features in the image. These corner detectors are not scale-invariant. Scale invariance is important when you use set of images with different scales of object. Detected corner in non-scaled image can be well identifiable but when you zoom in the corner it exceeds pixel window. The corner breaks down into multiple pixels and it can be impossible to detect this corner again in zoomed image.

D. Lowe came up with SIFT algorithm in 2004 [2] which solves feature detection in different scales. SIFT algorithm finds features in scale-space, assign them orientation, compute their description and match keypoints (features) between images.

Scale-space extrema detection

Scale-space is created for image extrema detection. To find these extrema Laplacian of Gaussian (LoG) is computed for image with various σ values. LoG acts as blob detector. Various sizes of these blobs are found by σ change. SIFT use Difference of Gaussian (DoG) instead of LoG since LoG is costly. DoG is created as difference of Gaussian blurring of an image using different σ values, see figure 3.

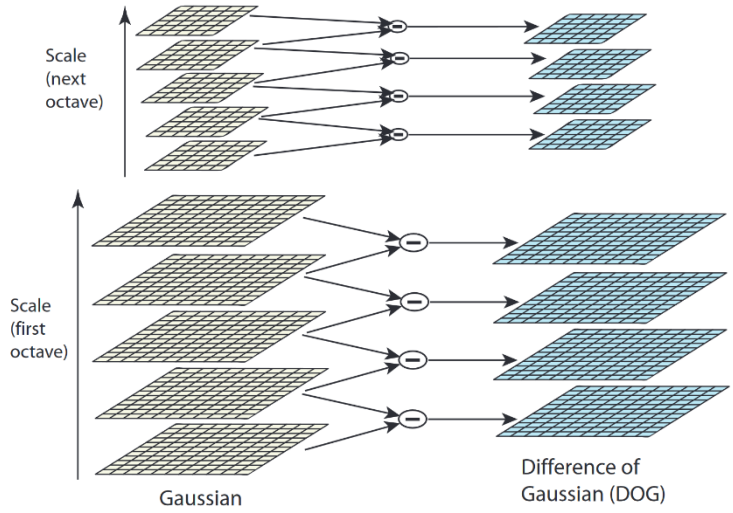

Figure 3: Image scale-space [2].

Features (potential keypoints) are found as local minima and maxima across the scale and space. Features are compared with its neighbors and with the same submatrix 3x3 in the next and the previous scales as shown in figure 4.

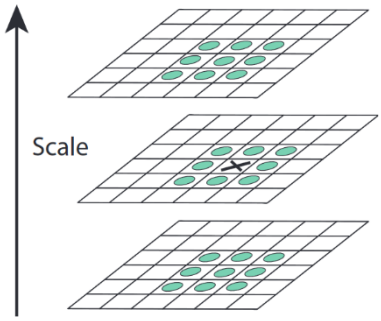

Figure 4: Maxima and minima of the DoG images comparison [2], where X mark is a local minima or maxima.

Accurate keypoint localization

Potential keypoint location is found as maxima or minima of 3D quadratic function fitted to local sample points. This extremum is tested to remove low contrast keypoints. DoG has great response along edges. To remove these keypoints Hessian matrix (2x2) is computed. Edges has one eigen value larger than the second. The ratio between eigen values is tested.

Orientation assignment

Orientation assignment of keypoint is performed on local image properties. Keypoint descriptor is computed relative to keypoint orientation to ensure invariance image in rotation. Scale-invariance is achieved by choosing the closest scale of Gaussian smoothed image to the scale of the keypoint. Gradient magnitude and orientation are computed based on pixel differences of each image sample. An orientation histogram is created from orientations which are weighted by gradient magnitudes and Gaussian-weighted circular window with a σ. The highest peak in histogram is detected. For any other peak above 80% of the highest peak a new keypoint with orientation is created. Finally, a parabola is fitted to peak and its closest surroundings to achieve better accuracy.

Keypoint descriptor

Descriptor is a vector which describes local image region around keypoint. Image gradient magnitudes and orientations are computed around keypoint location in Gaussian smoothed image. The coordinates of the descriptor and the gradient orientations are rotated relative to keypoint orientation to achieve orientation invariance. Gaussian weighting function with σ equals to one half of the width of the descriptor (circle in the figure 5). This function avoids sudden changes in the descriptor and gives less contribution to gradients far from the descriptor center. As shown in the figure 4 descriptor is on the right side and consists of 2x2 histogram array. Each array is orientation histogram of 4x4 region (sample array) where directions represent orientations, and the length of the arrow is magnitude. To avoid boundary, effect the trilinear interpolation is used to distribute each value of gradient. $1 - d$ is used as a weight, where d is distance of the sample from the center. The descriptor is created as a vector of all histograms. Best results are achieved with 4x4 histogram array created from 16x16 sample array. Then the descriptor is modified to reduce illumination change. The vector is normalized to unit length to remove contrast change and brightness change. However non-linear illumination change is not removed by normalization. The effect of non-linear illumination causes a large change in some magnitudes for soma gradients, but less for gradient orientations. Large gradient magnitudes are restricted to the value of 0.2 in normalized vector and then the vector is renormalized.

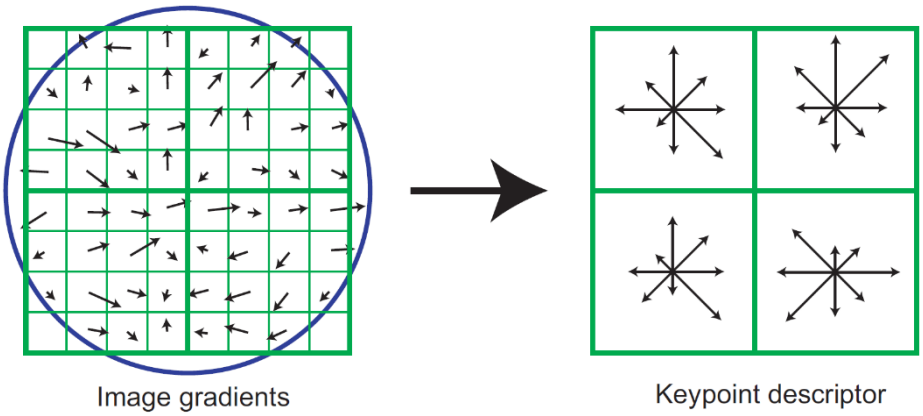

Figure 5: Keypoint descriptor creation [2].

Keypoint matching

The best candidate for the match is determined as a nearest neighbor. It is defined as minimum Euclidean distance for the descriptor vector. Some keypoints achieve incorrect match since they rise from background, noise or other reasons. In this case it is useful to use the ratio of the closest and the second closest distance for matching verification. If the ratio is greater than 0.8 the match is rejected. It reduces false matches for 90%.

Figure 6: SIFT keypoints in the image of statue cardinal Beran.

Important feature detectors and descriptors

SURF (Speeded Up Robust Features)

SURF [3] is faster feature detector and descriptor than SIFT. It simplifies some steps of SIFT. SURF approximates LoG with Box Filter which speed up calculation. Scale and location are derived from determinant of Hessian matrix. SURF uses wavelet responses in horizontal and vertical direction and applying gaussian weights. Image orientation is determined from a plot by calculating all responses in sliding window. SURF uses wavelet responses for feature descriptor. A feature neighborhood of 20s x 20s (where s is chosen size for feature) is used for descriptor computation which is divided into 4x4 subregions. It forms 64-dimension descriptor vector with extension to 128 dimensions.

FAST (Features from Accelerated Segment Test)

FAST [4] is a corner detector. It is intended as feature detector for SLAM (Simultaneous Localization and Mapping) technology with limited computational technology. It takes image pixel and creates circle around this point. Pixels laying at this circle are tested whether they are brighter or darker (threshold value) than chosen image pixel. This pixel is considered as corner if *n* pixels at the circle is brighter or darker than chosen pixel.

BRIEF (Binary Robust Independent Elementary Features)

BRIEF [5] creates unique set of pixel pairs in feature surroundings. Then pixel intensity is of these pairs is compared and results of comparison are stored into a bit-string (binary vector). Hamming distance is

used for matching descriptors. Hamming distance takes number of different (binary) positions when compares descriptors.

ORB (Oriented FAST and Rotated BRIEF)

ORB [6] uses FAST feature detector improved for feature orientation and BRIEF descriptor following this orientation. ORB is free and faster than SIFT and SURF and descriptor works better than SURF.

2. Estimation structure and motion

Epipolar geometry

Epipolar geometry is the geometry of stereovision. It defines geometric relationships between 3D points and their projections into the 2D image plane. These relationships are based on assumption of pinhole camera model. Figure 7 below shows the geometry of stereovision.

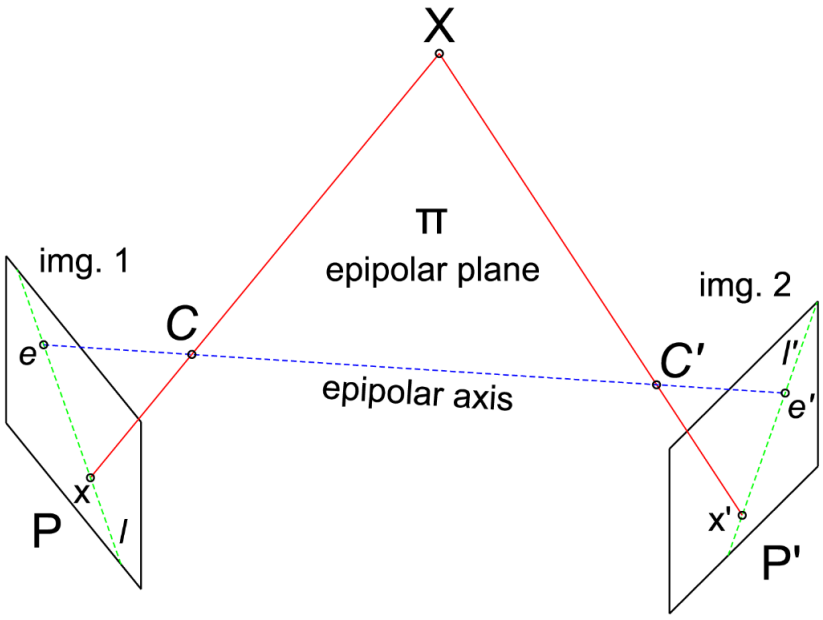

Figure 7: Epipolar geometry of stereovision.

Where *X* is 3D point, *C*, *C'* are projection centers, *x*, *x'* are image coordinates of 3D point *X*, *e*, *e'* are epipoles (2D coordinates), *l*, *l'* are epipolar lines and *P*, *P'* are projection matrices.

Projection matrix

Projection matrix defines projection of 3D points into an image plane (2D). Projection matrix *P* is multiplication of calibration matrix *K*, rotation matrix *R*, translation vector *T* and vector of 3D point coordinates *X*.

$$P_{3x3} = K_{3x3} \cdot [R_{3x3} \quad T_{3x1}] \begin{bmatrix} X \\ 1 \end{bmatrix}_{4x1} = K_{3x3} \cdot R_{3x3} \cdot [X_{3x1} - C_{3x1}]$$

Where *C* is 3D coordinates of projection center.

$$K = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Calibration matrix *K* is shown above. Where f_x is focal length for x axis, f_y is focal length for y axis, x_0, y_0 are coordinates of principal point and *s* is skew parameter. This is full calibration matrix which can be decreased. Focal lengths ratio f_y / f_x is aspect ratio and can be removed from computation by defining one parameter for focal length *f*. Skew parameter *s* can be set to 0.

Essential matrix

Essential matrix is a special form of fundamental matrix in sense of normalized image coordinates. Normalized coordinates are obtained by decomposition of projection matrix *P*. Let x are image coordinates and normalized coordinates are $\hat{x}$, then $\hat{x} = K^{-1} \cdot x = [R \quad T] \cdot X$ (*X* are homogenous coordinates). Matrix $[R \quad T]$ is called normalized camera matrix. Essential matrix has a form:

$$E = [T]_x R$$

Defining equation for Essential matrix is:

$$\widehat{x'}^T \cdot E \cdot \widehat{x} = 0$$

By substituting $\widehat{x'}$and $\hat{x}$ gives equation:

$$x'^T \cdot K'^{-T} \cdot E \cdot K^{-1} \cdot x = 0$$

$$x'^T \cdot F \cdot x = 0$$

Where *F* is fundamental matrix and relationship between Fundamental matrix and Essential matrix is:

$$E = K'^T \cdot F \cdot K$$

Essential matrix has 5 degrees of freedom, *R* and *T* has both 3 degrees of freedom, but there is scale ambiguity since Essential matrix is homogeneous quantity. Another constrain is that Essential matrix has 2 singular values equal, and 3rd is equal to zero [7].

Fundamental matrix

Fundamental matrix is defined below and must satisfy for any pair of matching point *x'*, *x* (homogenous image coordinates) in two images [7].

$$x'^T \cdot F \cdot x = 0$$

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0$$

After denotation a set of linear equation is obtained:

$$Af = \begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} f = 0$$

The solution is right null-space of A. It is done via SVD decomposition.

$$U \cdot D \cdot V^T = svd(A)$$

$$f = V_{:,9}$$

After reshaping *f* to form 3x3 (*F*) it is necessary to apply constraint to rank of *F* matrix. It is done by SVD cleanup.

$$U \cdot D \cdot V^T = svd(F)$$

$$F_{rank\ 2} = U \cdot \widetilde{D} \cdot V^T$$

Where $\widetilde{D}$ is *D* with last element set to 0.

Epipole and epipolar line

Once *F* matrix is reconstructed image coordinates of epipole and equation of epipolar line can be computed. Epipolar lines are defined:

$$\boldsymbol{l'} = \boldsymbol{F} \cdot \boldsymbol{x}, \boldsymbol{l} = \mathbf{F^T} \cdot \boldsymbol{x'}$$

Epipol coordinates in the left image are calculated:

$$[u, d] = eigs(F' \cdot F)$$

$$e = u(:,3)$$

e is defined by homogenous coordinates, so it is necessary to define scale factor by normalizing last element of *e* to 1. Epipole of right image is defined in a similar way:

$$[u, d] = eigs(F \cdot F')$$

$$e' = u(:,3)$$

Automatic estimation of Fundamental matrix

Modern approach uses automatic estimation of Fundamental matrix. The main role plays RANSAC algorithm [8]. RANSAC is robust iterative method which estimates model parameters from noise data. Input into RANSAC algorithm are features of left and right images together with their matches.

a) Compute Fundamental matrix from random sample of 8 correspondences.
b) Find inliers which satisfy *F*. Inliers are determined based on computed distance between point x_i and epipolar line. This distance is compared with threshold.

Solution is *F* with largest number of inliers. Afte that refine *F* by non-linear Least squares.

Reconstruction R and T from Fundamental matrix

First it is necessary to extract Essential matrix from Fundamental matrix. If camera calibration matrix is unknown it can be estimate from image metadata. Then Essential matrix is achieved from relation of Fundamental and Essential matrix. SVD of E is:

$$E = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T$$

Where $U = [u_1 \quad u_2 \quad u_3]$ and vector u_3 is nullspace of *E*. Thus $T = u_3$ or $-u_3$. First camera projection matrix is chosen as $P = [I\ 0]$ and second camera projection matrix has 4 possible solutions, as shown in the figure 8.

$$P' = [U \cdot W \cdot V^T + u_3] \mid [U \cdot W \cdot V^T - u_3] \mid [U \cdot W^T \cdot V^T + u_3] \mid [U \cdot W^T \cdot V^T - u_3]$$

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

If *det(R) = -1*, then *T = -T* and *R = -R*.

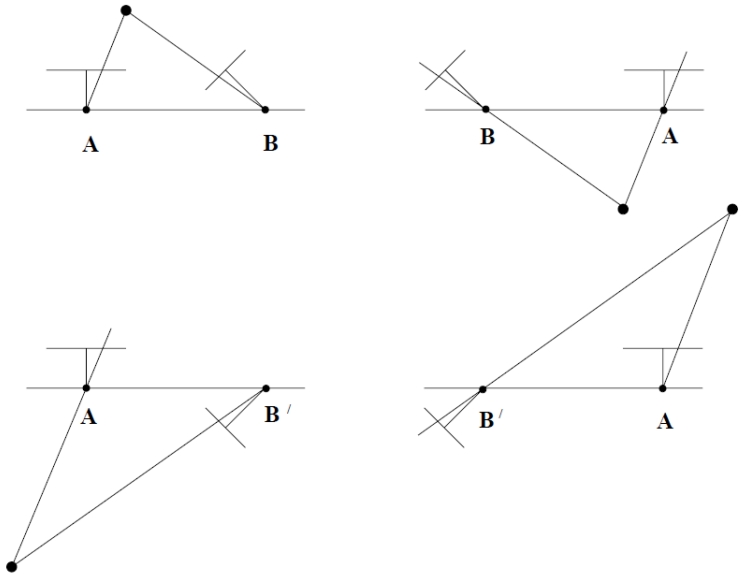

Figure 8: 4 solutions for reconstruction from *E* [7].

Right solution for *P'* can be achieved by point triangulation. 3D point must lie in front of image planes.

$$\begin{bmatrix} x_1 \\ 1 \end{bmatrix} = P \begin{bmatrix} X_1 \\ 1 \end{bmatrix}, \begin{bmatrix} x_2 \\ 1 \end{bmatrix} = P' \begin{bmatrix} X_1 \\ 1 \end{bmatrix}$$

Cross product of parallel vectors is equal to 0. Apply this rule to equation above we achieve system:

$$\begin{bmatrix} \begin{bmatrix} x_1 \\ 1 \end{bmatrix}_x P \\ \begin{bmatrix} x_2 \\ 1 \end{bmatrix}_x P' \\ \vdots \\ \begin{bmatrix} x_n \\ 1 \end{bmatrix}_x P_n \end{bmatrix} \cdot \begin{bmatrix} X_1 \\ 1 \end{bmatrix} = 0$$

It is necessary to use Least squares if *rank(F)* is greater than 2.

3. Bundle adjustment

Bundle adjustment is computational method for simultaneous refining 3D points in the scene, exterior orientation parameters (camera orientation parameters) and interior orientation parameters (camera calibration parameters). This process is computationally challenging. Bundle adjustment minimize distance between image point coordinates and reprojection of 3D point into the image plane. Below is basic equation written in Euclidean form:

$$x + v = \frac{P_{1:2}(x, R, T, c) \cdot X}{P_3(x, R, T, c) \cdot X}$$

Where *x* are image coordinates of point, *v* is correction, *P* is projection matrix, *R* is rotation matrix, *T* is translation vector, *c* are distortion parameters and *X* are 3D point coordinates.

System to solve is:

$$\Delta l + v = A \cdot \Delta x$$

Where Δl are observations (2D points) and Δx are unknowns.

Normal solution is shown below but it would be very tricky to solve this system.

$$\Delta x = (A^T \cdot \Sigma^{-1} \cdot A)^{-1} \cdot A^T \cdot \Sigma^{-1} \Delta l$$

Where Σ is block diagonal covariance matrix.

Since *A* matrix is sparse matrix, it is possible to decompose *A* matrix and vector of unknowns:

$$\Delta l + v = [C \quad B] \begin{bmatrix} \Delta k \\ \Delta t \end{bmatrix}$$

Where Δk are unknown 3D points and Δt are orientation parameters.

Normal matrix is then divided into 4 submatrices.

$$N = A^T \cdot \Sigma^{-1} \cdot A = \begin{bmatrix} C^T \\ B^T \end{bmatrix} \cdot \Sigma^{-1} \cdot [C \quad B] = \begin{bmatrix} C^T \cdot \Sigma^{-1} \cdot C & C^T \cdot \Sigma^{-1} \cdot B \\ B^T \cdot \Sigma^{-1} \cdot C & B^T \cdot \Sigma^{-1} \cdot B \end{bmatrix} = \begin{bmatrix} N_{kk} & N_{kt} \\ N_{tk} & N_{tt} \end{bmatrix}$$

Where N_{kk} is almost diagonal matrix containing 3x3 blocks for each point, N_{tt} is almost diagonal matrix consist of 6x6 blocks for camera orientation, N_{kt} and $N_{tk} = N_{kt}^T$ links camera orientations with point observations via 3x6 blocks and

$$N_{kk} = diag(N_{kiki}), N_{kiki} = \sum_{j \epsilon B_i} C_{ij} \cdot \Sigma_{ij}^{-1} \cdot C_{ij}^T$$

N_{kk} consist of submatrices N_{kiki} which summarizes all images where a 3D point is visible.

$$N_{tt} = diag(N_{tjtj}), N_{tjtj} = \sum_{i \epsilon P_j} B_{ij} \cdot \Sigma_{ij}^{-1} \cdot B_{ij}^T$$

N_{tt} consist of submatrices N_{tjtj} which summarizes all visible points in the image.

$$N_{kitj} = C_{ij} \cdot \Sigma_{ij}^{-1} \cdot B_{ij}^T$$

If system is rewritten by normal equations, we achieve:

$$\begin{bmatrix} N_{kk} & N_{kt} \\ N_{tk} & N_{tt} \end{bmatrix} \begin{bmatrix} \Delta k \\ \Delta t \end{bmatrix} = \begin{bmatrix} h_k \\ h_t \end{bmatrix}, \begin{bmatrix} h_k \\ h_t \end{bmatrix} = A^T \cdot \Sigma^{-1} \cdot \Delta l$$

Now can be system solved by adding matrix.

$$\begin{bmatrix} N_{kk}^{-1} & 0 \\ -N_{tk} \cdot N_{kk}^{-1} & I \end{bmatrix} \begin{bmatrix} N_{kk} & N_{kt} \\ N_{tk} & N_{tt} \end{bmatrix} \begin{bmatrix} \Delta k \\ \Delta t \end{bmatrix} = \begin{bmatrix} N_{kk}^{-1} & 0 \\ -N_{tk} \cdot N_{kk}^{-1} & I \end{bmatrix} \begin{bmatrix} h_k \\ h_t \end{bmatrix}$$

$$\begin{bmatrix} I & N_{kk}^{-1} \cdot N_{kt} \\ 0 & N_{tt} - N_{tk} \cdot N_{kk}^{-1} \cdot N_{kt} \end{bmatrix} \begin{bmatrix} \Delta k \\ \Delta t \end{bmatrix} = \begin{bmatrix} N_{kk}^{-1} \cdot h_k \\ h_t - N_{tk} \cdot N_{kk}^{-1} \cdot h_k \end{bmatrix}$$

Reduced normal system from the 2nd row is:

$$\bar{N}_{tt} \cdot \Delta t = \bar{h}_t, \bar{N}_{tt} = N_{tt} - N_{tk} \cdot N_{kk}^{-1} \cdot N_{kt}, \bar{h}_t = h_t - N_{tk} \cdot N_{kk}^{-1} \cdot h_k$$

N_{kk}^{-1} is easy to invert since it is possible to invert just each block of sparse diagonal matrix or use sparse solver. When Δt is determined it is easy to determine Δk from 1st row of reduced normal system:

$$\Delta k = N_{kk}^{-1} \cdot (h_k - N_{kt} \cdot \Delta t)$$

Multi-View Stereo

Multi-View Stereo algorithms are designed to reconstruct detailed 3D models from obtained images. MVS is solved as image\geometry consistency optimization problem. MVS focus on robust implementations of photometric consistency measures, and efficient optimization algorithms. MVS main algorithms to 3D reconstruction are Depth map reconstruction and Point-cloud reconstruction, another algorithms for data fusion and refinement are Volumetric data fusion and MVS mesh refinement [9].

Photo-consistency measures

For a given set of images and a 3D point (*p*) seen by all the images is photo-consistency of *p* seen by images I_i and I_j written as:

$$C_{ij(p)} = \rho(I_i(\Omega(\pi_i(p))), I_j(\Omega(\pi_j(p))))$$

Where $C_{ij(p)}$ is photo-consistency, $\rho(f, g)$ is similarity of vectors *f* and *g*, $\pi_i(p)$ is projection *p* on image *i*, $\Omega(x)$ is support domain (kernel) around point *x* and $I_i(x)$ are sampled image intensities within the domain (kernel window).

Example of rectangular 3x3 photo-consistency domain Ω which is centered around pixel point *e* is shown in the figure 9. Photo-consistency is comparing 1D vectors containing image intensity in the domain. Image intensities in the domain are ordered into 1D vector $\vec{f} = (a, b, c, d, e, f, g, h, i)$.

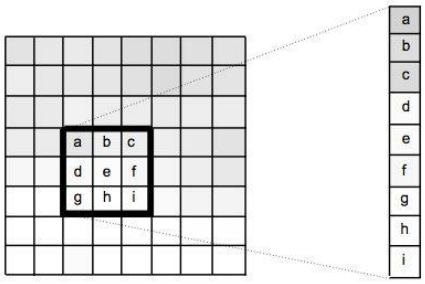

Figure 9: Photo-consistency domain around point *e* and transformation into vector [9].

For photo-consistency measures it can be used Sum of Squared Differences (SSD), Sum of Absolute Differences (SAD), Normalized Cross Correlation (NCC), Census, Rank and Mutual Information (MI).

Normalized cross correlation

Zero-mean normalized cross correlation (NCC) is one of the most MVS photo-consistency measures. It is invariant to changes in grain and bias. It is widely used when illumination and material invariance is necessary. NCC usually fails due to lack of texture or repetitive textures. The main reason to use NCC is its accuracy. The NCC similarity measure is:

$$\rho_{NCC}(f,g) = \frac{(f-\bar{f})\cdot(g-\bar{g})}{\sigma_f \cdot \sigma_g} \in [-1,1]$$

Where $\bar{f}$ is the mean of f and σ_f is standard deviation of f.

For handling color images, it is recommended to compute NCC independently per channel and compute average NCC score.

1. Depth map reconstruction

Depth map scene representation is very popular due to its flexibility and scalability into large models. Depth map can be reconstructed even for small number of neighboring images. Depth map can be imagined as an 2D array of 3D points. A scene represented by depth maps can be thought as merged 3D point cloud model. Depth map reconstruction takes a set of images and image parameters adjusted by SfM, and compute 3D geometry for reference image.

Winner-Takes-All algorithm

Is one of a simple depth map reconstruction algorithm. Photo-consistency values are computed along epipolar line for each neighboring image. Depth value is computed from the highest photo-consistency score which is achieved from a set of photo-consistency functions (neighboring images), see figure 10.

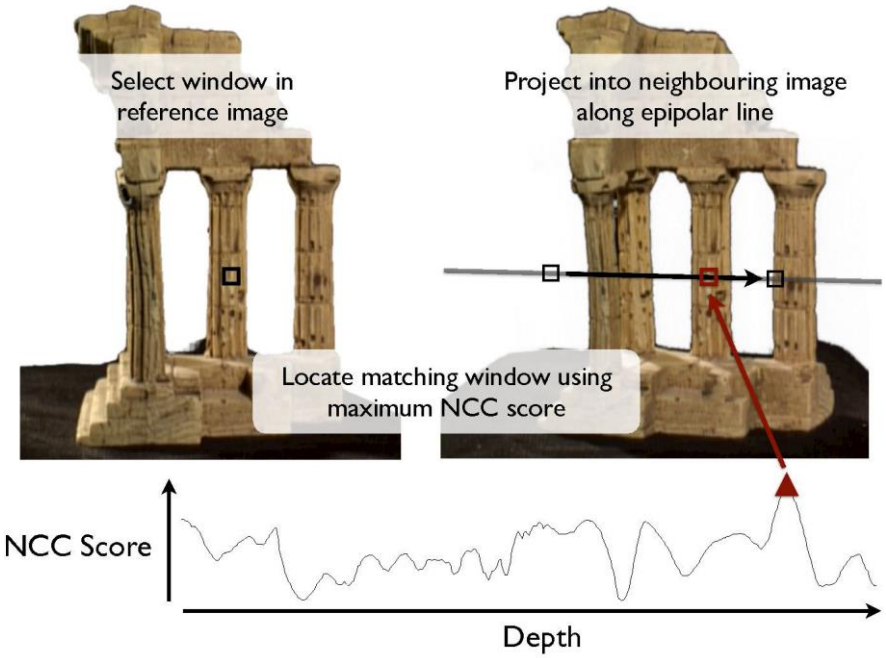

Figure 10:Winner-Takes-All algorithm [9], the highest NCC score of photo-consistency function is used for depth reconstruction.

Robust Photo-Consistency Depth maps

Occlusions and other effects bring noise to photo-consistency function. For pixel in the reference image algorithm computes photo-consistency functions of neighboring images. Local maxima are identified from a set of photo-consistency functions. Robust photo-consistency function $C^R(d)$ is given:

$$C^R(d) = \sum_k C_k \cdot W \cdot (d - d_k)$$

Where W is a kernel function (weight).

Simple average gives incorrect depth, while robust photo-consistency gives right depth, see figure 11.

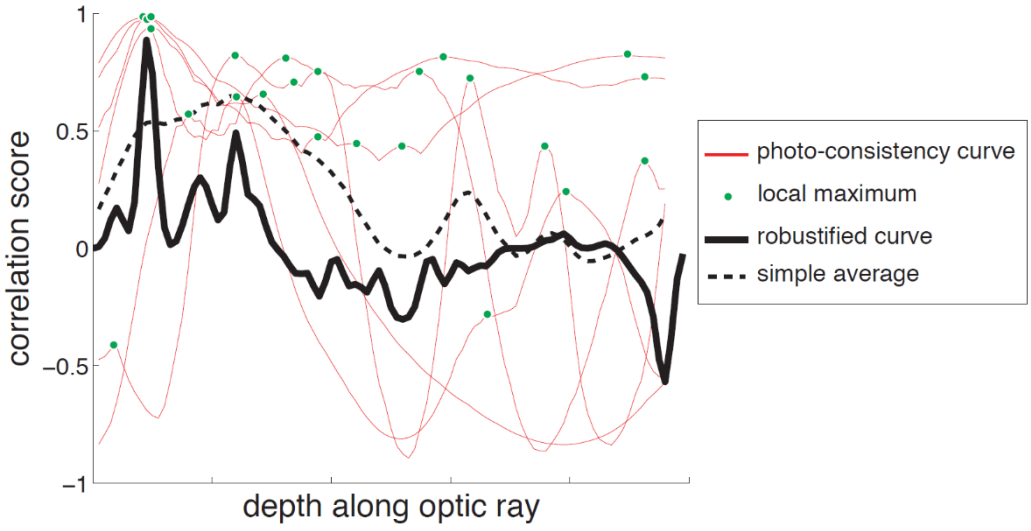

Figure 11:Robust photo-consistency function [9].

Depth map reconstruction another approaches

Thresholding photo-consistency scores can be another approach. Algorithm ignores photo-consistency scores below a chosen threshold. Markov Random Field (MRF) Depth maps is a combined solution that considers photo-consistency and spatial consistency together. Spatial consistency uses an assumption that neighboring pixels have similar depths.

The main disadvantage of depth maps is how to merge them into 3D scene. The quality of depth map is decreasing with depth discontinuities and occlusions. The accuracy of the depth map is inverse to the distance to the surface. Optimization of depth maps is very large and computationally expensive.

2. Point cloud reconstruction

Point cloud or patched based surface is less computationally expensive for optimization. Point cloud reconstruction algorithms are using spatial consistency assumption and expand point cloud on the surface during reconstruction process.

Patch based reconstruction

Patch can be imagined as a tangent plane to reconstructed surface. For patch *p* algorithm determines its position ***c(p)*** and normal vector ***n(p)***. Photo-consistency function is extended and consist of position of patch and surface normal.

Initial step

First step use detected features and triangulate them (if they are not yet triangulated). For initialization ***c(p)*** is achieved from triangulation and ***n(p)*** is set to image projection center. Detailed description of algorithm is in the figure 12, for more details visit [10].

Input: Features detected in each image.
Output: Initial sparse set of patches P.

Cover each image with a grid of $\beta_1 \times \beta_1$pixel^2 cells;
$P \leftarrow \phi$;
For each image I with optical center O
  For each feature f detected in I and lying in an empty cell
    $F \leftarrow$ {Features satisfying the epipolar consistency};
    Sort F in an increasing order of distance from O;
    For each feature $f' \in F$
      $R(p) \leftarrow I$; $T(p) \leftarrow \{J | N(p,R(p),J) \geq \alpha_0\}$;
      $c(p) \leftarrow$ 3D point triangulated from f and f';
      $n(p) \leftarrow$ Direction of optical ray from $c(p)$ to O;
      $n(p), c(p) \leftarrow \text{argmax}\, \bar{N}(p)$;
      $S(p) \leftarrow \{J | N(p,R(p),J) \geq \alpha_0\}$;
      $T(p) \leftarrow \{J | N(p,R(p),J) \geq \alpha_1\}$;
      If $|T(p)| \geq \gamma$
        register p to the corresponding cells in $S(p)$;
        exit innermost *For* loop, and add p to P.

Figure 12: Initial feature matching algorithm [10].

Expansion

Algorithm identifies neighboring cells of patch *p* which do not contain any patches. For each empty cell a new patch *p'* is created. Normal of patch *p'* is set as equal to patch *p*. Set of images where the patch *p'* is visible is set as the same for patch *p*. Position ***c(p')*** is initialized as point where viewing point intersects patch *p* plane. Optimization can be started. During optimization patch position and normal of the patch are adjusted and set of images where new patch is visible is updated. Figure 13 shows patch expansion algorithm, more details in [10].

Input: Patches P from the feature matching step.
Output: Expanded set of reconstructed patches.

Use P to initialize, for each image, Q_f, Q_t, and its depth map.
While P is not empty
  Pick and remove a patch p from P;
  For each image $I \in T(p)$ and cell $C(i,j)$ that p projects onto
    For each cell $C(i',j')$ adjacent to $C(i,j)$ such that $Q_t(i',j')$
    is empty and p is not *n-adjacent* to any patch in $Q_f(i',j')$
      Create a new p', copying $R(p'), T(p')$ and $n(p')$ from p;
      $c(p') \leftarrow$ Intersection of optical ray through
            center of $C(i',j')$ with plane of p;
      $n(p'), c(p') \leftarrow \text{argmax}\bar{N}(p')$;
      $S(p') \leftarrow \{$Visible images of p' estimated by the
            current depth maps $\} \cup T(p')$;
      $T(p') \leftarrow \{J \in S(p') | N(p', R(p'), J) \geq \alpha_1\}$;
      If $|T(p') < \gamma|$, go back to *For*-loop;
      Add p' to P;
      Update Q_t, Q_f and depth maps for $S(p')$;
Return all the reconstructed patches stored in Q_f and Q_t.

Figure 13: Patch expansion algorithm [10].

Filtering

For filtering a distance along the normal is compared to a chosen threshold (γ_d).

$$|(\boldsymbol{c}(\boldsymbol{p}) - \boldsymbol{c}(\boldsymbol{p'})) \cdot n(p)| + |(\boldsymbol{c}(\boldsymbol{p}) - \boldsymbol{c}(\boldsymbol{p'})) \cdot n(p')| < \gamma_d$$

Reference:

[1] *OpenCV: OpenCV-Python Tutorials*. OpenCV documentation [online]. Dostupné z: https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html. [cit. 2024-07-30].

[2] LOWE, David G. Distinctive Image Features from Scale-Invariant Keypoints. Online. *International Journal of Computer Vision*. 2004, roč. 60, č. 2, s. 91-110. ISSN 0920-5691. Dostupné z: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[3] BAY, Herbert; TUYTELAARS, Tinne a VAN GOOL, Luc. SURF: Speeded Up Robust Features. Online. In: LEONARDIS, Aleš; BISCHOF, Horst a PINZ, Axel (ed.). *Computer Vision – ECCV 2006*.

Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, s. 404-417. ISBN 978-3-540-33832-1. Dostupné z: https://doi.org/10.1007/11744023_32.

[4] ROSTEN, Edward a DRUMMOND, Tom. Machine Learning for High-Speed Corner Detection. Online. In: LEONARDIS, Aleš; BISCHOF, Horst a PINZ, Axel (ed.). *Computer Vision – ECCV 2006*. *Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, s. 430-443. ISBN 978-3-540-33832-1. Dostupné z: https://doi.org/10.1007/11744023_34.

[5] CALONDER, Michael; LEPETIT, Vincent; STRECHA, Christoph a FUA, Pascal. BRIEF: Binary Robust Independent Elementary Features. Online. In: DANIILIDIS, Kostas; MARAGOS, Petros a PARAGIOS, Nikos (ed.). *Computer Vision – ECCV 2010*. *Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, s. 778-792. ISBN 978-3-642-15560-4. Dostupné z: https://doi.org/10.1007/978-3-642-15561-1_56.

[6] RUBLEE, Ethan; RABAUD, Vincent; KONOLIGE, Kurt a BRADSKI, Gary. ORB: An efficient alternative to SIFT or SURF. Online. In: *2011 International Conference on Computer Vision*. IEEE, 2011, s. 2564-2571. ISBN 978-1-4577-1102-2. Dostupné z: https://doi.org/10.1109/ICCV.2011.6126544.

[7] HARTLEY, Richard a ZISSERMAN, Andrew. *Multiple view geometry in computer vision*. 2nd ed. Cambridge: Cambridge University Press, 2003. ISBN 978-0521540513.

[8] FISCHLER, Martin A. a BOLLES, Robert C. Random sample consensus. Online. *Communications of the ACM*. 1981, roč. 24, č. 6, s. 381-395. ISSN 0001-0782. Dostupné z: https://doi.org/10.1145/358669.358692.

[9] FURUKAWA, Yasutaka a HERNÁNDEZ, Carlos. Multi-View Stereo: A Tutorial. Online. *Foundations and Trends® in Computer Graphics and Vision*. 2015, roč. 9, č. 1-2, s. 1-148. ISSN 1572-2740. Dostupné z: https://doi.org/10.1561/0600000052.

[10] FURUKAWA, Yasutaka a PONCE, Jean. Accurate, Dense, and Robust Multiview Stereopsis. Online. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2010, roč. 32, č. 8, s. 1362-1376. ISSN 0162-8828. Dostupné z: https://doi.org/10.1109/TPAMI.2009.161.