Anatomy of a Photogrammetry Pipeline

Contents

1 Keypoint Identification

Given a set of images $I_1, ..., I_n$, which we will assume to be of grayscale $0, ..., 255$ intensities, the first step of the pipeline is to identify distinctive locations, i.e. keypoints. This chapter will be split in two sections, 'keypoint extraction' will focus on identifying keypoint location and sizes, whereas 'keypoint description' will illustrate how each keypoint may be supplied with an accurate and discriminative descriptor vector.

1.1 Keypoint Extraction

Blob Detection. Within the computer vision domain, Li et al. (2015) classified different feature detectors as edge detectors, corner detectors, and blob detectors. For keypoint extraction purposes usually blob detection algorithms are used.
Consider two random binary image patches in figure 1. Let the red squares denote candidate keypoints. The 3×3 patches to the side of the image patches denote the extracted candidate keypoints, and variations created by slightly shifting it to either side. Let (x_0, y_0) denote the centre of the candidate keypoint. Now, for all shift vectors (u, v), a good 3×3 blob keypoint would maximise:

$$\sum_{i,j \in \{-1,0,1\}} |I(x_0 + i, y_0 + j) - I(x_0 + i + u, y_0 + j + v)|. \tag{1}$$

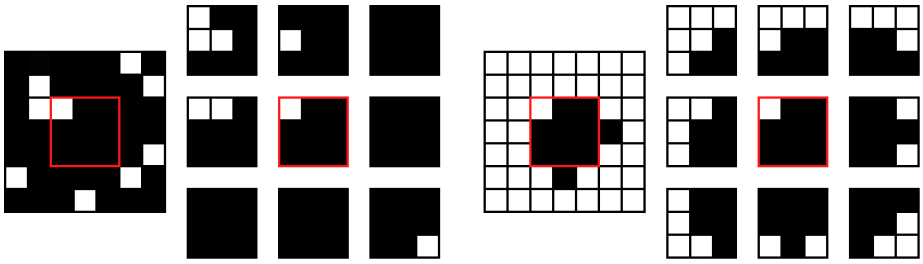

Figure 1: Two random binary image patches (7×7), with candidate keypoints marked in red. To the side are patches (3×3) that denote the keypoint, created by using different shift vectors (u, v) where $u, v \leq 1$.

Considering only sift vectors where $u, v \leq 1$, the candidate keypoint on the left produces absolute differences of $\{2, 2, 1, 1, 1, 1, 2\}$ in natural reading order. On the other hand, the second keypoint produces $\{5, 3, 3, 3, 3, 3, 3, 4\}$, making it a better blob keypoint.

Laplacian of Gaussian. In practice it is difficult to explicitly compute the difference as described in equation 1. A multitude of methods for finding such locations by different means exist. Li et al. (2015) identify the Laplacian of

Gaussian (LoG) as a classical method for blob detection. Consider an image $I(x, y)$, a Gaussian scale-space can first be constructed as:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y), \tag{2}$$

where $*$ denotes the convolution operator, and:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}. \tag{3}$$

Consequently the Laplace operator is applied to the scale space as:

$$\nabla^2 L(x, y, \sigma) = L_{xx}(x, y, \sigma) + L_{yy}(x, y, \sigma). \tag{4}$$

Now, extrema in $\nabla^2 L$ will correspond to blob keypoints in I of radius $r = \sqrt{2\sigma}$. Intuitively one may think of it as the difference in equation 1 being large when the partial derivatives are large at the image location.

Difference of Gaussian. A remaining problem of the LoG keypoint detection algorithm is the computation of $\nabla^2 L$ in practice. Presumably the most common approach to overcome this problem is the Difference of Gaussian (DoG), as first introduced within the popular scale invariant feature transform (SIFT) framework (Lowe, 2004). Consider again the scale-space of an image $I(x, y)$ as defined in equation 2. The DoG-space can be defined as:

$$\begin{aligned} D(x, y, \sigma) &= \big(G(x, y, k\sigma) - G(x, y, \sigma)\big) * I(x, y), \\ &= L(x, y, k\sigma) - L(x, y, \sigma), \end{aligned} \tag{5}$$

where $k > 1$. This DoG-space can be seen as an approximation to $\nabla^2 L$:

$$\nabla^2 L(x, y, \sigma) \approx \frac{\sigma}{\Delta\sigma}\Big(L(x, y, \sigma + \Delta\sigma) - L(x, y, \sigma)\Big) \approx D(x, y, \sigma). \tag{6}$$

Thus, extrema in $D(x, y, \sigma)$ again correspond to keypoints of radius $r = \sqrt{2\sigma}$ in I. For the explicit relation between the DoG-space and the LoG, see Lindeberg (2015, p. 31).

Discrete Difference of Gaussian. In order to handle a range of different σ values, a pyramid approach is normally used (Lowe, 2004). First, the scale-space $L(x, y, \sigma)$ is discretely computed as $L^{o,s}$, where o denotes the number of the octave, and s the number of the scale. A configuration used by SIFT is shown in figure 2.

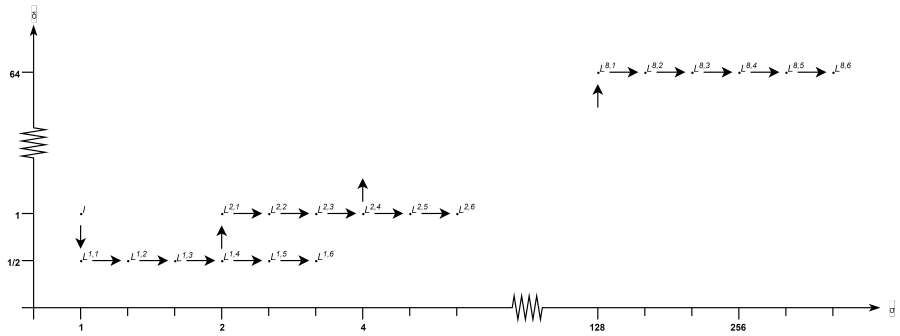

Figure 2: Default configuration of the discrete scale-space used by SIFT.

Here, the σ-axis denotes the scale, and the δ-axis denotes the inter-pixel distance. An arrow up or down corresponds to sub-sampling or expansion of the image using bi-linear interpolation, respectively. Furthermore, an arrow to the right denotes convolving the image with the Gaussian defined in equation 3, where $\sigma = \sqrt[3]{2}$. Thus, three sequential arrows to the right double the scale, since (Lowe, 1999, p. 3):

$$G(x, y, \sqrt[3]{2}) * \left(G(x, y, \sqrt[3]{2}) * \left(G(x, y, \sqrt[3]{2}) * I(x, y) \right) \right) = G(x, y, 2) * I(x, y). \quad (7)$$

Suppose we consider $L^{o,s}$ that corresponds to some discrete scale σ_0. Now, $L^{o,s+1}$ corresponds to the scale $\sqrt[3]{2}\sigma_0$. Thus a discrete DoG-space can be computed using equation 5 with $k = \sqrt[3]{2}$ as:

$$D^{o,s} = L^{o,s+1} - L^{o,s}. \quad (8)$$

The resulting configuration for the discrete DoG-space is depicted in figure 3.

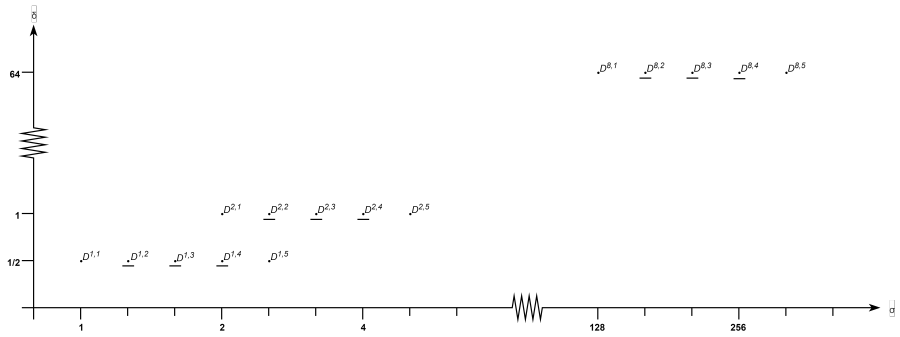

Figure 3: Default configuration of the discrete DoG-space used by SIFT.

Again, one is interested in extrema in the discrete DoG-space, as these correspond to keypoint locations in our image. To find discrete extrema, a pixel in

$D^{o,s}$ is compared to its 26 neighbours. Note that a pixel has eight neighbours in $D^{o,s}$ itself, and both nine in $D^{o,s-1}$ and $D^{o,s+1}$. Note also that this excludes the first and last scale from each octave. The discrete spaces eligible for extrema detection are underlined in figure 3. When some pixel (x_0, y_0) in $D^{o,s}$ that corresponds to the scale σ_0 is such an extrema, (x_0, y_0, σ_0) is considered as a keypoint.

Keypoint Refinement. The previously described scanning for discrete extrema is sensitive to noise and may produce unstable keypoints (Rey Otero and Delbracio, 2014). Therefore, the locations of keypoints is usually revised, and outliers are filtered. First, the position can be refined by locally fitting a three-dimensional quadratic function to the DoG-space as originally proposed by Brown and Lowe (2002). This is especially important for keypoints at higher octaves, as the sub-sampling has created large inter-pixel distances δ here. Next, low contrast keypoints are discarded based on some threshold:

$$\textbf{if } |D(x_0, y_0, \sigma_0)| < C_{DoG} \textbf{ then discard } (x_0, y_0, \sigma_0) \text{ as keypoint.} \tag{9}$$

As described before, for keypoint in a photogrammetry context we are interested in blob features, and not so much in edge features. However, the discrete DoG-space approach may unintentionally pick up on edge features. As a last step these are filtered based on the Hessian matrix at the keypoint location:

$$H_{(x_0,y_0,\sigma_0)} = \begin{bmatrix} D_{xx}(x_0, y_0, \sigma_0) & D_{xy}(x_0, y_0, \sigma_0) \\ D_{yx}(x_0, y_0, \sigma_0) & D_{yy}(x_0, y_0, \sigma_0). \end{bmatrix} \tag{10}$$

In practice this matrix can be computed using a finite difference scheme (Rey Otero and Delbracio, 2014). The eigenvalues of H will be close to equal for a blob keypoint, while an edge will have a dominant gradient direction and thus a large ratio between eigenvalues. The last filter is defined as:

$$\textbf{if } \frac{\lambda_{max}}{\lambda_{min}} > C_{Edge} \textbf{ then discard } (x_0, y_0, \sigma_0) \text{ as keypoint,} \tag{11}$$

where λ denotes the eigenvalues of $H_{(x_0,y_0,\sigma_0)}$. In practice the eigenvalues of the Hessian need not actually be computed, as only the ratio is relevant, which can be derived from the trace and determinant of the Hessian, see Rey Otero and Delbracio (2014, p. 382).

1.2 Keypoint Description

At this point a set of keypoints and their scale is known, i.e., we have a set of keypoints as (x_0, y_0, σ_0). The keypoints are assigned a description such that corresponding keypoints in different images can be matched later. First, a dominant orientation is assigned, after which a description vector is created.

Keypoint Orientation. For some keypoint (x_0, y_0, σ_0) a histogram of gradient orientations is created around the keypoint in the image $L^{i,j}$ that is closest to the scale σ_0. Typically this histogram has 36 bins, while each contribution is

weighted by the gradient magnitude as well as a Gaussian that has its origin in the keypoint as has a sigma of $1.5\sigma_0$. Finally, the histogram can be smoothed, for example by iteratively applying a three-tap box filter, see Rey Otero and Delbracio (2014). Define h_i as the value of bin i that corresponds to the orientation θ_i. A reference orientation is selected for a keypoint if:

$$\begin{cases} h_i > h_{i^-}, & \text{where } i^- = (i-1)\text{mod}(36), \\ h_i > h_{i^+}, & \text{where } i^+ = (i+1)\text{mod}(36), \\ h_i \geq 0.8 \cdot \max_k(h_k). \end{cases} \tag{12}$$

Notice that this selects an orientation if it is a local maximum that is at least larger as 80% of the global maximum. Therefore, multiple orientations may be selected, in which case the keypoint is 'split' in multiple keypoints at the same location with different orientations. Finally, the keypoint orientation for $(x_0, y_0, \sigma_0, \theta_0)$ is refined as:

$$\theta_0 = \theta_i + 10\left(\frac{h_{i^-} - h_{i^+}}{h_{i^-} - 2h_i + h_{i^+}}\right), \tag{13}$$

where θ_i is the selected maximum in equation 12.

Keypoint Description Vector. For each keypoint $(x_0, y_0, \sigma_0, \theta_0)$ a square patch is extracted that has a size corresponding to the scale σ_0, such that the reference orientation θ_0 now points upwards, see figure 4.

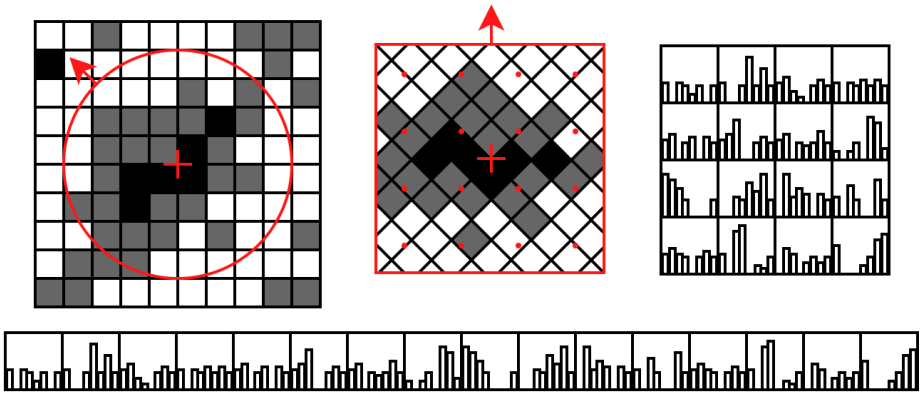

Figure 4: Given a keypoint (first image) with reference orientation, a square patch is extracted (second image) that has a normalised orientation. Next, gradient orientation in this square patch is described by an array of histograms (third image), which will be converted to a description vector $\boldsymbol{f}$ (fourth image).

The description vector will be created out of surrounding gradient orientations. First, an array of four by four histograms is created in the extracted patch, that represent orientation by eight bins. Every gradient orientation value in

the extracted patch corresponds bi-linearly to the centres of the four nearest histograms. Furthermore, a contribution to a histogram is also split linearly to its two nearest corresponding bins. The array of histograms is then used to create a 128-dimensional ($4 \times 4 \times 8$) feature vector, see figure 4. Once the feature vector $\boldsymbol{f}$ is constructed for a keypoint, each component is first revised as $\boldsymbol{f}_k = \min(\boldsymbol{f}_k, 0.2 \cdot \|\boldsymbol{f}\|)$ as to reduce the impact of illumination changes (Rey Otero and Delbracio, 2014). After this the vector is normalised such that $\|\boldsymbol{f}\|_2 = 512$. Finally, the vector is converted to an 8-bit integer vector as $\boldsymbol{f}_k = \min(\lfloor \boldsymbol{f}_k \rfloor, 255)$ in order to lower computational costs.

2 Keypoint Matching

At this point we both our initial set of images $I_1, ..., I_n$ and a set of keypoints $(x_0, y_0, \sigma_0, \theta_0)$ each with some descriptor $\boldsymbol{f}$ for each image. The goal of keypoint matching is to find images that overlap each other, and identify a set of keypoints in the first image that corresponds to another set of keypoints in the second image.

2.1 Two Nearest Neighbours

For a single keypoint $(x_0, y_0, \sigma_0, \theta_0)$ with descriptor $\boldsymbol{f}$ we are essentially searching for the nearest neighbour to $\boldsymbol{f}$ in the space of all descriptors (except those from the same image). For such a match it is useful to evaluate the quality of the matching, however, using for example the distance between the vectors will not suffice as the descriptor space is non-linear. Therefore, one often searches for two nearest neighbours (2-NN) to $\boldsymbol{f}$ instead of one. When:

$$\|\boldsymbol{f} - \boldsymbol{f}_{n_1}\|_2 < C_{match} \|\boldsymbol{f} - \boldsymbol{f}_{n_2}\|_2 , \tag{14}$$

we accept $\boldsymbol{f}_{n_1}$ as a match to $\boldsymbol{f}$, where $\boldsymbol{f}_{n_1}$ is the nearest neighbour, $\boldsymbol{f}_{n_2}$ is the second nearest neighbour, and C_{match} is for example equal to 0.8. This criterion is effective in assessing the quality of a match, however, it may kill keypoint matches in repetitive structures (Lowe, 2004).

2.2 Approximate Nearest Neighbours

In a typical application one often extracts in the order of 10.000 keypoints per image. When the number of images grows large as well, it is desirable to speed-up the procedure of finding the two nearest neighbours. This may be done using approximate nearest neighbour (A-NN) searches, or by matching images beforehand, thus also possibly creating an approximate nearest neighbour result.

Approximate Vector Matching. Using the classical A-NN approach of space partitioning by the means of building a tree, an approximate match to $\boldsymbol{f}$ can be found in the complete set of descriptors. Muja and Lowe (2009) identify two methods for building this tree, either hierarchical k-means trees, or randomised

$k-d$ trees. They conclude that hierarchical k-means is best suited for a general purpose, while $k-d$ trees may perform better on certain datasets, therefore, we will focus on hierarchical k-means trees. Given the set of all keypoints, k-means clustering is applied, which determines k cluster centres. Using these cluster centres, the space can be partitioned using Voronoi diagrams. After this, each partitioned space can be divided into k new partitions again, see figure 5.

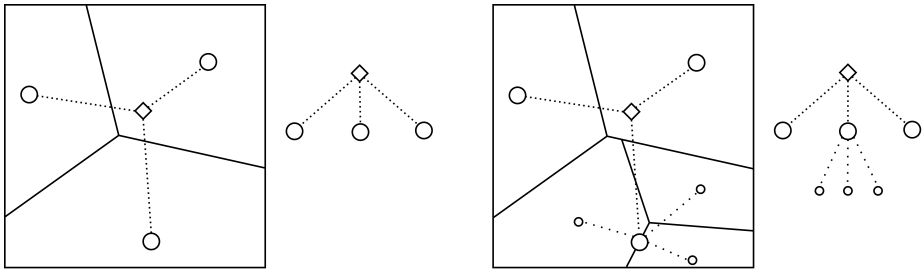

Figure 5: Creation of the hierarchical k-means tree using $k = 3$. A circle represents a centre of one of the k-means clusters, the diamond represents the top of the tree. The tree corresponding to the hierarchical clustering at each step is shown to the right.

This process may be carried out for a certain number of layers, or until each partitioned space contains k or less descriptors. Having constructed this tree once for the set of all descriptors, A-NN searches may for example be carried out by starting on the leaf corresponding to $\boldsymbol{f}$ and traversing up the tree for a fixed number of nodes. Thus one compares only a limited set of candidate matches to $\boldsymbol{f}$.

Image Matching. Alternatively, one can limit the set of the descriptors that need to be compared to $\boldsymbol{f}$ by matching the image in which $\boldsymbol{f}$ was found to a limited set of different images. Image matching may be carried out by creating a vocabulary tree (Nist and Stew, 2006). This tree is again build by using for example hierarchical k-means clustering. Once a tree is built for the complete set of keypoint descriptors, and image descriptor may be defined for each image by passing the descriptors of all it's keypoints to the tree. The image descriptor is defined as the number of keypoint descriptors that end up at a certain node:

$$q_i = n_i \cdot w_i, \tag{15}$$

where q_i is the i-th position of the image descriptor $\boldsymbol{q}$, n_i is the number of keypoint descriptors that end up on leaf i of the tree, and optionally, w_i may be some sort of weighing of the leafs. Now, a distance between two images can be defined as: $d(I_1, I_2) = \left\| \boldsymbol{q}_1 / \left\| \boldsymbol{q}_1 \right\| - \boldsymbol{q}_2 / \left\| \boldsymbol{q}_2 \right\| \right\|$.

2.3 Geometric Filtering

Consider two image I_1 and I_2 that look at the same scenery. Consider some point in the scenery that is observed in the first image at location $\hat{\boldsymbol{p}}_1$ and in

the second image at $\hat{\boldsymbol{p}}_2$. All such corresponding points can be related by the essential matrix E as:

$$\hat{\boldsymbol{p}}_1 E \hat{\boldsymbol{p}}_2 = 0. \tag{16}$$

The essential matrix will be discussed in more detail in the next chapter, but it can be computed when at least 7 such correspondences are known. This relationship is commonly used in a random sample consensus (RANSAC) framework to remove outlier keypoint matches. Consider again two images I_1 and I_2 with n keypoint locations $\hat{\boldsymbol{p}}_1^{0,\dots,n}$ and $\hat{\boldsymbol{p}}_2^{0,\dots,n}$ respectively, such that $\hat{\boldsymbol{p}}_1^i \leftrightarrow \hat{\boldsymbol{p}}_2^i$ is a matched keypoint, etc. In the RANSAC framework 7 matched keypoints are randomly selected, and the essential matrix E' is derived from these supposed correspondences. Next, the number of inliers for this supposed essential matrix E' is determined as:

$$\hat{\boldsymbol{p}}_1^i E' \hat{\boldsymbol{p}}_2^i < \varepsilon, \tag{17}$$

for some threshold error ε. The random set of matched keypoints that produces the largest number of inliers is assumed to be correct. All the inliers for this set, and thus also the random set itself, is kept, while the other matched keypoints are now discarded. The parameter ε may be set by a user, or alternatively, one may determine it *a contrario*, see Moulon et al. (2012).

3 Structure from Motion

Currently a number of correspondences, i.e. matched keypoints, is known between the given set of images. These matched keypoints have been filtered such that the geometric relationship they define is consistent. The next step, Structure from Motion (SfM), aims to further understand this geometric relationship. First, key definitions will be given that describe this geometric relationship, such as the essential matrix. After that the incremental structure from motion algorithm will be described, which allows us to find these geometric relations. Alternatively, different approaches exist such as global SfM (Moulon et al., 2013), hierarchical SfM (Toldo et al., 2015), or multi-stage SfM (Shah et al., 2015).

3.1 The Essential Matrix

First, the construct of the essential matrix E will be derived in 3-dimensional space. Next, the relation between 3- and 2-dimensional coordinates, as well as homogenised coordinates will be explained. Finally, it is shown that the same essential matrix also relates points in 2-dimensional space.

Three-Dimensional Space. Consider two cameras in three dimensional space, and their separate 3-dimensional coordinate systems (x_1, y_1, z_1), or $\boldsymbol{x}_1$, and (x_2, y_2, z_2), or $\boldsymbol{x}_2$, respectively. Consider also a point in space that is seen by both cameras, see figure 6. Let $\boldsymbol{p}_1$ denote the position of this point with respect to the coordinate system of the first camera, and $\boldsymbol{p}_2$ the position of this point with respect to that of the second camera. Furthermore, let $\boldsymbol{t}$ denote the vector from the origin of the first coordinate system to the second coordinate system.

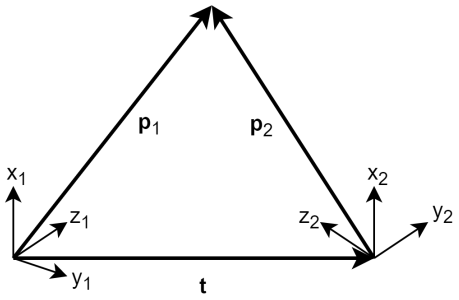

Figure 6: Two cameras and their 3-dimensional coordinate systems, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, and a single point in space that is seen by both cameras, $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$.

Since three points in space are always co-planar, that is, there exists a geometric plane on which all points lie, we can formulate a coplanarity conditions for $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, and $\boldsymbol{t}$, with respect to the first coordinate system as:

$$\boldsymbol{p}_1 \cdot (\boldsymbol{t} \times \boldsymbol{p}_2) = 0. \tag{18}$$

However, now $\boldsymbol{p}_2$ is a vector in the first coordinate system $\boldsymbol{x}_1$. In order to allow for the vector to be in its 'own' coordinate system, one needs to incorporate the rotation between the two coordinate systems. Consequently, there exists a 3×3 rotation matrix R between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ such that:

$$\boldsymbol{p}_1 \cdot (\boldsymbol{t} \times R\, \boldsymbol{p}_2) = 0, \tag{19}$$

where now, $\boldsymbol{p}_1$ is a vector in $\boldsymbol{x}_1$, and $\boldsymbol{p}_2$ is a vector in $\boldsymbol{x}_2$. Combining both $\boldsymbol{t}$ and R, the 3×3 essential matrix E is defined as $E = \boldsymbol{t} \times R$, such that:

$$\boldsymbol{p}_1^T E \boldsymbol{p}_2 = 0. \tag{20}$$

Three- and Two-Dimensional Space. Consider one camera and its corresponding coordinate system (x, y, z), or $\boldsymbol{x}$, see figure 7. In this coordinate system, the z-axis denotes the direction in which the camera is pointed. Consider also the image plane of the camera, which has coordinate system $(\hat{x}, \hat{y})$, or $\hat{\boldsymbol{x}}$. The image plane is located a focal length, f, behind the camera. However, note that for geometric reasons it may as well be depicted in front of the camera, as is done in the figure.

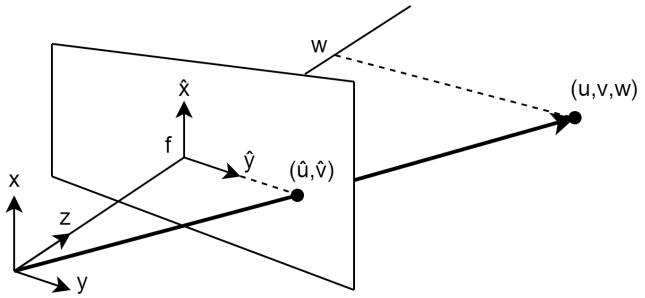

Figure 7: Relationship between a 3-dimensional coordinate system of a camera and the 2-dimensional coordinate system of the image plane corresponding to that camera.

A point (u, v, w) in the 3-dimensional coordinate system is related to a pixel $(\hat{u}, \hat{v})$ in the 2-dimensional coordinate system by:

$$\begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix} = \frac{f}{w} \begin{bmatrix} u \\ v \end{bmatrix}, \tag{21}$$

as $w > 0$ is in the viewing direction. Usually, cameras are normalised such that $f = 1$. In general, one can thus homogenise the two coordinate systems $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ as:

$$\begin{aligned} \begin{bmatrix} \hat{x} \\ \hat{y} \\ 1 \end{bmatrix} &= \frac{1}{z} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \\ \hat{\boldsymbol{x}} &= z^{-1} \boldsymbol{x}. \end{aligned} \tag{22}$$

Two-Dimensional Space. Consider again two cameras and their separate coordinate systems $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Consider also the corresponding images planes $\hat{\boldsymbol{x}}_1$ and $\hat{\boldsymbol{x}}_2$. Furthermore, suppose that the coordinate systems have been homogenised such that equation 22 holds. Now, consider a point that is observed by both cameras as $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, respectively. Previously, it was shown that these points are related by the essential matrix in 3-dimensional space as $\boldsymbol{p}_1^T E \boldsymbol{p}_2 = 0$. It can be shown that the corresponding pixels in the image planes $\hat{\boldsymbol{x}}_1$ and $\hat{\boldsymbol{x}}_2$ are related by the same essential matrix E:

$$\begin{aligned} \boldsymbol{p}_1^T E \boldsymbol{p}_2 &= 0, \\ \frac{1}{p_1^z p_2^z} \boldsymbol{p}_1^T E \boldsymbol{p}_2 &= 0, \\ (p_1^z)^{-1} \boldsymbol{p}_1^T E (p_2^z)^{-1} \boldsymbol{p}_2 &= 0, \\ \hat{\boldsymbol{p}}_1^T E \hat{\boldsymbol{p}}_2 &= 0, \end{aligned} \tag{23}$$

where p_1^z denotes the z-component of $\boldsymbol{p}_1$. Furthermore, it was used that $(p_1^z)^{-1} \boldsymbol{p}_1 = \hat{\boldsymbol{p}}_1$, as follows from the homogenised coordinates.

3.2 Incremental Structure from Motion

In incremental structure from motion, one starts with two cameras, defines rotation and translation between these cameras, and triangulates keypoints into space. Now, cameras may iteratively be added, for which we can follow the same steps. Additionally, when adding cameras bundle adjustment is usually performs, which optimises error over a larger set of cameras.

Initial Pair of Cameras. First, a connectivity graph G' is constructed, in which each camera is a node, and an edge is present between cameras when the images share a sufficient number of matched keypoints. Let G denote the largest connected graph within G'. The initial pair of cameras is selected as an edge in G that has both a large number of matched keypoints, but also a sufficiently wide baseline, i.e., a sufficiently large $\boldsymbol{t}$, see figure 6. This is required for reliable 3-dimensional reconstruction.
For this selected pair of cameras, the essential matrix E must be computed. Consider a matched keypoint $\hat{\boldsymbol{p}}_1 \leftrightarrow \hat{\boldsymbol{p}}_2$ between the images, equation 23 may be written as:

$$\begin{bmatrix} \hat{p}_1^x\hat{p}_2^x & \hat{p}_1^x\hat{p}_2^y & \hat{p}_1^x & \hat{p}_1^y\hat{p}_2^x & \hat{p}_1^y\hat{p}_2^y & \hat{p}_1^y & \hat{p}_2^x & \hat{p}_2^y & 1 \end{bmatrix} \begin{bmatrix} E_{11} \\ E_{12} \\ \vdots \\ E_{33} \end{bmatrix} = 0, \tag{24}$$

where $\hat{p}_1^x$ denotes the location of the keypoint in the first image along the x-axis, etc. Formulating it such as in equation 24, additional keypoints may be added to the leftmost matrix as extra rows. Now, we have $A\boldsymbol{x} = 0$, where $\boldsymbol{x}$ corresponds to E. The non-trivial solution can be found as the eigenvector corresponding to the zero eigenvalue of A^TA. Consider the singular value decomposition of A as $A = U\Sigma V^T$. Now, $\boldsymbol{x}$ is the column of V corresponding to the null eigenvalue on the diagonal of Σ, thus, the rightmost column of V in ordered single value decomposition.
Recall that the essential matrix consists of translation and rotation as $E = \boldsymbol{t}\times R$, see equation 19. Decomposing E again using singular value decomposition as $E = M\Sigma N^T$, we have that $\boldsymbol{t}$ is either m_3 or $-m_3$, where m_3 is the last column of M. Furthermore, R is equal to either MWN^T or MW^TN^T, where:

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{25}$$

Thus, there exist a total of four options for pose once the essential matrix is found. However, three of these correspond to a scenario in which the scene is behind the cameras. Thus, the correct pose can be uncovered by triangulating keypoints and verifying that all z values are positive. This is true since the z-axis corresponds to the viewing direction, see figure 7.

Next Best Camera. Having uncovered relative pose for two cameras and triangulated keypoints relative to that, an additional camera is added. Suppose a

visited edge and its nodes is added to the set V. The next camera is selected as an edge in the cut set $C = \{(u, v) \in G \mid u \in V, v \in G \setminus V\}$ that has the highest number of keypoints matched to the set of already triangulated keypoints. Now, $\boldsymbol{t}$ and R are estimated as before, and additional keypoints may be triangulated. Note that keypoints are triangulated based on the relative position of two cameras. When iteratively adding more and more cameras, error in pose may accumulate and cause incorrect triangulation. Therefore, bundle adjustment is usually performed after adding an additional camera. Bundle adjustment refers to the large optimisation problem of minimising all projection errors of the keypoints. More formally, consider a total of n keypoints and m cameras. Let v_{ij} be an indicator function that is equal to one when keypoint i is observed by camera j, and zero otherwise. Let the pose of camera j be denoted by the vector $\boldsymbol{e}_j$. Furthermore, let the location in 3-dimensional space of the keypoint i be given by $\boldsymbol{p}_i$. The function $P(\boldsymbol{p}_i, \boldsymbol{e}_j)$ denotes the 2-dimensional location of the keypoint i in the image plane of camera j as a result of the chosen projection location $\boldsymbol{p}_i$ and pose $\boldsymbol{e}_j$. On the other hand, let $\hat{\boldsymbol{x}}_{ij}$ denote the location of keypoint i in the image plane of camera j as previously determined by the keypoint identification process. Finally, let $d(\cdot, \cdot)$ denote the 2-dimensional euclidean distance. The bundle adjustment minimisation problem is given as:

$$\min_{\boldsymbol{e}_j, \boldsymbol{p}_i} \left(\sum_{i=1}^{n} \sum_{j=1}^{m} v_{ij} \; d\Big(P(\boldsymbol{p}_i, \boldsymbol{e}_j), \hat{\boldsymbol{x}}_{ij}\Big)^2 \right). \tag{26}$$

In this minimisation problem, the previously determined $\boldsymbol{t}$ and R serve as initialisation for $\boldsymbol{e}_j$, while $\boldsymbol{p}_i$ is initialised by the triangulation.
Bundle adjustment may be performed after adding a certain number of new cameras, and may be both global or locally. The latter here refers to optimising not the complete set of cameras, but for example only recently added cameras and their neighbours, thus keeping n and m small.

4 Depthmap Estimation

After structure from motion, the relative poses of cameras are known, as well as the locations of keypoints they share in 3-dimensional space relative to the cameras. This 3-dimensional point cloud is often too sparse for the purposes of either meshing or orthographic projections, thus extra points have to be added. This is achieved by computing depthmaps. First, key definitions for depthmap computations are given. Next, the algorithmic principles of depthmap or disparity map computation are given. And finally it is explained how this relates to a denser point cloud.

4.1 Epipolar Geometry

The principles of depthmap computation are largely based on the concept of epipolar geometry. The epipolar geometry can be seen as the geometric rela-

tionship between images that is defined by the essential matrix.

Epipolar Lines. Consider two cameras and their corresponding images I_1 and I_2 which have homogenised coordinate systems $\hat{\boldsymbol{x}}_1$ and $\hat{\boldsymbol{x}}_2$, respectively. Consider also a known essential matrix E that relates these images, as previously defined in equation 23. Suppose we were to fix some point $(\hat{u}_1, \hat{v}_1)$ in image I_1. Filling this in:

$$\begin{bmatrix} \hat{u}_1 \\ \hat{v}_1 \\ 1 \end{bmatrix}^T E \begin{bmatrix} \hat{x}_2 \\ \hat{y}_2 \\ 1 \end{bmatrix} = 0, \\ a\hat{x}_2 + b\hat{y}_2 + c = 0, \tag{27}$$

we find that the point $(\hat{u}_1, \hat{v}_1)$ must correspond to a pixel on the given line in image I_2. Here, $a = \hat{u}_1 E_{11} + \hat{v}_1 E_{21} + E_{31}$, etc. Intuitively one may think of it as follows, depending on the depth of the actual point corresponding to $(\hat{u}_1, \hat{v}_1)$, the location where it is observed in I_2 varies, see figure 8.

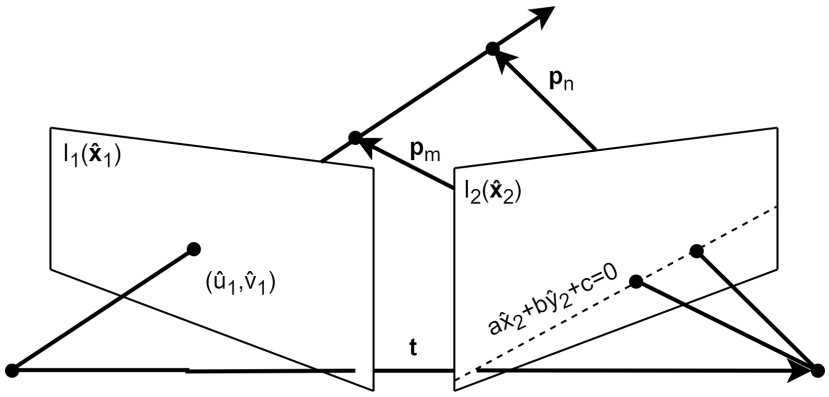

Figure 8: The point (a_1, b_1) in image I_1 corresponds to an epipolar line $ax_2+by_2+c = 0$ in image I_2. The exact correspondence to the point can be found along the line, and depends on the actual depth of the structure that is observed.

Conversely, any point in image I_2 that lies on the line $a\hat{x}_2 + b\hat{y}_2 + c = 0$, corresponds to one unique epipolar line in image I_1. This is true since the plane that spans either one of the set of vectors $\{\boldsymbol{t}, \boldsymbol{p}_m\}$ and $\{\boldsymbol{t}, \boldsymbol{p}_n\}$ will intersect the image plane I_1 on the same line. Thus, there exist pairs of epipolar lines in images I_1 and I_2, such that any point on one of the lines should exists somewhere on the other line.

Rectified Epipolar Geometry. Consider two images I_1 and I_2, their essential matrix E, and the epipolar lines that this essential matrix E defines. There exist two transformations H_1 and H_2, such that for $\tilde{I}_1 = H_1(I_1)$ and $\tilde{I}_2 = H_2(I_2)$ we

have a new essential matrix E^*, where:

$$E^* = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}. \tag{28}$$

An overview of this step, also referred to as 'rectification' of the epipolar geometry, is shown in figure 9.

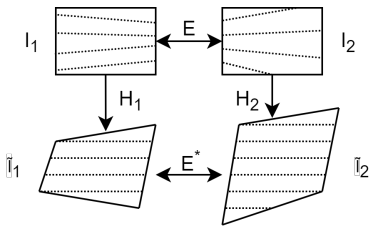

Figure 9: Images I_1 and I_2 are related by the essential matrix E. There exist two transformations H_1 and H_2, such that $\tilde{I}_1 = H_1(I_1)$ and $\tilde{I}_2 = H_2(I_2)$ are related by E^* as defined in equation 28.

Note that these transformations result in horizontal and aligned epipolar lines, also called 'scan-lines'. This is true since in $\tilde{I}_1$ and $\tilde{I}_2$ with coordinate systems $\tilde{\boldsymbol{x}}_1$ and $\tilde{\boldsymbol{x}}_2$ respectively, we have:

$$\begin{aligned} \begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \\ 1 \end{bmatrix}^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \\ 1 \end{bmatrix} &= 0, \\ \begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \\ 1 \end{bmatrix}^T \begin{bmatrix} 0 \\ 1 \\ -\tilde{y}_2 \end{bmatrix} &= 0, \\ \tilde{y}_1 &= \tilde{y}_2. \end{aligned}$$

The required transformations H_1 and H_2 for this operation can be defined as transformations of the images that map the so-called epipole to infinite point $(1, 0, 0)^T$. Here, the epipole of image I_1 is the point on the baseline $\boldsymbol{t}$ where the epipolar lines of I_1 would intersect. This transformation has many degrees of freedom, but is commonly constructed by requiring it to be a rigid transformation as far as is possible (Hartley, 1999).

4.2 Disparity Maps

As previously illustrated in figure 8, the location on the epipolar lines is dependent on the depth. Thus, an understanding of depth may be achieved by finding correspondences along these epipolar lines. Given images with a rectified

epipolar geometry, one can look along the scan-lines to find disparities. A large disparity indicates the object is close to the camera, and vice versa.

Matching Costs. Consider a pixel $\boldsymbol{p} = (p^x, p^y)$ in an image I_1 with intensity $I_1(\boldsymbol{p})$. Consider also a possible correspondence to this pixel in a second image I_2, $\boldsymbol{q}$ with intensity $I_2(\boldsymbol{q})$. Assume images I_1 and I_2 share a rectified epipolar geometry, then, $\boldsymbol{q}$ can be defined as $\boldsymbol{q} = (p^x + d, p^y)$, with d as disparity. For this candidate correspondence $\boldsymbol{q}$, or alternatively, for this candidate disparity d, Birchfield and Tomasi (1998) defined a cost function as:

$$c(\boldsymbol{p}, d) = \min_{d-\frac{1}{2} \leq \tilde{d} \leq d-\frac{1}{2}} \left| I_1\Big((p^x, p^y)\Big) - I_2\Big((p^x + \tilde{d}, p^y)\Big) \right|, \tag{29}$$

where the sub-pixel intensities in image I_2 are achieved by linear interpolation. Instead of the relatively simple Birchfield-Tomasi cost, the matching cost may be defined differently. For example, Hirschmüller (2005) defined a notable cost function based on mutual information, which is insensitive to recording and illumination changes. However, given the context, and for illustrative purposes, the Birchfield-Tomasi cost will be used here instead.

Disparity Computation. Simply minimising the above-mentioned matching cost often results in disparity maps that are subject to noise, and additionally, individual scan-lines are unrelated, thus a streaking effect is likely to occur (Hirschmüller, 2005). Therefore, additional constraints are often added that enforce a smoothness of the disparity map. This may be achieved by formulating a global energy function E for some disparity map D as:

$$E(D) = \sum_{\boldsymbol{p}} \left(c\Big(\boldsymbol{p}, D(\boldsymbol{p})\Big) + \sum_{\boldsymbol{q} \in N_{\boldsymbol{p}}} S\Big(D(\boldsymbol{p}), D(\boldsymbol{q})\Big) \right), \tag{30}$$

where c denotes a matching cost function such as equation 29, $D(\boldsymbol{p})$ denotes the disparity d assigned to pixel $\boldsymbol{p}$, $N_{\boldsymbol{p}}$ denotes all the neighbouring pixels of $\boldsymbol{p}$, and:

$$S\Big(D(\boldsymbol{p}), D(\boldsymbol{q})\Big) = \begin{cases} 0 & \text{if} \quad |D(\boldsymbol{p}) - D(\boldsymbol{q})| = 0, \\ K_1 & \text{if} \quad |D(\boldsymbol{p}) - D(\boldsymbol{q})| = 1, \\ K_2 & \text{if} \quad |D(\boldsymbol{p}) - D(\boldsymbol{q})| > 1, \end{cases} \tag{31}$$

where K_1 and K_2 are fixed numbers with $K_2 > K_1$. Thus, a penalty is introduced to the global energy function E when the disparities of neighbouring pixels are not equal. When the disparity change is small, at most one pixel, the penalty introduced by K_1 is relatively small. This is particularly helpful for preserving slanted or curved surfaces where the disparity ought to change gradually (Hirschmüller, 2005). Now, the problem can be formulated as finding the disparity map D that minimises $E(D)$. This minimisation problem may for example be solved in practice by formulating it as a graph cut problem, see Boykov et al. (2001) and Boykov and Kolmogorov (2004).

Disparity Refinement. Suppose we compute disparity maps D_1 and D_2 corresponding to a set rectified images I_1 and I_2, now consistency and occlusions can be checked. An occlusion can be defined as a region that is seen in one image, but is occluded, that is, not seen, in the other image. The disparities are refined as:

$$D_1(\boldsymbol{p}) = \begin{cases} D_1(\boldsymbol{p}) & \text{if} \quad |D_1(\boldsymbol{p}) - D_2(\boldsymbol{q})| \leq 1, \\ \text{occluded} & \text{otherwise.} \end{cases} \tag{32}$$

Here, $\boldsymbol{q}$ is the pixel in the second image that corresponds to $\boldsymbol{p} = (p^x, p^y)$ and is defined as:

$$\boldsymbol{q} = \Big(p^x + D_1(\boldsymbol{p}), p^y\Big). \tag{33}$$

This consistency check enforces the uniqueness constraint, such that mappings are one-to-one only (Hirschmüller, 2005). Additionally, steps such as median filters may be applied at different stages to remove outliers or peaks to further refine the results.

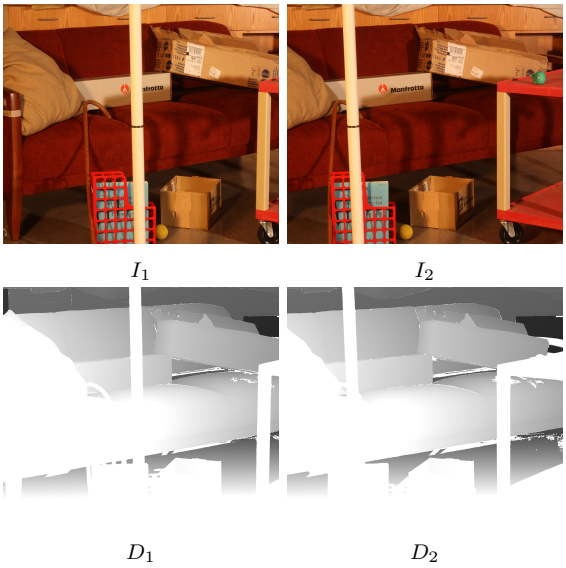

Figure 10: Example from the `vision.middlebury.edu/stereo/` page of two depthmaps D_1 and D_2 corresponding to I_1 and I_2 (Scharstein et al., 2014). Note that here, the images I_1 and I_2 share a known rectified epipolar geometry to begin with. Note also that for example the edge of the couch in I_1 cannot be found in I_2, hence, to fill such a part of D_1 supplementary images (shifted towards the other side) were required.

4.3 Dense Clouds

Given the sparse cloud of triangulated keypoints and camera poses as a result of structure from motion, the depthmaps may be used to add points to the sparse

cloud, thus creating a dense cloud. Suppose we have a known depthmap $D(\hat{\boldsymbol{x}})$ that looks at one of our previously triangulated keypoints $\boldsymbol{u}$. Note that thus $\boldsymbol{u}$ and $\hat{\boldsymbol{u}}$ are known. Now, consider $\hat{\boldsymbol{v}}$, an arbitrary pixel in our known depthmap that is not a keypoint. Using the relation between $D(\hat{\boldsymbol{v}})$ and $D(\hat{\boldsymbol{u}})$, as well as the known keypoint, the location of $\boldsymbol{v}$ can be estimated, see figure 11.

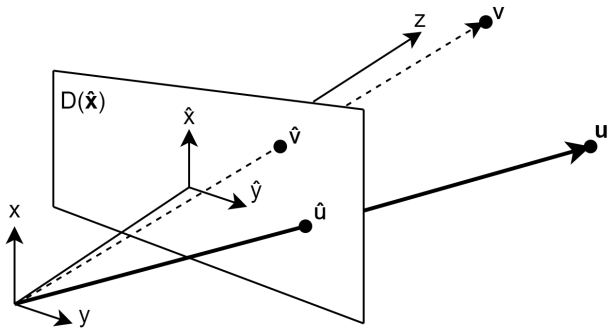

Figure 11: Process of filling in the sparse cloud using depthmaps. The pixel $\hat{\boldsymbol{v}}$ is used to generate an extra point $\boldsymbol{v}$ in the point cloud by referencing it to a known keypoint $\boldsymbol{u}$ using the depthmap D.

Now, depending on the resolution used while creating the depthmaps, up to a point per pixel may be added to the point cloud, thus creating the dense cloud. Note how regions that are occluded in one specific depthmap may be filled in from different depthmaps. Therefore, it is not advisable to use interpolation when creating depthmaps (Hirschmüller, 2005).

5 Orthographic Projections

In aerial imaging applications, all camera positions form approximately a plane. Additionally, the orientation of each camera is similar as well. Therefore, it is possible to position a plane directly below and parallel to the plane that the cameras span. This plane is split up in cells of the desired resolution, which should correspond to the 'resolution' of the dense cloud. Now, the dense cloud may be projected onto this plane at an angle of 90 degrees. For each cell, the height of projected values is selected as a means to eliminate remaining outliers in the depthmaps (Hirschmüller, 2005). The result of projecting the elevation values onto this common plane is referred to as a *digital elevation model* (DEM). It should be remarked that the resolution of depthmaps need not be equal to that of the original images, and is often lower. Therefore, when computing the *ortho-image*, a mesh need to be constructed first. This be done based on either -or a combination of- the point cloud and elevation model. Given such a mesh, all image intensities are projected to the same plane again, see figure 12.

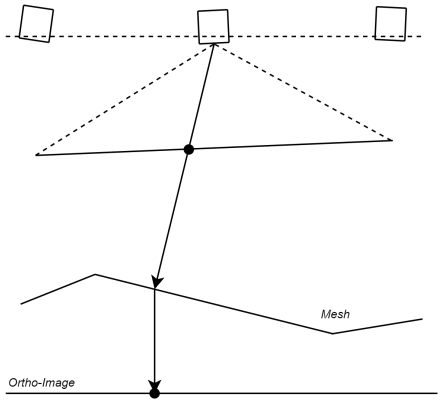

Figure 12: Side-view of images intensities being projected through the mesh onto the plane that will become the *ortho-image*. Notice that this plane is parallel to the span of camera locations.

The plane is again split up in cells, which may be of a different resolution, however, as that of the elevation model. For each cell centre, an intensity is calculated by bi-linearly interpolating the nearby projected colour intensities. Usually, only the projection of the closest camera is used here, leading to an interpolation of four values. The result of using this closest camera approach is referred to as an *orthomosaic* since it can create so-called 'seam-lines' when two cameras are equally close. Alternatively, one may for also choose to interpolate a multitude of projected intensities. Or choose to average or fade the result of the two closest cameras.

References

Birchfield, S. and C. Tomasi
1998. A Pixel Dissimilarity Measure That Is Insensitive to Image Sampling. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 20, NO. 4, APRIL 1998*, 20(4):401–406.

Boykov, Y. and V. Kolmogorov
2004. An Experimental Comparison of Min- Cut and Max-Flow Algorithms for Energy Minimization in Vision. 26(9):1124–1137.

Boykov, Y., O. Veksler, and R. Zabih
2001. Fast Approximate Energy Minimization via Graph Cuts 1 Energy minimization in early vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239.

Brown, M. and D. Lowe
2002. Invariant Features from Interest Point Groups. Pp. 1–23.

Hartley, R. I.
1999. Theory and practice of projective rectification. *International Journal of Computer Vision*, 35(2):115–127.

Hirschmüller, H.
2005. Accurate and efficient stereo processing by semi-global matching and mutua information. *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, II(2):807–814.

Li, Y., S. Wang, Q. Tian, and X. Ding
2015. A survey of recent advances in visual feature detection. *Neurocomputing*, 149(PB):736–751.

Lindeberg, T.
2015. *Image Matching Using Generalized Scale-Space Interest Points*, volume 52. Springer US.

Lowe, D. G.
1999. Object Recognition from Local Scale-Invariant Features. P. 150.

Lowe, D. G.
2004. Distinctive Image Features from Scale-Invariant Keypoints. *Journal of Computer Vision*, Pp. 1–28.

Moulon, P., P. Monasse, and R. Marlet
2012. Adaptive Structure from Motion with a contrario model estimation 1 Introduction 2 Structure from Motion the classical pipeline. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 7727 LNCS(PART 4):1–14.

Moulon, P., P. Monasse, and R. Marlet
2013. Global fusion of relative motions for robust, accurate and scalable

structure from motion. *Proceedings of the IEEE International Conference on Computer Vision*, Pp. 3248–3255.

Muja, M. and D. G. Lowe
2009. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP 2009 - Proceedings of the 4th International Conference on Computer Vision Theory and Applications*, 1:331–340.

Nist, D. and H. Stew
2006. Scalable Recognition with a Vocabulary Tree.

Rey Otero, I. and M. Delbracio
2014. Anatomy of the SIFT Method. *Image Processing On Line*, 4:370–396.

Scharstein, D., H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling
2014. High-resolution stereo datasets with subpixel-accurate ground truth. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8753(1):31–42.

Shah, R., A. Deshpande, and P. J. Narayanan
2015. Multistage SFM: Revisiting incremental structure from motion. *Proceedings - 2014 International Conference on 3D Vision, 3DV 2014*, Pp. 417–424.

Toldo, R., R. Gherardi, M. Farenzena, and A. Fusiello
2015. Hierarchical structure-and-motion recovery from uncalibrated images. *Computer Vision and Image Understanding*, 140:127–143.