# STRUCTURE FROM MOTION

by

Akash Chandra Shekar

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Computer Science

Charlotte

2018

Approved by:

Dr. Andrew Willis

Dr. Min Shin

Dr. Srinivas Akella

Dr. Hamed Tabkhi

# ABSTRACT

AKASH CHANDRA SHEKAR. Structure From Motion. (Under the direction of DR. ANDREW WILLIS)

The Structure from Motion (SfM) systems use knowledge of a camera's image formation model and a time-sequence of images from the camera to estimate camera motion and 3D scene structure. Formulations of this problem decompose the problem into two sub-problems: (1) solving fore the camera trajectory and solving for the depth of 3D scene points observed by the camera in multiple images. Solutions are provided by finding correspondences between $(x, y)$ locations in images take from different poses taken at distinct times in the camera trajectory. Traditional SfM find these correspondences by extracting and tracking features, e.g., corner points, match local regions within images. However, these approaches provide very sparse geometric descriptions of the scene structure and omit much of the geometric information in the image. Other , semi-dense, approaches attempt to find large collections of correspondences by solving for pixel-level correspondences which provide more geometric information about the scene. Unfortunately, unlike features, pixel values may are susceptible to viewpoint, illumination, and other image formation phenomena which requires semi-dense approaches to consider photometric correction in order to robustly determine correspondences.

# ACKNOWLEDGEMENTS

If you decide to have a acknowledgements page, your acknowledgement text would go here.

The Acknowledgement page should be brief, simple, and free of sentimentality or trivia. It is customary to recognize the role of the advisor, the other members of the advisory committee, and only those organizations or individuals who actually aided in the project. Further, you should acknowledge any outside source of financial assistance, such as GASP grants, contracts, or fellowships.

# DEDICATION

If you decide to have a dedication page, your dedication text would go here.

The Dedication page, if used, pays a special tribute to a person(s) who has given extraordinary encouragement or support to one's academic career.

# INTRODUCTION

If you decide to have an introduction page, your introduction text would go here.

Depending on the discipline or the requirements of the student's advisory committee, an Introduction may be included as a preliminary page.

TABLE OF CONTENTS

LIST OF FIGURES viii

LIST OF TABLES ix

LIST OF ABBREVIATIONS x

CHAPTER 1: INTRODUCTION 1

CHAPTER 2: RELATED WORK AND BACKGROUND INFORMATION 8

2.1. Image alignment 8

2.2. Two unknows of the SfM 10

2.2.1. Camera Trjectory 10

2.2.2. Depth Mapping 15

2.3. Non linear Optimization 27

CHAPTER 3: LSD SLAM 29

CHAPTER 4: Methodology 33

4.0.1. Time and Spatial Sampling Issues 33

4.0.2. Image Registration Issues 35

4.0.3. Resolving the Unknown SfM Scale 37

4.0.4. RGBD and SfM Depth Fusion 38

CHAPTER 5: Results 40

REFERENCES 44

# LIST OF FIGURES

FIGURE 2.1: Thin lens camera model 16

FIGURE 2.2: Pin-hole camera model 17

FIGURE 2.3: Epipolar geometry 21

FIGURE 4.1: An overview of the proposed depth fusion algorithm. 34

FIGURE 5.1: Results for three experiments are shown. Images shown are organized into separate columns. Column (a) shows a grayscale image of the scene (b) shows the sensed RGBD depth image (c) shows the SfM-estimated depth image (d) shows the fused image and (e) shows the standard deviation for fused depths (in $m$.). The fused image has been color-coded as follows: (white) denotes depth locations sensed only by the RGBD sensor, (yellow) denotes depth locations only sensed via SfM, (red) denotes fused (RGBD+SfM) depth locations and (black) denotes depth locations without RGBD or SfM measurements. 41

# LIST OF TABLES

TABLE 5.1: Ratios of depth measurements to total depths 40

# LIST OF ABBREVIATIONS

ECE An acronym for Electrical and Computer Engineering.

# CHAPTER 1: INTRODUCTION

The RGB-D camera captures depth information along with the RGB color information for every pixel. These cameras have the projector to projects and sensors to capture the infrared light. The depth informations can be obtained mainly by two different approaches, one is by projecting the known pattern of infrared light speckles and analyzing the deformation of light reflected back by the object, this method is called the structured light method and in another approach the depths are calculated by measuring the time taken by the infrared ray to reflect back to the sensors, this is called the Time-of-Flight technology. In both the case the infrared rays are play a crucial role, however the limitations of RGB-D camera it that it fails to work outdoor because of whitewash of infrared rays from sun, even indoor the RGB-D fails around the transparent and specular objects with high reflective index. We propose a method to take advantage of the Structure from Motion(SfM) to estimate depth of scene whenever the RGB-D fails to provide the required depth. The partial depths provided by the RGB-D camera can be used as a ground truth to jump-start the SfM, also these ground truth depths helps estimate the scale of the environment which is otherwise not possible with the SFM alone. The synergy of RGB-D and SfM provide better depth estimation.

Structure from Motion (SfM) is the photogrammetric process of estimating the three-dimensional structure of a scene from a set of two-dimensional images, this is achieved by tracking the motion of the camera generating these images and correlating information within the time-sequence of images. SfM has many applications in the field of robotics including augmented reality, geoscience and ***[put more here]***. In robotics, SfM is applied as one approach for visual odometry, the process of estimating

the egomotion of an robot using only the inputs of cameras attached to it, this is used highly in self driving car and unmanned aerial vehicle. In the field of augmented reality, SfM used to estimate the lost depths of the objects from images, this depth information enables to augment the virtual objects into the real world and even allows to interact with augmented objects. [depth map not defined, basic physical interaction vague].

The point in 3D can be estimated by triangulation of 2D projections from two or more images, i.e with the known camera orientation, we can back-project 2D points from images onto 3D environment and try to determine their intersection. In order to achieve that we need find a set of points in one image which can be identified as the same points in another image, this is called as image correspondence. Traditionally this was done by feature extraction and matching[1]. The feature extraction is the process of selecting the key points in the image which are unique and distinguishable and represent them in a efficient descriptor,which are later used for matching in the other image. There are many possible choices for features and descriptors, like SIFT, SURF, ORB etc. However in more recent implementations[**?**], instead of features, the image intensities are used directly for matching. Once the correspondence are established, the intersection between back-projected 3D points is estimated by minimize the re-projection error, which is done by imposing the epipolar constraint, which states that the image points and 3D points should be in a same plane. In order to address the noise in estimating the correspondence, a Gaussian model is assumed with zero mean, hence minimizing the re-projection error gives the maximum likelihood estimate of 3D point. On the other hand, with the known image correspondence and their 3D points, we can estimate the camera pose and orientation with the similar procedure. Where we try to solve for relative camera ordination which minimizes the re-projection error between known 3D and 2D point.

However, SfM is a chicken and egg problem with both depth and camera orientation

unknown.This again tries to minimize the re-projection error between the observed 2D image points and predicted (re-projected) 2D image point,which is expressed as the sum of squares of a large number of nonlinear, real-valued functions. Thus, the minimization is achieved using nonlinear least-squares algorithms like Guass-Newton or Levenberg–Marquardt. This process is also called as Bundle adjustment, as it is analogous to the process of optimally adjusting bundles of light rays originating from each 3D feature and converging them on each camera's optical center.

Over the years many real time approaches have be suggested to tackle this problem, and these approaches are broadly classified as Direct vs. Indirect, Dense vs. Sparse and Incremental vs. Global methods.

Direct vs. Indirect: All the SfM approaches have a underlaying probabilistic model in order to address the noise in measurements and camera representation, which uses the Maximum Likelihood approach to maximize the probability of obtaining the actual measurements, this differentiation of SfM is based on how the probabilistic models are defined. The direct method as the name specifies, directly uses the sensor values (pixel values) – light received from a certain direction over a certain time period – as as input for probabilistic model, hence this method tries to minimize the photometric error. This is analogous to the process of image alignment, whose goal is to minimize the sum of squared error between two images, the template T and the image I warped back onto the coordinate frame of the template.

$$\sum_{x} [I(\omega(x, p) - T(x)]^2 \tag{1.1}$$

1.1 is minimized with respect p, which in case of SFM are the camera orientation, this minimization is a process of non linear optimization.

On the other hand, the indirect method uses the intermediate representation of sensor measurements as input for probabilistic model, the sensor values can be either represented as sparse set feature key-points, this need a efficient feature detectors like

Scale Invariant Feature Transform (SIFT) to find the valid the correspondence between images, and the methods like Random Sample Consensus algorithm (RANSAC) are applied in order to avoid the outliers. An alternative to using key features, is to find the dense regularized optical flow, which tries to match the selected points from one image to another, assuming that both images are part of a sequence and relatively close to one another. Both this representation tries to minimize the geometric errors.

Dense vs. Sparse: As the name suggests the sparse methods uses only selected set of independent points, where as dense methods attempt to use and reconstruct all pixels in the 2D image domain. On the same line there is a intermediate approaches (semi-dense) tries to use the largely connected and well-constrained subset of points if not all of them. The choice of this method affects the addition of geometry prior, in the sparse formulation, since there is no notion of neighborhood, and geometry parameters (keypoint positions) are conditionally independent. Dense (or semi-dense) approaches on the other hand exploit the connectedness of the used image region to formulate a geometry prior, typically favoring smoothness.

Incremental vs. Global:Incremental SfM[2] begins by first estimating the 3D structure and camera poses of just two cameras based on their relative pose. Then additional cameras are added on incrementally and 3D structure is refined as new parts of the scene are observed. On the other hand global SfM[3] considers the entire problem at once. Here the objective is to estimate the global camera poses and 3D structure by removing outliers and by applying an averaging scheme [cite ???]. [why is this discussed? are their results different? is one of interest to you?] [cited eariler][isn't the LSD a incremental method]

LSD SLAM [**?**] is a semi-dense, direct method [you have not defined sparse methods, dense methods, direct methods, or indirect methods do this in a review of approaches above – similar to that from Direct Sparse Odometry article (https://arxiv.org/pdf/1607.02565.pdf) from TUM] [have done changes] optimizes the geometry directly on the image intensi-

ties, which enables using all information in the image.In addition to higher accuracy and robustness in particular in environments with little key-points, this provides substantially more information about the geometry of the environment.Images are aligned by Gauss -Newton minimization of the photometric error.

$$E(\xi) = \sum_i (I_{ref}(P_i) - I(\omega(p_i, D_{ref}(p_i), \xi)))^2$$

Where $D_{ref}$ is the estimated depth of reference frame,$\xi \in se(3)$ is Lie-algebra representation of rigid body motion and $\omega$ is the affine wrap function. The above error function gives the maximum-likelihood estimator for $\xi$ assuming i.i.d. Gaussian residuals. $\delta\xi^{(n)}$ is computed for each iteration by solving for the minimum of Gauss-Newton second-order approximation of $E$: [no description of image warping needed here, briefly discuss image warping for direct methods as a methods for image pixel matching see Baker2004 http://www.ncorr.com/download/publications/bakerunify.pdf]

$$\delta\xi^{(n)} = -(J^T J)^{-1} J^T r(\xi^{(n)})$$

with

$$J = \frac{\partial r(\epsilon \circ \delta\xi^{(n)})}{\partial \epsilon}$$

The new estimate is then obtained by multiplication with the computed update

$$\xi^{(n+1)} = \delta\xi^{(n)} \circ \xi^{(n)}$$

The overall system is composed of three major components, tracking, depth map estimation and map optimization.

The tracking component continuously estimates the rigid body pose with respect to the current key-frame, using the pose of the previous frame as initialization.LSD

slam tracks new frame by minimizing the variance-normalized photometric error

$$E_p(\xi_{ji}) = \sum_{p \in \Omega_{D_i}} \left\| \frac{r_p^2(p, \xi_{ji})}{\sigma^2_{r_p(p,\xi_{ji})}} \right\|_\delta$$

with

$$r_p(p, \xi_{ji}) \coloneqq I_i(p) - I_j(\omega(p, D_i(p), \xi_{ji}))$$

$$\sigma^2_{r_p(p,\xi_{ji})} \coloneqq 2\sigma_I^2 + \left( \frac{\partial r_p(p, \xi_{ji})}{\partial D_i(p)} \right)^2 V_i(p)$$

where $\|.\|$ is the Huber norm

$$\left\|r^2\right\|_\delta \coloneqq \begin{cases} \frac{r^2}{2\delta} if \ \|r\| \leq \delta \\ , \|r\| - \frac{\delta}{2} \ otherwise \end{cases}$$

applied to the normalized residual. The depth map estimation component uses tracked frames to either refine or replace the current key-frame. Depth is refined by filtering over many per-pixel, small-baseline stereo comparisons coupled with interleaved spatial regularization. If the camera has moved too far, a new key-frame is initialized by projecting points from existing, close-by key-frames into it. Each key-frame is scaled such that its mean inverse depth is one, which enables more small-baseline stereo comparisons. For every new key-frame added, the possibility of loop closure is checked by performing the reciprocal tracking check. The map optimization component is responsible for the updating depth map into global map, it detect loop closure and scale drift by estimating similarity transform (sim(3)) to close by existing key-frames.The global map is represented as a pose graph consisting of key-frames as vertices's with 3D similarity transforms as edges, elegantly incorporating changing scale of the environment and allowing to detect and correct accumulated drift. Each key-frame consists of a camera image, an inverse depth map and variance of the inverse depth.Edges between key-frames contain their relative alignment as simi-

larity transform, as well as corresponding covariance matrix. The map, consisting of a set of key-frames and tracked sim(3)-constraints, is continuously optimized in the background using pose graph optimization. The error function that is minimized is defined by (W defining the world frame)

$$E(\xi_{W1}....\xi_{Wn}) \coloneqq \sum_{(\xi_{ji},\Sigma_{ji})\in\varepsilon} (\xi_{ji} \circ \xi_{Wi}^{-1} \circ \xi_{Wj})\Sigma_{ji}^{-1}(\xi_{ji} \circ \xi_{Wi}^{-1} \circ \xi_{Wj})$$

# CHAPTER 2: RELATED WORK AND BACKGROUND INFORMATION

This article proposes fusion of SfM-estimated depths with depths captured from an RGBD image sensor. This section is dedicated to discussing the relevant aspects of the SfM algorithm and the sensed RGBD measurements necessary to explain the proposed fusion method. Specifically, this section reviews the theoretical details of the SfM algorithm, methods for processing depth images including computing depth im- ages for arbitrary camera poses, and details existing knowledge regarding the sensor measurement noise for RGBD depth measurements

## 2.1 Image alignment

The image alignment is the process of tranforming (wraping) one image with respect to another image, with a goal to minimizng the total differance between the intensites [**?**].

$$\sum_{x} [I(\omega(x)) - T(x)]^2 \tag{2.1}$$

In eq 1.1 the wrap function $\omega$ defroms image I along the image T in an atempt to minmize the differance between them. The wrap function which achive this goal is estimated incremental by perfroming gradiant descent.In order to achive this, we need to find the correspondace between images.

#### 2.1.0.1 Solving for Image Pixel Correspondences

There are generically two different approaches for finding corresponding observations of the same 3D surface location in multiple images referred to as *direct* and *indirect* [**?**]. In this discussion, we refer to the correspondence problem as a source-to-target matching problem. Let $\mathbf{I}_t(x, y)$ denote an image recorded at time $t$ and $\mathbf{I}_{t+\Delta t}(x, y)$ denote a subsequent image measured at time $t + \Delta t$. The correspondence

problem seeks to find a map that transforms pixels from the original $(x, y)$ coordinate field of $\mathbf{I}_t$ to new coordinate positions $(x', y')$ in $\mathbf{I}_{t+\Delta t}$ such that $\mathbf{I}_t(x, y)$ and $\mathbf{I}_{t+\Delta t}(x', y')$ correspond to images of the same 3D scene point.

### Indirect Methods

Indirect methods compute this mapping by detecting special $(x, y)$ locations referred to as features locations with a purpose-built feature detection algorithm, e.g., the Harris corner detector [**?**]. A description of the image patch in the vicinity of each detected $(x, y)$ location is computed using some feature descriptor algorithm, e.g., Lowe's SIFT descriptor [4]. Feature descriptors seek to provide a vector of values from the image patch data that is invariant to the image variations that occur during camera motion. These include but are not limited to the following effects: illumination variation, affine and/or projective invariance, photometric invariance (brightness constancy), and scale invariance. Popular feature descriptors often prioritize scale and affine invariance as their strengths. The invariance property allows for correspondences to be computed by finding the mapping from the feature descriptor set calculated from image $\mathbf{I}_t(x, y)$ to the feature descriptor set calculated from image $\mathbf{I}_{t+\Delta t}(x, y)$.

### Direct Methods

Direct methods on the other hand typically iteratively solve for a set of transformation parameters that best align a pair of images by the minimization of pixel-wise errors. An image warping function, $\omega(\mathbf{x})$, maps a pixel location, $\mathbf{x} = [x, y]^t$, in the original coordinate field to new coordinate positions, $\mathbf{x}'$, such that both locations correspond to images. A classical solution to this problem is given by the Lucas-Kanade-Tomassi (LKT) camera tracking algorithm [**?**].

## 2.2 Two unknows of the SfM

The SfM algorithm uses a time sequence of images from a moving camera to recover the 3D geometry of objects viewed by the camera. While this problem can be solved without a calibrated camera, reconstruction accuracy will adversely affected. This work assumes that the camera calibration [5] parameters are known. The SfM algorithm can be broken down into two key steps:

1. estimation of the camera pose, i.e., position and orientation, at the time each image was recorded,

2. estimation of the 3D structure of the scene.

As previously mentioned, typical SfM systems solve (1) by computing a map that associates pixels from the original $(x, y)$ coordinate field to new coordinate positions $(x', y')$ such that both locations correspond to images of the same 3D scene point and (2) via multi-view 3D surface reconstruction algorithm, e.g., bundle adjustment [6]. In the following sections we provide an overview of aspects of the SfM algorithm necessary for the development of the proposed RGBD-SfM depth fusion algorithm.

### 2.2.1 Camera Trjectory

#### Rigid Body Motion

The Camera trajectoryis a rigid body motion. We need an efficient model to represent and compute this rigid body motion. The camera position is represented by a 3D Vector in an Euclidean space, this camera position is chosen to represent origin of the world frame and specify the translation and rotation of the scene relative to this frame. The rigid body motion itself is composed of a rotation and translation.

Traditionally, rotation is represented by a $3 \times 3$ special orthogonal matrix called rotational matrix. Special Orthogonal matrix SO(3) are the matrix which satisfy $R^T R = RR^T = I$ and have a determinant of $+1$.

$$SO(3) = \{R \in \mathbb{R}^{3\times3} \mid R^T R = I, det(R) = +1\}$$

The rotation transformation of the coordinates $X_c$of a point p relative to frame C to its coordinates $X_w$ relative to frame W is

$$X_w = R_{wc} X_c$$

Also, because the rotational matrix are orthogonal, we have$R^{-1} = R^T$, on this line the inverse transformation of coordinates are

$$X_c = R_{wc}^{-1} X_w = R_{wc}^T X_w$$

The continuous rotation of a camera is described as a trajectory $R(t) : t \to SO(3)$ in the space $SO(3)$.When the starting time is not t = 0, the relative motion between time $t_2$ and time $t_1$ will be denoted as $R(t_2, t_1)$. The composition law of the rotation group implies

$$R(t_{2,}t_0) = R(t_{2,}t_1) \times R(t_{1,}t_0), \vee t_0 < t_1 < t_2 \in R$$

On the other hand the translation is represented by a $T \in R^3$,$1 \times 3$ vector which adds the translation values in each dimension.With this, the complete rigid body motion is represented by

$$X_w = R_{wc} X_c + T_{wc} \tag{2.2}$$

However, the above equation is not linear but affine. We may convert this to linear by using homogeneous coordinates, where we append 1 for $1 \times 3$ vector and make it a $1 \times 4$ vector,

$$\bar{X} = \begin{bmatrix} X \\ 1 \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{bmatrix} \in \mathbb{R}^4$$

With this new notation for point, we can rewrite the transformation from equation 6 as following

$$\bar{X}_w = \begin{bmatrix} X_w \\ 1 \end{bmatrix} = \begin{bmatrix} R_{wc} & T_{wc} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ 1 \end{bmatrix} = \bar{g}_{wc}\bar{X}_c \tag{2.3}$$

where the $4 \times 4$ matrix $\bar{g}_{wc} \in R^{4\times4}$is called the homogeneous representation of the rigid-body motion.

The set of all possible configurations of a rigid body can then be described by the space of rigid-body motions or special Euclidean transformations called SE(3)

$$SE(3) = \{\bar{g} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \mid R \in SO(3), T \in R^3\} \subset \mathbb{R}^{4\times4} \tag{2.4}$$

Finally, one other dimension we could consider is the scale, this can represented by a Similarity transformation $Sim(3)$, which is the composition of a rotation, translation and a uniform rescaling,

$$Sim(3) = \{\bar{g} = \begin{bmatrix} R & T \\ 0 & s^{-1} \end{bmatrix} \mid R \in SO(3), T \in R^3, s \in R\} \subset \mathbb{R}^{4\times4} \tag{2.5}$$

Exponential Map

The special orthogonal group in three dimensions can be represented by a $3 \times 3$ rotation matrix $R \in SO(3)$, which must satisfy the constraint $R^TR = I$, this implies that the $SO(3)$ transformations leaves the quantity $x^2 + y^2 + z^2$ invariant.

The group $SO(3)$ has 9 parameters, but the invariance of the length produces six independent conditions, leaving three free parameters, Hence, the dimension of the space of rotation matrices $SO(3)$ should be only three, and six parameters out of the nine are in fact redundant.We can use this to have better representation of Rigid body motion.

The $SO(3)$, $SE(3)$ and $SIM(3)$are categorized under the special group called the Lie group, which represents the smooth differentiable manifolds. Every Lie group has a tangent space at identity called the Lie algebra, which is a vector space used to study the infinitesimal transformations.

For rotation group $SO(3)$ its lie algebra $so(3)$ is represented by

$$so(3) \doteq \left\{\hat{\omega} \in \mathbb{R}^{3\times 3} \mid \omega \in \mathbb{R}^3\right\}$$

where $\hat{\omega}$is a skew symmetric matrix for the vector $\omega$

The map from the space $so(3)$ to $SO(3)$ is called the exponential map.

$$exp : so(3) \to SO(3); \hat{\omega} \mapsto e^{\hat{\omega}}$$

$$R(t) = e^{\hat{\omega}t} \tag{2.6}$$

The equation 2.6 represents a rotation around the axis $\omega \in \mathbb{R}^3$ by an angle of $t$ radians. And the inverse mapping is obtained by logarithm of SO(3)

$$log : SO(3) \to so(3); log(R) \mapsto \hat{\omega}$$

We can extend this to full rigid body motion which also involves the translation along with the rotation, for rotation group $SE(3)$ its lie algebra $se(3)$ is represented by .

$$se(3) \doteq \left\{ \hat{\xi} = \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix} \mid \hat{\omega} \in so(3), v \in \mathbb{R}^3 \right\} \subset \mathbb{R}^{4\times4}$$

with $v(t) = \dot{T}(t) - \hat{\omega}(t)T(t) \in \mathbb{R}^3$

Similarly, the exponential map from the space $se(3)$ to $SE(3)$is given by

$$exp : se(3) \to SE(3); \hat{\xi} \mapsto e^{\hat{\xi}}$$

and as before inverse to the exponential map is defined by logarithm

$$log : SE(3) \to se(3); log(g) \mapsto \hat{\xi}$$

[THIS IS TOO SIMILAR to source work elsewhere, bordering on plagiarism – REMOVE!, you should not be going into this much detail for the differential properties of rotations as this is not something you are comfortable with, write down the most important relationships, i.e., the differential of the rotation and why you need it for SfM, you will want to define sim3 transforms and how they are distinct from se3 transforms then leave this topic] [I wrote this, because I had done some study on exponential map between lie group and lie algebra and vis versa, I was hoping to keep this, but let me know your thoughts ] [exponential map of SIM(3) is way too complicated and I am working on it, I will incorporate it going forward. ]

## Visual Odometry

With this efficiant represenation for rigid body tranformation eq 2.5,the trajectory of camera is determined by the process of direct image alignment 2.1, i.e we can incrementaly find the relative pose $\xi \in SIM(3)$that minimize the photometric error between image pixels.

$$\widehat{\xi} = \min_{\xi} \sum_{\mathbf{x}} \left(\mathbf{I}(\omega(\mathbf{x}, \xi)) - \mathbf{T}(\mathbf{x})\right)^2 \tag{2.7}$$

In equation (2.7), the warp function, $\omega(\mathbf{x}, \xi)$, maps pixel locations, $\mathbf{x}$, in the template to pixel locations in the image $\mathbf{I}(\mathbf{x})$ using the warp transformation parameters $\xi$. For direct correspondence estimation, $\xi$ is a pose transformation of the viewing camera represented as a unknown 6x1 vector. We then seek the camera pose transformation parameters, $\widehat{\xi}$, that minimize equation which provides the camera pose change that best explain the differences in these two of the same scene. Given two images and the camera pose change between them, one can take the information in one image, and through the warp function and map these values into the viewpoint of the other image establishing a correspondence. In this case the theoretical difference between expected and observed values for the image pair is zero if the camera pose change is known exactly and sensor noise and other outside influences are ignored.

### 2.2.2 Depth Mapping

#### Camera model

The 2D image is formed by capturing the light energy (irradiance) for every pixel, this process can be mathematically represented by thin lens camera model, which describes the relationship between the three-dimensional coordinators to its projection onto the image plane. The thin lens model is represented by a optical axis and a perpendicular plane called the focal plane.The thin lens itself is characterized by its focal length and diameter, the focal length is the distance from optic center to where all the ray intersect the optic axis, while the point of intersection itself is called the focus of the lens. One of the important properties to consider is that the rays entering the lens through optic center are undeflected while the rest of the rays are. With this model the irradiance on the image plane is obtained by the integration of all the energy emitted from region of space contained in the cone determined by the

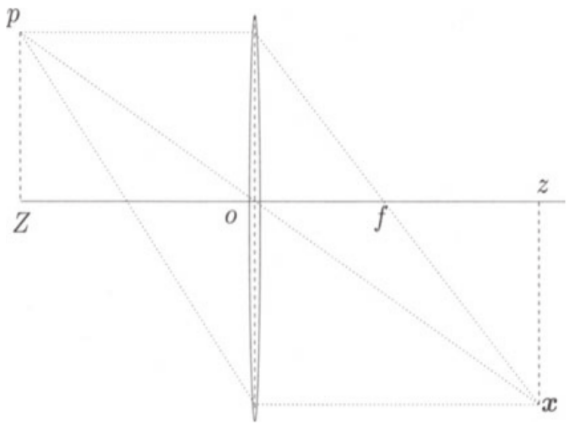


Figure 2.1: Thin lens camera model

geometry of the lens.

Using similar triangles, from above figure, we obtain the following fundamental equation of the thin lens

$$\frac{1}{Z} + \frac{1}{z} = \frac{1}{f}$$

For the simplification of calculation, we consider a Ideal camera model called the pinhole camera model, here the aperture of a thin lens is assumed to decreased to zero, all rays are forced to go through the optical center o, and therefore they remain undeflected.Consequently,as the aperture of the cone decreases to zero, the only points that contribute to the irradiance at the image point $x = [x, y]$ are on a line through the center 0 of the lens. If a point p has coordinates $X = [X, Y, Z]$ relative to a reference frame centered at the optical center 0, with its z-axis being the optical axis (of the lens), then it is immediate to see from similar triangles in Figure that the coordinates of p and its image x are related by the so-called ideal perspective projection.

$$x = -f\frac{X}{Z} \tag{2.8}$$

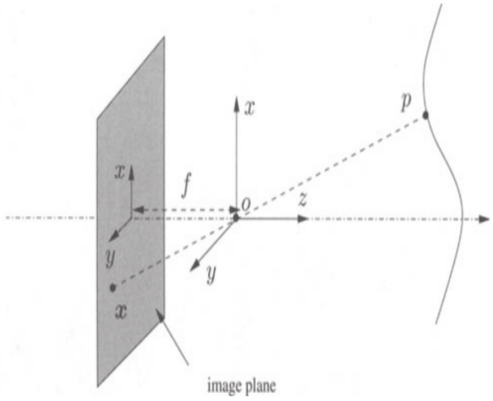


Figure 2.2: Pin-hole camera model

$$y = -f\frac{Y}{Z} \tag{2.9}$$

[write projection as a matrix transform similar to ROS][have it represented below]This mapping of 3D point to 2D is called projection and is represented by $\pi$

$$\pi : R^3 \rightarrow R^2$$

This is also written as $x = \pi(X)$.

The negative sign in the 2.8 and 2.9 makes the object appear upside down on the image plan, we can handle this by moving the image plane to front of optic center to $\{z = -f\}$ this will make $(x, y) \rightarrow (-x, -y)$.There for the equation 2 and 3 changes to

$$x = f\frac{X}{Z}$$

$$y = f\frac{Y}{Z}$$

This can be represented in matrix form as

$$x = \begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \end{bmatrix}$$

In homogeneous coordinates, this relationship can be modified as

$$Z \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

The above equation can be decomposed into

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

with

$$K_f = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \in R^{3\times3}, \Pi_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times4}$$

The matrix $\Pi_0$is a standard projection matrix.

With the rigid body transformation from 2.3, we can represent the overall geometric model for an ideal camera as below

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{2.10}$$

Where $\lambda$ is the unknown scale factor.

However, the above equation represents the ideal model where the retinal frame centered at the optical center with one axis aligned with the optical axis.But in practice, this does not true and the origin of the image coordinate frame typically in the upper-left corner of the image.we need to address this relationship between the retinal plane coordinate frame and the pixel array in our camera model. This can be represented by a special matrix as following

$$K_s = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3}$$

with these new parameters the we can represent the interstice parameters of camera as following

$$K = K_s K_f = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} fs_x & s_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$$

where

- $o_x$: x-coordinate of the principal point in pixels,
- $o_y$: y-coordinate of the principal point in pixels,
- $fs_x = \alpha_x$ : size of unit length in horizontal pixels,
- $fs_y = \alpha_y$ : size of unit length in vertical pixels,
- $\frac{\alpha_x}{\alpha_y}$: aspect ratio $\sigma$,
- $fs_\theta$: skew of the pixel, often close to zero.

Now, the ideal camera model 2.10can be updated as

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{2.11}$$

In the matrix notation,

$$\lambda x = K\Pi_0 g X_0 \tag{2.12}$$

To summarize, 2.12 represents the projection of three-dimensional coordinates $X_0$ by camera with orientation (extrinsic parameters)$g$ and intrinsic parameters K, onto two-dimensional coordinate $x$ with known scale $\lambda$

In addition to above linear distortion, if a camera has a wide field of view, there will be a significant distortion along radial directions called radial distortion. Such a distortion can be models by

$$x = x_d(1 + a_1 r^2 + a_2 r^4)$$

$$y = y_d(1 + a_1 r^2 + a_2 r^4)$$

where $(x_d, y_d)$ are the coordinates of the distorted points, $a_1, a_2$are the coefficients of radial distortion and $r$ is the radius of the radial distortion.

Epipolar geometry

Consider two images of the same point p from two camera position with relative pose $(R, T)$, where $R \in SO(3)$ is the relative orientation and $T \in \mathbb{R}^3$is the relative position,then if $X_1, X_2 \in \mathbb{R}^3$ are the 3-D coordinates of a point p relative to the two camera frames, by the rigid-body transformation we have

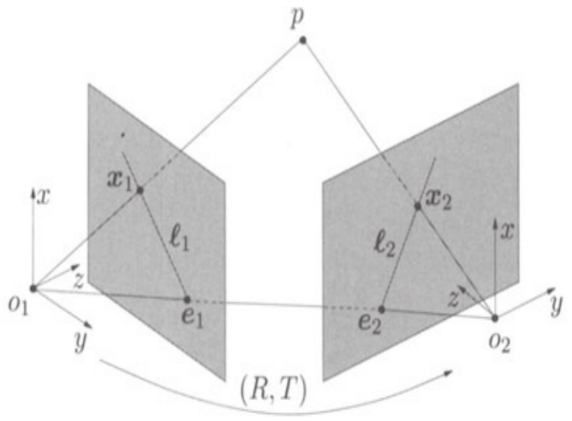


Figure 2.3: Epipolar geometry

$$X_2 = RX_1 + T$$

Now,let $x_1, x_2 \in \mathbb{R}^3$be the homogeneous coordinates of the projection of the same point p in the two image planes with respective unknown scales of $\lambda_1$and $\lambda_2$.

$$\lambda_2 x_2 = R\lambda_1 x_1 + T$$

By multiplying both the side by $\hat{T}$

$$\lambda_2 \hat{T} x_2 = \hat{T} R \lambda_1 x_1$$

By multiplying both the side by $x_2$

$$0 = x_2^T \hat{T} R \lambda_1 x_1 \tag{2.13}$$

This is the epipolar constraint and the matrix $E = \hat{T}R$ is called the essential matrix.It encodes the relative pose between the two cameras. Geometrically, it impose that the The vector connecting the first camera center $o_1$ and the point p, the vector connecting $o_2$ and p, and the vector connecting the two optical centers $o_1$ and $o_2$

clearly form a triangle. Therefore, the three vectors lie on the same plane.Hence the their triple product which measures the volume of the parallelepiped is zero.

#### 2.2.2.1 The eight-point linear algorithm

With epipolar constrain between two images, we should be able to retrieve the relative pose of the cameras. The eight-point linear algorithm is a simple closed-form algorithm, it consists of two steps: First a matrix E is recovered from a number of epipolar constraints; then relative translation and orientation are extracted from E.

The entries of E are denoted by

$$E = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \in \mathbb{R}^{3\times 3}$$

The matrix E is reshaped into vector $E \in \mathbb{R}^9$

$$E = [e_{11}, e_{21}, e_{31}, e_{12}, e_{22}, e_{32}, e_{13}, e_{23}, e_{33}]^T$$

and for $x_1 = [x_{1,}y_1, z_1]^T \in \mathbb{R}^3$and $x_2 = [x_{2,}y_2, z_2]^T \in \mathbb{R}^3$ define

$$a = [x_1x_2, x_1y_2, x_1z_2, y_1x_2, y_1y_2, y_1z_2, z_1x_2, z_1y_2, z_1z_2] \in \mathbb{R}^9$$

With these new notations, we can rewrite the 2.13 as below

$$a^T E = 0$$

this representation emphasizes the linear dependence of the epipolar constraint on the elements of the essential matrix.

Now,with a set of corresponding image points $(x_1^j, x_2^j), j = 1, 2, ...., n$ we can define a matrix $\chi \in \mathbb{R}^{n\times 9}$ associated with these measurements to be

$$\chi = \left[a^1, a^2, ..., a^n\right]^T$$

In the absence of noise, the vector E satisfies

$$\chi E = 0$$

In order to obtain the unique solution, the rank of the matrix $\chi \in \mathbb{R}^{n\times 9}$ needs to be exactly eight. This should be the case when we have $n \geq 8$ "ideal" corresponding points, hence the name The eight-point linear algorithm.

Because of errors in correspondences, we try to find the E that minimizes the least-squares error function $\|\chi E\|^2$.We do this by choosing eigenvector of $\chi^T\chi$ that corresponds to its smallest eigenvalue.

Once we have E, we need to extract the pose ($R \in SO(3)$ and $T \in \mathbb{R}^3$) from E, we know that [**?**]a nonzero matrix $E \in \mathbb{R}^3$ is an essential matrix if and only if E has a singular value decomposition $(SVD)E = U\Sigma V^T$with

$$\Sigma = diag\left\{\sigma, \sigma, 0\right\}$$

for some $\sigma > 0$ and $U, V, \in SO(3)$

With this we can obtain two relative pose

$$(\hat{T}_1, R_1) = (UR_Z(+\frac{\pi}{2})\Sigma U^T, UR_Z^T(+\frac{\pi}{2})V^T)$$

$$(\hat{T}_2, R_2) = (UR_Z(-\frac{\pi}{2})\Sigma U^T, UR_Z^T(-\frac{\pi}{2})V^T)$$

Among these one with that gives the meaningful (positive) depth are selected as the valid pose.

With $R_Z(\pm\frac{\pi}{2}) = \begin{bmatrix} 0 & \pm 1 & 0 \\ \pm 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Structure Reconstruction

One remaining thing to find is the position of points in three-dimension by recovering their depths relative to each camera frame.With the estimated pose (Translation is T is defined up to the scale $\gamma$) and point correspondence, we have

$$\lambda_2^j x_2^j = \lambda_1^j R x_1^j + \gamma T$$

for $j = 1, 2, 3..., n$

where, $\lambda_1$and $\lambda_2$are depths with respect to the first and second camera frames, respectively. One of the depths is redundant, if $\lambda_1$ is known, we can estimate $\lambda_2$ as a function of $(R, T)$.Hence we can eliminate, say, $\lambda_2$ from the above equation by multiplying both sides by $\hat{x_2}$

$$0 = \lambda_1^j x_2^j R x_1^j + \gamma x_2^j T$$

This is represented as linear equations

$$M^j \bar{\lambda^j} = \left[\hat{x_2^j} R x_1^j, \hat{x_2^j} T\right] \begin{bmatrix} \lambda_1^j \\ \gamma \end{bmatrix} = 0$$

where $M^j = \left[\hat{x_2^j} R x_1^j, \hat{x_2^j} T\right] \in \mathbb{R}^{3\times 2}$and $\bar{\lambda^j} = \begin{bmatrix} \lambda_1^j \\ \gamma \end{bmatrix} \in \mathbb{R}^2$ for $j = 1, 2, 3..., n$

since all n equations above share the same $\gamma$; we define a vector $\vec{\lambda} = [\lambda_1^1, \lambda_1^2, ..., \lambda_1^2, \gamma]$ and a matrix $M \in \mathbb{R}^{3n\times(n+1)}$as

$$M = \begin{bmatrix} \hat{x}_2^1 R x_1^1 & 0 & 0 & 0 & 0 & \hat{x}_2^1 T \\ 0 & \hat{x}_2^2 R x_1^2 & 0 & 0 & 0 & \hat{x}_2^2 T \\ 0 & 0 & \ddots & 0 & 0 & \vdots \\ 0 & 0 & 0 & \hat{x}_2^{n-1} R x_1^{n-1} & 0 & \hat{x}_2^{n-1} T \\ 0 & 0 & 0 & 0 & \hat{x}_2^n R x_1^n & \hat{x}_2^n T \end{bmatrix}$$

Then the equation

$$M \overrightarrow{\lambda} = 0$$

determines all the unknown depths up to a single universal scale. The linear least-squares estimate of $\overrightarrow{\lambda}$ is simply the eigenvector of $M^T M$s that corresponds to its smallest eigenvalue.

### Solving for Scene Geometry

There are generically two different approaches for 3D reconstruction referred to as *sparse* and *dense*. Sparse methods reconstruct the 3D scene geometry only for a select subset of the entire image data [1]. This subset is often corner locations or locations marked by some type of feature extraction, e.g., SIFT or SURF. [4, 7] This results in a sparse description of the 3D scene in terms of a point cloud. In contrast, dense methods [**?**] reconstruct as many 3D geometric locations as possible and seek to provide a complete description of the 3D scene.

Sparse reconstructions often benefit from having a lower computational cost but provide few 3D measurements. Dense reconstructions have higher computational cost but provide a much more complete description of the 3D scene. Dense reconstruction techniques have seen much recent interest, although a highly accurate, dense, and real-time SfM approach has remained elusive.

A third class of algorithms, referred to as *semi-dense* algorithms [**?**], seeks to strike

a compromise between the sparse and dense methods. The reconstruction techniques used are most similar to dense methods, however, only a subset of all image pixels are reconstructed. These approaches leverage the high accuracy of dense reconstruction techniques, but are sparse enough to allow for real-time operation.

Here, reconstruction is limited to those pixels which possess high intensity gradient values. These regions often correspond to scene geometries such as edges, corners, and curves and to other areas of the scene that are highly textured. The thought here is that regions of the image that possess large changes in intensity convey more information than regions that possess less, thus semi-dense reconstructions provide a compressed version of the total scene

RGBD Measurement Noise

The proposed fusion algorithm relies on experimental studies of accuracy and noise for RGBD sensor measurement, e.g., the Kinect sensor. Research in [**?**] shows that a Gaussian noise model provides a good fit to observed measurement errors on planar targets where the distribution parameters are mean 0 and standard deviation$\sigma_Z = \frac{m}{2f_x b}Z^2$ for depth measurements where $\frac{m}{f_x b} = -2.85e^{-3}$ is the linearized slope for the normalized disparity empirically found in [**?**]. Since 3D the coordinates for $(X, Y)$ are a function of both the pixel location and the depth, their distributions are also known as shown below:

$$\begin{aligned} \sigma_X &= \tfrac{x_{im}-c_x+\delta_x}{f_x}\sigma_Z = \tfrac{x_{im}-c_x+\delta_x}{f_x}(1.425e^{-3})Z^2 \\ \sigma_Y &= \tfrac{y_{im}-c_y+\delta_y}{f_y}\sigma_Z = \tfrac{y_{im}-c_y+\delta_y}{f_y}(1.425e^{-3})Z^2 \\ \sigma_Z &= \tfrac{m}{f_x b}Z^2\sigma_{d'} = (1.425e^{-3})Z^2 \end{aligned} \tag{2.14}$$

These equations indicate that 3D coordinate measurement uncertainty increases as a quadratic function of the depth for all three coordinate values. However, the quadratic coefficient for the $(X, Y)$ coordinate standard deviation is at most half that in the depth direction, i.e., $(\sigma_X, \sigma_Y) \approx 0.5\sigma_Z$ at the image periphery where $\frac{x-c_x}{f} \approx 0.5$,

and this value is significantly smaller for pixels close to the optical axis.

## 2.3 Non linear Optimization

In practice, because of the noise in image correspondence and other errors we cannot measure the actual coordinates but only their noisy versions, say

$$\widetilde{x}_1^j = x_1^j + \omega_1^j$$

$$\widetilde{x}_2^j = x_2^j + \omega_2^j$$

where $x_1^j$and $x_2^j$are the ideal image coordinates and $\omega_1^j = \left[\omega_{11}^j, \omega_{12}^j, 0\right]^T$and $\omega_2^j = \left[\omega_{21}^j, \omega_{22}^j, 0\right]^T$are localization errors (called residuals) in the correspondence.Therefore, we need a way to optimize the parameters $(x, R, T)$ that minimize this errors.

One of the minimalistic approach to optimality is to minimize the squared 2-norm of residuals, if we choose the first camera frame as the reference

$$\phi(x, R, T, \lambda) = \sum_{j=1}^{n} \left\|\omega_1^j\right\| + \left\|\omega_2^j\right\|^2 = \sum_{j=1}^{n} \left\|\widetilde{x}_1^j - x_1^j\right\|^2 + \left\|\widetilde{x}_2^j - \pi(R\lambda_1^j x_1^j + T)\right\|^2$$

The above error is often called the "re-projection error", since $x_1^j$ and $x_2^j$are the recovered 3-D points projected back onto the image planes. This process of minimizing the above expression for the unknowns $(R, T, x_1, \lambda)$ is known as bundle adjustment[6].

One of the many ways to minimize this squared error is the Gauss-Newton method. Gauss-Newton is a iterative method for finding the value of the variables which minimizes the sum of squares,it starts with the initial guess and this method does not need the second derivatives (Hessian matrix) of the of function, which is often expensive and sometimes not possible to compute, instead the Hessian is approximated with the Jacobian matrix of the function.

For the least-square function of form

$$\min_x \sum_i r_i(x)^2$$

Gauss-Newton method iteratively solves

$$x_{t+1} = x_t - H^{-1}g$$

with gradient g

$$g_j = 2\sum_i r_i \frac{\partial r_i}{\partial x_i}$$

and the Hessian H

$$H_{jk} = 2\sum_i r_i \left(\frac{\partial r_i}{\partial x_j}\frac{\partial r_i}{\partial x_k} + r_i \frac{\partial^2 r_i}{\partial x_i \partial x_k}\right)$$

by dropping the second order term we can approximate the Hessian matrix with Jacobian matrix

$$H_{jk} = 2\sum_i J_{ij} J_{ik}$$

with

$$J_{ij} = \frac{\partial r_i}{\partial x_j}$$

The modification to Gauss-Newton method is The Levenberg-Marquardt algorithm,

$$x_{t+1} = x_t - (H + \lambda I)^{-1} g$$

this modification is a hybrid between the Newton method ($\lambda = 0$) and a gradient descent step size $1/\lambda$ for $\lambda \longrightarrow \infty$

# CHAPTER 3: LSD SLAM

We are using LSD-SLAM [**?**] as our SfM implementation, it is a *semi-dense*, *direct* method which optimizes the geometry directly on the image intensities. LSD-SLAM provides as output a reconstruction of the observed 3D environment as a pose-graph of specially designated RGB image keyframes with associated semi-dense depth images. Direct correspondences are found between every RGB frame and each RGB keyframe to estimate the keyframe-to-RGB-frame relative camera pose. This is achieved by Levenberg-Marquardt optimization of the photometric error between the RGB image pair.

$$E(\xi) = \sum_i (I_{ref}(P_i) - I(\omega(p_i, D_{ref}(p_i), \xi)))^2 \tag{3.1}$$

Equation (3.1) shows the specific image alignment object function and formalizes the form for equation (2.1). Here, the warp function relies on the current estimate of depth $D_{ref}$ to determine the relative pose $\xi \in sim(3)$. The reference depth $D_{ref}$ is the depth associated with current key frame, these depth values could be initialized either with random values or with the depth measurements from a RGBD sensor to initiate the process. For every new frame tracked, the depths associated with the keyframe are continuously refined by performing the adaptive baseline stereo 3D reconstruction [**?**] between new tracked frames and the stack of previously tracked frames. When there is drastic pose change between the tracked frame and keyframe, the current tracked frame is promoted as a keyframe and depth map from previous keyframe is propagated to the new keyframe and regularized. Concurrently, all the keyframes are added as a nodes to a pose graph that stores the relative pose between the keyframes

as edges/constraints [8]. The pose graph stores the global trajectory of the camera in the 3D scene and an optimization algorithm processes the pose graph to improve the camera pose estimates as new image correspondences are found by image matching and loop closures.

The depth computation process involves finding the epiploar lines between new tracking frame and referance frames, along which the disparity for the pixle is determined. This process have two error sources, the error on disparity itself called the geometric disparity error and the error which encodes intensity differances called the photomatirc disparity error, The former error accounts for the magnitude of the image gradiant along the epliplor line , while the later one on the angle between the image gradient and the epipolar line.

Point Cloud Reconstruction

Measured 3D $(X, Y, Z)$ positions of sensed surfaces can be directly computed from the intrinsic camera parameters and depth image values. Here, the $Z$ coordinate is directly taken as the depth value and the 3D $(X, Y)$ coordinates are computed using the pinhole camera model. In a typical pinhole camera model 3D $(X, Y, Z)$ points are projected to $(x, y)$ image locations [9], e.g., for the image columns the $x$ image coordinate is $x = f_x \frac{X}{Z} + c_x - \delta_x$. However, for a depth image, this equation is reorganized to "back-project" the depth into the 3D scene and recover the 3D $(X, Y)$ coordinates as shown by equation (3.2)

$$\begin{aligned} X &= (x + \delta_x - c_x)Z/f_x \\ Y &= (y + \delta_y - c_y)Z/f_y \\ Z &= Z \end{aligned} \tag{3.2}$$

where $Z$ denotes the sensed depth at image position $(x, y)$, $(f_x, f_y)$ denotes the camera focal length (in pixels), $(c_x, c_x)$ denotes the pixel coordinate of the image center, i.e., the principal point, and $(\delta_x, \delta_y)$ denote adjustments of the projected pixel coordinate

to correct for camera lens distortion.

### Point Cloud Re-Projection

Depth images can be simulated for camera sensor in arbitrary poses by "re-projection." For discussion, assume that depth image $\mathbf{d}(x, y)$ has been recorded in the "standard" camera/optical coordinate system where the origin corresponds to the camera focal point, the $z$-axis corresponds to the depth/optical axis extending out into the viewed scene, the $x$-axis points towards the right and spans the image columns and the $y$-axis points downward and spans the image rows.

Let $\mathbf{R}$ denote the 3D rotation that rotates the coordinate axes of the standard coordinate system to align with the same axes of a second camera having arbitrary pose. Similarly, Let $\mathbf{t}$ denote the 3D translation vector describing the position of the focal point of a second camera having arbitrary pose. Using this notation, the re-projection algorithm consists of the following three steps:

1. Back-project $\mathbf{d}(x, y)$ to create an $(X, Y, Z)$ point cloud (as described in § 3),

2. Transform, i.e., rotate and translate, each point $\mathbf{p}_i = [X, Y, Z]^t$ in the point cloud to generate a new point $\mathbf{p}_i^{'} = [X^{'}, Y^{'}, Z^{'}]^t$ that lies in a standard optical coordinate system centered on the second camera's focal point and having orientation that aligns with corresponding $x, y, z$-axes using equation (3.3),

$$\mathbf{p}_i^{'} = \mathbf{R}(\mathbf{p}_i - \mathbf{t}) \tag{3.3}$$

3. Re-project the $(X, Y, Z)$ point cloud using the pinhole camera equations to compute the new depth image $\mathbf{d}^{'}(x, y) = Z$ using equation (3.4).

$$\begin{aligned} x &= f_x(\tfrac{X^{'}}{Z^{'}}) - \delta_x + c_x \\ y &= f_y(\tfrac{Y^{'}}{Z^{'}}) - \delta_y + c_y \\ Z &= Z^{'} \end{aligned} \tag{3.4}$$

Typically, the re-projected point cloud measurements fall at non-integer locations in the new depth image and the values of $\mathbf{d}'(x, y)$ must then be interpolated via bilinear interpolation or some other interpolation scheme (nearest neighbor).

# CHAPTER 4: Methodology

The proposed fusion approach applies the semi-dense monocular reconstruction approach referred to as Large Scale Direct (LSD) SLAM [**?**]. The LSD-SLAM algorithm solves the SfM problem using a *direct* method to compute pixel correspondences and a *semi-dense* method for 3D reconstruction. We select this approach as it does not require or impose any prior knowledge about the scene structure as required by *dense* reconstruction methods and it gives more 3D estimates than *sparse* approaches while having similar computational cost.

The LSD-SLAM SfM algorithm consists of the following three components:

1. A tracking component that estimates the pose of the camera
2. A depth map estimation component that estimates semi-dense depth images for keyframes
3. A map optimization component that seeks to create a 3D map of the environment that is self-consistent.

This work utilizes the first two components to explore fusion of RGBD depth images with SfM depth images. For this work, the map optimization component (3) is not used. Figure 4.1 depicts an overview of the proposed depth fusion algorithm.

### 4.0.1 Time and Spatial Sampling Issues

As mentioned previously, RGBD sensors measure depth at a rate of 30 frames per second (fps) and LSD-SLAM computes depth images only for *keyframes* which is a sparse subset of the measured RGB frames. Further, keyframes are not generated uniformly in time but created when the SfM algorithm detects criteria required to

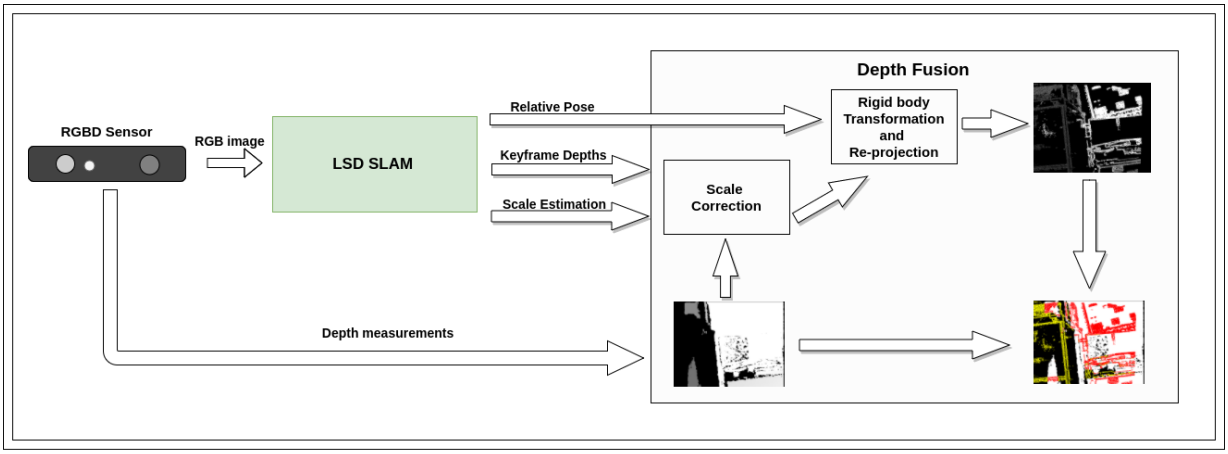


Figure 4.1: An overview of the proposed depth fusion algorithm.

create a new keyframe. This condition is triggered when the current camera pose is too far from the most recent keyframe camera pose and when the current frame tracking result is "good" in the sense that the image warping correspondence objective function suggests an accurate or low-error result. As a result, SfM-estimated depths exist only for those RGB images designated as keyframes.

Further, the spatial distribution of SfM-estimated depths within SfM keyframes are localized to only those pixels having "good" 3D reconstruction characteristics. In this sense, the quality of the depth estimate depends on accurately matching pixels along epipolar lines inscribed in the image. The matching performance here is best when there is a significant change in the image intensities along the epipolar line. Hence, 3D depth reconstruction is limited to those pixels that lie at sharp intensity changes, i.e., "edge" pixels, and further limited to those "edge" pixels that lie on edges that are roughly perpendicular to the direction of the epipolar line (see [**?**] for details).

The LSD-SLAM algorithm estimates depth at "good" pixel positions as a 1-dimensional Gaussian distribution specified as a mean image $\mu_{SfM}(x, y)$, i.e., the estimated depth image, and a variance image $\sigma^2_{SfM}(x, y)$ such that the RGB keyframe pixel at location $\mathbf{I}(x, y)$ is estimated to have depth $\mu_{SfM}(x, y)$ with uncertainties given by $\sigma^2_{SfM}(x, y)$. In this sense, the keyframe image $\mathbf{I}(x, y)$ augmented with the estimated depth image

$\mu_{SfM}(x, y)$ is analogous in format to sensed RGBD image data. Yet, the uncertainties for the image $\mu_{SfM}(x, y)$ are given by the image $\sigma^2_{SfM}(x, y)$ rather than the experimentally validated uncertainties discussed in § 2.2.2.1.

### 4.0.2 Image Registration Issues

Fusing depth measurements requires knowledge of the correspondence between the depth measurements generated from the RGBD sensor and the SfM algorithm. For SfM keyframes this correspondence is trivial due to the fact that RGBD sensors support hardware registration. Hardware registration co-locates the RGBD depth image measurements, $\mathbf{d}_{rgbd}(x, y)$, and RGB appearance values, $\mathbf{I}(x, y)$. Hence, for hardware-registered RGBD depth images, $\mathbf{d}_{rgbd}(x, y)$ is the measured depth of the surface having RGB pixel $\mathbf{I}(x, y)$. Similarly, SfM-estimated depths for an RGB keyframe, $\mu d_{SfM}(x, y)$, are the depths for the surface having RGB pixel $\mathbf{I}(x, y)$. Hence fusion is accomplished by fusing the measurements at corresponding $(x, y)$ locations in the RGBD depth image, $\mathbf{d}_{rgbd}(x, y)$, and the SfM depth image, $\mu\mathbf{d}_{SfM}(x, y)$.

Depth correspondences for RGB images that are not SfM keyframes must be computed from one or more SfM keyframe depth images. This article uses the most recent, i.e., closest-in-time, keyframe to generate co-registered SfM depth images for arbitrary RGB images. To do so, the depth image from the most recent keyframe, $\mathbf{d}_{SfM}(x, y)$, is "back-projected" to create a 3D point cloud of SfM measurements. Using the estimated keyframe-to-camera pose change, the 3D measurements are then re-projected into the RGB camera image plane using the 3D projection equations for the camera provided via camera calibration. The resulting depth image, $\widetilde{\mathbf{d}}_{SfM}(x, y)$ is then co-registered with the RGBD depth image $\mathbf{d}_{rgbd}(x, y)$ and RGB image $\mathbf{I}(x, y)$.

However, due to the noise in depth computation and consequent trajectory estimation, the depth regestration is not alway correct, this should be adressed as early as possible to avoid the drift in their computations. We try to achive better depth registration by estimating the transformation error between the point clouds of sen-

sor depth measurments and LSD depth estimation. Since the LSD estimations are not to the scale, we also need to estmate the scale for better regestation of depths. The Umeyama's Least-Square estimation [10] shows us how to estimate these $Sim(3)$ transformation parameters efficiantly. In his work, Umeyama shows how to estimates the rotation, translation and scale between two point clouds with known correspondance. In order to avoid errors, we use only the valid depths in both the point cloud for estimation.By using $\mathbf{pc}_{rgbd}$and $\mathbf{pc}_{SfM}$to represent the valid point clouds of sensor measurments and LSD estimation respectivly, we comput the mean, variances and covariance of both them as following.

$$\mu_{rgbd} = \frac{\sum \mathbf{pc}_{rgbd}}{n}$$

$$\mu_{SfM} = \frac{\sum \boldsymbol{pc}_{SfM}}{n}$$

$$\sigma^2_{rgbd} = \frac{\sum \left\| \boldsymbol{pc}_{rgbd} - \mu_{rgbd} \right\|^2}{n}$$

$$\sigma^2_{SfM} = \frac{\sum \left\| \boldsymbol{pc}_{SfM} - \mu_{SfM} \right\|^2}{n}$$

$$\Sigma_{SfM,rgbd} = \frac{\sum (\boldsymbol{pc}_{rgbd} - \mu_{rgbd})(\boldsymbol{pc}_{SfM} - \mu_{SfM})^T}{n}$$

With singular value decomposition of $\Sigma_{SfM,rgbd}$ as $UDV^T$,the optimum rotation matrix $R$ which achieves the minimum error between point clouds is given by

$$R = USV^T$$

$$S = I \begin{cases} I & det(\Sigma_{SfM,rgbd}) \geq 0 \\ diag(1,1,-1) & det(\Sigma_{SfM,rgbd}) < 0 \end{cases}$$

The scale adjustment $c$ is computed by

$$c = \frac{1}{\sigma^2_{SfM}} trace(DS)$$

With scale and rotation established, the translation parameters are determined as

$$t = \mu_{rgbd} - cR\mu_{SfM}$$

with the $Sim(3)$ parameters we trasanform LSD point clouds before projecting them to find $\widetilde{\mathbf{d}}_{SfM}(x, y)$

$$\boldsymbol{pc}_{\tilde{\boldsymbol{SfM}}} = cR\boldsymbol{pc}_{SfM} + t$$

Using these techniques co-registered SfM depth images can be computed for a general RGBD image. When RGBD images correspond to SfM keyframes the registration is "automatic," i.e., no computation is necessary. In all other cases, a co-registered SfM depth image must be computed by reconstructing a 3D point cloud from a keyframe and then projecting the point cloud into the target RGB camera image using the estimated camera calibration and keyframe-to-camera-frame relative pose parameters.

### 4.0.3 Resolving the Unknown SfM Scale

The methods described in previous sections detail how co-registered SfM depth measurements are computed for every sensed RGBD frame. However, as discussed in previously, SfM depth images intrinsically have an unknown scale, $\alpha$, which reflects the fact that the solution for the scene structure is not geometrically unique, i.e., the

same scene structure can be observed at a infinite number of distinct scales. Therefore fusion requires the scale of the SfM depth image to fit the scale of the real-world scene measured by the RGBD camera.

Given that the depth measurements for the RGBD depth image are co-registered with the SfM estimated depth image the scale parameter can be directly estimated by minimizing the sum of the squared depth errors [11] between the SfM depth image and the RGBD depth image. Let $V$ denote the set of $(x, y)$ positions that have valid depth measurements for "standard" fusion as described in 4.0.4. Equation (4.1) shows the error function used to compute the unknown scale value and equation (4.2) shows the solution $\widehat{\alpha}$ that minimizes this error.

$$e(\alpha) = \sum_{(x,y)\epsilon V} \|\mathbf{d}_{rgbd}(x, y) - \alpha \mathbf{d}_{SfM}(x, y)\|^2 \tag{4.1}$$

$$\widehat{\alpha} = \sum_{(x,y)\epsilon V} \frac{\mathbf{d}_{rgbd}(x, y)}{\mathbf{d}_{SfM}(x, y)} \tag{4.2}$$

### 4.0.4 RGBD and SfM Depth Fusion

For fusing measurements we consider the structured-light measurement of the RGBD sensor to generate a distribution for the unknown true depth of the scene surfaces at each RGBD $(x, y)$ pixel in the depth image. These measurements are considered to be independent and identically distributed to the measurements of the true unknown depth of the scene surfaces from the registered SfM estimated depths. With these assumptions, solving the depth fusion problem is equivalent to estimating the posterior distribution of the true scene depth at each $(x, y)$ position given the distributions for the RGBD and SfM depth values.

Fortunately, previous sections show that Gaussian models are appropriate distributions for both the RGBD and SfM depth values and the parameters of these models are either known (see § 2.2.2.1) or estimated continuously (see § 4.0.1). When both

distributions are Gaussian, the posterior distribution can be found analytically and is a well-known result used in pattern recognition and other prediction frameworks, e.g., the Kalman filter as discussed in [12]. Specifically, let the Gaussian noise for RGBD depth at position $(x, y)$ be represented as $\mathcal{N}(\mathbf{d}_{rgbd}, \sigma^2_{rgbd})$ and the Gaussian noise for the co-registered SfM depth image at position $(x, y)$ be $\mathcal{N}(\mu_{SfM}, \sigma^2_{SfM})$. The posterior distribution on the unknown true depth at position $(x, y)$ is also Gaussian and let $\mathcal{N}(\mu_{fused}, \sigma^2_{fused})$ denote the mean and variance parameters of this distribution. Equations (4.3) and (4.4) provide optimal estimates of the mean and variance of the fused depth at position $(x, y)$. The best estimate of the fused depth is given by the highest probability value in the posterior distribution which is the mean fused image, $\mu_{fused}(x, y)$.

$$\mu_{fused} = \frac{\mathbf{d}_{rgbd}\sigma^2_{SfM} + \mu_{SfM}\sigma^2_{rgbd}}{\sigma^2_{SfM} + \sigma^2_{rgbd}} \tag{4.3}$$

$$\sigma^2_{fused} = \frac{\sigma^2_{rgbd}\sigma^2_{SfM}}{\sigma^2_{rgbd} + \sigma^2_{SfM}} \tag{4.4}$$

# CHAPTER 5: Results

We conducted experiments with different setups to test and analyze the depth fusion algorithm. Experiments recorded RGBD image sequenced from ORBBEC Astra RGBD sensors and applied LSD-SLAM to their image streams using their factory-provided intrinsic camera calibration parameters. Each experiment included approximately 60 seconds of RGBD image data at the rate of 30 fps. Experimentally recorded RGB images were processed offline by the LSD-SLAM algorithm to generate SfM depth images. Experiments initialize the LSD-SLAM algorithm with the first recorded depth image from the RGBD sensor to facilitate the initial scale approximation. The output from the LSD-SLAM algorithm consisting of the relative pose for every tracked frame and the depth map for each keyframe was then captured to disk. The fusion algorithm was then run offline on the recorded RGBD image stream and LSD-SLAM output files to generate the results shown in this section.

Experiment 1 depicts a indoor office scene at the university. This scene includes specular and dark surface structures at close range that are not measured by the RGBD sensor. Yet, the SfM algorithm estimates depths at a number of locations (on the podium) where ther are significant intensity changes. These additional depths are

Table 5.1: Ratios of depth measurements to total depths

| Experiment | RGBD-only depths (%) | SfM-only depths (%) | Fused depths (%) |
|---|---|---|---|
| 1 | 67.7 | 6.7 | 25.5 |
| 2 | 55.5 | 10.0 | 34.3 |
| 3 | 68.1 | 15.1 | 16.6 |
| 4 | 51.0 | 25.7 | 23.1 |
| 5 | 68.3 | 8.8 | 22.8 |
| Average | 62.12 | 13.26 | 24.46 |

(a) (b) (c) (d) (e)

Figure 5.1: Results for three experiments are shown. Images shown are organized into separate columns. Column (a) shows a grayscale image of the scene (b) shows the sensed RGBD depth image (c) shows the SfM-estimated depth image (d) shows the fused image and (e) shows the standard deviation for fused depths (in $m$.). The fused image has been color-coded as follows: (white) denotes depth locations sensed only by the RGBD sensor, (yellow) denotes depth locations only sensed via SfM, (red) denotes fused (RGBD+SfM) depth locations and (black) denotes depth locations without RGBD or SfM measurements.

evident in the fused results, which includes SfM-only depth measurements in regions in the vicinity of image edges. The experiment demonstrates that depth fusion can improve depth images by obtaining depths from surfaces not measurable by the RGBD sensor.

Experiment 2 depicts a hallway in the UNCC EPIC building that includes both specular surfaces and high intensity illumination from overhead lights. The experiment is a second example showing that depth fusion bolsters over all depth measurement performance by providing the depths when RGBD sensors fail. The SfM takes advantage of the patterns on the surface and estimates the depths irrespective of the nature of its reflectance properties and color.

Experiment 3 depicts the UNCC faculty conference hall. Here a number of scene structures lie beyond the measurement range of the RGBD sensor. Yet, the SfM algorithm is able to estimate the depth of these scene structures (albeit at high variance) providing depths that would otherwise not be possible.

Experiment 4 depicts indoor to outdoor transition. By initializing the LSD SLAM indoor, we get the scale estimated. We can see that in outdoor, when the sensor measurements fails because of the infrared ray flooding, depth estimates from LSD SLAM are used. Often because of IR flooding the sensor measurements gets corrupted, in those scenarios we can consider replacing entire corrupted depth sensor measurements by LSD depth estimation instead of fusion for better results.

Experiment 5 depicts indoor with well lit conditions, even though the sensors are supposed work completely fine in this scenario, there are rare chances of the them failing at steep surfaces. During such failures, depth fusion comes handy.

Table 5.1 quantifies the amount of additional depth information provided via RGBD-SfM depth image fusion. When we average over the five experiments discussed approximately 62% of the fused depths originate from the RGBD sensor alone, approximately 13% original from the SfM algorithm alone, and approximately 24% of the

fused depths results from RGBD-SfM fusion. The addition of 13% of novel depth data is a significant contribution. Further, fused depths account for roughly 24% of depth image data and the measurement error for all of these measurements will be reduced by the fusion. Variance reduction will be greatest for low-variance SfM depth estimates which are typically in textured scene locations close to the camera. Yet, we note that by inspection of equation (4.4), it is theoretically impossible for the variance of any fusion result to increase. Another obvious observation from the Table 5.1 is that the SfM has a largest contribution in Experiment 4 (outdoor scenario) since sensor fails outdoors and it has least contribution in Experiment 5 (well lit indoor) where sensor is at its best.

REFERENCES


[1] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, (Nara, Japan), November 2007.

[2] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3d," *ACM Trans. Graph.*, vol. 25, pp. 835–846, July 2006.

[3] K. Wilson and N. Snavely, *Robust Global Translations with 1DSfM*, pp. 61–75. Cham: Springer International Publishing, 2014.

[4] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, Nov 2004.

[5] Y. M. Wang, Y. Li, and J. B. Zheng, "A camera calibration technique based on opencv," in *The 3rd International Conference on Information Sciences and Interaction Sciences*, pp. 403–406, June 2010.

[6] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment - a modern synthesis," in *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV '99, (London, UK, UK), pp. 298–372, Springer-Verlag, 2000.

[7] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Underst.*, vol. 110, pp. 346–359, June 2008.

[8] R. KÃŒmmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3607–3613, May 2011.

[9] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2003.

[10] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, pp. 376–380, Apr. 1991.

[11] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[12] P. S. Maybeck, T. In, A. Form, O. B. Means, O. Mechanical, I. Photocopy, and M. P. S, "Stochastics models, estimation, and control: Introduction," 1979.