Computer Vision

Review Paper

Yongyun Song

October 21, 2022

Hanyang University
Electrical Engineering (Major)
Mathematics (Dual Major)

Abstract

This is the review paper of Computer Vision and Deep Learning.

Chapter 1 consists of fundamental mathematical formulas, which are the essential for the computer vision. Moreover, there is a epipolar geometry in order to handle 3D location and principle points.

Chapter 2 reviews the state-of-the-art computer vision dissertations, such as Structure from Motion, NeRF and ORB-SLAM.

Struecture from Motion is the traditional 3D reconstruction method, which is still be used a lot. We can understand the 3D reconstruction pipeline through this paper.

NeRF is the novel view synthesis from sparse data, which is the most efficient method to create augmentation data with optical rays.

ORB SLAM is Feature-Based SLAM (Simultaneous Localization and Mapping) method in robotics

Chapter 3 introduces pytorch that is an open source machine learning framework and opencv that provide real-time optimized Computer Vision library.

Contents

Chapter 1

Introduction

1.1 Basics

Mathematics is a prerequisite for dealing with computer vision. Given a set of lighting conditions scene geometry, surface qualities, and camera optics, it is crucial to comprehend the image production process that resulted in a certain image.

1.1.1 Homogeneous Coordinate

$$\begin{array}{ccc} (x, y) & \Leftrightarrow & (x, y, 1) \\ \\ \text{Cartesian} & & \text{Homogeneous} \end{array}$$

We can use matrices to represent translations with this homogeneous coordinate. Furthermore, it enables us to represent the depth division in perspective projections.

1.1.2 Transform

Intrinsic matrix represents camera internal parameters such as focal length, principal point and skew coefficient. Extrinsic matrix represents camera external parameters. It is euclidean transformation that combines rotation and translation. In this process, it is converted from a world coordinate system to a camera coordinate system. These two matrices are called calibration matrix.

$$s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & skew\_cf_x & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = A[R \mid t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

This matrix shows how the calibration matrix can be parameterized K, we can combine the intrinsic matrix to create a single 3 $\times 4$ camera matrix.

$$\mathbf{P} = \mathbf{K}[\mathbf{R} \mid t]$$

1.1.3 SIFT

Detecting features at the finest stable scale possible may not be appropriate in many situations. Fine-scale features, for example, may not exist when matching images with little high-frequency detail (e.g., clouds).

One solution is to extract features at different scales, for example, by performing the same operations at different resolutions in a pyramid and then matching features at the same level. This method is appropriate when the images being matched do not have large scale changes, such as when matching consecutive aerial images taken from an airplane or stitching panoramas taken with a fixed-focal-length camera.

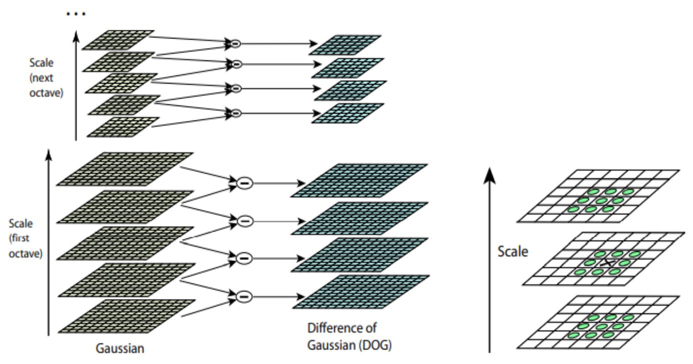


Figure 1.1: SIFT scale invarient

A sub-octave Difference of Gaussian pyramid is used to detect scale-space features. To generate Difference of Gaussian images, adjacent levels of a sub-octave Gaussian pyramid are subtracted. By comparing a pixel to its 26 neighbors, extrema (maxima and minima) in the resulting 3D volume are detected.

Rotational invariance and orientation estimation.

In addition to dealing with scale changes, most image matching and object recognition algorithms need to deal with (at least) in-plane image rotation. One way to deal with this problem is to design descriptors that are rotationally invariant, but such descriptors have poor discriminability, i.e., they map different looking patches to the same descriptor.

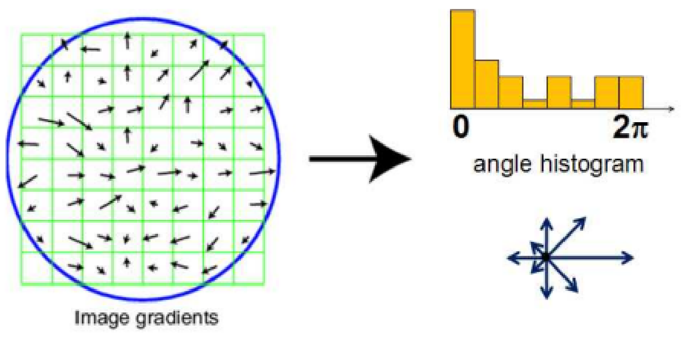


Figure 1.2: Feature descriptor

A better method is to estmiate a dominant orientation at each detected keypoint. Once the local orientation and scale of a keypoint have been estimated, a scaled and oriented patch around the detected point can be extracted and used to form a feature descriptor.

The simplest possible orientation estimate is the average gradient within a region around the keypoint. If a Gaussian weighting function is used, this average gradient is equivalent to a first-order steerable filter, i.e., it can be computed using an image convolution with the horrizontal and vertical derivatives of Gaussian filter. To make this esitmate more reliable, it is usually preferable to use a larger aggregation window (Gaussian kernel size) than detection window.

Examining the histogram of orientations computed around the keypoint is a more reliable technique.

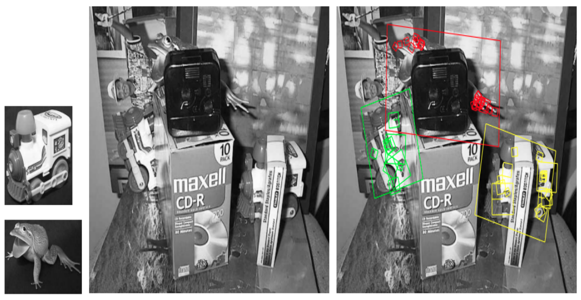


Figure 1.3: SIFT recognition

Despite the objects we want to detect change in scale and rotation, the SIFT algorithm can detect feature and feature descriptor for finding orientation.

1.1.4 Epipolar Geometry

Epipolar geometry is the geometry of stereo vision. When two cameras view a 3D scene from two distinct positions, there are a number of geometric relations between the 3D points and their projections onto the 2D images that lead to constraints between the image points. These relations are derived based on the assumption that the cameras can be approximated by the pinhole camera model.

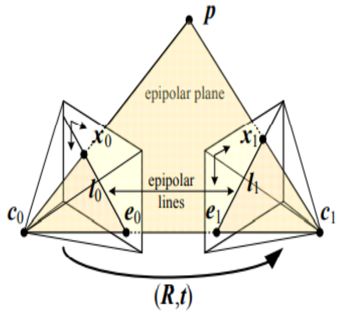


Figure 1.4: Epipolar geometry

1.1.5 Triangulation

The problem of determining a point's 3D position from a set of corresponding image locations and known camera is known as triangulation. One of the simplest ways to solve this problem is to find the 3D point p that lies closest to all of the 3D rays corresponding to the 2D matching feature locations $\mathbf{x}_j$ observed by cameras $\mathbf{P}_j = \mathbf{K}_j[\mathbf{R}_j|\mathbf{t}_j]$, where $\mathbf{t}_j = -\mathbf{R}_j\mathbf{c}_j$ and $\mathbf{c}_j$ is the jth camera center. As you can see in Figure 1.1, these rays originate at $\mathbf{c}_j$ in a direction $\hat{\mathbf{v}}_j$ = $N(\mathbf{R}_j^{-}1\mathbf{K}_j^{-}1\mathbf{x}_j)$, where $N(\mathbf{v})$ normalizes a vector v to unit length. The nearset point to p on this ray, which we denote as $\mathbf{q}_j = \mathbf{c}_j + d_j\hat{\mathbf{v}}_j$, minimizes the distance

$$\left\|\mathbf{q}_j - \mathbf{p}\right\|^2 = \left\|\mathbf{c}_j + d_j\hat{\mathbf{v}}_\mathbf{j} - \mathbf{p}\right\|^2$$

which has a minimum at $d_j = \hat{\mathbf{v}}_\mathbf{j}\dot{(}\mathbf{p} - \mathbf{c}_j)$. Hence,

$$\mathbf{q}_j = \mathbf{c}_j + (\hat{\mathbf{v}}_j\hat{\mathbf{v}}_j^T)(\mathbf{p} - \mathbf{c}_j) = \mathbf{c}_j + (\mathbf{p} - \mathbf{c}_j)_{\|}$$

it is used by axis rotation equation, and the squared distance between p and $\mathbf{q}_j$ is

$$r_j^2 = \left\| (\mathbf{I} - \hat{\mathbf{v}}_j\hat{\mathbf{v}}_j^T)(\mathbf{p} - \mathbf{c}_j) \right\|^2 = \| (\mathbf{p} - \mathbf{c}_j)_\perp \|^2$$

The optimal value for p, which lies closest to all of the rays, can be computed as a regular least squares problem by summing over all the $\mathbf{r}_j^2$ and finding the optimal value of p,

$$\mathbf{p} = \left[\sum_j (\mathbf{I} - \hat{\mathbf{v}}_j\hat{\mathbf{v}}_j^T) \right]^{-1} \left[\sum_j (\mathbf{I} - \hat{\mathbf{v}}_j\hat{\mathbf{v}}_j^T)\mathbf{c}_j \right]$$

1.1.6 Bundle Adjustment

The most accurate way to recover structure and motion is to perform robust non-linear minimization of the measurement (re-projection) errors, which is commonly known in the computer vision communities as bundle adjustment. When projection model is $\hat{p}_{ij} = \pi(\xi_i, \mathbf{X}_j)$, the cost function is:

$$\xi^*, \mathbf{X}^* = argmin \sum_i \sum_j \mathbf{e_{ij}^T e_{ij}} = argmin \sum_i \sum_j (\mathbf{p_{ij}} - \hat{\mathbf{p_{ij}}})^T(\mathbf{p_{ij}} - \hat{\mathbf{p_{ij}}}), \quad \mathbf{e_{ij}} = \mathbf{p_{ij}} - \hat{\mathbf{p_{ij}}}$$

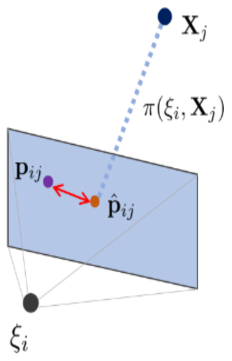


Figure 1.5: Bundle Adjustment

To optimize camera poses and 3D points, $\mathbf{x} = [\xi_1, \cdots, \xi_m, \mathbf{X}_1, \cdots, \mathbf{X}_n]$, two section can be divided.

$$\mathbf{x} = [\mathbf{x}_c \mathbf{x}_p]^T \; where \; \mathbf{x}_c = [\xi_1, \cdots, \xi_m] \in \mathbb{R}^6 m, \; \mathbf{x}_p = [\mathbf{X}_1, \cdots, \mathbf{X}_n] \in \mathbb{R}^{3m}$$

Jacobian $\mathbf{J}$ is derived to make an approximation of Talyor so that The formula is non-linear.

$$\mathbf{e}(\mathbf{x}+\Delta\mathbf{x}) = \mathbf{e}(\mathbf{x}) + \mathbf{J}\Delta\mathbf{x} \textit{ where } \mathbf{J} = [\mathbf{J_c}|\mathbf{J_p}]$$

$\mathbf{J_c}$ is the camera pose from Jacobian, $\frac{\partial \mathbf{e}^T\mathbf{e}}{\partial \xi}$ *and* $\mathbf{J_p}$ is the 3D point from Jacobian, $\frac{\partial \mathbf{e}^T\mathbf{e}}{\partial \mathbf{X}}$

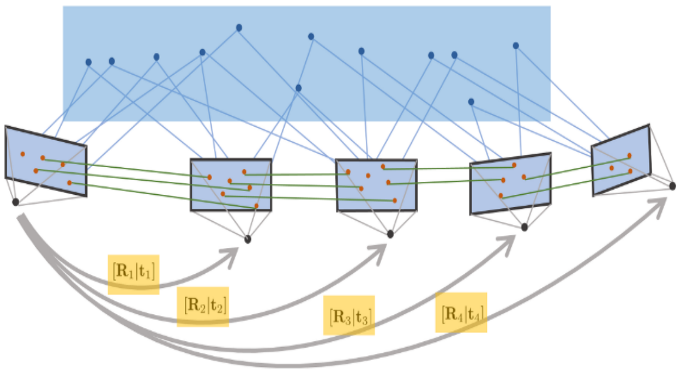

$[\mathbf{R}_1|\mathbf{t}_1]$
$[\mathbf{R}_2|\mathbf{t}_2]$
$[\mathbf{R}_3|\mathbf{t}_3]$
$[\mathbf{R}_4|\mathbf{t}_4]$

Figure 1.6: Least Squares for BA

Chapter 2

Computer Vision

2.1 COLMAP

COLMAP is currently one of the most widely used, as it can reconstruct extremely large scenes. It utilizes Structure from Motion technique that improves 3D reconstruction with multiple view geometry. By projecting a 3D structure onto a series of images acquired from various angles, the SfM technique reconstructs the 3D structure. Feature extraction and matching are frequently the first steps, followed by geometric verification. The generated scene graph serves as the basis for the reconstruction step, which incrementally registers additional images, triangulates scene points, filters outliers, and fine-tunes the reconstruction using bundle adjustment after starting the model with a carefully chosen two-view reconstruction.

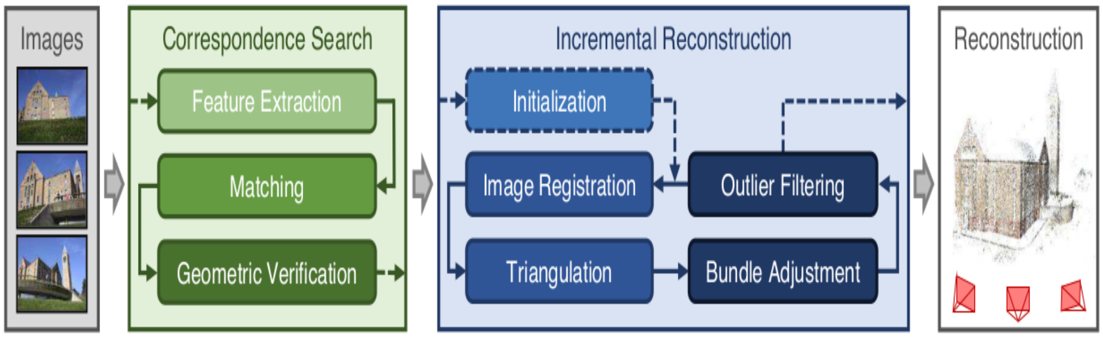


Figure 2.1: COLMAP framwork

In correspondence Search stage, there are three steps in which features are extracted with SIFT method, images are matched by discovering overlapping image pairs, and the potentially overlapping image pairs are verified. Since matching is based solely on appearance, it is not guaranteed that corresponding features actually map to the same scene point. So, we can describes the relation using the trifocal tensor, which is extended to three views compared to epipolar geometry. Since the correspondences from matching are often outlier-contaminated, robust estimation techniques, such as RANSAC are required in order to remove outlier.

Overall, Input is multi images taken from different view points and output is scene graph.

Continuously, there are 5 steps in Incremental Reconstruction stage in which it is initialization, image registration, triangulation, bundle adjustment (BA), and outlier filtering. The scene graph serves as the input during reconstruction and camera pose estimates serves as the output. $\mathcal{P} = \{\mathbf{P}_c \in \mathbf{SE}(3) \mid c = 1...N_p\}$ and reconstructed scene structure as a set of points $\mathcal{X} = \{\mathbf{X}_k \in \mathbb{R}^3 \mid k = 1...N_X\}$.

Initializing from a dense location in the image graph with many overlapping cameras typically results in a more robust and accurate reconstruction.

Image Registration is started by solving the Perspective-n-Point (PnP) problem using feature correspondences.

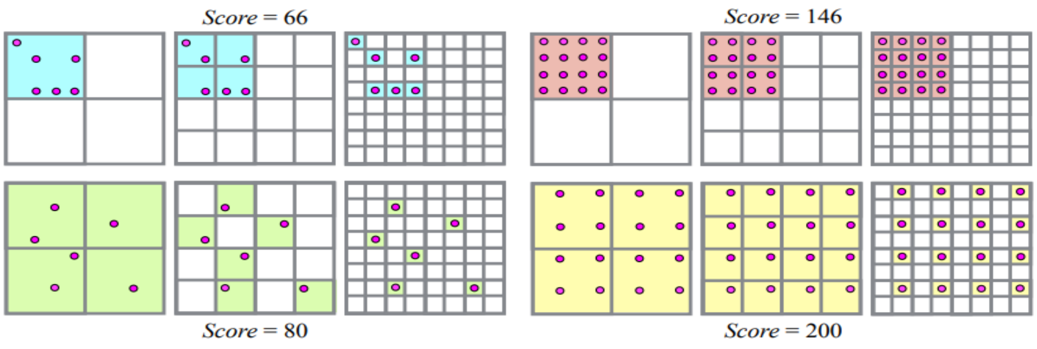


Figure 2.2: Scene graph augmentation for calibration

A multi-model geometric verification technique called ”scene graph augmentation” adds the relevant geometric connection to the scene graph for calibration. To perform calibration ,we firstly estimate a fundamental matrix. If at least N_F inliers are found, the image pair is taken into account as geometrically verified. Next, we determine the number of homography inliers N_H so that categorize the transformation for the same image pair. It also estimate an essential matrix and its number of inlieres N_E. if $N_E/N_F < \varepsilon_{HF}$, we assume correct calibration.

Next, Best View Selection is used to choose the image that sees most triangulated points with the aim of minimizing the uncertainty in camera resection. Triangulation is a crucial step in SfM to find out 3D point with stability. But,Uncertainties in the camera pose affect triangulated points and vice versa. Additional triangulations may improve the initial camera pose through increased redundancy. Without further refinement, SfM usually drifts quickly to a non-recoverable state.

To solve this problem, bundle adjustment is handling camera problems waived by searching for the appropriate principle points. Without further refinement, SfM usually drifts quickly to a non-recoverable state. BA is the joint non-linear refinement of camera parameters $\mathbf{P}_c$ and point parameters $\mathbf{X}_k$ that minimizes the re-projection error.

$$E = \sum_j \rho_j(\|\pi(\mathbf{P_c}, \mathbf{X_k} - \mathbf{x}_j\|_2^2)$$

π projects scene points to image space and a loss function ρ_j is potentially down-weight outliers.

2.2 NeRF

One of the most intriguing uses of computer vision in recent years has been image-based rendering. Using numerous perspectives of an environment, image-based rendering combines 3D reconstruction methods from computer vision with rendering methods from computer graphics to produce interactive photo-realistic experiences such as Photo Tourism system. It results for synthesizing novel views of complex scenes by optimizing an underlying continuous volumetric scene using a sparse set of input views.

Rendering Equation

$$L_{out}(\mathbf{p}, \mathbf{v}, \lambda) = L_{emit}(\mathbf{p}, \mathbf{v}, \lambda) + \int_{\Omega} BRDF(\mathbf{p}, \mathbf{r}, \mathbf{v}, \lambda) \cdot (\mathbf{n}^T \mathbf{r}) d\mathbf{r}$$

The BRDF (Bidirectional Reflectance Distribution Function) is a four-dimensional function that describes the proportion of each wavelength that arrives at an incident direction.

$$L_r(\hat{\mathbf{v}}_\mathbf{r}; \lambda) = \int L_i(\hat{\mathbf{v}}_\mathbf{r}; \lambda) f_r(\hat{\mathbf{v}}_\mathbf{i}, \hat{\mathbf{v}}_\mathbf{r}, \hat{\mathbf{n}}; \lambda) cos^+ \theta_i d\hat{\mathbf{v}}_\mathbf{i}$$

the product of incoming light can be integrated $L_i(\hat{\mathbf{v}}_\mathbf{r}; \lambda)$ with the BRDF.

NeRF

Neural Radiance Fields (NeRF) enable ray-traced rendering of glossy 3D models made from numerous input images by extending the rendering algorithm to take into account pixel ray directions as well as output continuous values for opacities and radiance.

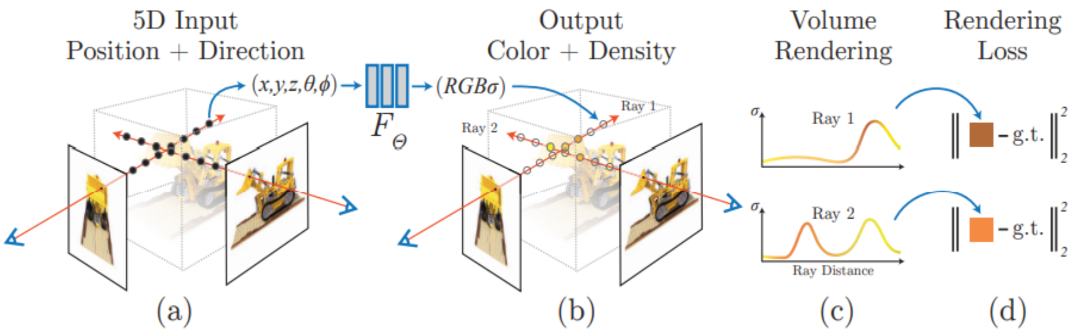


Figure 2.3: Radiometric field for view synthesis

Neural Radiance Field Scene Representation and differentiable rendering procedure have 4 steps.

First, the NeRF feeds those locations into an MLP (Multi Layer Perceptron) to produce a color and volume density. To accomplish this, the MLP F_{theta} first processes the input 3D coordinate $\mathbf{x}$ with 8 fully-connected layers (using ReLU activation and 256 channels per layer), and outputs σ and a 256-dimensional feature vector. This feature vector is then concatenated with the camera ray's viewing direction and passed to one additional fully-connected layer (using a ReLU activation and 128 channels) that output the view-dependent RGB color.

$$F_\theta \ : \ (\mathbf{x}, \mathbf{d}) \ \rightarrow \ (\mathbf{c}, \sigma)$$

Second, it uses volume rendering techniques to composite these values into an image. It renders the color of any ray passing through the scene. The volume density $\sigma(\mathbf{x})$ can be interpreted as the differential probability of a ray terminating at an infinitesimal particle at location $\mathbf{x}$. The expected color $C_{(\mathbf{r})}$ of camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ with near and far bounds t_n and t_f is:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d}))dt, where T(t) = exp\left(- \int_{t_n}^{t} \sigma(\mathbf{r}(s))ds) \right).$$

It numerically estimate this continuous integral using quadrature. Deterministic quadrature, which is typically used for rendering discretized voxel grids, would effectively limit our representation's resolution because the MLP would only be queried at a fixed discrete set of locations. Instead, it uses a stratified sampling approach where we partition $[t_n, t_f]$ into N evenly-spaced bins and then draw one sample uniformly at random from within each bin:

$$t_i \sim \mathcal{U}\left[t_n + \frac{i-1}{N}(t_f - t_n), tn + \frac{i}{N}(t_f - t_n)\right].$$

Although it uses a discrete set of samples to estimate the integral, stratified sampling enables it to represent a continuous scene representation because it results in the MLP being evaluated at continuous positions over the course of optimization.

$$\hat{C}_c(\mathbf{r}) = \sum_{i=1}^{N_c} w_i c_i, w_i = T_i(1 - exp(-\sigma_i \delta_i)).$$

where $\delta_i = t_{i_1} - t_i$ is the distance between adjacent samples.

Third, this rendering function is differentiable, so it can optimize scene representation by minimizing the residual between synthesized and ground truth observed images. There are three tricks such as positional encoding, hierarchical volume sampling, and implemtatiion details.

Positional encoding shows $\gamma(p)$ is a mapping from $\mathbb{R}$ into a higher dimensional space $\mathbb{R}^{2L}$. Formally, the encoding function is:

$$\gamma(p) = (sin(2^0 \pi p), cos(2^0 \pi p), \cdots, sin(2^{L-1} \pi p), cos(2^{L-1} \pi p))$$

Hierarchical volume sampling simultaneously optimize two networks: one "coarse" and one "fine".

$$\hat{C}_c(\mathbf{r}) = \sum_{i=1}^{N_c} w_i c_i, \quad w_i = T_i(1 - exp(-\sigma_i \delta_i)).$$

Implementation details represent ground truth camera poses, intrinsics, and bounds for synthetic data, and COLMAP structure-from-motion package to estimate these parameter for real data.

$$\mathcal{L} = \sum_{r \in R} \left[||\hat{C}_c(\mathbf{r}) - C(\mathbf{r})||_2^2 + ||\hat{C}_f(\mathbf{r}) - C(\mathbf{r})||_2^2 \right]$$

transparancy is the same from different view points only rgb color to be predicted as a function of both location and viewing direction.

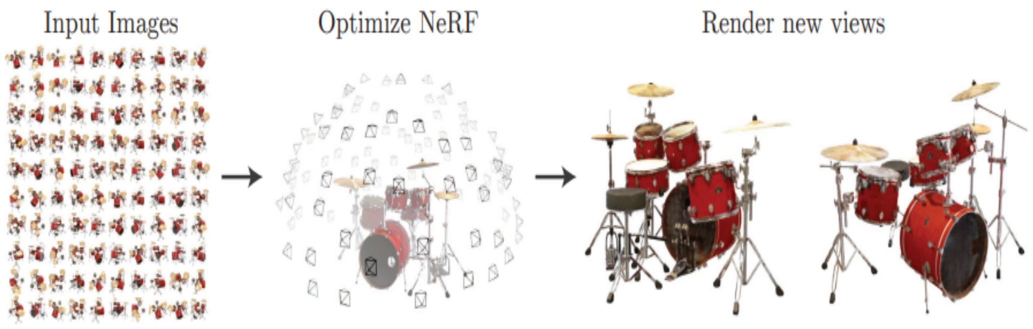


Figure 2.4: Rendering

It represents a method that optimizes a continuous 5D neural radiance field representation (volume density and view-dependent color at any continuous location) of a scene from a set of input images. It uses techniques from volume rendering to accumulate samples of this scene representation along rays to render the scene from any viewpoint. Here, it visualizes the set of 100 input views of the synthetic Drums scene randomly captured on a surrounding hemisphere, and we show two novel views rendered from optimized NeRF.

2.3 ORB-SLAM

SLAM is an incredibly rich and rapidly evolving field of research, full of challenging robust optimization and real-time performance problems.

ORB (Oriented FAST and Rotated BRIEF)

Fast is feature detector to get a one feature from a one keypoint compared to SIFT that get a diverse feature from a one keypoint. Moreover, BRIEF descriptor is binarized to decrease the computational cost. It is effective to combine these two method because it reduce not only the computational cost and time complexity.

Bags of Words

Bag of Words is commonly used for image categorization and image retreval, but recently used for the object and scene. First, feature is extracted by like SIFT (feature extraction algorithm). And then, it create codebook as a dictionary by including just important words to categorize the object and image. Last but not least, the image can be represented by codebook and this codeword can represent the feature with the histogram.

Visual Odometry

Knowing where you are seems to be easy, but it is challenging for robot to know where it is. Camera is used to find a trajectory from essential matrix. Furthermore, there are several sensor used ,such as wheel encoder to know how far it has moved by counting the number of wheel revolutions, IMU sensor with accelerometer, and gyro sensor,Magnetometer.

Monocular ORB-SLAM

Monocular SLAM (Simultaneous Localization And Mapping) was initially solved by filtering. In that approach every frame is processed by the filter to jointly estimate the map feature locations and the camera pose. It cause fulls of linearization errors. On the other hand keyframe-based approaches estimate the map using only selected frames allowing to perform more costly but accurate bundle adjustment optimizations, as mapping is not tied to frame-rate.

The ORB-SLAM uses feature based method (e.g., SURF AND BRIEF)that can be more accurate than direct methods.

There are Parallel three threads. First, Tracking is to Find where you are by map. Second, Local mapping is to find keyframe to do mapping. Lastly, Loop closing is to fix the loop.

BA (Bundle Adjustment) is essential that there is a difference between projection point from 3D point and ground truth to minimize reprojection error. In tracking stage, Motion only BA is used. In local mapping stage, Local BA is used. Lastly, Full BA is used to optimize covisible keyframes in loop closing stage.

The most recent trend in SLAM has been the integration with visual-inertial odometry (VIO) algorithms, which combine higher-frequency inertial measurement unit (IMU) measurements with visual tracks, which serve to remove low-frequency drift. Because IMUs are now commonplace in consumer devices.

I. TRACKING

What it needs for pose estimation is 2D coordinates of a few points, 3D locations of the same points (World Coordinates) and Intrinsic parameters of the camera. ORB Extraction is to extract FAST corners 8 scale levels. Initial Pose Estimation from Previous Frame is to extract constant velocity motion model to predict the camera pose and perform a guided search of the map points observed in the last frame. Cosequently, t-1 data is used for t data if tracking is well in pose estimation stage. IF not, global relocalization with bags of words is used to find similar keyfram using solve PnP.

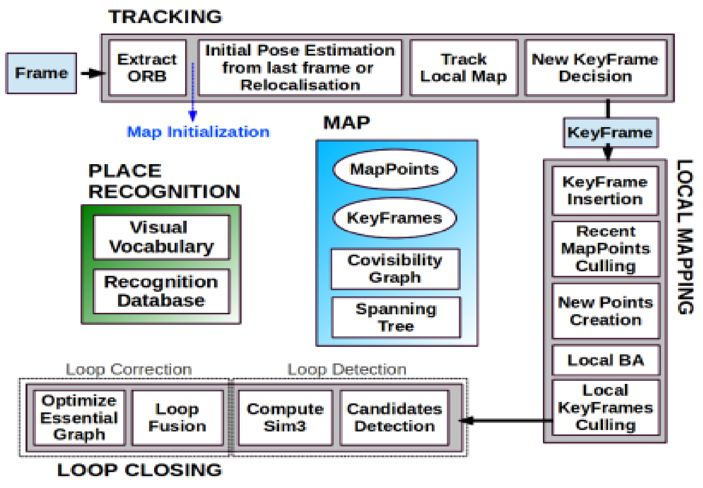


Figure 2.5: ORB SLAM framework

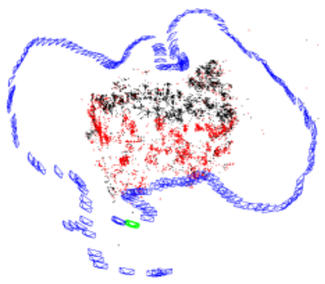
II. LOCAL MAPPING

Local Mapping is used with Covisibility Graph to express similar feature. And then, it removed outlier map points and create new map point with covisibility graph. The methods are used such as Triangulate ORB pairs to check Depth, parallax, reprojection error, scale consistency, Local BA to optimize key frame.

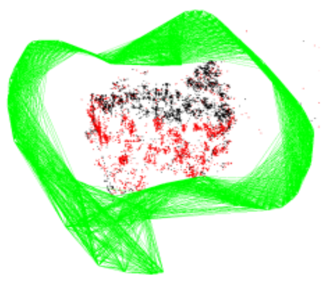
III. LOOP CLOSING

Loop Closing uses Place Recognition with Bag of words that uses visual vocabulary by using Recognition Database and then conducts Loop Detection.

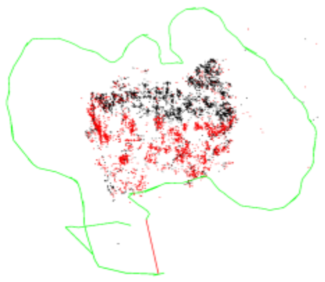
Finally, output is Map Data, such as MapPoints, KeyFrames, Covisibility Graph & Essential Graph and Spanning Tree.



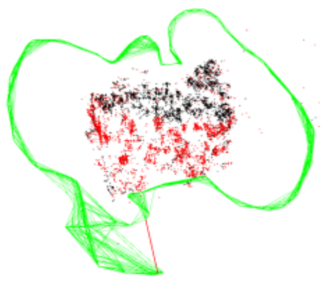
(a) KeyFrames (blue), Current Camera (green), MapPoints (black, red), Current Local MapPoints (red)



(b) Covisibility Graph



(c) Spanning Tree (green) and Loop Closure (red)



(d) Essential Graph

Figure 2.6: ORB SLAM graphs

Chapter 3

Examplification

3.1 SolvePnP

SolvePnP is useful method to find camera pose. Solvepnp -1st - fixed camera + moved motion - pose estimation -2nd - fixed motion + moved camera - camera location estimation

Rodrigues' rotation fomular is an efficient algorithm for rotating a vector in space, given an axis and angle of rotation.

$$\mathbf{p}_{rot} = \mathbf{p} cos\theta (\mathbf{v} \times \mathbf{p}) sin\theta + \mathbf{v}(\mathbf{v} - \mathbf{p})(1 - cos\theta)$$

```
// matching pairs

vector<Point3f> objectPoints; // 3d world coordinates

vector<Point2f> imagePoints; // 2d image coordinates


// camera parameters

double m[] = {fx, 0, cx, 0, fy, cy, 0, 0, 1}; // intrinsic parameters

Mat A(3, 3, CV_64FC1, m); // camera matrix


double d[] = {k1, k2, p1, p2}; // k1,k2: radial distortion, p1,p2:
    tangential distortion

Mat distCoeffs(4, 1, CV_64FC1, d);
```

```
// estimate camera pose

Mat rvec, tvec; // rotation & translation vectors

solvePnP(objectPoints, imagePoints, A, distCoeffs, rvec, tvec);


// extract rotation & translation matrix

Mat R;

Rodrigues(rvec, R);

Mat R_inv = R.inv();


Mat P = -R_inv*tvec;

double* p = (double *)P.data;


// camera position

printf("x=%lf, y=%lf, z=%lf", p[0], p[1], p[2]);
```

Appendix A

References

[1] Computer Vision: Algorithms and Applications, Richard Szeliski

[2] https://demuc.de/papers/schoenberger2016sfm.pdf, COLMAP

[3] https://arxiv.org/pdf/2003.08934.pdf, Nerf

[4] https://arxiv.org/pdf/1502.00956.pdf,ORB-SLAM

[5] https://pytorch.org/tutorials/, pytorch tutorials

[6] https://darkpgmr.tistory.com/99, solvePnP

[7] https://en.wikipedia.org/wiki/Rodrigues%27_rotation_formula, Rodrigues fomular

[8] https://en.wikipedia.org/wiki/Epipolar_geometry, epipolar geomtry

[9] https://www.youtube.com/watch?v=mth-SMWnfas, ORB SLAM tutorials

[10] https://alida.tistory.com/51?category=1061563, Bundle Adjustment

[11] https://groups.csail.mit.edu/drl/courses/cs54-2001s/odometry.html