FROM PHOTOS TO 3D DESIGN:

A PRODUCT SHAPE FAMILY DESIGN AND MODELING FRAMEWORK WITH IMAGE-BASED RECONSTRUCTION AND 3D CONVOLUTIONAL NEURAL NETWORKS

By

TIAN JIN

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Mechanical and Aerospace Engineering

Written under the direction of

Hae Chang Gea

And approved by

_____

_____

_____

_____

New Brunswick, New Jersey

October 2019

ABSTRACT OF THE DISSERTATION

From Photos to 3D Design: A Product Shape Family Design and Modeling Framework with Image-Based Reconstruction and 3D Convolutional Neural Networks

by TIAN JIN

Dissertation Director:

Hae Chang Gea

Current development of new product and product variation is largely driven by the increasingly sophisticated and demanding customers, and product variety plays a crucial role to gain customer satisfaction at a wider range. Our research focuses on the shape variety of product, which is inherently linked to both functional and aesthetic variety. The challenge that we are addressing here is the low conceptual design efficiency with cross-professional and back-and-forth communication due to limited design visualization tools during converting demands into ideas and then into 3D models.

In our research, a learning-based product shape family design and modeling framework is developed, which integrated image-based reconstruction and 3D shape learning to simplify the modeling process but meanwhile providing abundant flexibilities for 3D shape variation and thus improving conceptual design efficiency. With the reconstruction system and learning system, raw model generating process can be as simple as taking photos around design or redesign targets and making selections from predicted models. Two subsystems are developed for the reconstruction process, a Structure from Motion system that recovers camera motion and a sparse structure, and a Multi-View Stereo system generating denser matches and thus denser point clouds. The incremental reconstruction strategy is adopted, and an initial pair selection strategy and an extrinsic matrix correction measure are derived and utilized to provide better initial camera motion

estimation for any further global adjustment optimization. Our experiments shows improved accuracy and robustness of the reconstruction system. To understand what is in the reconstructed point cloud, a 3D convolutional neural network model is constructed to identify 3D shapes from point cloud data. The model is trained by labeled pre-processed point cloud data obtained from both reconstruction and CAD file sampling. The preprocessing includes normalization, down-sampling, manually labeling, and voxelization. The voxel representation made it possible to use convolutional-styled learning method. And the two-layered configuration of Convolution-ReLU-Max Pooling proved to have good performance in classifying 3D shapes.

A new concept of Product Shape Family is defined, and a hierarchical-structured library of product shape family tree is proposed so that classification can be done at different level with a smaller number of candidate classes. Chain rule is used to calculate the possibility at certain shape family and this way the user can be provided with multiple best guesses and select from them at different family generations. A modularized model is defined within every product shape family including internal modules that form the product shape platform and external modules that are optional. The editing process of the selected shape family model is simply selecting desired module and edit shape with pushing and pulling operations on predefined control points. The new design could form a new family branch in the library, or it can be sampled and preprocessed in the same manner as training data and fed back to training process along with newly scanned point clouds.

Two examples are presented to demonstrate performance of the framework from reconstructing 3D point clouds from photos of common design objects to classifying their shape families, and to design with modularized 3D models. Our design and modeling framework reconciles the dilemma between functionality and user-friendliness of 3D modeling tools. In this way, new shape ideas or variants can be easily modeled and visualized in real time and in 3D for non-CAD-users, which improves communication efficiency and hence improves conceptual design efficiency.

Acknowledgements

Throughout my graduate study and writing of this dissertation, there are many I would like to acknowledge, who helped me and contributed to this research in one form or another. First of all, I would like to thank God for giving me the motivation, perseverance, and ability to achieve what I have done. And I would like to thank my family for always giving me the warmest encouragement and continual love. Especially I would like to thank my wife Yuxi Chen for bringing light into my life, for her love and support throughout my research, and for providing me with immeasurable inspiration and motivation. I also would like to extend my thanks to my daughter Atarah Jin for being such a considerate baby, bringing me a joyful start every single day.

I would like to give special thanks to my advisor, Dr. Hae Chang Gea. Words can't express how grateful I feel to have him as my mentor, guidance, and friend, giving me countless help and enlightening discussions not just on this research but also on my personal growth in professional development as well as understanding of "life optimization". I would also like to thank all the other committee members, Dr. Howon Lee, Dr. Xiaoli Bai, and Dr. Weihong Guo, for their sincere and professional comments and suggestions.

Thanks to my colleagues in the Design Automation Lab at Rutgers University, particularly, Wei Song, Huihui Qi, and Euihark Lee for giving me great amount of help in teaching Computer-Aided Design and thus having a better understanding of 3D modeling tools. Thanks to Xiang Yang who brought me into image-based reconstruction and taught me the fundamentals. And thanks to Kazuko Fuchi, Yi Tan, Xing Liu, Bowen Huang, Bo Lei, Jiaming Li, Jingyu Zhang, Debiao Meng, and Hui Ma for sharing their thoughts and ideas for my research work.

Dedication

To Yuxi,

my amazing wife,

for her love, patience, and encouragement,

and for her sacrificial care for me and our child,

and to our beloved daughter,

Atarah Yutian Jin,

who indeed is a treasure from the Lord.

Table of Contents

List of Tables

List of Illustrations

CHAPTER 1

INTRODUCTION

Life nowadays has smart and on-the-go as two of its significant characteristics, owing to fast technological advances of digital devices. To survive in today's highly competitive and volatile markets, companies from almost all industries are innovating or re-engineering their product towards better convenience for their customers. At the same time, the innovating or re-engineering process itself has been put a lot of focus on to achieve higher efficiency during product development. State-of-the-art technologies and tools are developed and integrated so that new iterations of product can be designed, prototyped, tested, manufactured, and be ready on shelves at a speed that keeps up with the trends of customer preferences.

In this chapter, we start with a discussion on the efficiency of current conceptual shape design process and the limits of current idea presenting formats. Then the idea of our design and modeling framework is presented along with a list of our contributions. The chapter ends with a layout of the rest of the dissertation.

1.1 Efficiency for Conceptual Shape Design

Customers today are becoming increasingly sophisticated and demanding. They are sensitive to nuances and differences in products and are not easily attracted only by good product functionality or low price. Thus, product variety plays a crucial role to gain satisfaction for a wider

range of customers, but it requires companies to design and manufacture new product or variation of existing product much more frequently while still keeping the iconic product shape or feature for branding purpose. Apart from speed and efficiency in converting ideas into a launched product, the process to generate design ideas must rely heavily on a thorough consumer research. It has become a common practice for nearly all businesses to be customer-centric and to involve consumers into the design process like the use of consumer panels or even co-designing product with customers.

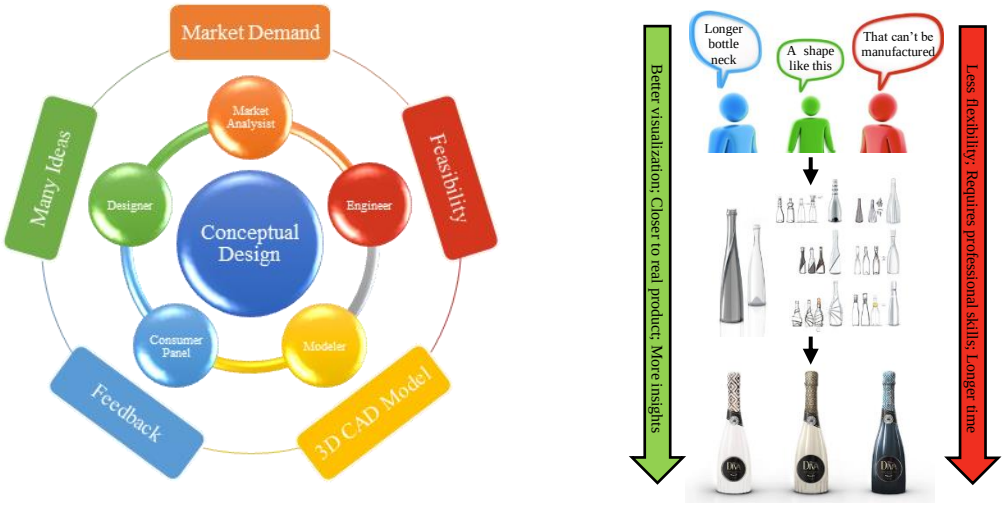


Figure 1-1 Cross-professional combination and the design presenting formats[1] during a design process

Our research focuses on the shape variety of product, which is inherently linked to both functional and aesthetic variety. The challenge that we are addressing here is the design efficiency during the conceptual design phase, which usually involves marketing analysists who have obtained data and ideas from consumer research and market research, selected consumers who potentially provide feedbacks to designs and consumer behaviors, product designers who convert ideas into many possible design sketches, CAD modelers who transfer 2D design sketches into 3D CAD models for viewing and prototyping, and engineers who evaluate feasibility and providing insights on aspects like material, structure, operation, and manufacturing process. The complicated combination, as shown in Figure 1-1, makes sure all aspects of a new design is taken into

consideration. And it is important to have back and forth exchange of opinions among these groups to shape the ideas into desired and also feasible designs. However, the efficiency of this process is low, mostly because every group has their own specialty and is communicating in their own language, which is limited to verbal description with pictures of inspirations especially at early stages. The media gets more visualized after designers draw some sketches with imagination and start to modify them. And finally, with 3D models rendered with material, color, and lighting, the designs can be viewed from all angles and then get further retouched or edited in 3D.

It will be ideal if 3D format is used throughout the design process, where the groups with no experience in CAD are able to share insights based on a more holistic view of the designs, to have a more expressive way to present ideas and potential designs, and to apply quick edits on the models in real time during a discussion session. However, during the conceptual design phase, most design ideas are presented in the form of 2D sketches. Speed is the key factor here and designers are well-trained on converting market demands into many design ideas and presenting them with drawings and sketches. Compared with 3D models, 2D sketches are so far easier and faster for designers to create due to their skillful wrist motion for generating intricate complex curves, shading techniques for vivid 3D effects, and the flexibility that comes with pencil and more recently with powerful vector graphics editing software like Adobe Illustrator [2] and Photoshop[3].

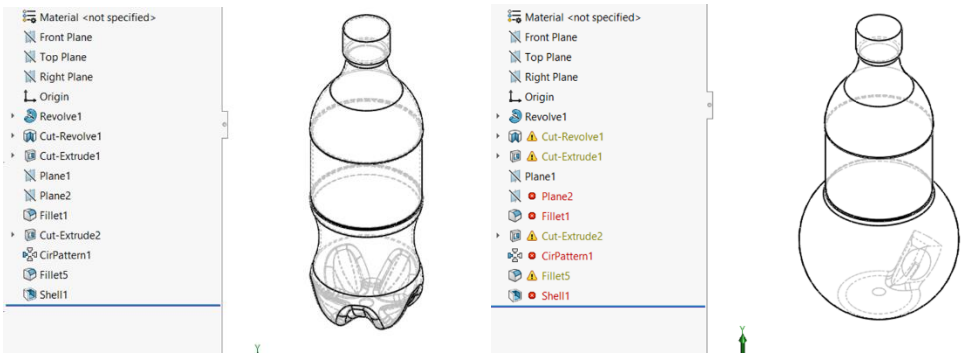


Figure 1-2 Example of feature-based modeling causing rebuild errors for later steps when changing earlier steps

On the contrary, current 3D CAD tools that adopted feature-based modeling methods requires intricate modeling techniques to create easy-to-change models, mostly because features

are applied chronologically and are dependents of previous features. Even with more recent direct modeling, lots of training is required to be able to model complex geometries from the beginning, not to mention modeling from a design idea that is only imaginary. Thus, current design process involves CAD modelers to understand the geometry presented on designers' 2D sketches, plan for a modeling logic, and then create 3D models that are as close to the sketches as possible. After a raw model is created, designers will review the model and give instructions on more details of refinement, where back-and-forth communication is almost inevitable.

To sum up, efficiency of conceptual shape design can be improved if 3D models can be easily created and edited by designers and any non-CAD-users involved in the design process, but currently there are no such tools available and the modeling methods of today's best CAD software requires lots of professional training and is error-prone for design editing.

1.2 Research Objectives

To address the issues mentioned above, we list the following objectives for our research:

(1) To develop a 3D design and modeling framework that assists design idea generation and speeds up idea-to-3D-model conversion;

(2) To improve non-CAD-user-friendliness of 3D design and modeling and grant easy access to creation and editing of 3D designs; and

(3) To improve conceptual design efficiency for product shape innovation.

1.3 Overview of Framework Components

There are mainly three components in our design framework, namely

(1) image-based reconstruction,

(2) deep learning on 3D shape, and

(3) product shape family design and modeling methodology.

The essential idea is to have an easier way to generate a 3D model from a design idea. We choose to use photogrammetry due to the popularity and capability of smart phones. The design process starts from taking photos around real products with any mobile device that has a camera. The reconstruction part consists of a Structure from Motion (SfM) system and a Multi-View Stereo (MVS) system, where SfM analyzes the photos to estimate motion of camera and triangulate a sparse structure of the scene, and MVS further generates a dense point cloud. To obtain a 3D model, the point cloud needs to be first understood by the system. A 3D convolutional neural network (CNN) model is built and trained with manually labeled point clouds to classify shape family of input point clouds. And finally, our 3D design and modeling methodology utilizes a hierarchical structure of shape family tree and transform the shape knowledge of the scan into a generic modularized 3D model, where users can start designing by creating shape variation of individual modules or swapping modules internally or externally.

1.4 Contributions from the Research

Our primary contribution is a product shape family design and modeling methodology integrating image-based reconstruction, hierarchical 3D shape learning, and modularized modeling into a design and modeling framework. Additional contributions from the dissertation include the following:

- Fully implemented Structure from Motion system, Multi-View Stereo system, and 3D shape learning system with 3D CNNs: Section 3.4, 4.2, and 5.3.
- Initial pair selection strategy and extrinsic matrix correction method for improved accuracy and robustness of SfM: Section 3.5.2, and 3.5.3.
- Solution for reconstructed data handling for training and testing, Section 5.3.1, 5.3.2, and 5.3.3.

- A dataset of manually-labeled pre-processed 3D point clouds that is ready-to-use for 3D geometry learning tasks: Section 5.3.3 and 6.3.2.
- A structure of 3D shape classification model using 3D CNNs and a few trained classifiers: Section 5.3.4 and 5.4.
- Shape dataset expanding method using corrupted samples from CAD models and datasets of manually labeled 3D point clouds of different containers and furniture: Section 5.3.3.
- Concept of product shape family and product shape platform: Section 6.1.2
- A hierarchical-structured modularized shape family library: Section 6.1.3.
- A new design modeling process using the framework: Section 6.3.4.

These contributions cannot be substantiated here in the first chapter, but we will revisit them after discussing all the methods and results.

1.5 Overview of the Dissertation

Having laid the foundation of our research by introducing the current problems and an overview of our solution in this chapter, the next chapter presents a review of modeling development, techniques, and methodologies of traditional solid modeling and reversed modeling. Special focus is put on reversed modeling in Section 2.2 to illustrate the efforts as well as opportunities towards new modeling strategies and better modeling efficiency.

The SfM system is introduced in Chapter 3. The related works and theories in each step of SfM is illustrated, starting from two-view to multi-view incremental reconstruction. Experiment results of the implemented system are analyzed to show the drifting problem caused by accumulation of errors. And our solutions are discussed and examined with examples. Following SfM, in Chapter 4 we introduce MVS system to generate a denser point cloud of the scan. Related works and theories are reviewed, and important steps for the MVS system like re-estimating

epipolar geometry and stereo rectification are illustrated with detailed derivation, and results from both rectification and from the whole MVS pipeline are examined to show the performance of our implemented reconstruction system.

To understand the reconstructed point clouds from Chapter 3 and 4, our 3D deep learning system is introduced in Chapter 5, starting from a review of research works on 2D and 3D classification using artificial neural networks. Details of similar image classification with CNNs are discussed as a starting point to move into 3D, and then our model for shape family classification and segmentation is introduced with details including preprocessing steps like normalization and voxelization, training and testing setup, and the layout of our 3D CNN model. Experiments are applied on simple geometries as well as complex design targets like different types of container or furniture. And both qualitative and quantitative results are examined to demonstrate the performance of our shape family identification system.

The design and modeling framework is introduced in Chapter 6 starting with a review of previous research on product family design. Then we introduce the concept of our hierarchical product shape family and the details of the design framework, including how reconstruction and classifiers are integrated to work with a model database, how a redesign or a new design can be generated from modularized model, and how the design framework can be improved and expanded through the use of it. Two design examples using the framework are presented to demonstrate the design process and capability.

The final chapter of the dissertation contains a summary of it, emphasizing answers to the research questions and resulting research contributions. Limitations of the design framework are analyzed and possible future works are discussed.

CHAPTER 2

MODELING AND REVERSE MODELING

3D modeling plays an important role in modern product development and manufacturing process. For decades, large amount of research work and company resources have been dedicated to improving the efficiency of 3D modeling so that the time for designing, prototyping, simulation, and optimization of product can be shortened. However, at the same time the capability of the modeling tools needs to keep improving to match the fast developing manufacturing technologies, to integrate new computer technologies, and to be able to generate more complex geometries. It has always been a challenge to reconcile the conflict between modeling functionality and user-friendliness.

In this chapter, we review the development and research work on both solid modeling and reverse modeling and look into reverse modeling methodologies based on different model format. Solid modeling is more traditional but has grown into a very powerful tool to connect design with manufacturing and product lifecycle management. But reverse modeling has gained particular interest in recent years thanks to the development of mobile devices and scanning technologies. We put special emphasis on reversed modeling methodology because we would like to take advantage of its potential for design automation and thus improve user-friendliness and more importantly design efficiency.

2.1 Solid Modeling

Since Patrick J. Hanratty developed the first commercial computer numerical control (CNC) programming tool Pronto in 1957 [4], CAD software has evolved from 2D engineering drawing drafting tools into 3D modelers that can create, modify, assemble, analyze, and optimize complex models, playing an important role in product lifecycle management processes. Nowadays CAD becomes widely used in many applications, including automotive, shipbuilding, and aerospace industries, industrial and architectural design, as well as animation, special effects in movies, and many more.

Major milestones include the invention of Non-Uniform Rational Basis Spline (NURBS) which formed the basis of 3D curve and surface modeling, and the development of Part and Assembly Description Language (PADL). In the late 1970s, Constructive Solid Geometry (CSG) and Boundary Representation (BREP) laid the foundations of solid modeling [5]. CSG describes a solid model using basic three-dimensional shapes such as cuboids and spheres or better known as primitives, and Boolean operations of these basic elements. And BREP models use surface definitions to describe the enclosed solid [4].

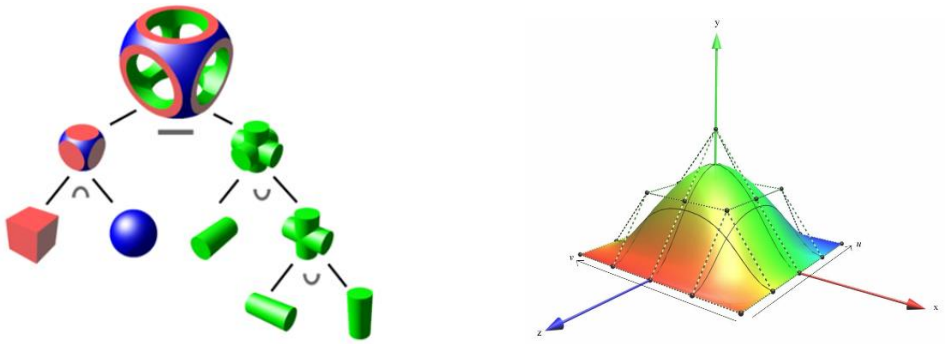


Figure 2-1 Example of Constructive Solid Geometry and NURBS surface [6]

Today, Autodesk (AUTOCAD, FUSION 360, MAYA), Dassault Systèmes (CATIA, SOLIDWORKS), PTC (Creo), and Siemens PLM Software (NX) are the top vendors that dominate the global CAD market [7], and almost all of them are in the category of Parametric feature-based modeling and Freeform surface modeling.

Parametric Feature-Based Modeling builds a model with features, which are parametric shapes associated with attributes such as the intrinsic geometric parameters (length, width, depth), position and orientation, geometric tolerances, material properties, and references to other features. Features cover the overall information flow of the design and production, helping product designs to be effectively transformed to manufacturing processes[8]. In these solid modelers, features are created based on sketches, part is composed of different features, and assembly is composed of parts. Very powerful indeed, but parametric modeling requires more skill in model creation and editing, since any modification on an early feature may cause later features to fail.

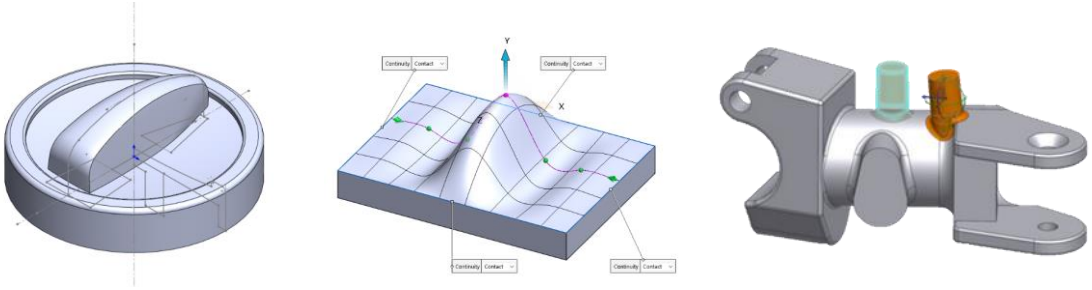


Figure 2-2 Example of feature-based modeling and freeform surface modeling in Solidworks [9], and Direct modeling in PTC Creo

Another method used in CAD is Freeform Surface Modeling, which is usually used to create aesthetic surfaces, for example consumer product packaging, or technical surfaces such as gas turbine blades. There are two basic methods in CAD software to create freeform surfaces. Spline curves can be swept along guideline or lofted to create 3D surface. Or as used in many animation design software, surfaces are generated from subdividing a mesh of control points, which can be manipulated to adjust the shape of the surface with connectivity and continuity maintained.

2.2 Reverse Modeling

The modeling process sometimes needs to go backwards. It is often necessary to produce a copy of a part when no original drawings or documentation are available. In other cases, we may want to re-engineer an existing part when analysis and modifications are required to construct a

new improved product.[10] Also for modern industrial design and especially automotive design, conceptual designers, in most cases artists instead of digital modelers, have the original design presented in the form of coarse industrial clay models, which must be scanned and fitted so that further clay milling can be accomplished. There are also cases when a new model will be designed based on an existing structure.

To acquire digital models, non-contact methods use light, sound, or magnetic fields to detect the surface of the object, while tactile methods touches the surface of the object using mechanical probes at the end of a robot arm.

Due to physical constraints of the acquisition systems, which are only able to measure a limited region of the object surface with discrete samples, captured data is usually in the form of a point cloud, and data from multiple scans must be combined into a complete measure. The captured raw data is also corrupted by noise and error from the observation process and thus needs to be further processed.

The obtained data cloud however is not enough to be handled by CAD systems until continuous bounding surfaces, either regular or free-form, that fit the discrete points are generated. For applications in mechanical engineering, where surface type may decide the way to manufacture it, a more precise surface representation is needed, rather than only collections of facets. That's why much research attention is focused on fitting primitives to a scanned data cloud.

Eventually the goal is to recreate an editable contiguous model of vertices, edges, and faces. Then the method to remodel it is decided by how the model is represented, for example by tuning parameters for feature-based models, changing locations of control points for free-form surface models, or by direct modification of mesh models.

2.2.1 Scanning Technologies

There are many scanning technologies to capture 3D data from an existing object, using different types of interaction with the surface of it.

Tactile methods like Coordinate Measuring Machine (CMM) sensing points on object surface with a probe, which is either connected to a programmed robot arm or a handheld device with its pose and position tracked. These methods are among the most robust, but the speed is also the slowest.

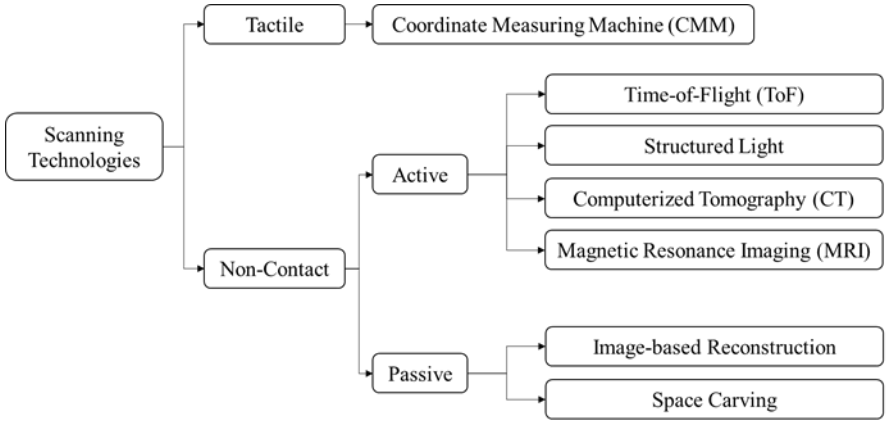


Figure 2-3 Classification of scanning technologies

Non-contact methods can be active or passive. Active methods include time-of-flight estimation and triangulation systems, which involves certain type of controlled energy beam and reflected energy detection[10, 11]. Time-of-flight technology[12, 13], most used for large scale object scanning, tracks the time between sending out a laser or ultrasonic pulse and receiving the reflected signal. With the speed of light and sound known, the distance can be calculated. Triangulation methods, which is more suitable for small scale scanning, use strips, grids, or other coded patterns of light projected onto a surface and detect the distorted light structure with a photosensitive device. The distance can then be deduced from the known angle and distance with geometric triangulation. It is very fast and full of details because it can obtain large amount of data within one single image frame. But it requires a lot of work to set up for a scan and very delicate calibration of devices. Special active methods like computerized tomography (CT) and magnetic

resonance imaging (MRI) emit and detect electromagnetic waves to reconstruct the inner structure of human tissues.

Currently there are many commercial scanning devices available in the market. Different types of active sensing devices are used on different distance range cases, like Lidar sensors for large-scale long-range scanning, time-of-flight sensors for mid-range objects, and structured-light devices for close range objects. Although these scanners can obtain very high precision data, cost (both financial cost for device and computational power and time cost to handle large amount of data) and flexibility are among the biggest issue. Also, if multiple active scanners are working at the same place and the same time, sensors will receive enormous amount of noise and making reconstructed model hard to handle.

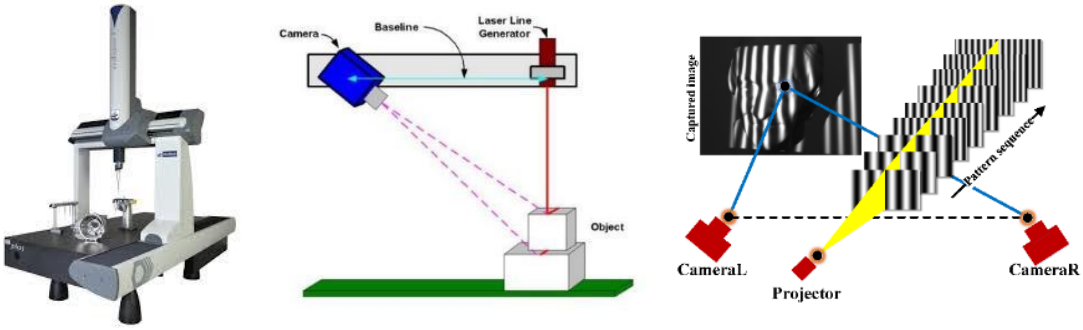


Figure 2-4 Example of CMM, time-of-flight, and structured light method for 3D scanning

Passive scanning methods do not use any artificial energy source but uses images alone to obtain three-dimensional location information. These image-based shape recovering methods achieve 3D data with triangulation from correspondents between images or depth data using methods such as shape from shading[14], texture[15], specularity[16], shape from contour[17], and shape from 2D edge gradients[18]. Most passive methods obtain shape from multiple images, however there are also techniques that achieve reconstruction of 3D shape from a single image or monocular video sequences [19-23]. Space carving is another powerful passive scanning method that treats space as voxels and carve background out from different angles[24].

Compared with active methods, passive scanning techniques are more flexible, widely applicable, portable, and affordable. Methods like image-based reconstruction with camera pose estimation and stereo disparity provides a low-cost solution that does not require additional devices but can be accomplished with portable mobile devices. Device interference would not be a concern in these methods. However, the scanning accuracy will be affected by poor calibrated camera parameters and misdetection or mismatching of corresponding pixels in different images. Also, since multiple images are required to contribute enough over-lapping view of the scene, a registration process is required to align newly triangulated data with reconstructed data set, which causes an accumulation of error throughout the reconstruction process. Moreover, depending totally on pixel matches from images, these methods are very limited when facing too dark/shiny scanning environment, texture-lacking surface, transparent surface, thin/fuzzy structure, non-Lambertian surfaces, patterned texture, and occlusion.

2.2.2 Modeling with Point Clouds

The scanned results come out in the form of a point cloud. Most commercial 3D point cloud processing software like CloudCompare [25] and Meshlab [26] focuses on point clouds display enhancement, registration, resampling, cleaning/smoothing, color/normal/scalar fields handling, measuring, statistics computation, segmentation, and surface/geometry fitting. More recently with the help of machine learning, tools like VRMesh [27] and Bentley Pointools [28] can be applied on large-scale scanned point clouds of terrestrial or infrastructural area to classify different types of object automatically and also provides brush tools to manually correct errant classifications. However, very few direct editing operations could be applied on the point cloud models except for selection, moving, and deleting, mostly because the applications are targeting at constructional site management or large-scale infrastructure planning.

Algorithms on point cloud direct processing are mostly designed for restoring the original scanned scene or object, like clouds registration [29] that merges multiple scanning data into one coordinate system and hole filling on scan-failing areas [30], or for semantically classifying point clouds by learning a large labeled 3D database [31-35]. A procedural 3D building editing method is reported in [36] that operates directly on point clouds to interactively create new point cloud by using procedural copy, paste, and resizing of semi-automatically segmented 3D building data. Compared with direct processing, most methods for redesigning or sculping a scanned point cloud usually start from creating a triangular mesh surface.

2.2.3 Model Representation

Usually the measured point samples need to be further reconstructed into a polygon mesh model for rendering, analysis, redesigning, or printing purpose. And in mechanical applications a more parameterized model is required for simulation and manufacturing purpose, like basic primitives, feature-based models, or free form surfaces. However, there are several difficulties. The point cloud is often sampled nonuniformly with various sparsity and sometimes redundancy, and positions of these samples are usually corrupted by sampling inaccuracy. Also due to occlusion or other accessibility constraints, some region will be missing and leave a hole on it. So, model reconstruction algorithms try to infer the topology of the mesh model, fit the noisy data, and detect and fill holes reasonably.

Based on the type of representation of the surface, most methods of model reconstruction from a point cloud can be classified into the following categories:

(1) Parametric: in these methods, a continuous surface is represented by stitching together a number of parametric surfaces that are described by parametric equations, for example NURBS[37] and Coons patches[38].

(2) Implicit: in these methods, for example Poisson surface reconstruction[39], sample points are fit into a smooth and approximated surface using the zero set of an implicit function, such as sum of radial basis[40] or piecewise polynomial functions[41].

(3) Combinatorial: most combinatorial methods are Delaunay-based[42, 43], using all or a subset of input points which are on the surface and connect them appropriately to form a mesh model composed of a collection of simple entities like points, edges, and polygons. Example algorithms are Alpha-Shape[44], Power Crust[45], and Cocone[46].

Other methods include deformable models[47] that deforming an initial surface so that it gives a good approximation of the given set of points, Moving Least Square approaches[48] define a surface as an invariant set of a project operator, which is a numerical optimization step on a locally constructed implicit function, Bayesian model[49] uses Bayes' rule to infer a reconstruction of maximum probability.

Commercial CAD software are usually feature-based modelers, thus the reconstructed meshes needs to be segmented into groups based on information like surface normal to obtain surfaces, edges, and vertices, and then to be classified into different basic geometric features[50], like extrusion, revolve, sweep, etc. Some other algorithms fit the cloud directly into geometric primitives like plane, cylinder, sphere, torus, etc., or parametric surface as mentioned above. All these methods make the point cloud or surface model into a higher-level shape model, with easier access to be edited through feature parameters or surface control points.

Mesh model reconstruction algorithms that approximate the input cloud can be categorized based on different priors assumed on the dense point cloud or the scanned shape, including surface smoothness, visibility, volumetric smoothness, geometric primitives, global regularity, data-driven, etc. Algorithms that assume local smoothness [48, 51, 52] generate smooth output only in regions with data, while global smoothness methods [43, 53, 54] ensure entire model is smooth, which is better at handling missing data. And piecewise smooth methods [55-57] recover sharp features first

and ensure smoothness away from these areas. Volumetric smoothness assumption [58-60] ensures the local shape thickness varies smoothly to handle challenging missing data. For CAD and architecture models, geometric primitives (or volumetric primitives [61]) can be detected, fitted [62], and merged into a model [63]. Some hybrid methods exist to combine primitive and non-primitive portions together [64, 65]. Shape regularity, like symmetry [66], repetition [67], or canonical relationships [68] (coplanar, orthogonality, coaxial, etc.) are shown to have great use handling severe defects of a point cloud. Data-driven (or semantic) methods [69, 70] are more flexible by using a database of known shapes to help perform reconstruction. With the help of user knowledge, user-driven methods could gather very reliable information on feature [71], topology [72], structural repetition [73], and primitives [74].

Delaunay-based methods are another large research focus that form the triangulated mesh model by a subcomplex of the Delaunay triangulation with the help of Voronoi cells. These methods include tangent plane methods [42, 75] that assume neighbors of a point should not deviate too much from the tangent plane of the surface at that point, restricted Delaunay-based methods like crust [76] and cocone [77] that choose just a subset of Delaunay triangulations that is a good approximation of the unknown surface, inside-outside labeling methods like power crust [45], natural neighbors [78], wrap [79], convection [80], and empty balls methods like ball pivoting algorithm [81] and regular interpolant algorithm [82]. These methods guarantee geometric qualities of reconstruction, but are impractical for scanned real-world scenes with significant imperfections[83].

2.2.4 Mesh Model Editing

Unlike traditional solid modeling used in most CAD/CAM systems, which includes implicit modeling like CSG with primitives and procedural modeling like Parametric feature-based modeling, surface modeling or raw mesh modeling deal with surface patches or polygonal meshes

directly. Non-uniform rational basis spline (NURBS) surface modeling was dominant in early surface modeling systems and became the standard approach for industrial design, which defines surface with NURBS patches described by parametric equations and stitch them together with high level of smoothness. However, despite modern NURBS modeling tools like Solidworks [9], Autodesk Alias [84], and Rhino [85] stand out for their interface usability, NURBS modeling is still difficult for artist because of the smoothness constraint for rigid topological structures.

Another strategy to handle surface is Subdivision surface modeling, and the subdividing algorithm proposed by Catmull and Clark [86] remains dominant today and is integrated in almost all the entertainment 3D designing tool in industry, like Maya [87], Blender [88], 3DSMax[89], and AutoCAD [90]. SubD modelers allow designers to work on a low-resolution quad control mesh to manipulate the shape of the smooth surface, which gives designer much easier control over the trade-off between topological flexibility and surface smoothness. Also, the iterative algorithm makes it possible to create a multiresolution surface by storing edits at different resolutions using displacement vectors. However, the edit operations (usually pulling) on mesh vertices is tedious especially performed at a projected view. There is another easier version push/pull technology used in SketchUp[91], which extrudes sketches to form 3D shapes, mostly used in speed architecture or interior concept design.

Compared with SubD, digital sculpting is a much more convenient interface for designers by dragging a brush in 3D space and generate a specified displacement or hard surface on existing surface while preserving smoothness. Sculpting tools like ZBrush and MudBox made it possible to sculpt very high-resolution mesh models, and very quickly gained acceptance among designers especially in film and game industries.

Part-based surface modeling introduced the idea of surface parts of mesh models, where part is defined by any change (deformation) of a surface. And an ON operation was defined by the means of applying the change on the surface, which made the design interface more efficient and expressive.

Basic direct operations on a mesh model or a selected mesh group include local operations like resampling, discarding, erasing and filling, extruding, offsetting, deforming, sculpting, displacement stamping, fusion, and global operations like mirroring, transforming, cutting, and Boolean operations. However, most of these operations are developed for repairing a scanned mesh model or generating creative surface design for entertainment purpose with very little control of quantitative characters of the model.

2.3 Remarks

Although technologies and algorithms on each step of reverse engineering have been developed through the years, it is still challenging and time consuming to obtain an easy-to-edit model from an existing object, and even more challenging for a designer to be involved in the interactive reconstruction process.

When converting scanned point clouds into mesh models, choice upon smoothness and level of details is always a dilemma, considering most scanned data is not noise-free. Implicit methods tend to obtain smooth surfaces, but are very sensitive to noise, outliers, misalignment, and non-uniform sampling, thus details like sharp edges get smoothed, or undesired surfaces are generated. Combinatorial methods keep most of the details, but all the scanning inaccuracy will be shown on the reconstructed mesh model as well. Fitting primitives directly is only applicable to object with basic geometries like planes, spheres, cylinders, cones, and torii.

Choice of remodeling method depends on the type of data representation, either point cloud or mesh surface. In most cases scanned point clouds are converted into mesh models for editing, with parameterized features, control points, or simple triangle mesh. Feature-based models are difficult to modify due to features being constructed chronologically. Designers need to first understand how the model is constructed step by step, and then will be able to edit sketches and parameters in certain steps. Also, editing previous features sometimes leads to rebuild error of later

features, which is very inefficient especially during conceptual design stage. The feature fitting process is usually tedious and time consuming when many features are included on the scanned object. Similarly, free-from surface modeling also requires a fitting process on point clouds for NURBS or Bezier patches with continuity satisfied at boundaries. More control points give more accuracy of the surface generated from original point clouds, but it also complicates the editing process. Moreover, before any of these fitting processes, segmentation must be applied on the point clouds, usually based on grouping of points with similar normal directions, which, if not done properly, may result in mistakenly treating portions of point cloud as not-a-feature. As for direct mesh editing, local refinement and global editing are both applicable, but the editing is more of a sculpting process, and it is challenging to quantify each editing step so that an accurate repeatable model can be generated.

User interaction can indeed provide valuable input containing higher-level knowledge of the object and geometry, and thus help guiding the reconstruction process on choice of feature, surface type, topology, or primitive. But the amount of user operations needs to be designed carefully since too many operations makes the process tedious, while insufficient operations cannot cover the reconstruction errors from automatic process.

In conclusion, we list the following main challenges for reverse modeling, which we try to address in our research.

(1) Robustness versus convenience in scanning object into point cloud;

(2) Methodology of editing reconstructed structure.

CHAPTER 3

STRUCTURE FROM MOTION

In our learning-based product shape family design and modeling framework, the first step is to obtain a three-dimensional expression of the product in a point cloud format so that learning and classification can later be applied on it to support the design process. And passive photogrammetry with mobile device provides the most flexibility for product of different sizes at any location.

Perspective projection, just like how our eyes work, is not trustworthy from a single view and provides very limited information of the scene. Even when both of our eyes are used to look at a scene from a certain angle, the information is still far from enough for our brain to process and reconstruct a reliable structure of the scene, not just because of the missing information of the occluded back side, but the ambiguity inherited from the projection mechanism, not even with knowledge of the possible structure since the prediction of the possible structure may actually be based on an optical illusion, like the examples shown in Figure 3-1.

(a) (b)

Figure 3-1 Examples of optical illusion for (a) a 2D image and (b) a 3D structure[92] viewed from different angles

For a human being, the solution is to simply walk around the scene so that all the views will be integrated by our brain to register a reliable structure, or if not viewing in this continuous manner, a few looks from different angles also provide enough information to reconstruct most of the details of the structure. This process, if replacing our eyes with perspective camera and our brain with a computer algorithm, is often referred to as multi-view Structure from Motion.

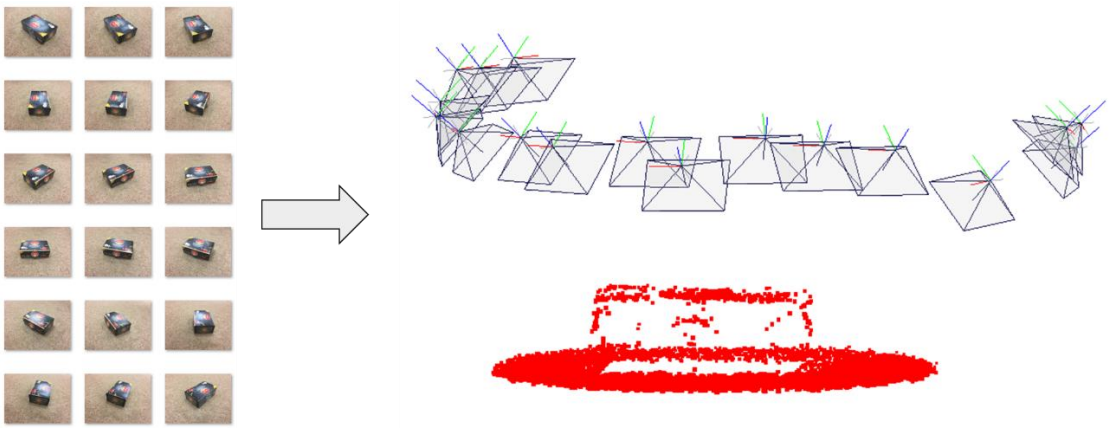


Figure 3-2 Input images and visualized output of Structure from Motion

As shown in Figure 3-2, several photos are taken around a box placed on the floor. Each photo captures some part of the box and every two photos have shared view of certain part of the box. For human being, because of the overlapping, it is easy to use our spatial imagination to roughly restore the relative viewing angles of each 2 photos and a possible structure. Similarly, for computer, these overlapping areas provide possibility to identify strong correspondence between pixels and thus can be used to estimate relative positions and poses of cameras, with which triangulation can be done to reconstruct 3D locations of points that are seen on two or more photos.

Starting from the seminal work of Longuet-Higgins [93] that first introduced a linear method based on point correspondences, large amount of research work in computer vision has been focused on Structure from Motion (SfM) as a starting point to recover 3D scenes from 2D images. In this chapter, theories and algorithms are elaborated along with related works on each step of SfM, namely keypoint or feature detection and matching, camera calibration, camera location and pose estimation, 3D coordinate triangulation, and camera registration. Issues came up

in the registration process are addressed with correction methods so that a better initial set can be obtained for the bundle adjustment process.

3.1 Keypoint Detection and Matching

The passive reconstruction process has only image pixels as its input, thus the pipeline starts with an analysis of pixels to identify features and then match them, which serve as the basis for calibration, camera estimation and structure estimation.

3.1.1 Related Works

Features, or interest points, are the key to relate images taken at the same scene. Feature detection tries to locate edges [94], corners [95, 96], or blobs [97, 98], using the image gradient matrix or Hessian matrix and looking for large changes or local extrema. Many early feature descriptors are derived from local image intensities, for example using derivatives of Gaussians of different order [99], filter banks [100], and phase information [101]. The most widely used descriptor, Scale-Invariant Feature Transform (SIFT) [97, 102], calculate descriptors using local gradient histograms sampled in the local neighborhood of a keypoint and has the advantage of invariance to changes in scale and rotation. Built upon SIFT, principle component analysis (PCA) was applied to image gradient in PCA-SIFT [103] and a more compact representation was derived. An extension of SIFT called Gradient Location and Orientation Histogram (GLOH) [104] combined it with the idea of Shape Context descriptor, which is based on an edge histogram sampled from a log-polar grid. To speed up computation of SIFT, Speeded Up Robust Features (SURF) [98] used box filters instead of Gaussian filters to derive descriptors. Other descriptors used learning approaches [105, 106] or training-based approaches[107, 108].

To match the extracted features between images, Euclidean distance between descriptor vector of features are calculated and thus a nearest neighbor can be found. Then a fixed threshold can be applied to identify matched pairs that have much closer Euclidean distance than other neighbor feature pairs.

3.1.2 Scale-Invariant Feature Transform

SIFT descriptor for robust matching was presented by Lowe [102] to extract highly distinctive features invariant to scale and rotation, affine distortion, change of perspective view, addition of noise, and change in illumination, that a single feature can be correctly matched with high probability against a large database of features from many images in the SfM process.

There are 4 major stages to extract SIFT descriptors from an image. First a search is applied over all scales and locations to identify potential interest points using a difference-of-Gaussian function. Then from the candidates, keypoints are selected based on measures of their stability. Each keypoint is assigned one or more orientations based on image gradient. And finally, local gradients at selected scale around each keypoint are measured and transformed into a descriptor representation.

Stage 1: Scale-space extrema detection

The scale space $L(x, y, \sigma)$ of an input image $I(x, y)$ is produced from convolution of a variable-scale Gaussian $G(x, y, \sigma)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y), \tag{3-1-1}$$

where $*$ is the convolution operation in x and y, and

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

For every 2 nearby scales separated by a constant factor k, the difference-of-Gaussian function convolved with the image, $D(x, y, \sigma)$, defined by

$$\begin{aligned} D(x, y, \sigma) &= \left[G(x, y, k\sigma) - G(x, y, \sigma)\right] * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned}, \tag{3-1-3}$$

can be computed and its scale-space extrema can be used to detect stable keypoint locations, where $L(x, y, \sigma)$ is the smoothed image. The reason to use D is that it provides a close approximation to the scale-normalized Laplacian of Gaussian, $\sigma^2\nabla^2 G$, whose extrema produces the most stable image features.

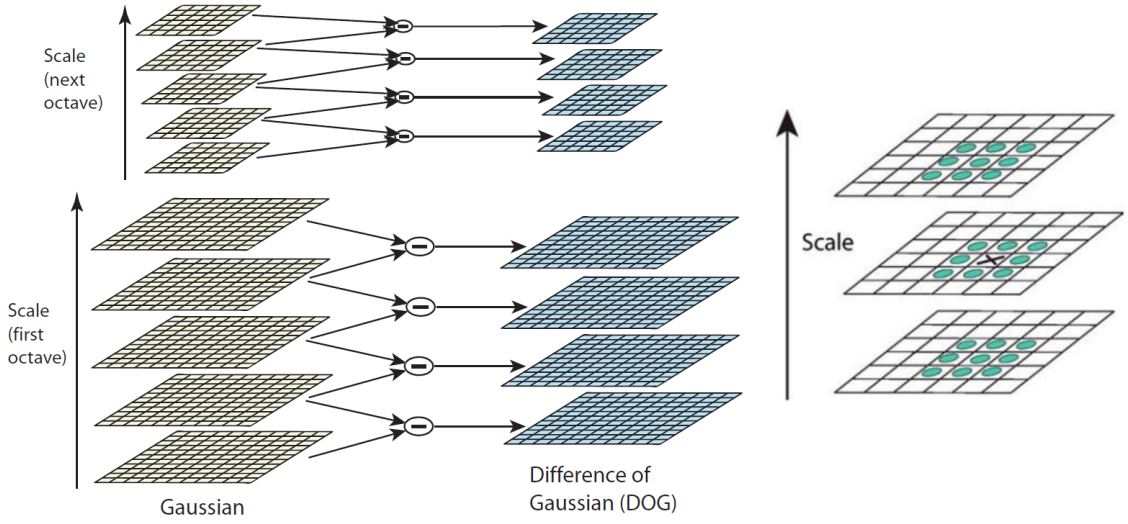


Figure 3-3 (a) DOG images from different octave of scale space (b) Extrema of DOG [102]

The approach to construct DOG images is shown in Figure 3-3 (a). Gaussians varied by a factor k in scale space are incrementally convolved with an initial image, producing octaves (doubling of σ) of blurred images and each octave is divided into an integer number of intervals. Then DOG images can be obtained by subtracting adjacent images. The same process will be applied to the next octave with Gaussian image down-sampled by taking every second pixel in each row and column. Then as shown in Figure 3-3 (b), local maxima and minima of $D(x, y, \sigma)$ is

detected by comparing each sample point to its 26 neighbors in 3×3 regions at the current and adjacent scales.

Stage 2: Keypoint localization

Instead of simply locate keypoint at the location and scale of the central sample point, a 3D quadratic function is fitted to the local sample points to determine the interpolated location of the extrema. The scale-space function $D(x, y, \sigma)$ is expanded up to quadratic terms

$$D(\mathbf{x}) = D + \frac{\partial D^{\mathsf{T}}}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^{\mathsf{T}}\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{3-1-4}$$

and, by taking derivative with respect to $\mathbf{x}$ and setting it to zero, the location of the extrema $\hat{\mathbf{x}}$ can be determined.

$$\mathbf{x} = -\left(\frac{\partial^2 D}{\partial \mathbf{x}^2}\right)^{-1}\frac{\partial D}{\partial \mathbf{x}} \tag{3-1-5}$$

The function value at the extrema

$$D(\mathbf{x}) = D + \frac{1}{2}\frac{\partial D}{\partial \mathbf{x}}\mathbf{x}, \tag{3-1-6}$$

can be used for rejecting unstable extrema with low contrast.

However, a poor peak in the DOG function could have a large principle curvature across the edge but a small one in the perpendicular direction. For further stability, principal curvatures of D can be computed from a Hessian matrix,

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix} \tag{3-1-7}$$

And the check of ratio r between eigenvalues, α and β, where $\alpha = r\beta$, can be easily accessed from checking

$$\frac{\mathrm{Tr}(\mathbf{H}^2)}{\mathrm{Det}(\mathbf{H})} = \frac{(r+1)^2}{r} \tag{3-1-8}$$

Stage 3: Orientation assignment

The scale of a keypoint are used to select the level of Gaussian blur. And for each blurred $L(x,y)$, the gradient magnitude $m(x,y)$, and orientation $\theta(x,y)$ can be computed with

$$\begin{aligned} m(x,y) &= \sqrt{\left[L(x+1,y)-L(x-1,y)\right]^2+\left[L(x,y+1)-L(x,y-1)\right]^2} \\ \theta(x,y) &= \tan^{-1}\left[\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}\right] \end{aligned} \tag{3-1-9}$$

Then within a region around each keypoint, gradient orientations are weighted by magnitude and by a Gaussian-weighted circular window with σ that is 1.5 times the scale of keypoint and added into a histogram with 36 bins. Peaks are located and local peaks within 80% of the highest peak are used to create new keypoints. And for better accuracy, interpolation with a parabola fit is applied on the 3 histogram values closest to each peak. As a result, every keypoint has been assigned a location, scale, and orientation.

Stage 4: Keypoint descriptor

As illustrated by Figure 3-4, image gradient magnitudes and orientations are calculated for every point around a keypoint within a certain range at the scale of the keypoint, where all the orientations are rotated relative to the keypoint orientation for the purpose of orientation invariance. Then a Gaussian weighting function with σ that is 1.5 times the width of the descriptor window to provide more weight on the gradients closer to keypoint and to avoid sudden changes. Then the 8×8 set of samples are divided into four 4×4 regions and orientation on 8 directions form the histograms such that a gradient sample still contributes to the same histogram even shifted up to four sample positions. And the descriptor vector is created from all the orientation histogram entries. For the example shown in Figure 3-4, the descriptor vector has $2\times2\times8=32$ elements. But in real case, 4×4 descriptors are computed from a 16×16 set of samples for best performance.

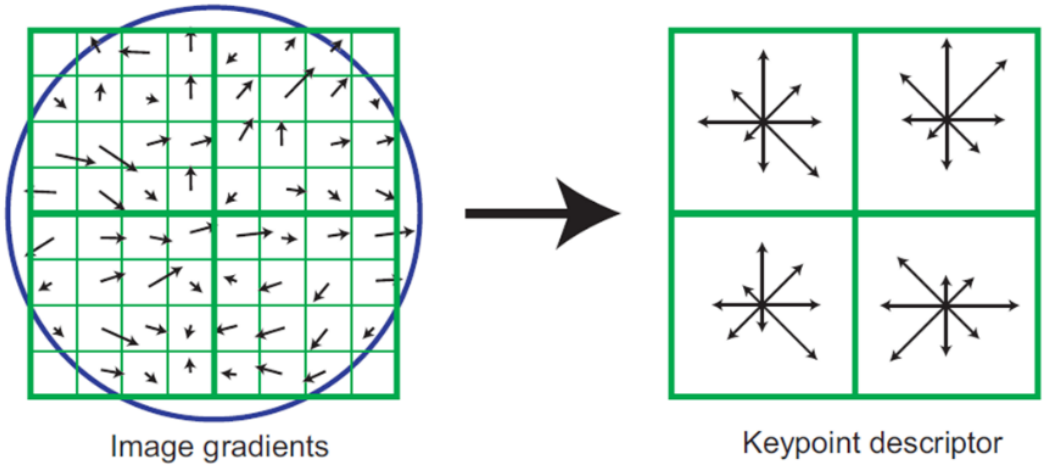


Figure 3-4 Example of 2×2 descriptor array computed from an 8×8 set of samples[102]

Final modifications on the vector are done to reduce effect of illumination change by normalizing the vector to unit length for change of image contrast, and by thresholding the values in the unit vector and renormalize to unit length for non-linear illumination changes.

3.1.3 Matching Keypoints with SIFT Descriptors

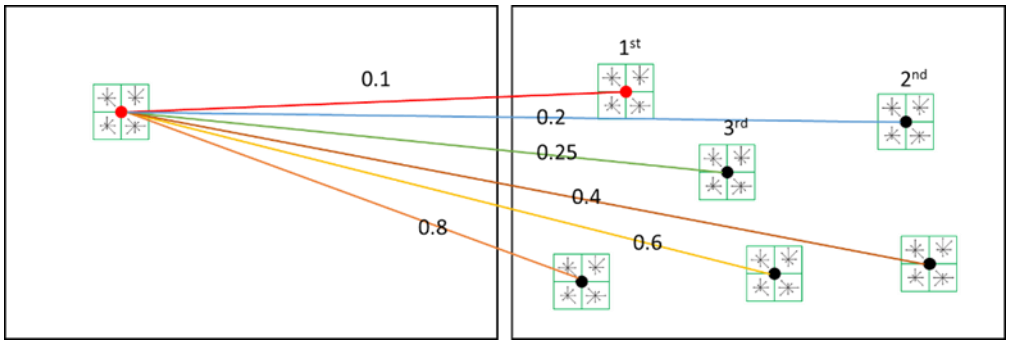


Figure 3-5 Matching of SIFT feature descriptors

The descriptor for each keypoint is a vector and thus the nearest neighbor can be identified with minimum Euclidean distance. But an image could contain many features with no correct matches to those on another image. An effective measure, instead of thresholding distances directly, is to compare the minimum distance to the second smallest distance, which guarantees the best match is significantly closer than the closest incorrect match.

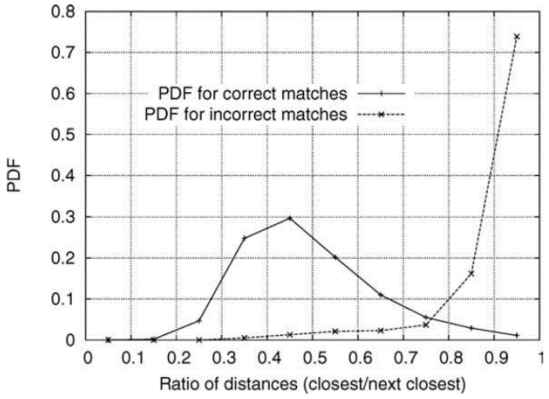


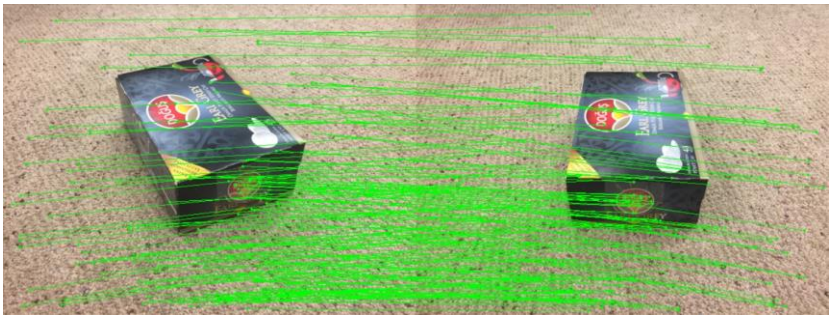
Figure 3-6 Probability of correct match against distance ratio[102]

Figure 3-6 shows that the ratio, if chosen at 0.8, meaning a match is identified when the closest Euclidean distance between the SIFT descriptor with its best match is smaller than 0.8 of that with its second-best match, 90% of false matches can be eliminated while less than 5% of correct ones being discarded.

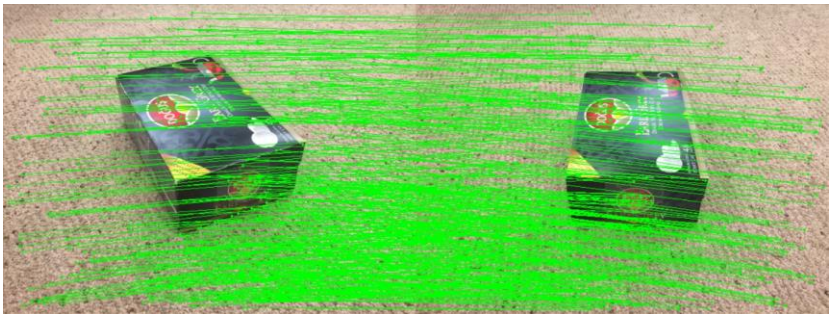
3.1.4 Experiment

To reconstruct a scene of a box placed on the floor, pictures are taken around it from different viewing angles. Figure 3-7 shows the results of matching keypoints using SIFT descriptors at closest-to-next-closest distance ratio 0.6, 0.7, 0.8, and 0.9.

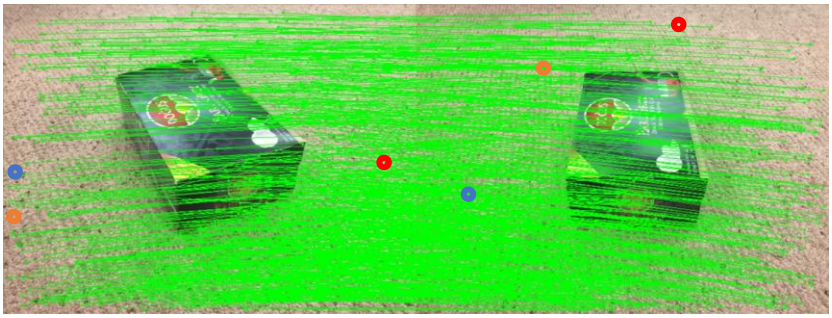
As indicted in 3.1.3, larger ratio results in a greater number of matches. However, Figure 3-7 (c) with ratio at 0.8 starts to show obvious wrong matches (keypoint pairs in colored circles) with the green connecting line going across other lines at a large angle, and much more wrong matches are seen in Figure 3-7 (d) with ratio at 0.9. Thus, for better robustness in later SfM process, we test a list of ratios between 0.65 to 0.75 and define a measurement to decide the best ratio to use for identifying matches between all image pairs. The measurement is also used to decide the initial images to start the incremental reconstruction for better accuracy, which will be illustrated in Section 3.5.2.



(a)



(b)



(c)

(d)

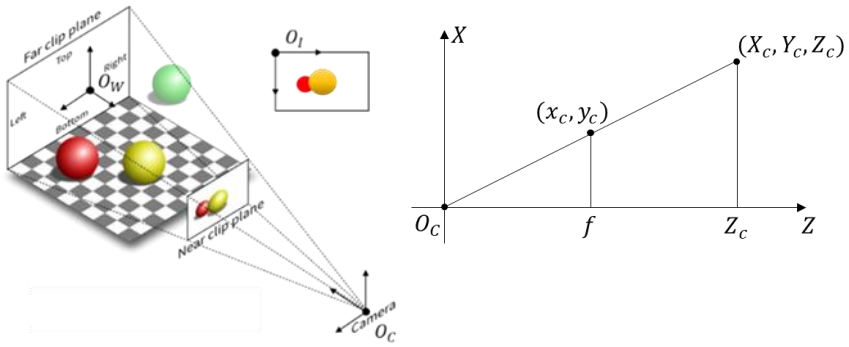
Figure 3-7 Matched features on 2 images of a box from different viewing angles with distance ratio at (a) 0.6, (b) 0.7, (c) 0.8, and (d) 0.9. Number of matches are respectively 176, 414, 689, and 1121.

3.2 Camera Model

The mathematical formulation of the camera model is indeed another foundation of the whole reconstruction process. How a real scene is transformed through the lens and aperture into pixels on an image provides crucial if not the most important model to go backwards and retrieve 3D information from 2D pixels.

3.2.1 Pinhole Camera Model

Most modern camera systems use lenses and apertures to map color information of 3D scenes onto an image sensor and then transform sensor readings into an image with different RGB color at different pixel locations. In the simplest pinhole camera model, we assume that no lens is used, aperture is zero and projection is perspective. The illustration of a pinhole camera model is shown in Figure 3-8. On the left side is the 3D view of a pinhole camera originated at O_C taking a photo of three balls on the ground in world coordinate system originated at O_W. And the final image is on a 2D image plane originated at O_I on its top left corner. And coordinate relations if projected onto XZ plane is shown on the right side.



(a) (b)

Figure 3-8 Pinhole camera model viewed (a) in 3D and (b) in 2D

The perspective projection mapping as shown in Figure 3-8 (b) gives

$$x_C = \frac{f_x}{Z_C} X_C, \quad y_C = \frac{f_y}{Z_C} Y_C \tag{3-2-1}$$

If we express all the points using homogeneous coordinate, then we have

$$\begin{bmatrix} x_C \\ y_C \\ w_C \end{bmatrix} = \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} = \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] \begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} \tag{3-2-2}$$

The matrix $[\mathbf{I} \mid \mathbf{0}]$ added before 3D point coordinates suggests that the world coordinate is now considered as the same as the camera coordinate system. Notice that here all the calculation is done in the camera coordinate system. All the 2D points on the camera sensor plane, $Z = f$, if scaled and shifted to image coordinate, then we have the pixel location of the mapped point on the final image as

$$\begin{bmatrix} x_I \\ y_I \\ w_I \end{bmatrix} = \begin{bmatrix} 1/s_x & 0 & O_x \\ 0 & 1/s_y & O_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_C \\ y_C \\ w_C \end{bmatrix} = \begin{bmatrix} f_x/s_x & 0 & O_x \\ 0 & f_y/s_y & O_y \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] \begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} \tag{3-2-3}$$

where f_x and f_y are the camera focal length (or for a pinhole camera, the distance from pinhole to image sensor), s_x and s_y are the scaling factors, and (O_x, O_y) is coordinates of the image center or the principal point, the point that principal axis (Z axis here) pass through. By defining the intrinsic camera matrix, or camera calibration matrix $\mathbf{K}$ as

$$\mathbf{K} = \begin{bmatrix} f_x/s_x & 0 & O_x \\ 0 & f_y/s_y & O_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3-2-4}$$

Then the equation for a pinhole camera to map any 3D point expressed in the camera coordinate system to a pixel on a 2D image is

$$\mathbf{x} = \mathbf{K}[\mathbf{I} \mid \mathbf{0}]\mathbf{X}_C \tag{3-2-5}$$

If a 3D point is expressed in a world coordinate system that is not the camera coordinate system, then a rotation and translation can be found to transform the point from world coordinate system to camera coordinate system with $\mathbf{P} = [\mathbf{R} \mid \mathbf{T}]$ such that

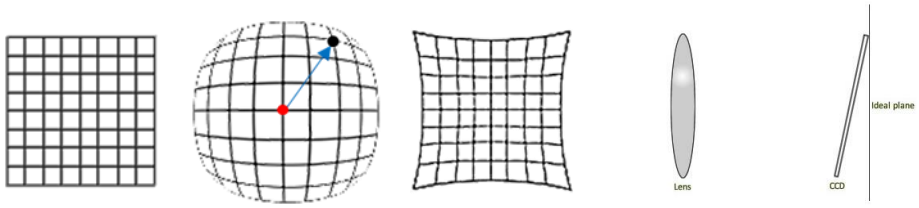
$$\mathbf{X}_C = \mathbf{R}_W^C \mathbf{X}_W + \mathbf{T}_W^C = \left[\mathbf{R}_W^C \mid \mathbf{T}_W^C\right] \begin{bmatrix} \mathbf{X}_W \\ 1 \end{bmatrix} \tag{3-2-6}$$

And thus, we replace the matrix $[\mathbf{I} \mid \mathbf{0}]$ in equation (3-2-5) and obtain the general equation to map any 3D point in any world coordinate system onto a 2D pixel on an image with the pinhole camera model.

$$\mathbf{x} = \mathbf{K}[\mathbf{R} \mid \mathbf{T}]\mathbf{X}_W \tag{3-2-7}$$

And $\mathbf{P} = [\mathbf{R} \mid \mathbf{T}]$ is called the extrinsic camera matrix.

3.2.2 Lens Distortion



(a) (b)

Figure 3-9 (a) Radial distortion (b) Tangential distortion

Two types of lens distortion are usually considered to correct a pinhole camera model, the radial distortion and the tangential distortion. Radial symmetrical distortion is most commonly encountered due to the symmetry of lenses. The example of "Barrel" distortion and "Pincushion" distortion are shown in Figure 3-9 (a). These distortions increase as the square of distance from the center and can be corrected with an even-ordered polynomial model[109]

$$\begin{aligned} x_{corrected} &= x\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \\ y_{corrected} &= y\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \end{aligned} \tag{3-2-8}$$

Tangential distortion, as shown in Figure 3-9 (b), occurs when lens is not perfectly parallel to the imaging plane, and can be modeled with [110]

$$\begin{aligned} x_{corrected} &= x + \left[2p_1 xy + p_2 \left(r^2 + 2x^2 \right) \right] \\ y_{corrected} &= y + \left[2p_2 xy + p_1 \left(r^2 + 2y^2 \right) \right] \end{aligned} \tag{3-2-9}$$

3.3 Camera Calibration

From previous discussion of the camera model and different types of distortion, the purpose of the calibration process of a camera is to find its intrinsic parameters in **K** , namely the focal length f , scaling factors s_x and s_y , and principal point $\left(O_x, O_y\right)$, together with its distortion coefficients k_1, k_2, k_3, p_1, and p_2.

3.3.1 Related Works

The calibration technique estimates the intrinsic parameters of a lens and sensor of a camera. The pinhole camera model used perspective projection with the parameters of focal lengths, optical centers, size of the pixel, and a skew coefficient. And lens distortion model includes parameters of radial distortion coefficients and tangential distortion coefficients. Roughly there are two categories of calibration techniques, photogrammetric calibration and auto-calibration (or self-calibration). Photogrammetric calibration makes use of a calibration object with known geometry and dimension in 3D space. Early approaches used two or three planes orthogonal to each other[111] or a plane with known precise translation as calibration object[112]. Other calibration objects include points on the same line[113], vanishing points of orthogonal directions [114], and the most popular two-dimensional patterns[115] like matrices of circles or chessboard pattern. Auto-calibration determines intrinsic parameters directly from multiple uncalibrated images, first introduced by Maybank and Faugeras [116] where Kruppa equations were used. Later methods include bundle-

adjustment like methods [117, 118], estimating the absolute dual quadric over many views [119], auto-calibration of a stereo rig [120], and stratified approach that first retrieve affine calibration using modulus constraint and then upgrade to Euclidean [121].

3.3.2 Calibrating with Planar 2D Patterns

As reported in [115], the corners of a 2D pattern like a checkerboard placed on a planar surface can be detected and used to estimate intrinsic camera parameters and distortion coefficients. If we assume one of the corners is origin and the pattern plane sits on $Z = 0$ plane, then from the camera projection equation (3-2-7) we have

$$\begin{bmatrix} x_I \\ y_I \\ w_I \end{bmatrix} = \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 \mid \mathbf{t} \end{bmatrix}\begin{bmatrix} X_W \\ Y_W \\ 0 \\ 1 \end{bmatrix} = \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}\begin{bmatrix} X_W \\ Y_W \\ 1 \end{bmatrix} = \mathbf{H}\begin{bmatrix} X_W \\ Y_W \\ 1 \end{bmatrix} \tag{3-3-1}$$

The corner point and its corresponding image point is related by a homography

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix} = \lambda \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \tag{3-3-2}$$

where λ is an arbitrary scaler. For any given 2D pattern, the homography between the model plane and its image can be estimated[115]. Since $\mathbf{r}_1$ and $\mathbf{r}_2$ are from a rotational matrix and thus are orthogonal, i.e., $\mathbf{r}_1^{\mathsf{T}}\mathbf{r}_2 = 0$, and $\mathbf{r}_1^{\mathsf{T}}\mathbf{r}_1 = \mathbf{r}_2^{\mathsf{T}}\mathbf{r}_2$, we have the following 2 constraints given one homography,

$$\begin{aligned} \mathbf{h}_1^{\mathsf{T}}\mathbf{K}^{-\mathsf{T}}\mathbf{K}^{-1}\mathbf{h}_2 &= 0 \\ \mathbf{h}_1^{\mathsf{T}}\mathbf{K}^{-\mathsf{T}}\mathbf{K}^{-1}\mathbf{h}_1 &= \mathbf{h}_2^{\mathsf{T}}\mathbf{K}^{-\mathsf{T}}\mathbf{K}^{-1}\mathbf{h}_2 \end{aligned} \tag{3-3-3}$$

To solve for the parameters in $\mathbf{K}$, a new matrix $\mathbf{B}$ is defined as $\mathbf{B} = \mathbf{K}^{-\mathsf{T}}\mathbf{K}^{-1}$. Then $\mathbf{B}$ is symmetric and can be defined by a vector

$$\mathbf{b} = \begin{bmatrix} B_{11} & B_{12} & B_{22} & B_{13} & B_{23} & B_{33} \end{bmatrix}^{\mathsf{T}} \tag{3-3-4}$$

Thus, we have

$$\mathbf{h}_i^{\mathsf{T}}\mathbf{B}\mathbf{h}_j = \mathbf{v}_{ij}^{\mathsf{T}}\mathbf{b} \tag{3-3-5}$$

where $\mathbf{v}_{ij} = \begin{bmatrix} h_{i1}h_{j1} & h_{i1}h_{j2}+h_{i2}h_{j1} & h_{i2}h_{j2} & h_{i3}h_{j1}+h_{i1}h_{j3} & h_{i3}h_{j2}+h_{i2}h_{j3} & h_{i3}h_{j3} \end{bmatrix}^{\mathsf{T}}$.

Then constraints in (3-3-3) can be rewritten as

$$\begin{bmatrix} \mathbf{v}_{12}^{\mathsf{T}} \\ (\mathbf{v}_{11}-\mathbf{v}_{22})^{\mathsf{T}} \end{bmatrix}\mathbf{b} = \mathbf{V}\mathbf{b} = \mathbf{0} \tag{3-3-6}$$

where $\mathbf{V}$ is a $2n \times 6$ matrix for n number of images. And the solution of it is the eigenvector associated with the smallest eigenvalue of $\mathbf{V}^{\mathsf{T}}\mathbf{V}$. Once $\mathbf{b}$ is obtained, the parameters from intrinsic matrix $\mathbf{K}$ can be uniquely extracted.

3.4 Incremental Reconstruction

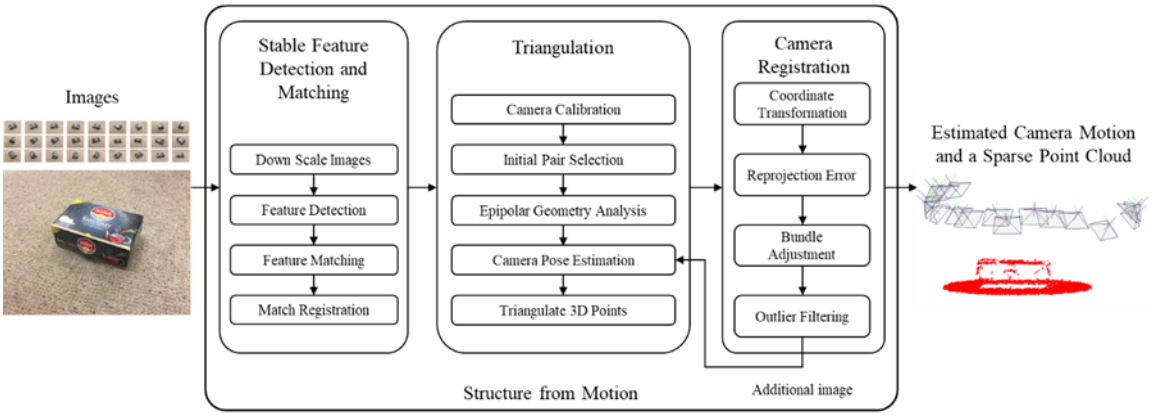


Figure 3-10 Workflow of Incremental Structure from Motion

From previous processes, a sequence of photos taken around the same scene can be related to each other through matched SIFT features. And the intrinsic parameters and distortion coefficients are estimated from calibration. Then the idea for incremental structure from motion is to start from two images to estimate their relative camera locations and poses, and then

incrementally adding more images and recover the locations and poses of them one at a time until all images are processed.

The pipeline of our SfM process is show in Figure 3-10, from which the location and poses of all cameras will be estimated, along with a 3D point cloud that corresponds to strong matching features detected on 2D images.

3.4.1 Related Works

Earliest eight-point algorithm [93] estimates essential matrix from eight epipolar constraints and then solve for translation and rotation from it. This linear method is fast and easy to implement but is extremely susceptible to noise. Thus [122] proposed a normalization on coordinates for 8-point algorithm and performance was improved. However, the error from each image pair pose estimation introduces biases for the entire reconstruction and, if not handled properly, may not be able to recover from bundle adjustment.

A sequential pipeline for SfM was presented in [123]. First, features are extracted and pairwise matched. Then random sampling and consensus (RANSAC) [124] is used to robustly estimate essential matrices between pairs of images, discard outliers, and then estimate pairwise rotation and translation. Then, starting with the pair with the least outliers and adding one image at a time, bundle adjustment [125] is applied repeatedly. However, sequential approaches may rely on the sequence of adding images, and drifting may appear as error building up. Skeletal approaches [126, 127] apply reconstruction on a small subset of images selected by a graph algorithm or a "bag-of-words" approach, and then add remaining images using pose estimation[109]. Pose-graph approaches estimate the absolute poses of each camera that best fits the relative pairwise measurements, where orientation of cameras can be estimated first [128] and then use it to solve for camera locations with least unsquared deviations (LUD) method [129] or ShapeFit [130].

From the extracted feature matches along with estimated camera calibrations and poses, 3D points can be simply triangulated from the corresponding pixels from image pairs. Research work on structure estimation include incremental and global method, and joint structure and motion estimation. Majority of algorithms refine 3D structure, camera poses, and camera calibrations together, known as bundle adjustment, which seeks to minimize or total reprojection error. The nonlinear cost function with large nonlinear parameter space can be minimized with several numerical optimization algorithms [131]. And a Levenberg Marquardt (LM) algorithm-based optimization was proposed in [125] to handle very large dataset. The incremental bundle adjustment methods scale poorly as the number of images grows and can drift or fall into bad local minima. An alternative global approach [132] used a discrete Markov random field (MRF) formulation coupled with a continuous LM refinement to compute an coarse initial estimate of all the camera poses and 3D points, and then improve the solution and solve for a structure using bundle adjustment, which proved to be more robust and efficient with similar or better results.

3.4.2 Epipolar Geometry

The intrinsic projective geometry between two views only depends on camera parameters, both intrinsic and extrinsic, which can be expressed with the fundamental matrix $\mathbf{F}$. Through the following discussion on epipolar geometry, the approach to estimate this intrinsic geometry will be derived and thus lead to the estimation of camera poses.

As shown in Figure 3-11, $\mathbf{x}$ and $\mathbf{x}'$ represent the same 3D point $\mathbf{X}$ from 2 views. These three points and the camera centers are coplanar. The lines connecting $\mathbf{X}$ and the two image points respectively also lie on this plane. The line connecting two camera centers is called baseline. And the intersecting point of the baseline with an image plane is called an epipole, which is also the image of the other camera center. Any plane containing the baseline is called an epipolar plane. And the intersection line of an epipolar plane with an image plane is called the epipolar line for the

corresponding 2D point on the other image. Thus in the example in Figure 3-11, the corresponding point of $\mathbf{x}$ on the second image must lie on the epipolar line $\mathbf{l}'$.

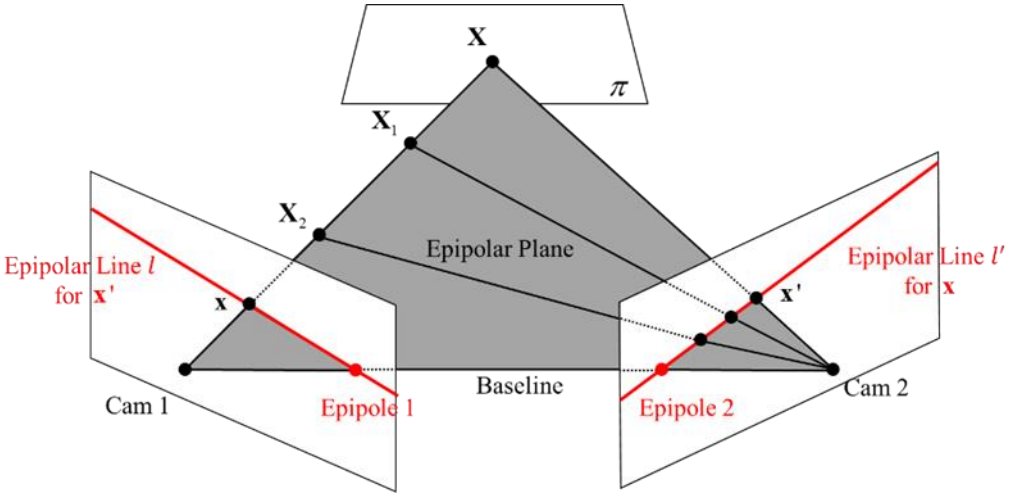


Figure 3-11 Illustration of epipolar geometry

Consider a transfer plane π that does not pass either of the camera centers. The ray connecting first camera center and $\mathbf{x}$ intersect π at $\mathbf{X}$, which can be projected to $\mathbf{x}'$ on the other image located on the epipolar line $\mathbf{l}'$ (suggesting that $\mathbf{x}'^{\mathrm{T}}\mathbf{l}' = 0$). Thus, there exist a homography $\mathbf{H}_\pi$ mapping a set of all such points in the first image and the corresponding points on the other image. i.e., $\mathbf{x}' = \mathbf{H}_\pi \mathbf{x}$. And the line $\mathbf{l}'$ in homogeneous coordinates can be calculated from $\mathbf{l}' = \mathbf{e}' \times \mathbf{x}' = [\mathbf{e}']_\times \mathbf{x}'$, then we have $\mathbf{x}'^{\mathrm{T}}[\mathbf{e}']_\times \mathbf{x}' = \mathbf{x}'^{\mathrm{T}}[\mathbf{e}]_\times \mathbf{H}_\pi \mathbf{x} = 0$. By defining the fundamental matrix $\mathbf{F} = [\mathbf{e}]_\times \mathbf{H}_\pi$, we obtain

$$\mathbf{x}'^{\mathrm{T}}\mathbf{F}\mathbf{x} = 0 \tag{3-4-1}$$

The fundamental matrix $\mathbf{F}$ is a 3×3 homogeneous matrix with 7 degrees of freedom (8 for any homogeneous matrix since scale is irrelevant and $\det(\mathbf{F}) = 0$ removes one). The plane π is not required for $\mathbf{F}$ to exist. The equation (3-4-1) is the condition for any pair of correspondents in two images.

Also. from the definition of $\mathbf{F}$, we obtain the equations to calculate the epipoles and epipolar lines for each 2D point as follows,

$$\begin{aligned} \mathbf{l} = \mathbf{x}'^{\mathsf{T}}\mathbf{F}, \quad \mathbf{l}' = \mathbf{F}\mathbf{x} \\ \mathbf{F}\mathbf{e} = \mathbf{0} = \mathbf{F}^{\mathsf{T}}\mathbf{e}' \end{aligned} \tag{3-4-2}$$

The estimation of $\mathbf{F}$ can be done with the matched 2D points from 2 images. A term by term multiplication of equation (3-4-1) can be derived and simplified considering the one degree of freedom removed by scaling as

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} = -1 \tag{3-4-3}$$

And if n pair of correspondents are given, we have

$$\begin{bmatrix} x_1'x_1 & x_1'y_1 & x_1' & y_1'x_1 & y_1'y_1 & y_1' & x_1 & y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n'x_n & x_n'y_n & x_n' & y_n'x_n & y_n'y_n & y_n' & x_n & y_n \end{bmatrix} \begin{bmatrix} f_{11} \\ \vdots \\ f_{32} \end{bmatrix} = \begin{bmatrix} -1 \\ \vdots \\ -1 \end{bmatrix} \tag{3-4-4}$$

From equation (3-4-4), all the elements of the fundamental matrix can be estimated with more than 8 pairs of corresponding points from two images using a maximum likelihood estimator or the RANSAC algorithm.

If the intrinsic camera matrix $\mathbf{K}$ is known, then the image point expressed in normal coordinates, $\hat{\mathbf{x}} = \mathbf{K}^{-1}\mathbf{x}$, can be inserted into the perspective camera mapping equation (3-2-7) and result in $\hat{\mathbf{x}} = [\mathbf{R} \mid \mathbf{T}]\mathbf{X}_W$. Here the $[\mathbf{R} \mid \mathbf{T}]$, which is the extrinsic camera matrix, is also called the normalized camera matrix. And the fundamental matrix corresponding to a pair of normalized cameras is called the essential matrix $\mathbf{E}$, defined similarly with $\mathbf{F}$ by

$$\hat{\mathbf{x}}'^{\mathsf{T}}\mathbf{E}\hat{\mathbf{x}} = 0 \tag{3-4-5}$$

Inserting $\hat{\mathbf{x}} = \mathbf{K}^{-1}\mathbf{x}$ and $\hat{\mathbf{x}}' = \mathbf{K}'^{-1}\mathbf{x}'$ into equation (3-4-5), we obtain the relation between fundamental matrix and essential matrix as

$$\mathbf{E} = \mathbf{K}'^{\mathsf{T}}\mathbf{F}\mathbf{K} \tag{3-4-6}$$

Essential matrix $\mathbf{E}$ has the form $\mathbf{E} = [\mathbf{T}]_{\times}\mathbf{R}$ and has 5 degrees of freedom (3 for both rotation and translation, but one removed due to scaling). From essential matrix, the camera extrinsic matrix can be estimated up to scale and a four-fold ambiguity, and thus it can be used to estimate the relative poses of cameras.

In contrast, the fundamental matrix $\mathbf{F}$ only represents the projective relationship between two cameras. If rewriting $\mathbf{PX} = \mathbf{PH}\left(\mathbf{H}^{-1}\mathbf{X}\right)$, it can be derived that pairs of extrinsic matrices, if right multiplied by a projective transformation matrix $\mathbf{H}$, share the same fundamental matrix $\mathbf{F}$. Extrinsic camera matrix can indeed be retrieved from $\mathbf{F}$, but only up to right multiplication by a projective transformation. So, in our SfM pipeline, camera calibration matrix $\mathbf{K}$ and fundamental matrix $\mathbf{F}$ are first estimated and then use calculated essential matrix $\mathbf{E}$ to retrieve the camera locations and poses.

3.4.3 Estimation of Extrinsic Parameters

Assume the first camera has extrinsic matrix $\mathbf{P_0} = [\mathbf{I} \mid \mathbf{0}]$. Then the goal here is to estimate second camera extrinsic matrix $\mathbf{P}_0^1 = [\mathbf{R}_0^1 \mid \mathbf{T}_0^1]$. If enough 3D points expressed in the first camera coordinate system have been reconstructed, location and pose of the second one can be estimated from an analysis on the 3D-to-2D projection equation, which will be discussed in 3.4.5. However, at the starting point of reconstruction, no 3D points had been estimated yet. But since the essential matrix has the form $\mathbf{E} = [\mathbf{T}]_{\times}\mathbf{R}$, by decomposing it the extrinsic parameters can be estimated.

It can be proved [109] that a 3×3 matrix is an essential matrix if and only if two of its singular values are equal and the third one is zero with the help of two matrices,

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \mathbf{Z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{3-4-7}$$

where $\mathbf{W}$ is orthogonal and $\mathbf{Z}$ is skew-symmetric. Thus, the singular value decomposition of an essential matrix can be written as $\mathbf{E} = \mathbf{U}\,\mathrm{diag}(1,1,0)\mathbf{V}^{\mathsf{T}}$. Using the matrices $\mathbf{W}$ and $\mathbf{Z}$, it can also be proved that there are up to sign two possible factorization of $\mathbf{E} = [\mathbf{T}]_{\times}\mathbf{R}$,

$$\begin{aligned} [\mathbf{T}]_{\times} &= \mathbf{U}\mathbf{Z}\mathbf{U}^{\mathsf{T}} \\ \mathbf{R} &= \mathbf{U}\mathbf{W}\mathbf{V}^{\mathsf{T}} \text{ or } \mathbf{U}\mathbf{W}^{\mathsf{T}}\mathbf{V}^{\mathsf{T}} \end{aligned} \tag{3-4-8}$$

From the first equation we get up to scale $\mathbf{T} = \mathbf{U}\begin{bmatrix}0 & 0 & 1\end{bmatrix}^{\mathsf{T}} = \mathbf{u}_3$, the third column of $\mathbf{U}$. Thus, there are in total four possible solutions of the other camera $\mathbf{P}_0^1$,

$$\begin{aligned} \mathbf{P}_0^1 &= [\mathbf{U}\mathbf{W}\mathbf{V}^{\mathsf{T}} \mid \mathbf{u}_3] \\ \text{or } \mathbf{P}_0^1 &= [\mathbf{U}\mathbf{W}\mathbf{V}^{\mathsf{T}} \mid -\mathbf{u}_3] \\ \text{or } \mathbf{P}_0^1 &= [\mathbf{U}\mathbf{W}^{\mathsf{T}}\mathbf{V}^{\mathsf{T}} \mid \mathbf{u}_3] \\ \text{or } \mathbf{P}_0^1 &= [\mathbf{U}\mathbf{W}^{\mathsf{T}}\mathbf{V}^{\mathsf{T}} \mid -\mathbf{u}_3] \end{aligned} \tag{3-4-9}$$

The sign of $\mathbf{T} = \pm\mathbf{u}_3$ defines the translation direction of the second camera. And for the rotational part, by comparing the first and the third candidate, it can be verified that

$$\begin{aligned} [\mathbf{U}\mathbf{W}^{\mathsf{T}}\mathbf{V}^{\mathsf{T}} \mid \mathbf{u}_3] &= [\mathbf{U}\mathbf{W}\mathbf{V}^{\mathsf{T}} \mid \mathbf{u}_3]\begin{bmatrix}\mathbf{V}\mathbf{W}^{\mathsf{T}}\mathbf{W}^{\mathsf{T}}\mathbf{V}^{\mathsf{T}} & \mathbf{0} \\ \mathbf{0} & 1\end{bmatrix} \\ &= [\mathbf{U}\mathbf{W}\mathbf{V}^{\mathsf{T}} \mid \mathbf{u}_3]\begin{bmatrix}\mathbf{V}\,\mathrm{diag}(-1,-1,1)\mathbf{V}^{\mathsf{T}} & 0 \\ 0 & 1\end{bmatrix} \end{aligned} \tag{3-4-10}$$

where $\mathbf{V}\,\mathrm{diag}(-1,-1,1)\mathbf{V}^{\mathsf{T}}$ stands for a rotation about the baseline for $180°$. The difference of the four candidates are shown in Figure 3-12. And only case (a) results in all the reconstructed 3D points in front of both the cameras. This can be used to select the correct estimation by triangulate any pair of 2D points into a 3D point and inspect the sign of depth values in both cameras.

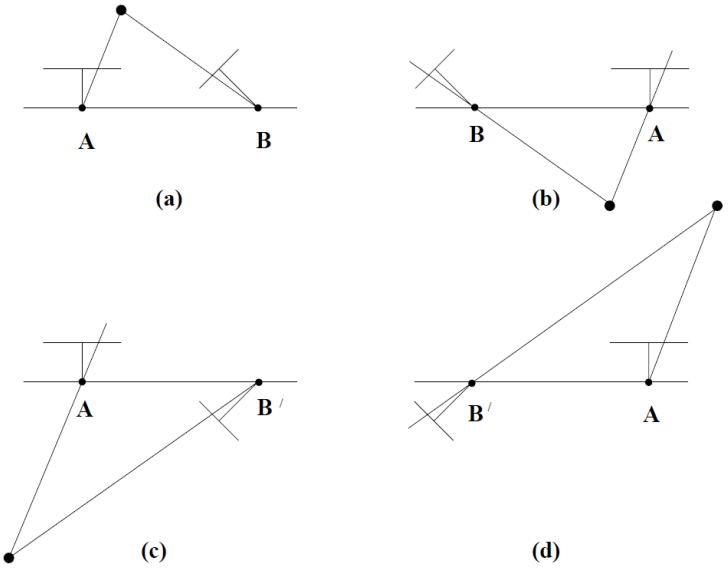


Figure 3-12 Possible solutions of camera location and pose estimated from essential matrix[109]

However, in practice, since the fundamental matrix as well as the intrinsic parameters are estimated with errors, a singular value decomposition of essential matrix often results in different singular values. The solution is to manually set them to be equal, recalculate the essential matrix back, and redo SVD on the newly obtained essential matrix for $\mathbf{U}$ and $\mathbf{V}$. Moreover, wrong match still exists and as a result, the false extrinsic matrices also generate some points in front of both cameras. The solution is to triangulate all the points with all the $\mathbf{P}_0^1$ candidates and choose the set that generates positive value for the greatest number of 3D points.

3.4.4 Triangulation

Once the intrinsic and extrinsic camera matrices are estimated, the basic perspective mapping equation (3-2-7) has only 3D coordinates as unknowns. In a normalized coordinate system, where $\hat{\mathbf{x}} = \mathbf{K}^{-1}\mathbf{x}$, for any 2D point in the i-th image, we have

$$\hat{\mathbf{x}}^i = \mathbf{X}^i = \mathbf{P}_0^i \mathbf{X}^0 \tag{3-4-11}$$

where the superscript 0 indicate that the 3D point is expressed in the first camera coordinate system. In order to solve for $\mathbf{X}^0$, the equation can be reformulated into an $\mathbf{A}\mathbf{X}^0 = \mathbf{B}$

type of equation by transforming the equivalence of two vectors into a cross product equals to zero form as shown below.

$$\hat{\mathbf{x}}^i \times \mathbf{P}_0^i \mathbf{X}^0 = \left[\hat{\mathbf{x}}^i\right]_\times \left[\mathbf{p}_0^{i\,1\mathrm{T}} \quad \mathbf{p}_0^{i\,2\mathrm{T}} \quad \mathbf{p}_0^{i\,3\mathrm{T}}\right]^\mathrm{T} \mathbf{X}^0 = \mathbf{0}$$

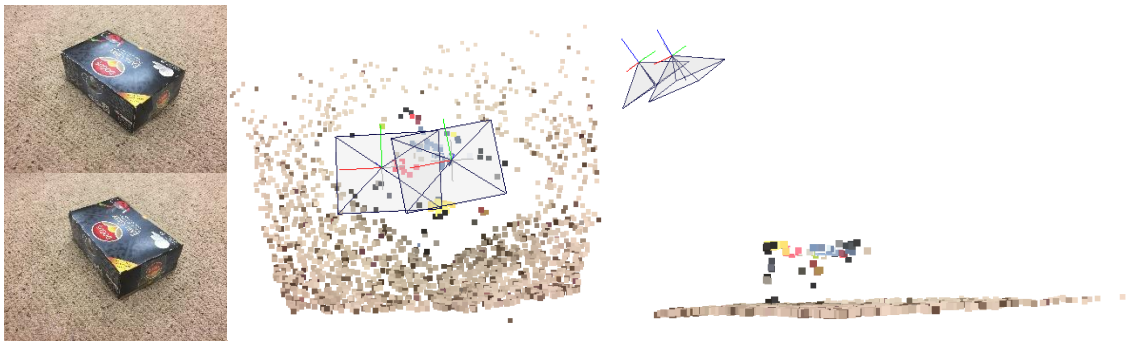
$$\Rightarrow \begin{bmatrix} 0 & -1 & \hat{y}^i \\ 1 & 0 & -\hat{x}^i \\ -\hat{y}^i & \hat{x}^i & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0^{i\,1\mathrm{T}} \\ \mathbf{p}_0^{i\,2\mathrm{T}} \\ \mathbf{p}_0^{i\,3\mathrm{T}} \end{bmatrix} \begin{bmatrix} X^0 \\ Y^0 \\ Z^0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3-4-12}$$

where $\hat{\mathbf{x}}^i = \left[\hat{x}^i \quad \hat{y}^i \quad 1\right]^\mathrm{T}$ and $\mathbf{p}_0^{i\,j\mathrm{T}}$ stands for the j-th row of $\mathbf{P}_0^i$. Factoring out the equation (3-4-12) the following equation can be obtained.

$$\begin{bmatrix} \hat{x}^i \mathbf{p}_0^{i\,3\mathrm{T}} - \mathbf{p}_0^{i\,1\mathrm{T}} \\ \hat{y}^i \mathbf{p}_0^{i\,3\mathrm{T}} - \mathbf{p}_0^{i\,2\mathrm{T}} \end{bmatrix} \mathbf{X}^0 = \mathbf{0} \tag{3-4-13}$$

The reason that only two equations are left is that the third one is a linear combination of the first two. Thus, for every pair of matched features, e.g. $\mathbf{x}^0$ and $\mathbf{x}^1$, there are 4 linear equations and thus can be used to triangulate the 3D coordinates with a maximum likelihood estimator.

For the following two images in Figure 3-13 (a), the estimated camera locations, poses, and triangulated 3D points are shown in (b) and (c). It can be seen in (c) that surfaces and edge locations are relatively correct, so are the camera locations and poses.



(a) (b) (c)

Figure 3-13 Resulting camera locations, poses, and the triangulated 3D points from 2 images

3.4.5 Camera Registration

From the previous discussion in this chapter, the steps to apply reconstruction from two images of the same scene (assuming same camera is used) are

(1) Down-sample all the pictures taken at the same scene;

(2) Use pictures of planar 2D pattern, down-sampled to the same size, to estimate intrinsic camera parameters in $\mathbf{K}$ and distortion coefficients;

(3) Apply distortion correction on all images;

(4) Detect SIFT features in each image and find matches;

(5) Insert point coordinates into equation (3-4-4) to construct $\mathbf{A}$ such that $\mathbf{Af} = \mathbf{1}$;

(6) Estimate entries of fundamental matrix $\mathbf{F}$ from matched feature pairs by solving $\mathbf{Af} = \mathbf{1}$ with RANSAC method to exclude outliers;

(7) Calculate essential matrix $\mathbf{E}$ from $\mathbf{E} = \mathbf{K}^T\mathbf{F}\mathbf{K}$;

(8) Perform SVD on $\mathbf{E}$ so that $\mathbf{E} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$.

(9) Recalculate $\mathbf{E} = \mathbf{U}\ \mathrm{diag}(1,1,0)\mathbf{V}^T$ and then perform SVD on new $\mathbf{E}$ to get new $\mathbf{U}$ and $\mathbf{V}^T$;

(10) Use equation (3-4-9) to calculate 4 candidate extrinsic camera matrices $\mathbf{P}_0^1$;

(11) For every pair of match, construct $\mathbf{A}$ from equation (3-4-13) such that $\mathbf{A}\mathbf{X}^0 = \mathbf{0}$. Perform SVD on $\mathbf{A}$ and 3D coordinates is the right null-space of $\mathbf{A}$.

If more than two pictures are taken, between each two adjacent images their relative position and pose can be estimated and a 3D point cloud corresponding to the SIFT feature matches can be generated. However, since from essential matrix the estimation of extrinsic camera matrix is only up to scale, simply merging 3D point clouds into one coordinate system with $\mathbf{P}_0^1$ does not give well-aligned results.

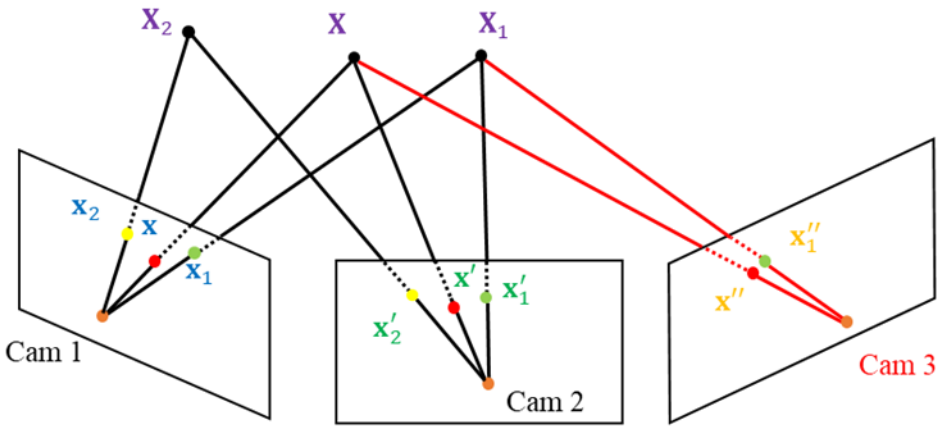


Figure 3-14 Demonstration of using a shared subset of previous 2D-3D pairs to estimate location and pose of the third camera

Another way to obtain the third camera matrix is by estimating it directly from the 2D-3D point pairs. From the first two images, a set of $\mathbf{x}_{01}^{0} \leftrightarrow \mathbf{x}_{01}^{1} \leftrightarrow \mathbf{X}_{01}^{0}$ is estimated. And from the second and the adjacent third image we can obtain matches $\mathbf{x}_{12}^{1} \leftrightarrow \mathbf{x}_{12}^{2}$, in which a subset $\mathbf{x}_{012}^{1} \leftrightarrow \mathbf{x}_{012}^{2}$ can be found that corresponds to $\mathbf{x}_{012}^{0}$ in the first image, a subset of $\mathbf{x}_{01}^{0}$, and represent the same 3D points $\mathbf{X}_{012}^{0}$, a subset of $\mathbf{X}_{01}^{0}$. As a result, a set of 2D-3D pairs $\mathbf{x}_{012}^{2} \leftrightarrow \mathbf{X}_{012}^{0}$ is obtained. The points relations are depicted in Figure 3-14.

From basic perspective mapping equation (3-2-7), we have the following equation that has all extrinsic parameters as unknowns.

$$\mathbf{x}_{012}^{2} = \mathbf{K}\mathbf{X}_{012}^{2} = \mathbf{K}\left[\mathbf{R}_{0}^{2} \mid \mathbf{T}_{0}^{2}\right]\mathbf{X}_{012}^{0} \tag{3-4-14}$$

We can use the same procedure as in the derivation of triangulation. But from equation (3-4-12) we factor out the equation and reformat it into another linear equation by setting extrinsic parameters as unknowns, which is

$$\begin{bmatrix} X^0 & Y^0 & Z^0 & 1 & 0 & 0 & 0 & 0 & -\hat{x}^i X^0 & -\hat{x}^i Y^0 & -\hat{x}^i Z^0 & -\hat{x}^i \\ 0 & 0 & 0 & 0 & X^0 & Y^0 & Z^0 & 1 & -\hat{y}^i X^0 & -\hat{y}^i Y^0 & -\hat{y}^i Z^0 & -\hat{y}^i \end{bmatrix} \begin{bmatrix} \mathbf{p}_0^{i1} \\ \mathbf{p}_0^{i2} \\ \mathbf{p}_0^{i3} \end{bmatrix}_{12\times 1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3-4-15}$$

where $\left[\mathbf{p}_0^{i1} \quad \mathbf{p}_0^{i2} \quad \mathbf{p}_0^{i3}\right]^{\mathsf{T}}$ is a vector composed of all the 12 entries of the extrinsic camera matrix, 9 for rotation and 3 for translation. Similarly, every 2D-3D pair provides 2 linear equations and thus with more than $5\frac{1}{2}$ such pairs the extrinsic parameters can be estimated with a maximum likelihood estimator up to scale.

In practice, all the shared matches are used for the estimation. But equation (3-4-15) is modified by setting the last entry of rotational part to be 1 and transform it into $\mathbf{A}\mathbf{p}_0^i = \mathbf{B}$, i.e.

$$\begin{bmatrix} X^0 & Y^0 & Z^0 & 1 & 0 & 0 & 0 & 0 & -\hat{x}^i X^0 & -\hat{x}^i Y^0 & -\hat{x}^i Z^0 \\ 0 & 0 & 0 & 0 & X^0 & Y^0 & Z^0 & 1 & -\hat{y}^i X^0 & -\hat{y}^i Y^0 & -\hat{y}^i Z^0 \end{bmatrix} \begin{bmatrix} {}_0^i r_{11} \\ {}_0^i r_{12} \\ {}_0^i r_{13} \\ {}_0^i t_1 \\ {}_0^i r_{21} \\ {}_0^i r_{22} \\ {}_0^i r_{23} \\ {}_0^i t_2 \\ {}_0^i r_{31} \\ {}_0^i r_{32} \\ {}_0^i t_3 \end{bmatrix} = \begin{bmatrix} \hat{x}^i \\ \hat{y}^i \end{bmatrix} \quad \text{(3-4-16)}$$

The stacked matrices $\mathbf{A}$ and $\mathbf{B}$ can then be used to obtain the 11 parameters.

By projecting reconstructed 3D points back to new 2D points, the estimated camera location and pose now produce new 3D points in the same scale of previous results. However, the estimated parameters are not eligible yet to be used in the SfM pipeline. If rearranged directly into an extrinsic matrix, newly triangulated 3D points from the second and third image can be merged with the results from the first pair of images easily by transforming them using the calculated $\mathbf{P}_0^2$. But if apply the same procedure to extend to more pictures, as shown in Figure 3-15, we start to see wrong estimation of cameras and distorted reconstructed 3D point clouds, which cannot be merged well with previous data. This drifting effect is fatal to our incremental SfM pipeline, and the reasons and solutions will be discussed in the following section.

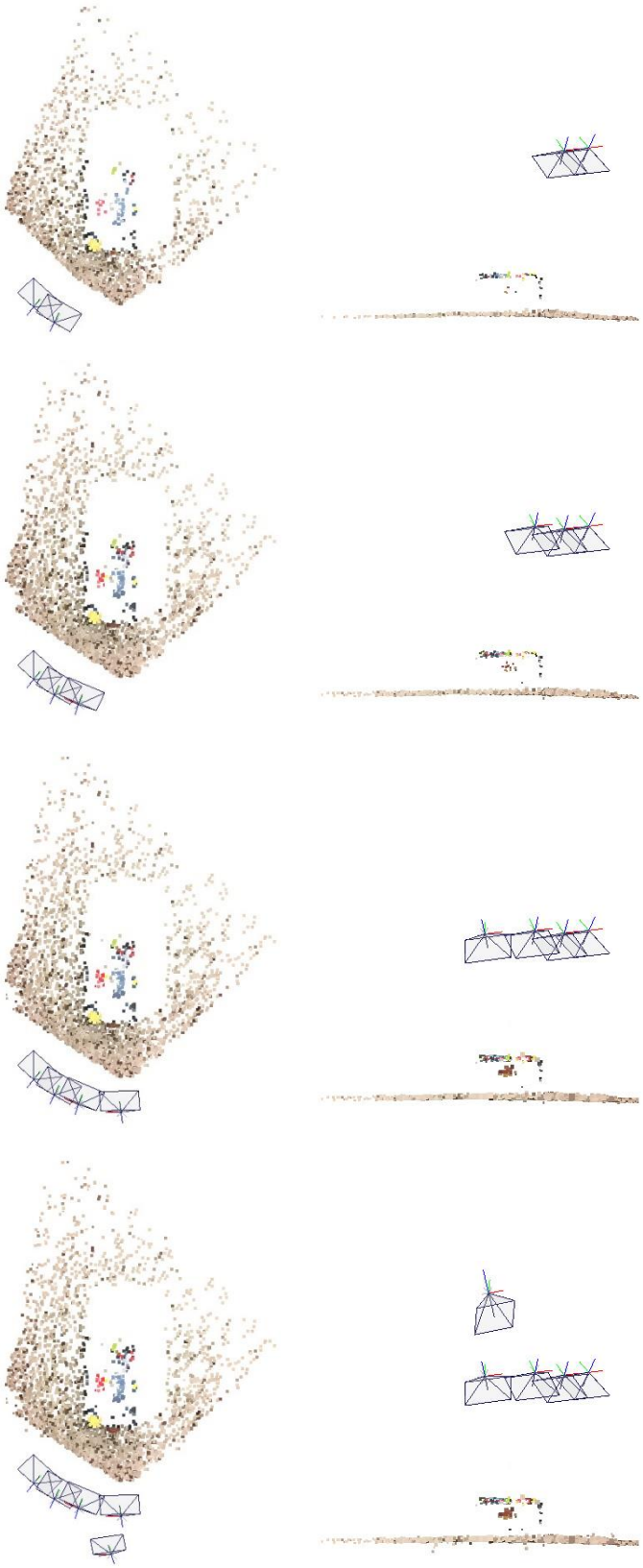


Figure 3-15 Reconstruction results with five images with additional camera solved by projection. Obvious errant camera and points shown in fifth reconstruction

3.5 Drifting

From the reconstruction results following steps discussed in the previous sections, it is apparent to see the trend of increasing level of distortion coming with more images added into the workflow. The reason can be traced back all the way to feature detection and matching between images, but in fact every step of estimation is calling in errors and the accumulation of it is doomed to have later point clouds drifting away from the previous clouds. An example of drifting in 3-image reconstruction is shown below in Figure 3-16.

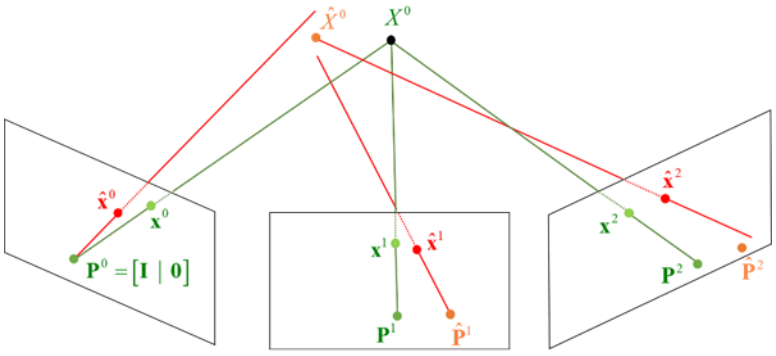


Figure 3-16 Drifting caused by mis-match and errors in reconstruction. Real 3D points, camera locations and poses, and projected 2D points locations are shown in green. Estimated features, cameras, and 3D points are shown in red and orange, where orange indicates final results of a three-image reconstruction.

In order to reduce the effect of drifting and try to reach a more robust structure, it is necessary to use more robust methods for estimation to reduce the impact of mis-matching and apply correction at the end of it. In this section, we discuss the measures taken so that the SfM pipeline can propagate with added images and still result in well-aligned point clouds.

3.5.1 RANSAC in Estimation

In the SfM pipeline, SIFT features from images are taken as measurements and thus will not be corrected. But the outliers (wrong matches) in them can be filtered out so that only pairs that are good enough will be used for estimation. The most commonly used approach is through Random Sample Consensus (RANSAC) algorithm[124].

The RASAC algorithm randomly and uniformly samples a number of elemental subsets that contains minimal number of measurements, from which an estimation can be applied to generate a structure. Within certain range of an estimated structure, number of measurements can be counted and the structure that includes the greatest number of measurements will be chosen as the inlier structure, and then all the outliers outside of the range are discarded. The number of elemental subsets to sample can be calculated with a probabilistic model shown in the following equation so that with probability p, e.g. $p = 0.95$, at least one random sample is free from outliers.

$$N = \frac{\log(1-p)}{\log\left[1-(1-e)^{s}\right]} \tag{3-5-1}$$

where e is probability that a measurement is and outlier, s is the number of measurements in a sample, N is the number of samples to calculate, and p is the desired probability that we get a good sample.

For every estimation in the SfM pipeline where the 2D image features are used, e.g. the estimations of fundamental matrix $\mathbf{F}$, RANSAC can be applied in the final maximum likelihood estimation.

3.5.2 Initial Pair Selection

From a series of pictures taken around a scene, two images need to be selected to initiate the reconstruction pipeline. However, a selection of these two images could be fatal to the later incremental reconstruction if it results in a structure with large error. One could use the pair with the greatest number of matches as initialization. But in our case, for every pair after extracted features are matched, all the steps for two-image reconstruction are conducted and obtain the two camera locations and a 3D point cloud, during which the feature outliers are detected and are not used in the estimation. Then all the inliers and outliers are triangulated into 3D points and then

reprojected back to 2D pixels and evaluate the pixel difference with original features. The pair that returns the least mean reprojected pixel difference is then selected as the initial pair for the pipeline.

The following figures show a comparison of reconstruction from two good first selection and poor ones, along with the mean of L-2 norm of pixel-wise reprojection error for each pair.

(a) (b) (c)

error = 0.553 error = 12.48 error=5.59

(d) (e) (f)

Figure 3-17 (a), (b), (c) Original 3 images of a box on the ground. (d) Reconstruction from first and second image, and (e) from first and third image, and (f) from second and third image

3.5.3 Correction on Extrinsic Matrix

As illustrated in 3.4.5, the estimated vector is not ready yet to be used directly as entries of the extrinsic matrix. The main reason comes from the constraints of a rotational matrix, i.e.,

$$\mathbf{R}^{\mathsf{T}} = \mathbf{R}^{-1}, \quad \text{and} \quad \det(\mathbf{R}) = \pm 1 \tag{3-5-2}$$

Assuming the rearranged matrix is $\left[\hat{\mathbf{R}} \mid \hat{\mathbf{T}}\right]$, the correction for the determinant constraint can be easily applied with normalization with

$$\left[\bar{\mathbf{R}} \mid \bar{\mathbf{T}}\right] = \frac{1}{\sqrt[3]{\det\left(\hat{\mathbf{R}}\right)}}\left[\hat{\mathbf{R}} \mid \hat{\mathbf{T}}\right] \tag{3-5-3}$$

However, the vector, if not using rotational angle and axis format or quaternion format and estimated using a nonlinear optimizer, it is not guaranteed that the output forms an orthogonal matrix.

To address this issue, we use the following steps to correct our estimation, assuming that the obtained $\left[\bar{\mathbf{R}} \mid \bar{\mathbf{T}}\right]$ is close to the true extrinsic matrix in its parameter space:

(1) Find the closest rotational matrix $\tilde{\mathbf{R}}$ from $\bar{\mathbf{R}}$ with the following constructed matrix $\mathbf{Q}$ [133],

$$\mathbf{Q} = \frac{1}{3}\begin{bmatrix} r_{11}-r_{22}-r_{33} & r_{21}+r_{12} & r_{31}+r_{13} & r_{23}-r_{32} \\ r_{21}+r_{12} & r_{22}-r_{11}-r_{33} & r_{23}+r_{32} & r_{31}-r_{13} \\ r_{31}+r_{13} & r_{23}+r_{32} & r_{33}-r_{11}-r_{22} & r_{12}-r_{21} \\ r_{23}-r_{32} & r_{31}+r_{13} & r_{12}-r_{21} & r_{11}+r_{22}+r_{33} \end{bmatrix} \tag{3-5-4}$$

and the eigenvector of $\mathbf{Q}$ that corresponds to the eigenvalue 1 is the sought quaternion of $\tilde{\mathbf{R}}$.

(2) Transform the quaternion expression into rotational matrix representation. And plug the rotational entries back into equation (3-4-16) and use least square method to obtain new translation vector $\begin{bmatrix}\tilde{t}_1 & \tilde{t}_2 & \tilde{t}_3\end{bmatrix}^{\mathsf{T}}$. Now the new extrinsic matrix becomes $\left[\tilde{\mathbf{R}} \mid \tilde{\mathbf{T}}\right]$.

(3) Transform the new rotational matrix into angle-axis representation. And use the obtained angel , axis, and translation vector as initial condition and solve equation (3-4-15) expressed with functions of angle θ and axis $\begin{bmatrix}u_x & u_y & u_z\end{bmatrix}^{\mathsf{T}}$ as shown in the following equation with a nonlinear solver to minimize the norm of the left production.

$$\begin{bmatrix} X^0 & Y^0 & Z^0 & 1 & 0 & 0 & 0 & 0 & -\hat{x}^0X^0 & -\hat{x}^0Y^0 & -\hat{x}^0Z^0 & -\hat{x}^0 \\ 0 & 0 & 0 & 0 & X^0 & Y^0 & Z^0 & 1 & -\hat{y}^0X^0 & -\hat{y}^0Y^0 & -\hat{y}^0Z^0 & -\hat{y}^0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X^0 & Y^0 & Z^0 & 1 & 0 & 0 & 0 & 0 & -\hat{x}^nX^0 & -\hat{x}^nY^0 & -\hat{x}^nZ^0 & -\hat{x}^n \\ 0 & 0 & 0 & 0 & X^0 & Y^0 & Z^0 & 1 & -\hat{y}^nX^0 & -\hat{y}^nY^0 & -\hat{y}^nZ^0 & -\hat{y}^n \end{bmatrix} \begin{bmatrix} \cos\theta + (1-\cos\theta)u_x^2 \\ -u_z\sin\theta + (1-\cos\theta)\,u_xu_y \\ u_y\sin\theta + (1-\cos\theta)u_xu_z \\ \tilde{t}_1 \\ u_z\sin\theta + (1-\cos\theta)u_xu_y \\ \cos\theta + (1-\cos\theta)\,u_y^2 \\ -u_x\sin\theta + (1-\cos\theta)u_yu_z \\ \tilde{t}_2 \\ -u_y\sin\theta + (1-\cos\theta)u_xu_z \\ u_x\sin\theta + (1-\cos\theta)u_yu_z \\ \cos\theta + (1-\cos\theta)u_z^2 \\ \tilde{t}_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ 0 \end{bmatrix} \quad (3\text{-}5\text{-}5)$$

(4) Convert obtained angle and axis back into rotational matrix form and combine it with optimized translation vector to form the final estimation of extrinsic camera matrix.

The process guaranteed the estimated $\mathbf{P} = [\mathbf{R} \mid \mathbf{T}]$ has its rotational part meeting the criteria of being a rotational matrix. And also, the initial condition is set up to be close to the real solution thus avoiding converging at local minimum during optimization.

We apply the correction at the same reconstruction illustrated at the end of section 3.4.5. The new results are shown in Figure 3-18. The location and pose of the fifth camera are relatively correct compared with previous results. However, the consequence of adjustment on camera matrix alone will cause shift on triangulated 3D points and thus it is observed that the drifting of 3D point cloud is even more obvious. And this drifting is going to be enlarged with more images added into the pipeline. And thus, a more holistic correction is needed to adjust more parameters at the same time, which will be discussed in the next section.

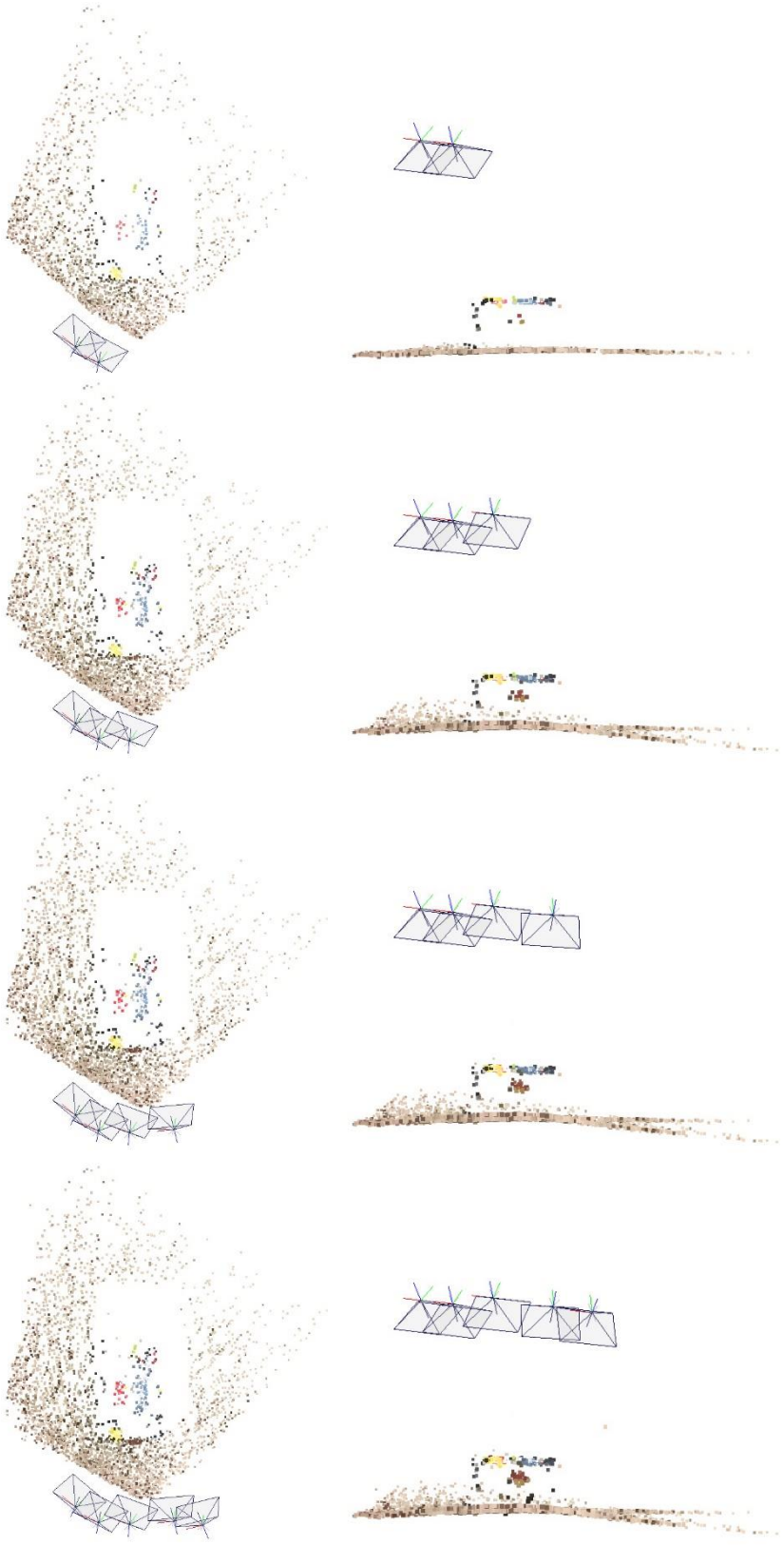


Figure 3-18 Reconstruction with 5 images using correction of camera matrix for every additional one.

3.5.4 Bundle Adjustment

Consider that in the perspective mapping equation, the 2D features are corrupted by noise, and all other three part, $\mathbf{K}$, $\mathbf{P}$, and $\mathbf{X}$ are estimated from total least square. The correction on $\mathbf{P}$ only addresses the constraint issue and seeks for a constrained solution with corrupted features and estimated $\mathbf{K}$ and $\mathbf{X}$, not mentioning that it is only applied on additional cameras beyond the first two. As the SfM process propagates, drifting may be negligible for the first a few images, but will soon become a fatal issue because from 2D-3D pairs with poor estimated 3D points, the resulted extrinsic parameters can be far from the true ones and thus not able to be fixed in the correction.

A bundle adjustment is then introduced as a final touch after processing every additional image on all the estimated parameters, i.e. intrinsic camera parameters, camera locations and poses, and coordinates of all 3D points. The measure of correctness of the estimation is defined by projecting all triangulated 3D points back to 2D images with resulted parameters and then check norm of pixel-wise difference between reprojected points and original 2D feature points. Then the objective of bundle adjustment is to minimize the sum of all the reprojection error, assuming the initial locations, poses, and 3D structure are close to the global optimum. Thus, we obtain the following objective function,

$$\min_{\mathbf{K}^i,\mathbf{P}_0^i,\mathbf{X}_j^0} \quad f = \sum_{i=1}^{n}\sum_{j=1}^{m} w_j^i \left\| \mathbf{x}_j^i - \mathbf{K}^i\mathbf{P}_0^i\mathbf{X}_j^0 \right\|^2 \tag{3-5-6}$$

where w_j^i is a window function and is 1 when the j-th 3D point $\mathbf{X}_j^0$ has a corresponding 2D feature $\mathbf{x}_j^i$ on the i-th image and is 0 otherwise.

For every camera, we have 4 variables for the intrinsic matrix (if every camera is considered different, otherwise we have 4 variables in total for it), and 6 variables for the extrinsic parameter if we include the rotation angle into normalized axis vector by multiplying them. And for every 3D point we have 3 coordinates. As a result, the optimization problem $10n + 3m$ variables (or $6n + 3m + 4$) with n images and m 3D points corresponding to 2D features on different images.

With usually more than 20 images for close range scanning and hundreds of thousands of points, we have a very high dimensional variable space and thus a huge Jacobian matrix, not to mention the thousands of cameras and millions of points cases for large reconstruction problems. However, it is easy to identify the following independences between variables:

(1) A 2D point from one image should be independent from the camera that takes another image, i.e., for a same 2D feature, the derivatives with respect to intrinsic and extrinsic parameters of the camera that captures this point are independent from those of another camera;

(2) For every 3D point, the derivatives with respect to 3D coordinates are independent from anything else but its own parameters.

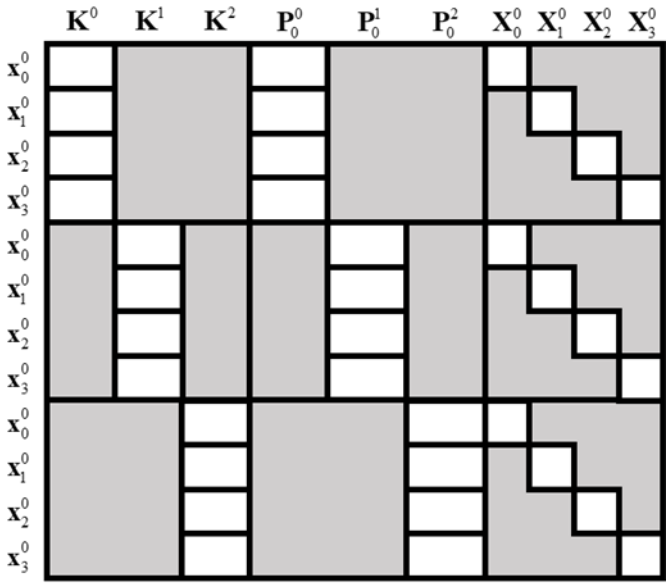


Figure 3-19 Sparsity of the Jacobian matrix for a bundle adjustment problem consisting of 3 cameras and 4 points.

These independence of variables results in a sparse Jacobian matrix, as shown in Figure 3-19, and can simplify the optimization process greatly. The entries in Jacobian matrix in each iteration can be calculated from a provided routine for slightly improved convergence or faster speed. But for simplicity it can also be evaluated from numerical difference and it is reported to have very little disadvantage.[109]

Commonly the Levenberg-Marquardt algorithm, a type of Trust Region method particularly suitable for large sparse problem, is used to solve the optimization problem by combining gradient descent method and Gauss-Newton method. If the objective function has decreased, the step size of the gradient descent part is then factored down so that the optimization is more inclined to a Gauss-Newton method for faster convergence. Otherwise, it is factored up to transit into more of a gradient descent method for a more rapid function decrease. The details of the sparse Levenberg–Marquardt algorithm can be found in [109]. And [131] provides a comprehensive review of bundle adjustment methods.

In our research, for every new image added, estimated parameters and newly triangulated 3D coordinates are used to form a new sparse structure of Jacobian matrix, which is then calculated from numerical difference in every iteration. The variables are then updated by LM method until convergence. Furthermore, from the optimal variables, new reprojection error for every 3D point projected onto corresponding 2D features can be used to filter out bad measurements and lower down noise scale. And the rest of the points and new camera parameters are used to estimate the next added image.

3.6 Results of SfM

We tested our SfM pipeline with images of a few scenes of commonly seen objects with basic geometries like cuboid and cylinder. In the first example, 44 photos were taken around a scene, where two cuboids and a toy of irregular geometry are placed on top of a flat box. Estimated camera poses and a sparse cloud corresponding to the stable SIFT feature points are shown in Figure 3-20 . The sparse point cloud consists of 9742 3D points. The relative positions of the objects are preserved, relative locations and poses between cameras are consistent with the real situation, and structures of the objects are correctly reconstructed. Average reprojection error is 0.86 pixel, suggesting a good estimation of camera locations, poses and 3D locations.

Figure 3-20 Reconstructed point cloud from SfM and camera poses

The second scene contains a box, a cylinder-shaped can, a cylinder-shaped jar, a bottle of Coca Cola, and a bottle of Pepsi. 23 images are taken around the scene, and as shown in Figure 3-21, the relative positions are preserved, and structures of the objects are correctly reconstructed. The sparse point cloud consists of 6235 3D points. Average reprojection error is 0.44 pixel, suggesting a good estimation of camera locations, poses and 3D locations.

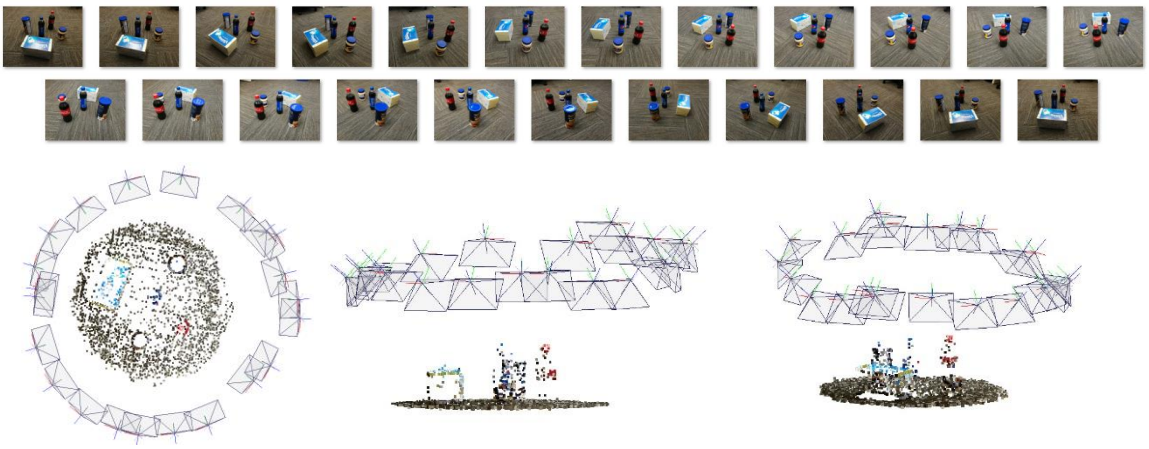


Figure 3-21 Images and reconstruction of a scene containing a box, a jar, a can, and two bottles.

3.7 Conclusions

In this chapter, we discussed the details of every step in our structure from motion pipeline, which starts from extracting and matching SIFT features. And then the pair that returns the least reprojection error is selected as a good pair to start the estimation of camera parameters and 3D point coordinates. An adjacent image is then added in and its camera extrinsic matrix is estimated by projecting the 3D points onto corresponding 2D features shared with previous image. Corrections are introduced to guarantee that extrinsic matrix satisfy its constraint to be orthogonal, providing a better initial condition for a more holistic correction, i.e. the bundle adjustment, on all the estimated parameters.

Our main contribution here is to design the methods to reduce the effect of drifting by selecting a good starting pair with the least reprojection error, and by applying correction on the estimated camera extrinsic parameters and thus providing bundle adjustment algorithm with a better initialization. From the two examples and the box example illustrated throughout this chapter, our SfM pipeline correctly restores the scene information including camera parameters and a low point-count structure that correspond to stable SIFT feature matches on the images. Drifting issue is well resolved by the introduced initial pair selection strategy and the correction methods combined with bundle adjustment algorithm.

However, the reconstructed structure is too sparse because the features used to triangulate 3D points are very sparsely detected for better robustness in the estimation. And the matching of SIFT features are conducted from a 2D search in the close neighbor of the previous feature. After the filtering of bad points with large reprojection error, only a small part of the detected features is left and thus much of the information of the scene is missing. But indeed, these sparse points lead to a good estimation of the extrinsic parameters for each camera and thus providing relative relations between every pair of images. And these relations can be used to obtain much denser matches with simpler and faster methods.

CHAPTER 4

MULTI-VIEW STEREO

Even for a human being, it is not easy to identify the object of each reconstructed structure simply from the results of SfM like what is shown in Figure 3-21. The reconstructed 3D point cloud needs to spread out as much of the surface area as possible with higher density so that the object class can be identified with higher accuracy. However, the SIFT features used to relate images are very sparse yet very reliable because they are extracted by analyzing the surroundings of pixels in different scales and then compared with neighboring descriptors for a best match. These strong matching features with low level of noise contribute to the robustness of the reconstruction process, along with the measures discussed in the previous chapter.

In this chapter, we move a step further to generate a more densely sampled point cloud from the results of SfM pipeline. We start with a review of related research work that targets at generating dense reconstruction either in form of point cloud or of space voxels. Then our pipeline of Multi-View Stereo (MVS) is introduced including the theories and details of the algorithms used in the process. We follow the box example throughout last chapter to illustrate the steps. And the examples from section 3.6 will be extended further to demonstrate the performance of our pipeline.

4.1 Related Works

Multi-View Stereo serves as an extension of the classic two-view stereo problem, which uses stereo matching to find dense correspondences between rectified image pairs to simplify the

2D matches search into a 1D searching problem [134] and thus also allowing correlation-based matching algorithms [135] to work. Some methods [111, 136] seek to find the rectification homographies that project the two images onto a plane that is parallel to the base line and containing the intersection line of the two image planes. Loop and Zhang [134] proposed a rectifying method by decomposing each homography into a projective, a similarity, and a shearing transform, which obtains results with minimized distortion.

Generally, MVS is a term given to a group of techniques that reconstruct depth maps, point clouds, voxels, or meshes using stereo correspondence from multiple calibrated images. Type of MVS algorithm includes voxel approaches like voxel coloring [137] and space carving [24, 138], deformable polygon meshes approach [139], multiple depth maps method [140] that fusing depth maps into a 3D model, and feature-based patch-growing methods [141] that outputs a dense collection of small oriented rectangular patches. A multi-view depth map linking method is proposed in [142], which solve for the two-view stereo problem between each neighboring pair for disparity, and then fusing all independent estimates into a common 3D model through controlled correspondence linking, which is more robust to changes in camera exposure, non-Lambertian surfaces, and passers-by.

4.2 MVS Pipeline

In our MVS process, we adopted the algorithm proposed by Richard Hartley[136] to rectify image pairs with estimated camera parameters from previous SfM process, and the details will be discussed in section 4.4. Then one-dimensional searches on the rectified image pairs are conducted with Semiglobal Block Matching (SGBM) for denser matches, which can then be triangulated into a denser point cloud and provide more details of the scanned scene. Figure 4-1 shows the pipeline of our MVS process.

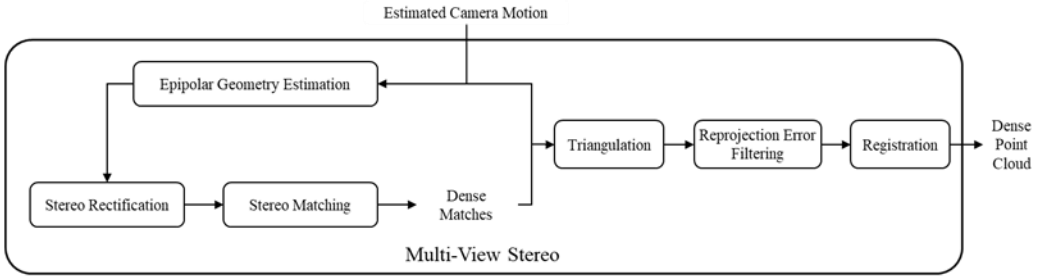


Figure 4-1 Workflow of MVS

For the box example, the input and output of the MVS process are shown in Figure 4-2. From estimated camera poses from SfM, the epipolar geometry can be obtained reversely. Then each image pair can be rectified by a warping operation with an estimated homography. The rectified images will have colinear epipolar lines and thus can reduce the matching process into a 1-denmentional searching problem. A disparity map is then generated from stereo matching, resulting in much denser correspondences. These dense matches can be triangulated, filtered, and registered in the same manner as in SfM. The output will be a dense point cloud that preserves most of the details of the scene.

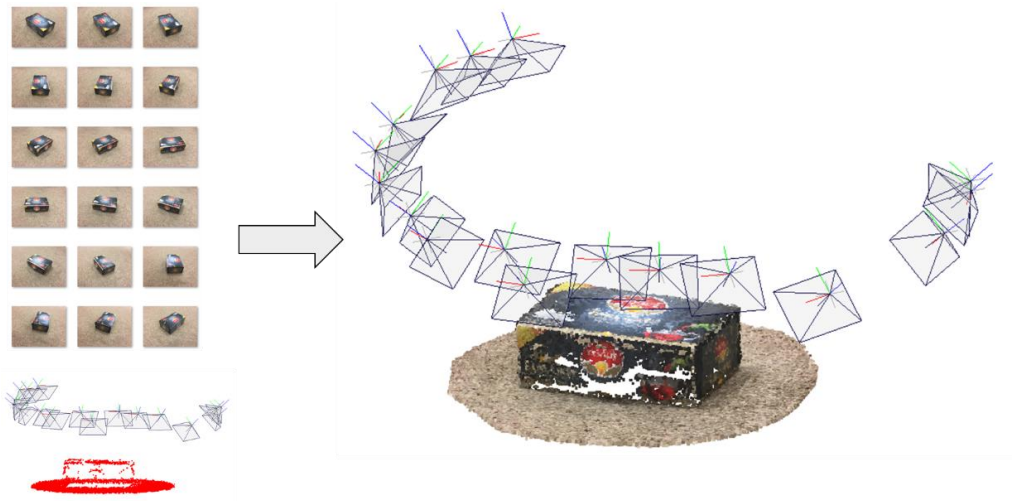


Figure 4-2 Input images, cameras and sparse point cloud from SfM, and visualized output of Multi-View Stereo

In the following sections, each step of the MVS process will be discussed in detail, and two examples will be shown at the end of this chapter to demonstrate the performance of our MVS pipeline.

4.3 Re-estimate Epipolar Geometry

With all the correction measures done in SfM, the resulting extrinsic camera parameters can be used backwards to retrieve more accurate fundamental matrices between each two images for later use of rectification.

Consider the first camera center as origin, and the estimated extrinsic matrix of the second camera to be $\mathbf{P}'$, the epipole $\mathbf{e}'$ on the second image is the projection of origin. From equation (3-4-2) and the basic projective mapping equation, we obtain the following equation to calculate fundamental matrix from corrected $\mathbf{K}$ and $\mathbf{P}'$.

$$\begin{aligned} & \mathbf{Fx} = \mathbf{l}' = [\mathbf{e}']_{\times} \mathbf{x}' \\ \Rightarrow \quad & \mathbf{F}\left(\mathbf{KP}^0\mathbf{X}\right) = [\mathbf{e}']_{\times} \left(\mathbf{KP}'\mathbf{X}\right) \\ \Rightarrow \quad & \mathbf{F} = [\mathbf{e}']_{\times} \mathbf{KP}'\left(\mathbf{KP}^0\right)^{-1} \end{aligned} \tag{4-3-1}$$

4.4 Stereo Rectification

To decrease the searching range of matches between two images to one dimensional, for any point on one image, the line on the other image that contains the match of it needs to be identified and located. By applying 2D projective transforms on both images, the rectification process seeks to find a pair of such transformation or homographies by which the two images can be warped onto the same plane, i.e.,

$$\bar{\mathbf{x}} = \mathbf{Hx}, \quad \bar{\mathbf{x}}' = \mathbf{H}'\mathbf{x}' \tag{4-4-1}$$

with the epipoles moved to infinity or $\mathbf{i} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$, and thus, the epipolar lines lie parallel to each other. New fundamental matrix $\bar{\mathbf{F}}$ has the form

$$\bar{\mathbf{F}} = [\mathbf{i}]_{\times} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \tag{4-4-2}$$

Substituting the two equations above into $\bar{\mathbf{x}}'^{\mathrm{T}} \bar{\mathbf{F}} \bar{\mathbf{x}} = 0$, we have

$$\mathbf{F} = \mathbf{H}'^{\mathsf{T}} [\mathbf{i}]_{\times} \mathbf{H} \tag{4-4-3}$$

Notice that there are infinite many pairs of homographies that satisfy equation (4-4-3) and the optimal one we seek to obtain is the pair that introduces the least distortion to both images.

Here we discuss two commonly used methods for rectification, one proposed by Loop and Zhang [134] for a more optimal solution, and another by Hartley [136] for a simpler but faster solution.

4.4.1 Rectification with Minimized Distortion

Consider the scale invariant homographies,

$$\mathbf{H} = \begin{bmatrix} \mathbf{u}^{\mathsf{T}} \\ \mathbf{v}^{\mathsf{T}} \\ \mathbf{w}^{\mathsf{T}} \end{bmatrix} = \begin{bmatrix} u_a & u_b & u_c \\ v_a & v_b & v_c \\ w_a & w_b & 1 \end{bmatrix}, \quad \mathbf{H}' = \begin{bmatrix} \mathbf{u}'^{\mathsf{T}} \\ \mathbf{v}'^{\mathsf{T}} \\ \mathbf{w}'^{\mathsf{T}} \end{bmatrix} = \begin{bmatrix} u'_a & u'_b & u'_c \\ v'_a & v'_b & v'_c \\ w'_a & w'_b & 1 \end{bmatrix} \tag{4-4-4}$$

where $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ can be considered as three lines on the first image, and $\mathbf{u}'$, $\mathbf{v}'$, and $\mathbf{w}'$ as three lines on the second image. The transformation of epipoles can then be written as

$$\begin{aligned} \mathbf{He} &= \begin{bmatrix} \mathbf{u}^{\mathsf{T}}\mathbf{e} & \mathbf{v}^{\mathsf{T}}\mathbf{e} & \mathbf{w}^{\mathsf{T}}\mathbf{e} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^{\mathsf{T}} = \mathbf{i} \\ \mathbf{H}'\mathbf{e}' &= \begin{bmatrix} \mathbf{u}'^{\mathsf{T}}\mathbf{e}' & \mathbf{v}'^{\mathsf{T}}\mathbf{e}' & \mathbf{w}'^{\mathsf{T}}\mathbf{e}' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^{\mathsf{T}} = \mathbf{i} \end{aligned} \tag{4-4-5}$$

which indicates that lines $\mathbf{v}$ and $\mathbf{w}$ go through $\mathbf{e}$, similarly $\mathbf{v}'$ and $\mathbf{w}'$ go through $\mathbf{e}'$, and thus all of them are epipolar lines. And it can be further proved that $\mathbf{v}$ and $\mathbf{v}'$, $\mathbf{w}$ and $\mathbf{w}'$ are two pair of corresponding epipolar lines.

Decompose the first homography into $\mathbf{H} = \mathbf{H}_a \mathbf{H}_p$, where $\mathbf{H}_p$ is a projective transformation that maps the epipole to infinity and is defined to be

$$\mathbf{H}_p = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ w_a & w_b & 1 \end{bmatrix} \tag{4-4-6}$$

and $\mathbf{H}_a$, an affine transformation, can be calculated from $\mathbf{H}\mathbf{H}_p^{-1}$. Further decomposing $\mathbf{H}_a$ into $\mathbf{H}_a = \mathbf{H}_s\mathbf{H}_r$, where $\mathbf{H}_r$ is a similarity transformation and has the form

$$\mathbf{H}_r = \begin{bmatrix} v_b - v_c w_b & v_c w_a - v_a & 0 \\ v_a - v_c w_a & v_b - v_c w_b & v_c \\ 0 & 0 & 1 \end{bmatrix} \tag{4-4-7}$$

and $\mathbf{H}_s$ is a shear transformation, which does not affect the rectification, and is defined as

$$\mathbf{H}_s = \begin{bmatrix} s_a & s_b & s_c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4-4-8}$$

Thus, the homography for the first image is decomposed into three part,

$$\mathbf{H} = \mathbf{H}_s\mathbf{H}_r\mathbf{H}_p \tag{4-4-9}$$

For the other homography $\mathbf{H}'$, similar decomposition is defined with prime notation.

For the projective part, given any direction $z = \begin{bmatrix} \lambda & \mu & 0 \end{bmatrix}^\mathsf{T}$ in the first image, we obtain the epipolar line $\mathbf{w} = [\mathbf{e}]_\times \mathbf{z}$ corresponding to $\mathbf{w}' = \mathbf{F}\mathbf{z}$ on the other image. Any point $\mathbf{p}_i = \begin{bmatrix} p_{i,x} & p_{i,y} & 1 \end{bmatrix}^\mathsf{T}$ on the first image will be transformed by $\mathbf{H}_p$ to $\begin{bmatrix} \frac{p_{i,x}}{w_i} & \frac{p_{i,y}}{w_i} & 1 \end{bmatrix}^\mathsf{T}$, where the weight $w_i = \mathbf{w}^\mathsf{T}\mathbf{p}_i$. Total distortion can be defined as

$$D = \sum_{i=1}^{n} \left[\frac{\mathbf{w}^\mathsf{T}(\mathbf{p}_i - \mathbf{p}_c)}{\mathbf{w}^\mathsf{T}\mathbf{p}_c} \right]^2 \tag{4-4-10}$$

where $\mathbf{p}_c$ is the average of points. Rewriting it into a matrix form, we obtain

$$D = \frac{\mathbf{w}^\mathsf{T}\mathbf{P}\mathbf{P}^\mathsf{T}\mathbf{w}}{\mathbf{w}^\mathsf{T}\mathbf{p}_c\mathbf{p}_c^\mathsf{T}\mathbf{w}} \tag{4-4-11}$$

where

$$\mathbf{P}=\begin{bmatrix} p_{1,x}-p_{c,x} & p_{2,x}-p_{c,x} & \cdots & p_{n,x}-p_{c,x} \\ p_{1,y}-p_{c,y} & p_{2,y}-p_{c,y} & \cdots & p_{n,y}-p_{c,y} \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{4-4-12}$$

Substitute $\mathbf{w}=[\mathbf{e}]_{\times}\mathbf{z}$, and $\mathbf{w}'=\mathbf{F}\mathbf{z}$ into equation (4-4-11) we obtain the sum of distortion,

$$\Sigma=\frac{\mathbf{z}^{\mathsf{T}}[\mathbf{e}]_{\times}^{\mathsf{T}}\mathbf{P}\mathbf{P}^{\mathsf{T}}[\mathbf{e}]_{\times}\mathbf{z}}{\mathbf{z}^{\mathsf{T}}[\mathbf{e}]_{\times}^{\mathsf{T}}\mathbf{p}_c\mathbf{p}_c^{\mathsf{T}}[\mathbf{e}]_{\times}\mathbf{z}}+\frac{\mathbf{z}^{\mathsf{T}}\mathbf{F}^{\mathsf{T}}\mathbf{P}'\mathbf{P}'^{\mathsf{T}}\mathbf{F}\mathbf{z}}{\mathbf{z}^{\mathsf{T}}\mathbf{F}^{\mathsf{T}}\mathbf{p}_c'\mathbf{p}_c'^{\mathsf{T}}\mathbf{F}\mathbf{z}}=\frac{\mathbf{z}^{\mathsf{T}}\mathbf{A}\mathbf{z}}{\mathbf{z}^{\mathsf{T}}\mathbf{B}\mathbf{z}}+\frac{\mathbf{z}^{\mathsf{T}}\mathbf{A}'\mathbf{z}}{\mathbf{z}^{\mathsf{T}}\mathbf{B}'\mathbf{z}} \tag{4-4-13}$$

where, if consider all the points on the image of resolution $w\times h$,

$$\begin{aligned} \mathbf{P}\mathbf{P}^{\mathsf{T}} &= \frac{wh}{12}\begin{bmatrix} w^2-1 & 0 & 0 \\ 0 & h^2-1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \mathbf{P}_c\mathbf{P}_c^{\mathsf{T}} &= \frac{1}{4}\begin{bmatrix} (w-1)^2 & (w-1)(h-1) & 2(w-1) \\ (w-1)(h-1) & (h-1)^2 & 2(h-1) \\ 2(w-1) & 2(h-1) & 4 \end{bmatrix} \end{aligned} \tag{4-4-14}$$

from which the coefficient matrices $\mathbf{A}$, $\mathbf{A}'$, $\mathbf{B}$, and $\mathbf{B}'$ can be calculated.

Since direction $\mathbf{z}$ is defined up to scale, μ can be set to 1 and result in the following minimization problem.

$$\min_{\lambda}\ \Sigma=\frac{\mathbf{z}^{\mathsf{T}}\mathbf{A}\mathbf{z}}{\mathbf{z}^{\mathsf{T}}\mathbf{B}\mathbf{z}}+\frac{\mathbf{z}^{\mathsf{T}}\mathbf{A}'\mathbf{z}}{\mathbf{z}^{\mathsf{T}}\mathbf{B}'\mathbf{z}} \tag{4-4-15}$$

An initial guess is first obtained by averaging the results of minimizing the two terms in Σ separately, from which the root can be found iteratively, and then inserted into $\mathbf{w}=[\mathbf{e}]_{\times}\mathbf{z}$, and $\mathbf{w}'=\mathbf{F}\mathbf{z}$ to obtain w_a and w_b.

Similarity transformation is then calculated to rotate the epipoles at infinity into alignment with $\mathbf{i}=[0\quad 0\quad 1]^{\mathsf{T}}$. Eliminating v_a and v_b in equation (4-4-3) we get

$$H_r = \begin{bmatrix} F_{32} - w_b F_{33} & w_a F_{33} - F_{31} & 0 \\ F_{31} - w_a F_{33} & F_{32} - w_b F_{33} & F_{33} + v_c' \\ 0 & 0 & 1 \end{bmatrix}$$
$$H_r' = \begin{bmatrix} w_b' F_{33} - F_{23} & F_{13} - w_a' F_{33} & 0 \\ w_a' F_{33} - F_{13} & w_b' F_{33} - F_{23} & v_c' \\ 0 & 0 & 1 \end{bmatrix} \quad (4\text{-}4\text{-}16)$$

Here v_c' is obtained such that minimum y coordinate in either image is zero.

Finally, on top of the rectification with projection and rotation, a shear transformation

$$\mathbf{S} = \begin{bmatrix} a & b & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4\text{-}4\text{-}17)$$

is introduced to reduce distortion caused by projective transformation. The approach is to examine the transformed midpoints of the four edges of the images,

$$\begin{aligned} \hat{\mathbf{a}} &= \mathbf{H}_r \mathbf{H}_p \begin{bmatrix} \frac{w-1}{2} & 0 & 1 \end{bmatrix}^{\mathsf{T}} \\ \hat{\mathbf{b}} &= \mathbf{H}_r \mathbf{H}_p \begin{bmatrix} w-1 & \frac{h-1}{2} & 1 \end{bmatrix}^{\mathsf{T}} \\ \hat{\mathbf{c}} &= \mathbf{H}_r \mathbf{H}_p \begin{bmatrix} \frac{w-1}{2} & h-1 & 1 \end{bmatrix}^{\mathsf{T}} \\ \hat{\mathbf{d}} &= \mathbf{H}_r \mathbf{H}_p \begin{bmatrix} 0 & \frac{h-1}{2} & 1 \end{bmatrix}^{\mathsf{T}} \end{aligned} \quad (4\text{-}4\text{-}18)$$

and to preserve the perpendicularity and aspect ratio of the two lines $\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{x}}_x & \hat{\mathbf{x}}_y \end{bmatrix}^{\mathsf{T}} = \hat{b} - \hat{d}$ and $\hat{\mathbf{y}} = \begin{bmatrix} \hat{\mathbf{y}}_x & \hat{\mathbf{y}}_y \end{bmatrix}^{\mathsf{T}} = \hat{c} - \hat{a}$ with

$$\begin{aligned} \left(\mathbf{S}\hat{\mathbf{x}}\right)^{\mathsf{T}} \left(\mathbf{S}\hat{\mathbf{y}}\right) &= 0 \\ \frac{\left(\mathbf{S}\hat{\mathbf{x}}\right)^{\mathsf{T}} \left(\mathbf{S}\hat{\mathbf{x}}\right)}{\left(\mathbf{S}\hat{\mathbf{y}}\right)^{\mathsf{T}} \left(\mathbf{S}\hat{\mathbf{y}}\right)} &= \frac{w^2}{h^2} \end{aligned} \quad (4\text{-}4\text{-}19)$$

Real solution can be found following with

$$a = \frac{h^2 \hat{\mathbf{x}}_y^2 + w^2 \hat{\mathbf{y}}_y^2}{hw\left(\hat{\mathbf{x}}_y \hat{\mathbf{y}}_x - \hat{\mathbf{x}}_x \hat{\mathbf{y}}_y\right)}, \quad b = \frac{h^2 \hat{\mathbf{x}}_x \hat{\mathbf{x}}_y + w^2 \hat{\mathbf{y}}_x \hat{\mathbf{y}}_y}{hw\left(\hat{\mathbf{x}}_x \hat{\mathbf{y}}_y - \hat{\mathbf{x}}_y \hat{\mathbf{y}}_x\right)} \quad (4\text{-}4\text{-}20)$$

up to sign, but it is preferred to choose the solution where a is positive.

It is important that same scale and y-translation be applied to both images So, the shear homography $\mathbf{H}_s$ is defined combining $\mathbf{S}$ with a uniform scaling factor that preserves the sum of image areas and a translation term so that rectification is preserved.

4.4.2 Fast Rectification

Another faster method is to choose a transformation homography that acts as far as possible as a rigid transformation in the neighborhood of a selected point, the image center for most of the case. The strategy is to choose $\mathbf{H}'$ first for the second image and then seek a matching homography to minimize the sum of square distance.

First, the center of the second image $\mathbf{x}_0$ is translated to origin (at corner) with $\mathbf{H}'_T$ and rotate the image with $\mathbf{H}'_R$ such that the epipole be rotated to $\mathbf{e}' = \begin{bmatrix} f & 0 & 1 \end{bmatrix}^{\mathsf{T}}$, then the following transformation

$$\mathbf{H}'_P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/f & 0 & 1 \end{bmatrix} \tag{4-4-21}$$

can take the epipole to $\begin{bmatrix} f & 0 & 0 \end{bmatrix}^{\mathsf{T}}$ at infinity. Meanwhile any other point $\begin{bmatrix} u & v & 1 \end{bmatrix}^{\mathsf{T}}$ is then mapped to

$$\begin{aligned} \begin{bmatrix} \hat{u} & \hat{v} & 1 \end{bmatrix}^{\mathsf{T}} &= \begin{bmatrix} u & v & 1-u/f \end{bmatrix}^{\mathsf{T}} \\ &= \begin{bmatrix} u\left(1+u/f+u^2/f^2+\cdots\right) \\ v\left(1+u/f+u^2/f^2+\cdots\right) \\ 1 \end{bmatrix} \end{aligned} \tag{4-4-22}$$

which has the following Jacobian matrix, suggesting a near identity mapping around origin.

$$\frac{\partial(\hat{u},\hat{v})}{\partial(u,v)} = \begin{bmatrix} 1+2u/f & 0 \\ v/f & 1+u/f \end{bmatrix} + O\left(u^2, v^2\right) \tag{4-4-23}$$

The origin can then be translated back to image center with $\mathbf{H}'_{-T}$. Thus, we have the transformation homography for the second image

$$\mathbf{H}' = \mathbf{H}'_{-T}\mathbf{H}'_{P}\mathbf{H}'_{R}\mathbf{H}'_{T} \tag{4-4-24}$$

The fundamental matrix can be factored as $\mathbf{F} = [\mathbf{e}']_{\times}\mathbf{M}$ for some non-singular $\mathbf{M}$. Then the transform homography $\mathbf{H}$ for the first image has the form $\mathbf{H} = \mathbf{A}\mathbf{H}_0$, where $\mathbf{H}_0 = \mathbf{H}'\mathbf{M}$ and $\mathbf{A}$ is an affine transformation of form

$$\mathbf{A} = \begin{bmatrix} a & b & c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4-4-25}$$

Since we have $\mathbf{e}'^{\mathsf{T}}\mathbf{F} = 0$, $\mathbf{e}'$ can be obtained from a singular value decomposition of the fundamental matrix $\mathbf{F} = \mathbf{U}\Sigma\mathbf{V}^*$ and $\mathbf{e}'$ is the last column of $\mathbf{U}$ corresponding to the 0 singular value. Also notice that $[\mathbf{e}']_{\times}\mathbf{e}' = \mathbf{0}$, then the choice of a non-singular transformation $\mathbf{M}$ that satisfies $\mathbf{F} = [\mathbf{e}']_{\times}\mathbf{M}$ can be

$$\mathbf{M} = [\mathbf{e}']_{\times}\mathbf{F} + \mathbf{e}'[1 \quad 1 \quad 1] \tag{4-4-26}$$

Now the goal is to obtain an optimal matrix $\mathbf{A}$ that minimizes disparity

$$\sum_i d\left(\mathbf{H}\mathbf{x}_i, \mathbf{H}'\mathbf{x}'_i\right)^2 \tag{4-4-27}$$

Writing $\hat{\mathbf{x}}_i = \mathbf{H}'\mathbf{M}\mathbf{x}_i$ and $\hat{\mathbf{x}}'_i = \mathbf{H}'\mathbf{x}'_i$, then the optimization problem becomes

$$\min_{a,b,c} \sum_i d\left(A\hat{\mathbf{x}}_i, \hat{\mathbf{x}}'_i\right)^2 \tag{4-4-28}$$

Then from the matched points $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, and consider that vertical differences are zero, thus the optimization problem is now simplified to

$$\min_{a,b,c} \sum_i \left(a\hat{u}_i + b\hat{v}_i + c - \hat{u}'_i\right)^2 \tag{4-4-29}$$

and can be solved with linear least square minimization.

4.4.3 Results

The second method of rectification is adopted due to its convenience and speed. At the same time the last step of the first method is also included to apply a shearing homography to the rectified images so that the distortion introduced by projective mapping can be further reduced.

For the same box reconstruction example, the first original image pair and the rectified result of it are shown in Figure 4-3. A few epipolar lines are marked across the image to show that the correspondences, after rectification, are aligned with very close if not the same y-coordinates. Also, distortion coefficients from calibration is considered here and resulting in the obvious distortion close to image edges. The results demonstrate a good rectification with small distortion from the original images and thus are used in the next step of MVS process.

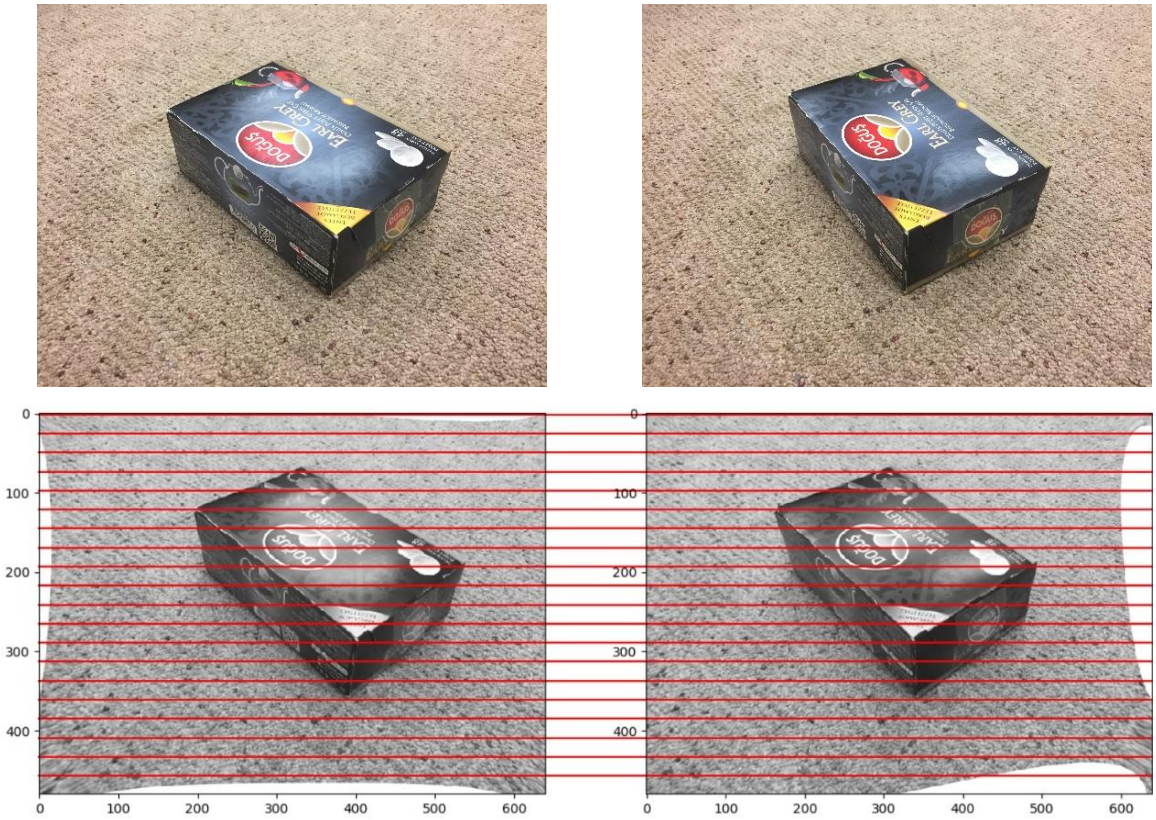


Figure 4-3 Original two images of a box and rectified images with sampled epipolar lines.

4.5 Stereo Matching

Now that the image pairs are stereo rectified, a much quicker 1D correlation-based search can be applied to locate the best match for each pixel or block that "wins" the second-best match at certain ratio, and as a result we obtain a much denser set of correspondences. These matches will have the same x coordinate and the difference on y direction. And as shown in Figure 4-4, larger y - difference indicates larger depth value. The collection of the difference at every pixel location is called the disparity map, which can be plotted with a gray-scale color for visualization.

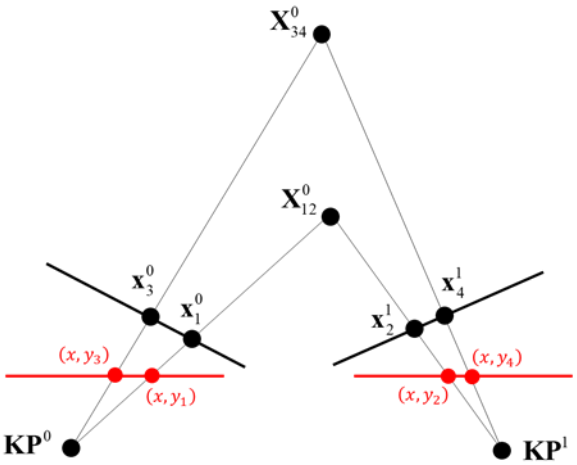


Figure 4-4 Stereo Matching on rectified images

We used OpenCV stereo matching module [143] directly with SGBM method, and the details of the algorithm can be found in [144]. As a result, the disparity map for the box example is obtained and shown in Figure 4-5, where black color indicates no match was found while darker color means further position from the rectified image plane. The matched set can then be warped back using the inverse of the rectifying homographies $\mathbf{H}^{-1}$ and $\mathbf{H}'^{-1}$, and thus the dense matches on original image pair can be obtained.

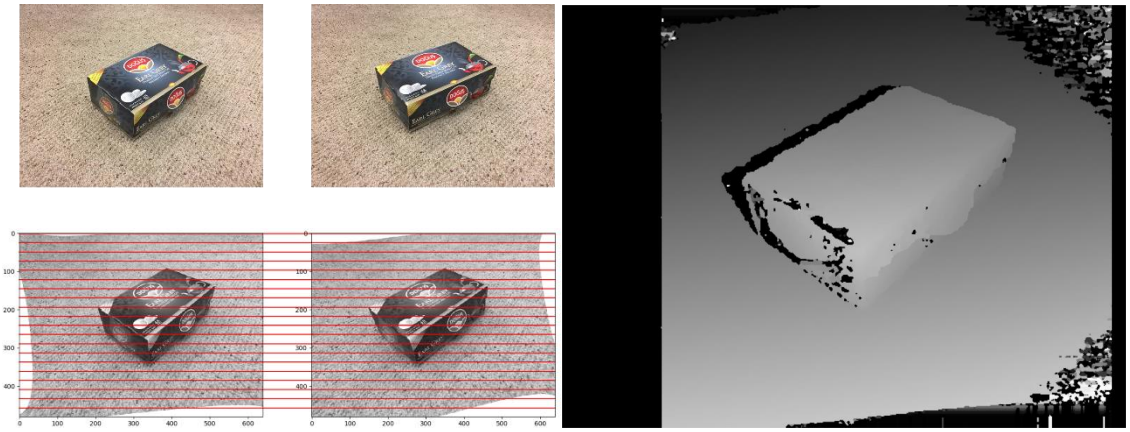


Figure 4-5 Images of box, rectified images, and obtained disparity map

4.6 Dense Point Cloud Reconstruction

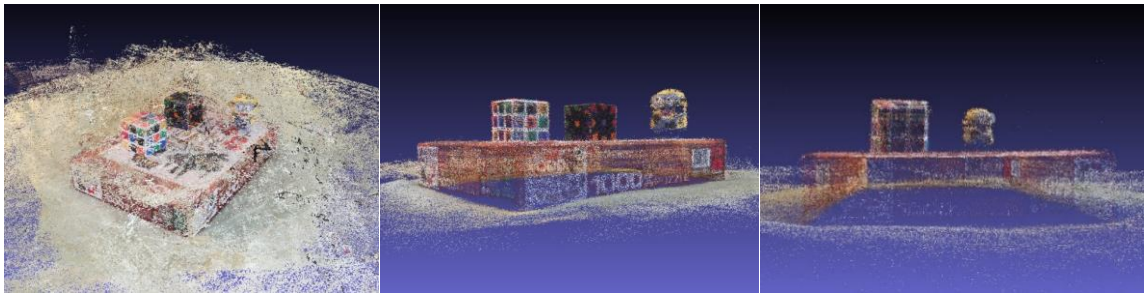
Notice that on the disparity map there are areas that are porous, unevenly colored, or have blocks of black color, indicating wrong matches or no matches found in those areas. Thus, we implement the following filters on the matches so that noise scale can be decreased: (1) all the pixels in disparity map with black color are filtered; and (2) all the pixels that are added from rectification process shown in white in the rectified images. (3) all pixels that are out of image when warped back.

The resulting matches for each pair of images are then stored in a data structure for easy tracking of repeats, every row of which contains the 3D coordinates, pixel color, and the 2D coordinates in every image that sees it. We still follow the incremental schema to triangulate and update the point clouds. Every time a new image is added, the dense matches are stacked into the data structure and non-repeats are used to triangulate new 3D points and then transformed into the world coordinate system. Then the reprojection error of all the 3D points onto all the images that see them will be calculated and yield a measurement for filtering out noisy results. Noise of the final results can be further decreased by leaving only 3D points that are seen by at least 3 images or 4 images.

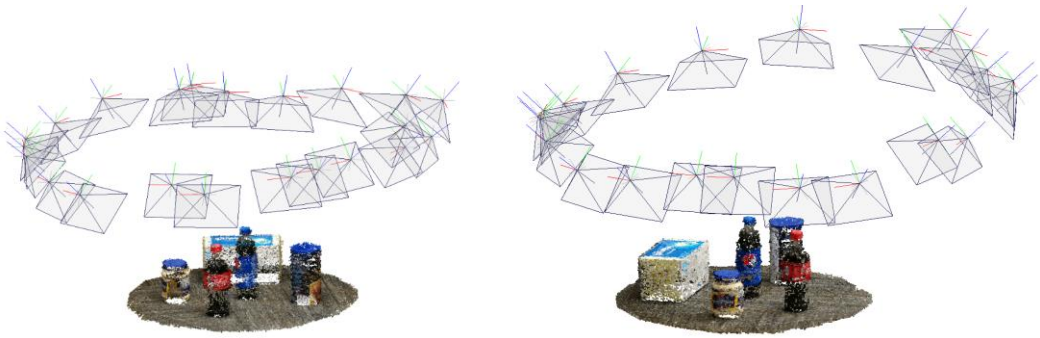
4.7 Results of MVS

Follow the two examples for SfM, the results are put into the MVS process and the following results of dense point clouds are obtained.

Figure 4-6 shows the reconstructed and filtered dense point clouds. For the first example, 3,649,180 points are generated from processing 43 images of size 960 by 720 pixels, 617,815 from which can be viewed by four or more cameras, and 81,662 from which are left after the reprojection error filter. For both examples, viewed from a distance, the point cloud provides a good sampling of the scene, shape of structures as well as sharp edges are distinguishable, and number of errant points are largely decreased from the filtering methods.



(a) (b) (c)



(d) (e)

(f) (g) (h)

Figure 4-6 Reconstructed dense point cloud. Example 1: (a) Raw registered data. (b) Points seen by at least 4 cameras. (c) Points after filter. Example 2: (d) (e) dense reconstruction results viewing from different angles. (f) (g) (h) Three other reconstructed results of commonly seen products

Now compared with the SfM results, these scenes and objects are easier to identify by human eyes. Although samples of surfaces are still corrupted and can be seen when closing in to details, especially on regions that provides less features. However, the goal here is not to reconstruct the scene or the object to extremely fine details on the surfaces, but to generate a point cloud that is dense enough for a classification algorithm to use for either training purpose or identification purpose.

4.8 Conclusions

In this chapter, each step of the Multi-View Stereo pipeline is discussed including the theories, algorithms, and results of examples. For every pair of images, epipolar geometries are first reversely obtained from the estimated camera matrices and are then used to calculate a pair of matching homographies to rectify the images by mapping epipoles to infinity so that epipolar lines are co-linear in both images and distortion from the projective mappings are minimized. Then 1D searches for dense matches using semiglobal block matcher are applied on the rectified images and result in much denser matching pairs, which are recorded, filtered, and used to triangulate 3D points. The final results are further filtered by reprojecting 3D points back to all images that has view of them, and by rejecting points that are seen by less than 3 images. The obtained results from MVS process provide much denser sampling of the scene and number of errant points are largely decreased from the filtering methods. The geometric shape and product type can be identified with human eyes directly from the dense point cloud. And thus, in our next step we seek to find the class and family of the scanned object from the reconstructed dense point cloud.

CHAPTER 5

SHPAE FAMILY IDENTIFICATION

In our research, the scanning and reconstruction of a scene or object is not just for visualization purpose, but more importantly to move a step closer to an image-based 3D modeling method, for which it is necessary to first understand what has been reconstructed, not just by the designer, but by the computer. However, for most of the scanned surface, the complexity and irregularity of them make it almost impossible to be fitted with a mathematical surface function of certain order.

For decades, semantic classification and segmentation have been explored for machines to mimic the biological neural networks operation on learning and identifying certain types of object to understand an input, like an image, through estimating the probabilities of it being of certain classes, like an animal or a person.

More recently with the increasingly common use of scanning tools, like photogrammetry, Lidar, and RGB-D camera, classification has been applied on obtained 3D point cloud data, mostly for the purpose of understanding surrounding environment of autonomous robots, cars, or drones for target grabbing, obstacle avoidance, or terrestrial labeling. The potential of 3D data classification is further explored after the introduction of supervised 3D Convolutional Neural Networks (CNNs), the 2D version of which had been a great success interpreting images due to easily parallelizable network architectures and availability of huge public image dataset.

In this chapter, we seek to build a classification model that can predict the shape family of our scanned objects using multilayer CNNs. We begin with a brief review of the foundational as well as the cutting-edge research work in artificial neural networks, and then discussed the mathematical derivation and implementation of both 2D CNNs and 3D CNNs. Structure of our 3D multilayer CNNs model is then introduced along with the details of data preprocessing, model training, evaluation and testing. At the end of this chapter, the performance of the trained model is tested with our reconstructed point clouds from SfM and MVS, and the result shows high accuracy in classification of shape family as well as the ability to be extended to identify more classes.

5.1 Related Works

The idea of artificial neural networks traces back to 1943 when Warren McCulloch and Walter Pitt [145] created a computational model called threshold logic about how neurons might work using electrical circuits. And with the later advancement in computer simulation, the idea is used on pattern recognition to identify 2D stimulus pattern like the “neocognitron” proposed in [146]. The spectacular comeback of neural networks arrived with the seminal work of Krizhevsky [147] and deep CNNs then become the core technique for learning-based classification and segmentation, not just for 2D images [148-150], but also on RGB-D images and more recently on 3D point clouds.

Before the use of CNNs, machine learning classifiers [151-153] use a combination of hand-crafted features for 3D data recognition and segmentation. After the success of CNNs on image analysis which extracts features automatically by learning, it is then extended to RGB-D images but only consider depth as an extra channel (2.5D) instead of the full 3D data [154, 155]. Also, for multiple RGB-D images of the same scene, it is difficult to integrate the segmentation information across different viewpoints.

One of the approaches that fully extends CNNs to 3D point cloud is achieved through the volumetric data representation, starting from the work in [156] applied on Lidar data. A generative Convolutional Deep Belief Network model, 3D ShapeNets, is proposed in [34] which learns voxelized CAD data and is able to apply object recognition and shape completion from depth maps. And VoxNet proposed in [33] uses volumetric occupancy grid representation with a 3D CNN architecture and can be applied to create fast and accurate detectors for 3D point cloud data from Lidar, RGB-D, and CAD models.

5.2 Image Classification with 2D Convolutional Neural Networks

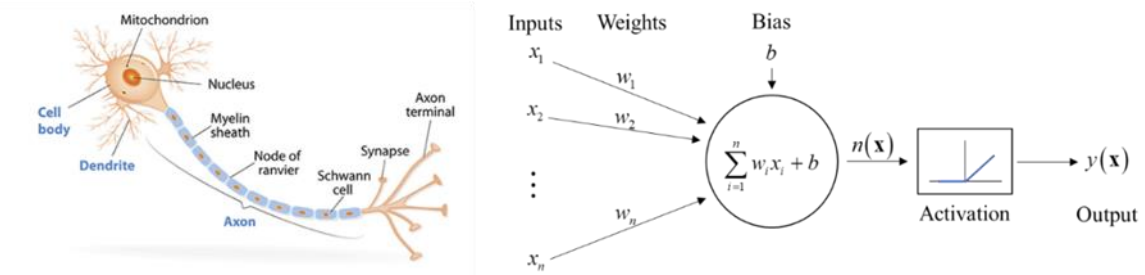


Figure 5-1 Model of a single neuron

Just like how a neuron receives information from many dendrites and then from a synapse either pass a synthesized information to the next neuron or not, the mathematical model of a single artificial neuron is setup similarly as a weighted sum of all the inputs plus a bias for the synthesizing part, and an activation function to decide either passing the result or not, as shown in Figure 5-1. The activation function, also known as the rectifier, is defined as the positive part of its argument. And a unit that employs the rectifier is called a rectified linear unit (ReLU), which is used to increase non-linearity. Various activation functions are also developed to address the smoothness and speed of convergence issues, e.g., tanh, sigmoid, Leaky ReLU, and Maxout.

5.2.1 Artificial Neural Networks

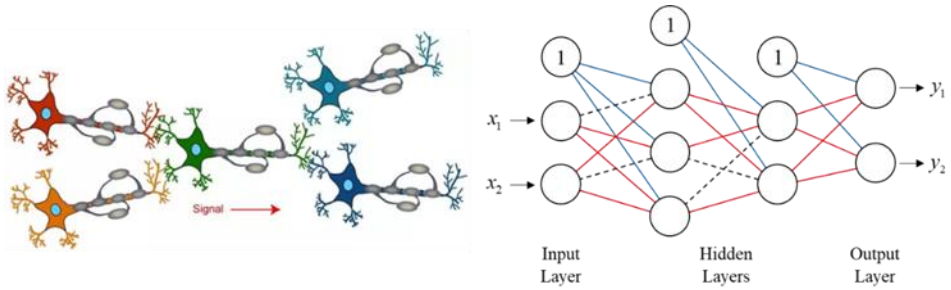


Figure 5-2 Model of multi-layer neural networks

The output of a neuron can then become the input of another neuron, and thus many neurons can be connected to build a network of neurons and arranged in layers to create an artificial neural network. The right side of Figure 5-2 shows an example of a neural network that takes two inputs and generates two outputs from four layers of neurons, where the passed weighted sum results of neurons are marked as red lines, unpassed results marked as black dashes, and bias inputs marked in blue lines.

Essentially the neural network is a universal approximator of a mapping from inputs to outputs using layers of weighted sum and nonlinear activation. The training process is then a repeated updating of the weights and biases from backpropagating output errors so that the model performance is optimized.

5.2.2 Training of Networks

For training purpose, the input $\mathbf{x}$ is associated with a provided target output t and can be compared with the neural network output y and evaluate the squared error.

$$E = \frac{1}{2}(t - y)^2 \tag{5-2-1}$$

Consider a single neuron j in the network. The outputs $\tilde{\mathbf{o}}$ from previous layer are fed into it, generating $n_j(\mathbf{x})$ from the weighted sum and neuron output o_j from the activation function φ.

$$o_j = \varphi\left(n_j\right) = \varphi\left(\sum_{k=1}^{N} w_{kj}\tilde{o}_k + b_j\right) \tag{5-2-2}$$

A commonly used example of activation functions is the sigmoid function defined as the following form along with its derivative:

$$\varphi(x) = \frac{1}{1+e^{-x}}, \quad \frac{d\varphi(x)}{dx} = \varphi(x)\left(1-\varphi(x)\right) \tag{5-2-3}$$

Compared with sigmoid and tanh, ReLU, defined as $\varphi(x) = \max(x, 0)$, reduces the likelihood of the gradient to vanish when $v_{lm}^{xyz} > 0$, and also increases sparsity when $v_{lm}^{xyz} < 0$. It is reported to have six times improvement in convergence compared to tanh [147] but could lead to the 'dying' of neurons. To fix the problem, Leaky ReLU gives a very small slope (e.g. 0.01) to the negative part and thus is $\varphi(x) = \max(x, 0.01x)$, PReLU assigned the slope to be a parameter and gives $\varphi(x) = \max(x, \alpha x)$, and the more generalized Maxout function uses $\varphi(x) = \max\left(w_1^{\mathrm{T}}x + b_1, w_2^{\mathrm{T}}x + b_2\right)$ with doubled number of parameters for each neuron. A graphical view of these activation functions is shown in Figure 5-3.

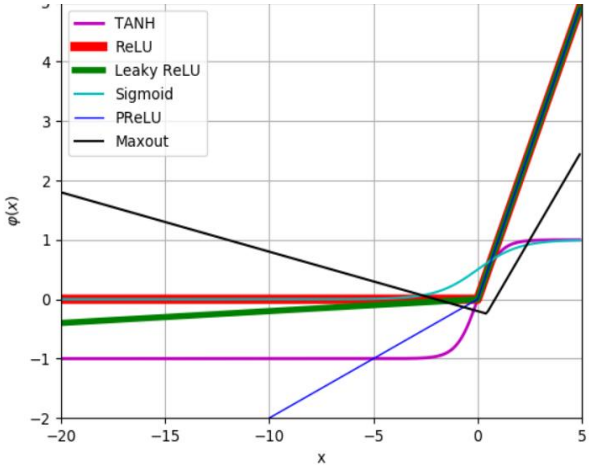


Figure 5-3 Different activation functions

In our experiment we use ReLU as our activation layer. The derivative of it is zero for negative x, one for positive x, and can be assigned to be 0 or 0.5 at point $x = 0$. But here we use

sigmoid function only for the sake of derivation. The partial derivative of the error with respect to each weight can be calculated using the chain rule:

$$\begin{aligned}\frac{\partial E}{\partial w_{ij}} &= \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial n_j}\frac{\partial n_j}{\partial w_{ij}} \\ &= \left[\frac{\partial E}{\partial o_j}\right]\left[o_j\left(1-o_j\right)\right]\left[\tilde{o}_i\right]\end{aligned} \tag{5-2-4}$$

The first term can be easily calculated if the neuron is the output neuron, meaning $o_j = y$ and thus $\partial E/\partial o_j = \partial E/\partial y = y - t$. But for a neuron from a hidden layer, the output of it will be inputs for neurons u, v, …, w from next layer, and thus the first term in the derivative is now

$$\begin{aligned}\frac{\partial E}{\partial o_j} &= \frac{\partial E\left(o_u\left(n_u\left(o_j\right)\right),\ o_v\left(n_v\left(o_j\right)\right),\ ...,\ o_w\left(n_w\left(o_j\right)\right)\right)}{\partial o_j} \\ &= \sum_{l=u}^{w}\frac{\partial E}{\partial o_l}\frac{\partial o_l}{\partial n_l}\frac{\partial n_l}{\partial o_j} \\ &= \sum_{l=u}^{w}\left[\frac{\partial E}{\partial o_l}\frac{\partial o_l}{\partial n_l}\right]w_{jl}\end{aligned} \tag{5-2-5}$$

Then the partial derivative of the error with respect to weight i of neuron j can be expressed recursively as follows.

$$\frac{\partial E}{\partial w_{ij}} = \left[\frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial n_j}\right]\tilde{o}_i = \delta_j\tilde{o}_i \tag{5-2-6}$$

where δ_j can be obtained from the all the derivatives with respect to the outputs o_l of next layer with

$$\delta_j = \begin{cases}\left(y-t\right)y\left(1-y\right), & \text{if } j \text{ is output neuron} \\ \sum_{l=u}^{w}\delta_l w_{jl} o_j\left(1-o_j\right), & \text{otherwise}\end{cases} \tag{5-2-7}$$

With the derivatives, gradient descent can be applied to update all the weights with a chosen learning rate η that describes step size along the gradient descent direction. And the updating on the weights are

$$\Delta w_{ij} = -\eta \delta_j \tilde{o}_i \tag{5-2-8}$$

5.2.3 2D Convolutional Neural Networks

In image processing, convolution operation can be used on an image with kernels for blurring, sharpening, edge detection, and more. The operation of a kernel $\mathbf{w}$ convoluting an image $\mathbf{I}(i, j)$ of size $m \times n$ can be written as

$$\mathbf{G}(i, j) = \mathbf{w} * \mathbf{I}(i, j) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \mathbf{w}(s,t) \mathbf{I}(i-s, j-t) \tag{5-2-9}$$

where the kernel has origin at center element and is of size $(2a+1) \times (2b+1)$. Outside the edge of the image is forced to pad zeros or same as edge entry, and the size of the output $\mathbf{G}(i, j)$ is either forced to be the same as the original image, or to be $(m-2a) \times (n-2b)$ if padding is not considered.

Table 5-1 Common kernel for 2D image processing

Function	Identity	Box Blur	Gaussian Blur	Sharpen	Edge Detection
Kernel	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$

Different kernel gives different effect on the original image, and commonly used kernels are listed in Table 5-1. The purpose of a convolutional neural network is to find the elements of a set of kernels, automatically through training, that discovers latent features of an image so that through multiple layers of perception the object in it can be identified at certain probability.

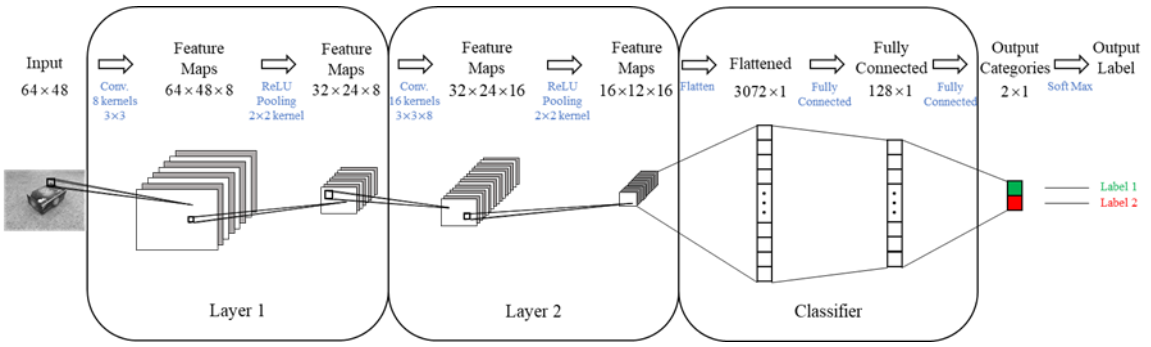


Figure 5-4 Example of a convolutional neural network structure for 2D image classification.

The convolution unit results in multiple feature maps with the same size of the input image. To reduce the number of parameters and thus reduce computation and control overfitting, a pooling layer as added after a convolution layer. Function like max-pooling, average pooling, or L2-norm pooling is used in a pooling unit so that every sub-block of the size of the kernel in a feature map will only generate 1 value.

Similar to the activation function mentioned in the neuron model, a ReLU layer is added between every convolution layer and pooling layer. And after several sets of Convolution-ReLU-Pooling layers, complex features are extracted with feature map size greatly reduced. The output feature maps can be flattened into a vector, which can be fed into a fully-connected neural network and generate predictions of the input image. An example of a convolutional neural network is shown in Figure 5-4, where the input image of size 64×48 is filtered by two layers of Convolution-ReLU-Pooling units to extract 16 feature maps of size 16×12, and then flattened to a vector and fed into a fully connected layer generating a prediction of two different classes. Finally, a SoftMax can be applied to generate the output label.

From an input image, the weights and biases in all the layers decide the output label. And the loss function can be defined as the mean of summed squared difference between predicted output y_i and label of training data t_i, i.e.,

$$E=\frac{1}{2N}\sum_{i=1}^{N}\left\|t_i-y_i\right\|^2 \tag{5-2-10}$$

In a backpropagation algorithm, all the weights and biases can be iteratively updated through gradient-based methods and generating models that provides higher prediction accuracy.

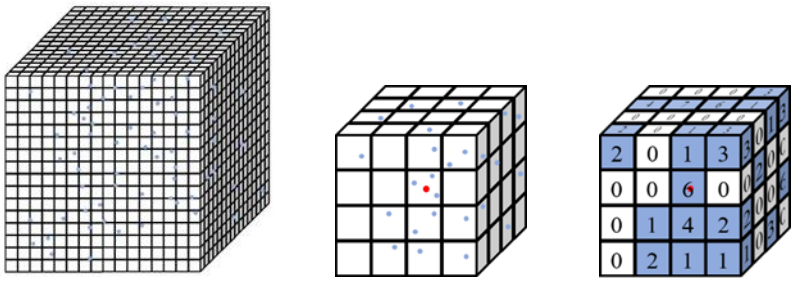
The use of CNNs has greatly improved the accuracy of image classification and segmentation especially with huge labeled datasets for model training. For example, models with 2D CNNs have about 99% accuracy in predicting handwritten digits from the MNIST database.

5.3 Identifying Shape Family with 3D CNNs

It is natural to move from 2D CNNs to 3D case since it is capable of extracting features automatically and these features could be even stronger for 3D data because of the added dimension. By extending image classification and segmentation to 3D to handle unordered point cloud, the reconstructed results from our SfM and MVS processes can be classified or segmented into different categories and thus helping the modeling or remodeling process.

5.3.1 Volumetric 3D Convolutional Neural Networks

In order for a convolutional kernel to be applied to a 3D point cloud in a similar manner as the 2D case, the datum with three coordinates must be preprocessed into a grid-like format, just like the pixels in an image. Volumetric representation can be used in this situation to generate voxel grids for the whole point cloud or around certain points of interest, as shown in Figure 5-5.



(a) (b) (c)

Figure 5-5 Voxelization (a) of whole 3D space, (b) of size 4×4 around a keypoint, and (c) occupancy as marked in blue and number of points in a voxel marked on surface

For classification of the whole point cloud, voxelization can be conducted by dividing the bounding box of the cloud into $N \times N \times N$ voxel grid with every voxel indexed. The occupancy of each voxel can be easily decided by setting it to be 0 if no point lies inside or 1 otherwise. Or the value of each voxel can be the number of points within it. Thus, the voxelization process generates an $N \times N \times N$ tensor to represent the original unordered point cloud. The voxelized data can then be processed in a similar manner as the 2D CNNs case.

For our case, the scanned 3D point cloud usually contains background and a plane surface at bottom like the ground or the tabletop. And there are also cases where multiple objects are scanned into one point cloud and need to be identified separately. The same voxelization procedure can be applied, but instead of on the whole point cloud, now on a fixed bounding area $[X-R, X+R] \times [Y-R, Y+R] \times [Z-R, Z+R]$ around every point (X, Y, Z) or some sampled point of interest, as shown in Figure 5-5 (b) and (c). For example, in one of our experiments, $N = 10$ and $R = 1$, then the voxel grids around a keypoint have 1000 voxels of size $0.2 \times 0.2 \times 0.2$. Then each point can be classified individually due to different point distribution around it within certain range, and thus a segmentation of the whole scanned point cloud can be obtained.

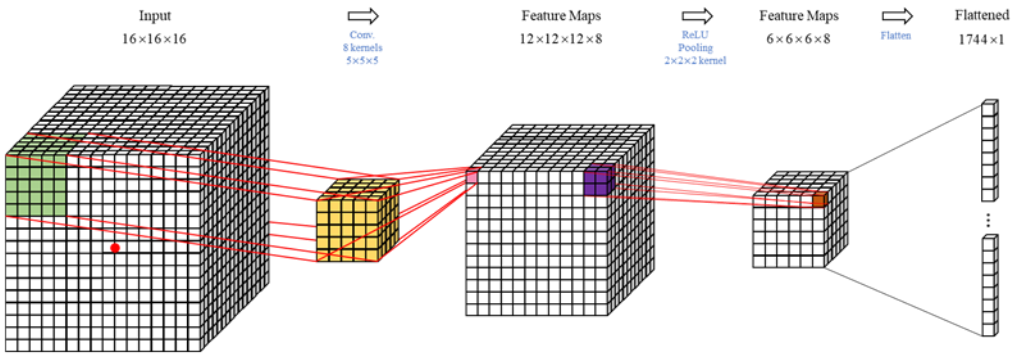


Figure 5-6 Example of a 3D Convolution-ReLU-Pooling layer with a $5 \times 5 \times 5$ kernel on $16 \times 16 \times 16$ voxel grids around a point without considering padding, and $2 \times 2 \times 2$ kernel for pooling, followed by flattening.

An example of one Convolution-ReLU-Pooling layer in 3D CNNs is shown in Figure 5-6, where a 3D convolution and 3D pooling will be applied on the voxel form of data and generating

flattened vectors and can then be fed into a fully connected neural network and predicting category of the voxel grids center.

In the l-th 3D convolutional layer, Q inputs of voxel grids or feature maps of size $n \times n \times n$ is convolved by d kernels of size $f \times f \times f$, which can be denoted as $C(n, d, f)$. As a result, on the m-th output feature map at location (x, y, z), the value is

$$v_{lm}^{xyz} = \sum_{q=1}^{Q} \sum_{i=0}^{f-1} \sum_{j=0}^{f-1} \sum_{k=0}^{f-1} w_{lmq}^{ijk} v_{(l-1)q}^{(x+i)(y+j)(z+k)} + b_{lm}, \quad m = 1, 2, ..., d \tag{5-3-1}$$

where w_{lmq}^{ijk} is the weight factor at position (i, j, k) of the kernel convolving the q-th input feature map to generate the m-th output feature map, and b_{lm} is the bias for the m-th output feature map.

After the convolutional layer, an activation layer is applied to increase nonlinearity. Just like the sigmoid function with updating function $\varphi(v_{lm}^{xyz}) = 1/1 + e^{-v_{lm}^{xyz}}$, a ReLU unit is simply updating each feature value with $\varphi(v_{lm}^{xyz}) = \max(v_{lm}^{xyz}, 0)$.

In the l-th 3D max pooling layer, on Q inputs of feature maps of size $n \times n \times n$, max pooling is applied through a $g \times g \times g$ kernels, which can be denoted as $P(n, g)$. As a result, on the m-th output feature map at location (x, y, z), the value is

$$v_{lm}^{xyz} = \max_{i,j,k \in \{0,1,...,g-1\}} v_{(l-1)m}^{(gx+i)(gy+j)(gz+k)}, \qquad m = 1, 2, ..., d \tag{5-3-2}$$

However, before voxelization, the input point clouds for training or testing purpose need to be normalized, like how the images need to be rescaled and down-sampled into the same size for 2D CNNs.

5.3.2 Preprocessing of Point Clouds

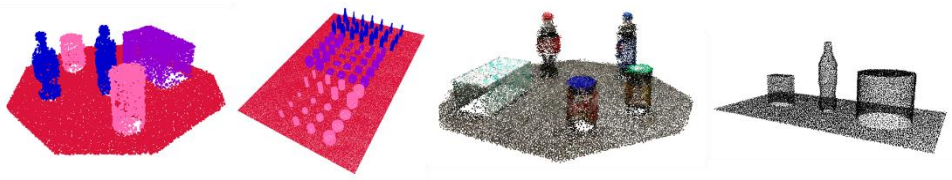
The preprocessing steps for point clouds include normalization, re-sampling, labeling, and voxelization. The reconstructed point clouds are up-to-scale, indicating that even two reconstructions of the same scene could have different scales. But the size of the voxel grids and size of each voxel are fixed, meaning that we have a 3D window of fixed size looking at different areas of the whole point cloud. If clouds are not normalized, we could end up having a window that observes only the cap of a scanned bottle or a chair that only takes 1% of the window volume but trying to identify the class of the target within the window. The true scale of the scanned point cloud can be recovered either through equaling a true measurement or a calibration object in real scene with the distance between the corresponding two reconstructed points, or from the knowledge of the true extrinsic calibration between two cameras. The second method has to involve calibrated devices like a stereo camera or set of cameras with fixed positions and poses. The first method can be very error-prone but here for the use of classification, a scale factor close to the true factor is sufficient for this task.

After being rescaled, a 3D point cloud could have very different point density from other scans. And density also varies a lot within the point cloud itself at different areas due to the distribution of strong image features and number of images taken from certain angles. For classification or segmentation of the scene, a simplified point cloud could still carry the shape information in a more uniformly distributed manner but at the same time largely reduce the computation. Thus, in our preprocessing stage, all the point clouds that are factored into the same scale are then down-sampled to have the same mean distance between closest points. The distance is chosen to be smaller than the voxel size so that the occupancy or point counts representation will still be valid.

Then for training purpose, every point in the rescaled and simplified point clouds are labeled manually for their categories, and then voxelized around every point within a bounding box as illustrated in section 5.3.1.

5.3.3 Classification and Segmentation System

As mentioned before, our system uses volumetric segmentation with 3D CNNs as the approach towards shape family identification, which can still be used to predict the class of a whole point cloud if only one object was scanned. The inputs of our segmentation system are the preprocessed and labeled point clouds that came from our scanning experiments and also from corrupted sampled CAD surface model to extend our training dataset. For evaluation purpose, we label the testing data as well. Some of our labeled point clouds and testing point clouds are shown in Figure 5-7. All the objects in the testing point clouds are of different size from the training models.



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

Figure 5-7 Preprocessed point clouds. Example of (a) labeled point cloud reconstructed from image; (b) labeled point cloud generated by corrupting sampled CAD surface model; (c) to-be-labeled point cloud reconstructed from image; and (d) to-be-labeled point cloud generated by corrupting sampled CAD surface model.

As shown in Figure 5-8, our system includes a batch training module to train a 3D convolutional neural network for classification and a testing module that use the trained model to predict label of each input.

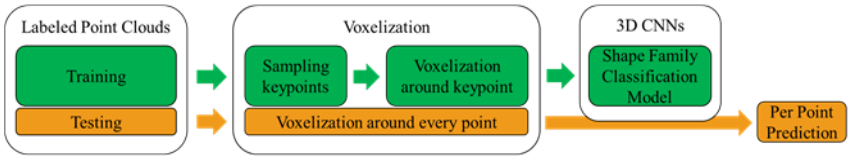


Figure 5-8 Training and testing the Segmentation Model

In the training module, a set of keypoints are randomly sampled from all labeled point clouds with balanced number in all the categories so that category bias from input dataset is eliminated. Then the previously mentioned voxelization process is applied around each of these keypoints to generate voxel grids that contains the point counts within each voxel. The voxel grids along with their labels (of the center keypoints) are then fed into the 3D convolutional neural

network, which predicts the probability of every point being of each category with two sets of convolution-ReLU-max pooling layers, a fully connected layer, and a softmax producing integer output class. The errors between all the predictions and the original labels are then used in the same manner as derived in section 5.2.2 to backpropagate through the model and update weights iteratively to train a prediction model. The process illustrated with simplified examples is shown in Figure 5-9.

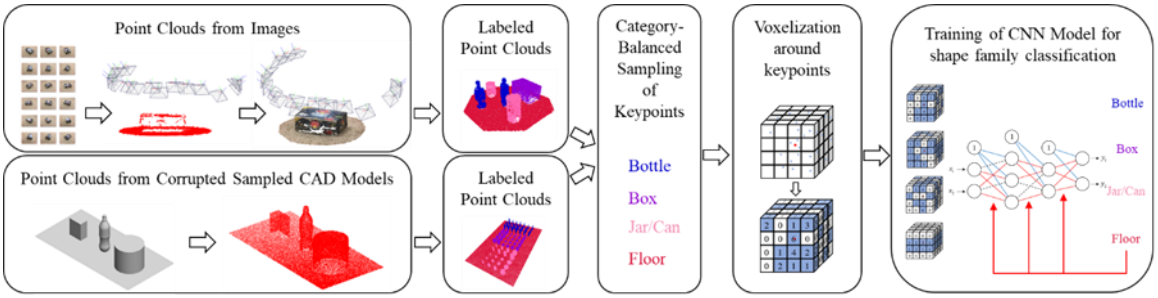


Figure 5-9 Process for training of 3D CNN model for shape family classification illustrated with simplified examples.

The testing module will take the preprocessed point clouds as input and apply voxelization around every point and feed the voxel grids into the trained network to generate per-point category prediction of input point clouds and thus resulting in segmentation of them, as shown in Figure 5-10. And the results are compared with ground truth labels to examine the performance.

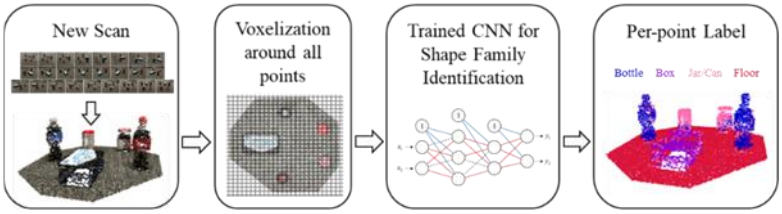


Figure 5-10 Process for testing trained 3D CNN model for shape family classification with newly scanned point cloud.

5.3.4 3D Convolutional Neural Network Layout

The architecture of our 3D CNN model includes two 3D convolution layers, two max pooling layers, and a fully connected layer. The output label is generated by softmax. Figure 5-11

shows the architecture with the example that choose $N = 20$ as size of the voxel grids generated around a keypoint (shown in blue in the picture). The $20 \times 20 \times 20$ input is first convolved with 32 $3 \times 3 \times 3$ kernels with random initialized values to generate 32 $20 \times 20 \times 20$ feature maps. After ReLU is applied to each value, these 32 feature maps are fed into a max pooling layer of kernel size $2 \times 2 \times 2$ to extract the largest size within each $2 \times 2 \times 2$ sub-block, and thus the size of the 32 output feature maps are reduced to half as $10 \times 10 \times 10$. The next convolutional layer takes these 32 feature maps and convolves them with 64 different kernels of size $3 \times 3 \times 3 \times 32$ and generate 64 feature maps of size $10 \times 10 \times 10$. And after that another max pooling layer downsize the feature maps to $5 \times 5 \times 5$. The 64 final feature maps of size $5 \times 5 \times 5$ are then flattened into an 8000×1 vector and fed into a fully connected layer with 1024 nodes, and then to the final output layer with 4 outputs, end up with the predicted label through a softmax.

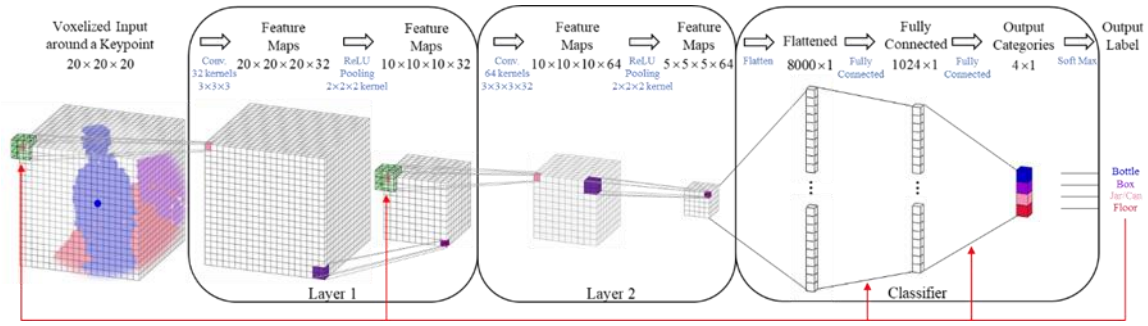
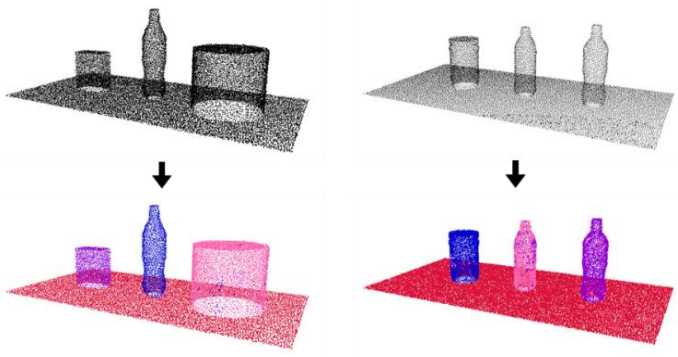


Figure 5-11 3D CNN architecture for classification.

5.4 Results of Shape Family Identification

We examine our shape family identification method with point clouds generated from both CAD model sampling and image-based reconstruction. And we present both qualitative results and quantitative results to demonstrate the performance of the system.

5.4.1 Qualitative Results



(a) (b)

Figure 5-12 Qualitative labeling results for CAD-generated point clouds
(a) Box, bottle, can, and floor; (b) Coke bottle, Pepsi bottle, jar, and floor

Figure 5-12 shows two qualitative results classifying CAD-generated point clouds into four different types of shape families, which are visualized with different color. For each different combination of classification targets, a specific model is trained with labeled point clouds of the target shapes and then used to test new input point clouds. We choose $N = 10$ and $R = 0.75$ for voxelization, meaning for every keypoint, we have 1000 cells of size $0.15 \times 0.15 \times 0.15$ around it. Our training set is a collection of corrupted CAD-generated point clouds of the target shape classes with different parameters. The parameters for the testing purpose are chosen to be different from any of the model in the training set. The results in the figure show clear separation across different types of object. And it suggests the potential for a hierarchical structured classifier to identify the shape family of a cluster and further identifying sub-families of them, just like in the example how a bottle could be identified out of different shapes and then further identified to be a Coke bottle, a Pepsi bottle, or by any chance a wrongly classified jar.

5.4.2 Quantitative Results

For quantitative results, we use three examples to demonstrate the performance of our shape family identification system. The testing set is manually labeled, and the classification results

are compared with ground truth labels to generate a confusion matrix which gives the probability of shape classes being each class available.

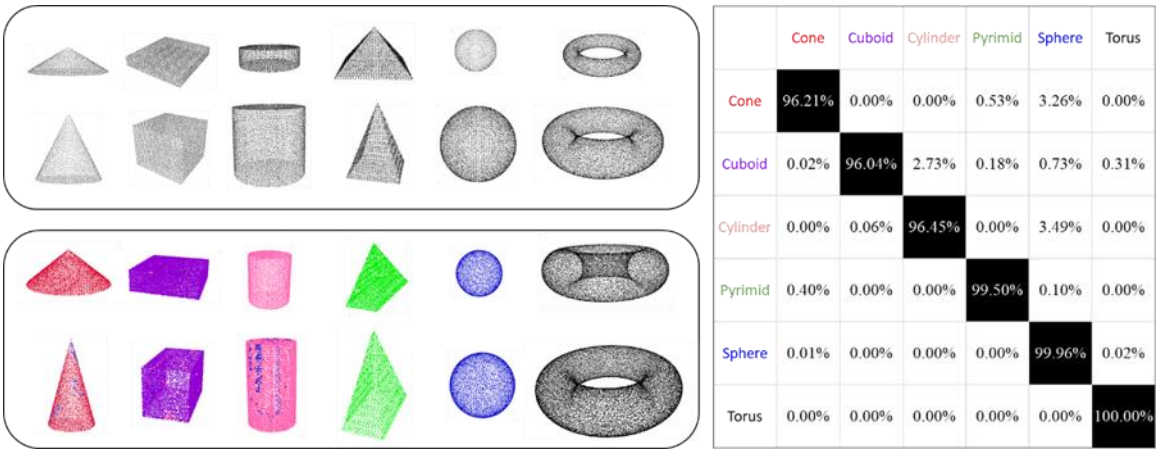


Figure 5-13 Quantitative results for six types of different 3D primitives: selected point clouds from training set, testing set, and the generated confusion matrix (darker color suggests higher prediction accuracy)

The first example is conducted on Gaussian-corrupted CAD-generated point clouds of 6 different types of 3D primitives: cone, cuboid, cylinder, pyramid, sphere, and torus. The top left section in Figure 5-13 shows two examples of labeled training clouds for each class of primitive. Each class in the training set has about 45 normalized point clouds with originally 50,000 points, which are sampled from CAD surface model and corrupted with Gaussian noise with standard deviation $\sigma = 0.003$, and then down-sampled with voxel size 0.02 to about 1000 to 7000 points depending on the physical size of the model. The parameters of the primitives vary within a range of common tabletop object size from about 50mm to about 300 mm. For voxelization, we choose $N = 10$ and $R = 0.5$, meaning for every keypoint, we have 1000 cells of size $0.1 \times 0.1 \times 0.1$ around it. The category-balanced sampling resulted in 102,543 points for each shape class, and thus the size of labeled training set is 615,258. The testing primitive point clouds have different parameters from all the training data. In total we have 63 testing point clouds, preprocessed in the same way as the training data and labeled manually as ground truth. The lower left section in Figure 5-13 shows two selected results for each shape class, and the right side chart shows the averaged confusion matrix for predicting each class of shape family. And the diagonal entries are all above

96%, suggesting a very high accuracy in predicting shape family of a point cloud of a single primitive.

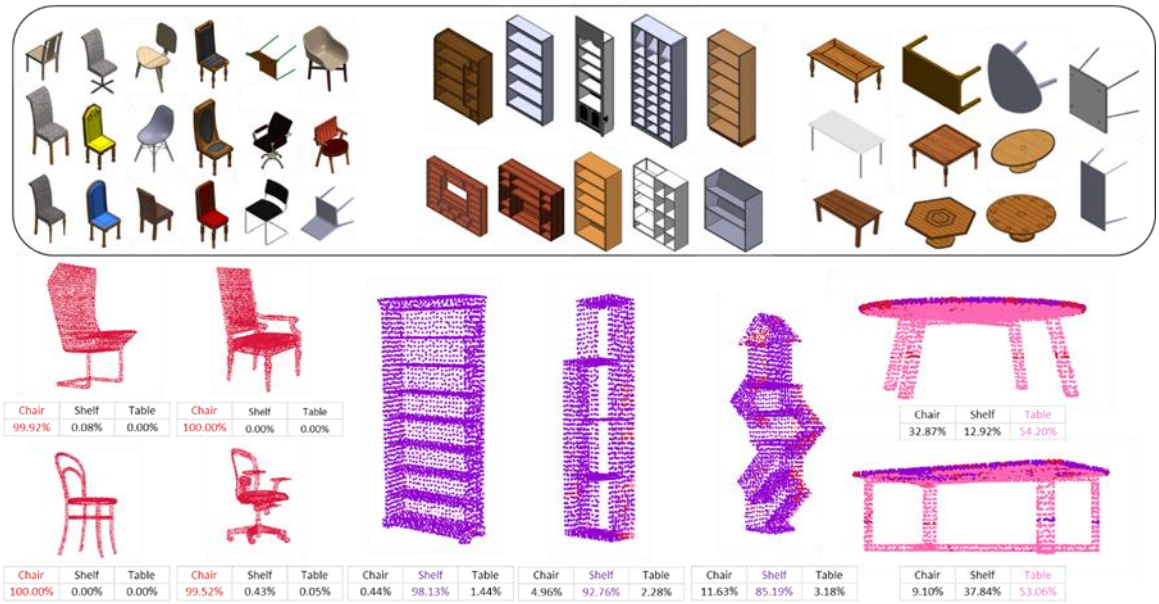
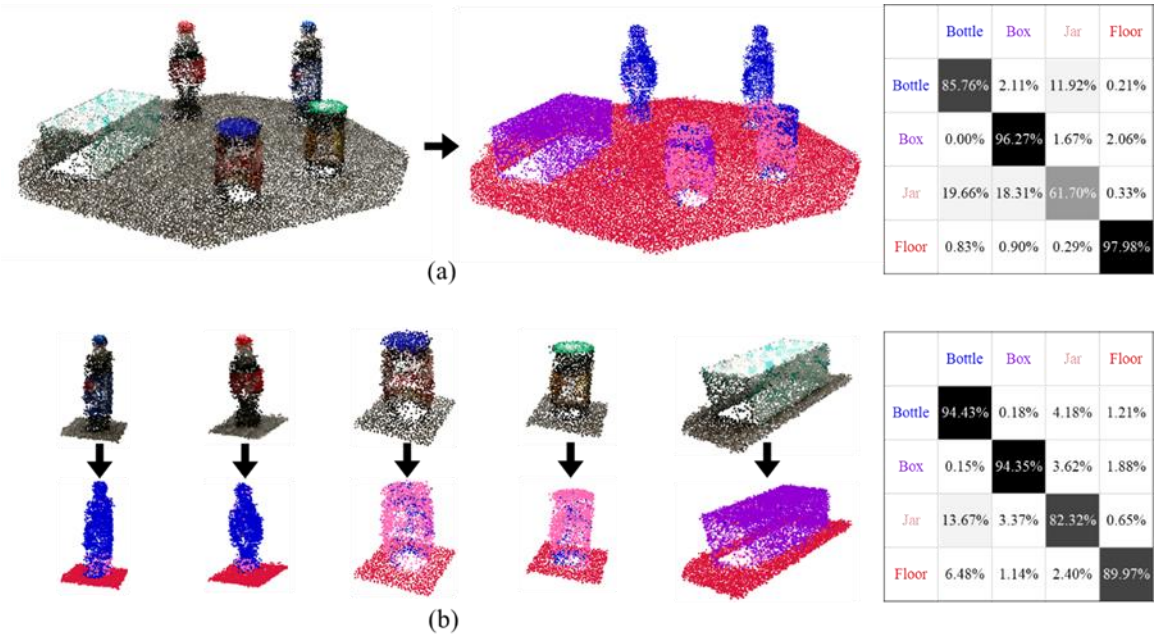


Figure 5-14 Quantitative results classification of chair, book shelf, and table: training set shown in CAD model, testing set shown in labeled point clouds, and the prediction for each class.

The second example is also on corrupted CAD-generated point clouds, but the shapes, i.e., chair, bookshelf, and table, are more irregular than basic primitives. Similar process is done as described for the primitive example. We have all the point clouds normalized to be confined in a unit sphere, and choose $N = 10$, $R = 1$, and $\sigma = 0.005$. To enlarge the training set, we also randomly rotate each point clouds 3 times, and resulted in 164,847 points for each class, i.e., 494,541 data for training. The training set (original CAD model), testing set (shown in point clouds), and percentage of prediction of each class are shown in Figure 5-14. The results show more than 99.5% accuracy in identifying chairs, more than 85% for shelves, and 53% for table. Although the accuracy is lower for tables, the dominating prediction of the cloud is still correct so that the whole cluster can be classified as a table.

The third example is on identifying point clouds generated from image-based reconstruction. Here for the preparation of point clouds, we choose $N = 20$, and $R = 2$. As

mentioned before, we also use corrupted CAD-generated point clouds with $\sigma = 0.02$ to enlarge our training set, ending up with 28,947 points after category-balanced sampling for each shape class, i.e. 115,788 labeled points for training. The testing point cloud is a newly scanned scene with all the objects different from any of the targets in the training data set. Figure 5-15 (a) shows the testing point cloud, the classified label for each point, along with the confusion matrix, which shows high accuracy prediction on box and floor, and less satisfying but still correct dominating prediction for bottles and jars. One of the reasons here is the small size of the training set, which could be enlarged by introducing more scanned and labeled point clouds of different shaped boxes, bottles, and jars of different sizes.



	Bottle	Box	Jar	Floor
Bottle	85.76%	2.11%	11.92%	0.21%
Box	0.00%	96.27%	1.67%	2.06%
Jar	19.66%	18.31%	61.70%	0.33%
Floor	0.83%	0.90%	0.29%	97.98%

	Bottle	Box	Jar	Floor
Bottle	94.43%	0.18%	4.18%	1.21%
Box	0.15%	94.35%	3.62%	1.88%
Jar	13.67%	3.37%	82.32%	0.65%
Floor	6.48%	1.14%	2.40%	89.97%

Figure 5-15 Quantitative results for identifying bottles, boxes, jars, and floor from scanned point cloud: (a) result of segmentation of the whole point cloud and confusion matrix, (b) results for single point cloud (separated from example (a)) and confusion matrix.

As shown in Figure 5-15 (b), the overall accuracy is raised up by separating the clusters of objects into different files and apply voxelization on each of them separately. The reason is that the choice of voxel size is selected so that majority of a cluster can be included inside the voxel grids, e.g., a keypoint at the cap area of the bottle should still have a voxel grids around it that covers the

bottom of the bottle. However, when the scanned point clouds are close to each other with distance smaller than R, part of other shape families are also included in the voxelized result and thus lowering the accuracy of prediction. But if only one object is scanned, or if the scanned targets are away from each other for about the size of itself, our system gives prediction for bottle, box, jar, and floor at accuracy higher than 82%.

5.5 Separation of Labeled Clusters

From the segmentation results, the labeled clusters can be extracted by first taking out the floor and then grouping the rest of points into clusters. Every cluster can then have a voting on the highest predicted class as the label for the whole cluster, assuming every cluster contains only one object of one shape family. A certain class of shape family will be linked to a generic surface model. One simple example is the model for a basic primitive like a cylinder with parameters like diameter and height.

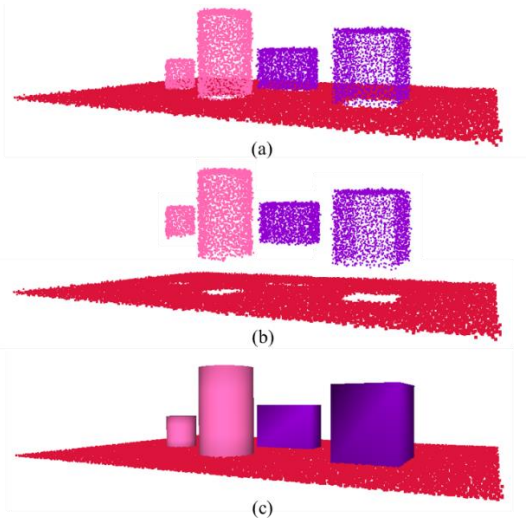


Figure 5-16 Example of separating a segmented point cloud into clusters after taking the floor out, and replacing clusters with mesh models with estimated locations and parameters of the dominant class.

Here we show an example of separating a segmented point cloud of cylinders, cuboids, and floor. After taking out the floor points, we randomly sample a point from the rest of the points and

obtain the k-nearest points around it within a certain distance. Then for every point within the distance repeat the operation to explore another layer of points until no points are within the range. The clustered points are taken out as a cluster and another point is sampled from the rest of points and repeat the same process until all points are explored and clustered. The dominant shape class of every cluster will be chosen as the class for it and all the points in this cluster are used to estimate the location and parameters of the generic model. The segmentation results, separated results, and the generated geometry meshes are shown in Figure 5-16.

5.6 Conclusions

In this chapter, we discussed the motivation and theories behind identification of 2D and 3D data with a bio-inspired artificial neural network model. And we presented our shape family identification process with a 3D convolutional neural network model for CAD-generated or scanned point cloud classification and segmentation. We use Gaussian-corrupted CAD-generated point clouds to enlarge the training dataset. And the steps of preprocessing, voxelization, training and testing setup methods, and the architecture of the CNN model are illustrated in detail.

The overall accuracy for CAD-generated results are higher due to easy access of large amount of training data with CAD software. For point clouds generated from image-based reconstruction, although with lower accuracy (about 61%), dominant prediction of shape family remains correct. And it is shown that accuracy can be further increased if the distance between objects are larger than the size of targets, or if scanned point clouds can be pre-separated into clusters of objects. In the end we also demonstrated the method to process the segmented data, which can be used to cluster a labeled point cloud into groups of points, vote for the dominant shape class for each cluster, and replacing points with a mesh model of parameters estimated from the points in the cluster.

CHAPTER 6

A LEARNING-BASED PRODUCT SHAPE FAMILY DESIGN AND MODELING FRAMEWORK

Nowadays the pressure of expanding product variety pushed by demanding customers and competitive market requires much faster conversion rate from market demands to design ideas and to new launches of products. In this chapter, we introduce our learning-based product shape family design and modeling framework targeting at increasing the efficiency of conceptual design or redesign of product shape by

(1) lowering the technical barrier for non-CAD-modelers to easily create 3D models with the help of image-based reconstruction and 3D shape family classifiers,
(2) simplifying design and editing process with a library of modularized 3D models, and
(3) providing an easy way for new design generation and modeling through combination of modules from different models.

We start with a review of some previous work on product family design, and the concept of our hierarchical product shape family is introduced. Then the details of the design framework are illustrated, including how image-based reconstruction and 3D CNN classifier described in previous chapters are integrated to work with a model database, how a redesign or a new design can be generated from modularized model, and how the design framework can be improved through the use of it to enlarge model database, to increase classifier capability and accuracy, and to expand

possibilities for product conceptual design. Design examples using this framework are presented at the end of this chapter to demonstrate the design process and capability.

6.1 Hierarchical Product Shape Family Design and Modeling Concept

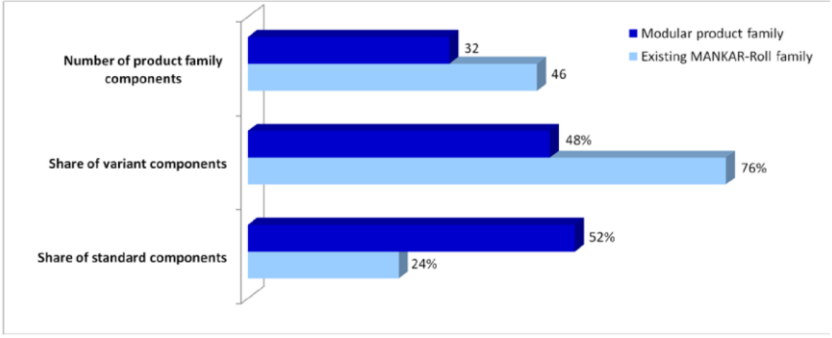
6.1.1 Product Family and Product Platform

The introduction of product family design and platform-based product development is to address the dilemma for companies to increase product variety for marketplace but with as little variety as possible to remain profitable. According to Simson et al [157], a product family can be defined as a group of related products that share common features, components, and subsystems; and satisfy a variety of market niches. And the set of common parameters, features, and/or components that remain constant within a product family is a product platform, from which different products of this family can be derived. Platform-based product development has been an effective way to strategically leverage the combination of platform parameter scaling and module swapping for product variety in a cost effective manner. The goal of most product family design approaches [157-159] is to identify the optimal commonality decision so that minimal cost is achieved within certain performance requirements. Some other approaches also take broader enterprise considerations into account, like market share [160] or sustainability [161].

Modularization plays an important role in product family design. PKT Institute developed an integrated approach for modular product family development [162], through design for variety and life phase modularization, to generate maximum external product variety using the lowest possible internal process and component variety. Figure 6-1 (a) shows an example of the MANKAR-Roll family of spraying system for herbicides, and Figure 6-1 (b) and (c) shows the design results of the approach with reduced number of components but additional adaptation to

specific needs of product life, and high share of standard components within the platform. A more systematic review on modular product design can be found in [163].

(a) (b)



(c)

Figure 6-1 (a) Product variants of the product family of herbicide spraying system MANKAR-Roll
(b) Modular sales concept of the MANKAR-Roll family with integrated aspect of ergonomics and corporate styling
(c) Results of the case study on spraying systems [162]

6.1.2 Product Shape Family and Product Shape Platform

In our research, similar to the ideas of product family and product platform, we define the product shape family as a group of products that share an iconic 3D shape but with small variations on both internal and external modules and parameters. The shared iconic 3D shape is the product shape platform, which consists of modules of parameterized complex geometries so that variants of the product shape family can be derived from scaling of parameters and swapping of modules.

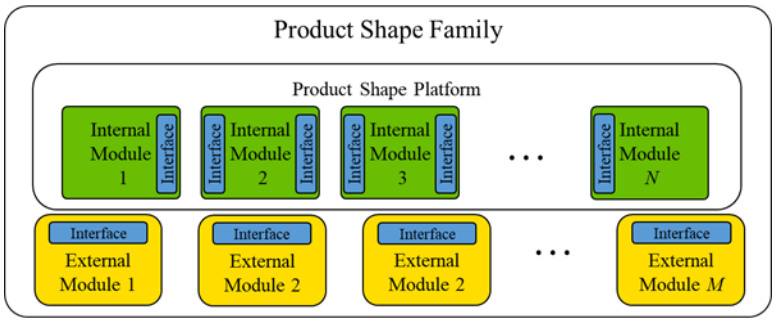


Figure 6-2 Modules of a product shape family

For example, bottles share an iconic shape that is rotational symmetric with a gradually narrowing opening mostly for pouring purpose, and thus can form a large product shape family with the platform that consists of modules like closure, neck, shoulder, body, and bottom. For each module, a profile can be defined with some control points and then be used to revolve the geometry around the central axis. Additional features could be included like twisting or bending of the whole shape module, or a circular patterned cut on it. Constrains can be predefined at the interface between each two modules to assure connectivity and surface smoothness. Also, optional external modules like dispensers can be used to further increase product variety.

There are many benefits of using shape family for conceptual product design and modeling. To start with, design and modeling are carried out in parallel, which gives users a much more holistic visualization of the design and any variation applied on it in real time. Secondly, users even with limited CAD experience can now avoid the laborious operations of setting up sketches and applying features to create an initial model but start directly on a generic platform model. Thirdly, editing of a design is now simplified to choosing desired geometry template for each module and dragging control points. Also, the iconic shape of the product family can be preserved but providing users with the freedom to create unlimited number of variants through combinations of changes on different modules. Moreover, it allows creative design ideas to be easily carried out like to switch off a seemingly important module of the product shape platform or to even replace it with a module from another shape family. Last but not least, the modularization of every product shape family as

well as the templates of every internal or external module can be reconfigured or expanded while causing minimum complication of user operation but additional options for more design ideas.

6.1.3 Hierarchical Product Shape Family Structure

Figure 6-3 Example of ancestor generations of a Coke bottle family branch.

Take the bottle shape family for example, there are many subclasses of bottle types which also share an iconic shape that can be easily recognized, like soda bottle, beer bottle, wine bottle, etc. Further going into the soda bottle family, Coca-Cola® bottles and Pepsi® bottles have their own unique shape and thus can be considered the next generation of product shape families. Towards the other direction, bottle family can also be considered as a part of a larger rigid container family, and further of a broader container family. Thus, we can consider Coke bottle as a small branch of a huge hierarchical product shape family tree. And the ancestor generations and some examples are shown in Figure 6-3.

One of the reasons to use a hierarchical family structure is that it provides different levels of predefined product shape platforms for different levels of design and modeling flexibility. Another reason is that the taxonomy based on product shape is intuitive to human since the shape pattern at each level of the tree is easily recognizable. Also notice that further downward of a branch shows less shape difference, thus we can expect good performance of a biomimetic classifier to tell the difference between families within a generation under the same family, and then move down into the next generation and apply classification recurrently.

6.2 Design Scenarios

We discuss two types scenarios, redesign and new design, to illustrate the need of a tool applying this framework and the potential improvement of design efficiency it could lead to.

The first type of scenarios is to redesign a product that already exists, but an easy-to-edit model is not accessible, like to redesign a Coke bottle or a dining chair. It could be commercially confidential and thus is not provided to a market analysist or a customer from a panel. Or the only accessible model is a CAD file which is tedious and difficult for non-CAD-users to edit, not to mention while talking about it. Or it could simply be a product from another company. In these cases, taking a few photos of the product with a smart phone would result in a generic model that is close to the target and easy to edit with its modularized setup and pushing-and-pulling style operations.

The other type is to design a brand new product that is totally different from the iconic shape of current product or similar competing products. But the design idea must come from an object of certain shape, like to design a chair that has the shape of a Coke bottle, or to design a bottle that has the shape of a fire pump. Or it could be a made-up shape in mind where no counterparts are at hand, but it won't be too hard to roughly create the shape with some industrial clay or even with playdoh. In all these cases, both objects from the target shape family and the inspiring source family can be scanned to obtain two models and through combination of modules a new design can be derived.

A special occasion is that when no modularized model of a desired product shape exists in the library, then the result of classification would provide the families with their shapes closest to that of the scanned product. And the module in the selected family that carries the desired shape can then be easily modified to a variant closer to the initial design idea.

6.3 Design and Modeling Framework

Our learning-based product shape family design and modeling framework consists of the following components, and the way they work together towards better design efficiency will be illustrated in detail in this section.

(1) an image-based 3D reconstruction tool,

(2) a library of different product shape family trees,

(3) a database that stores scanned point clouds and associated modularized models,

(4) series of 3D CNN-based shape family classifiers, and

(5) a modularized shape family design and modeling methodology.

6.3.1 Integration with Reconstruction and Classification

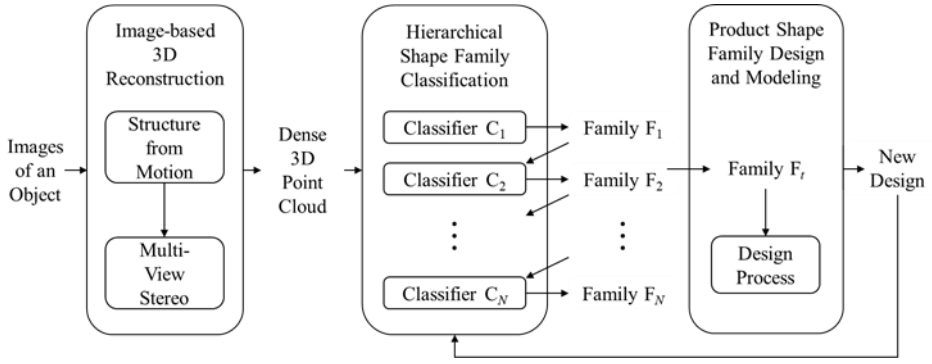


Figure 6-4 Integration of product shape family design with 3D reconstruction and 3D shape family classification.

There is large potential to integrate 3D reconstruction into the design and modeling process, but only if the scanned data can be transformed into an editable model. Since here our main drive is to simplify the design and modeling process, instead of applying surface reconstruction and feature extraction methods to generate a feature-based CAD model, usually through synthesizing detected surfaces, edges, and vertices information, we directly use a series of hierarchical 3D shape classifiers, as described in Chapter 5, to understand the raw scanned point clouds and to provide user with a few branches of most possible product shape families along with their generic shape

product models. Users can then select the desired product shape family to start the design process on a 3D model directly.

The 3D classifier not only works as the converter from scanned data to shape family models, but also benefits from every design process since the input scan as well as the new design can be fed back into the training process and provide additional labeled training data such that the classifier model can be updated to higher accuracy.

6.3.2 Shape Family Library and Model Database

Every product shape family tree consists of a number of descendent generations of product shape families. And for design purpose, the category spectrum of the product shape family trees should be as wide as possible to cover a broader range of different products. As a result, there are large number of nodes in the trees and each node represent a certain product shape family. The information of these trees can then form a library by linking each product shape family with its parent family and all of its child families. In this way the ancestors or descendants for any shape family can be traced easily.

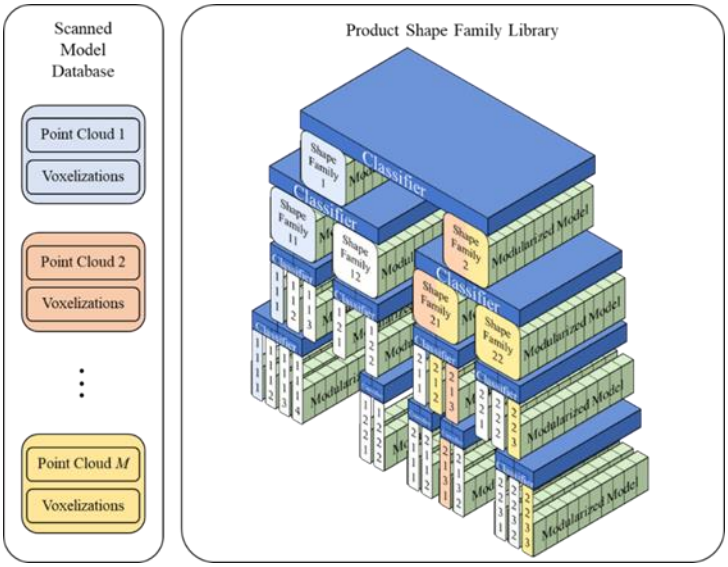


Figure 6-5 Visualization of the structure of Shape Family Library and Scanned Model Database

A specially trained classification model is linked to each product shape family to identify the shape sub-family right under it. For example, linked to bottle family there is a classifier to further identify whether a bottle is more of a milk bottle, a beer bottle, a wine bottle, a soda bottle, or a cosmetic bottle. Also linked to each product shape family are the related models, in forms of (1) 3D point clouds, (2) labeled voxelization results, and (3) a modularized shape model. All the 3D point clouds along with their labeled voxelization data form a training database for shape family identification. And each of them will have links to multiple product shape families, including every family in the branches related to all the products in the scanned point cloud. For example, if a scan contains a Coke bottle and a dining chair, then it is linked to every shape family in the Coke bottle branch (Coke bottle, soda bottle, bottle, rigid container, and container) and also every shape family in the dining chair branch (dining chair, chair, and furniture). Thus, the label of each voxelization data will also have a hierarchical structure, so that for training of a classifier for certain generation, the specific label will be used. The results from voxelization can be updated at any time from the raw 3D point clouds with a different set of voxelization parameters. The labeled results will be used repeatedly to further train the classifiers with newly added data.

Modularized shape model is another important section in the shape family tree library. For each shape family node, there is a modularized model specially defined for it. As introduced in section 6.1.2, every shape family has a product shape platform and a few external modules. For every internal module and external module, interfaces are predefined with the boundary connection constraints.

6.3.3 Hierarchical Product Shape Classification

Hierarchical classification in recent years has been applied on large-scale classification of textual[164, 165] and visual data [166, 167] due to the inherent hierarchical structure within the classes. In these cases, it would be very difficult for flat classification to cover hundreds of

thousands of classes especially when training set is highly unbalanced. On the other hand, hierarchical methods need to make intermediate decisions prior to reaching any final category and thus the accuracy may decrease due to error propagation. However, in our research, we are not training a series of classifiers for the best returning final winning label. Instead, all the classifiers are independent and are trained separately, and our objective is to facilitate the design process by listing a few of the most possible branches for user to select.

The classifier after each product shape family node will provide the probabilities of an input point cloud to be any of the shape families in the next generation. Apparently, it provides more flexibility to list a range of choices instead of only returning the winning shape family branch. We use chain rule to propagate predictions through all parent nodes to calculate the probability at each family node, i.e.,

$$\begin{cases} \hat{P}_k = P_k \ , & \text{for any ancestor node } k \\ \hat{P}_{i,j} = \hat{P}_i P_{i,j} \ , & \text{for node } i\text{'s subfamily node } j \end{cases} \tag{6-3-1}$$

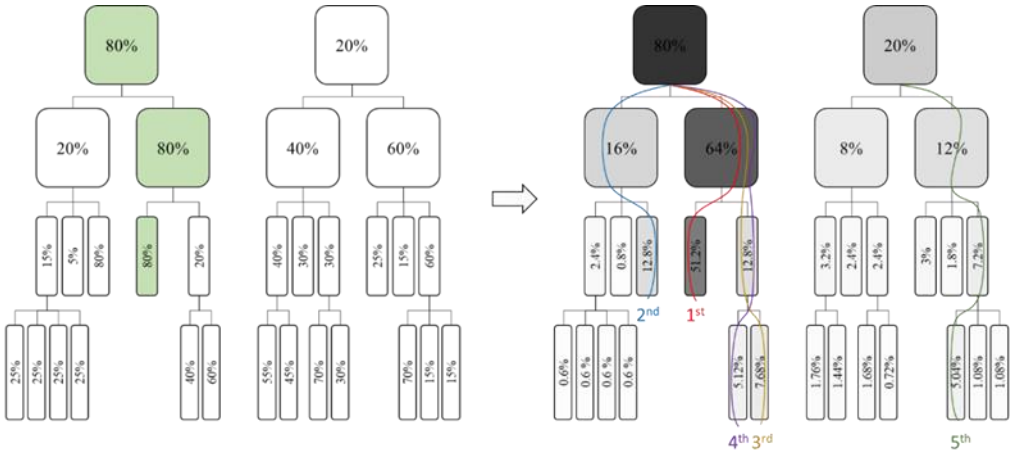


Figure 6-6 Comparison between returning winning shape families and returning the best 5 branches with probabilities calculated by chain rule. In the left image, winning blocks are shown in green. And in the right image, darker block suggests higher probability, and colored curves shows the best guesses.

The probabilities at all the end nodes, or leaves of the tree, are then sorted to return a few best guesses of product shape families along with all the parent nodes of them. An example illustrating the idea is shown in Figure 6-6.

6.3.4 Design and Modeling Process

The detailed steps of the design and modeling process is listed in Table 6-1, where both user and system operations along with the output are described for each design step. Notice that the order of the steps can be changed for editing modules and shapes, i.e. step 4 to 8. And through multiple editing steps, a new model can be generated.

Table 6-1 Steps for Design and Modeling with the Framework

Step	User Operation	System Operation	Output
1	Take pictures around existing products or clay models	3D Reconstruction	3D point clouds
2		Hierarchical classification	Best 5 family branches for each cloud, preview of shape platform
3	Select the desired model(s)	Load modularized shape models(s) and adjust to closet size	Shape model(s)
4	Select internal and external modules to delete, add, or swap	Connect modules with interface constraints	Shape model(s) with desired modules
5	Adjust global size by pushing or pulling	Update size of every individual module	Shape model(s) with desired size
6	Select a module to edit	Load geometries and control points	Preview of shape of the module and shape of whole model
7	Edit by pushing or pulling control points	Update geometries of the module	Preview of updated module shape and updated model shape
8	Select material texture, color, and decal on module surface	Update rendering	Rendered shape model
9		Sample points from new model surface	Point cloud model of the new design
10	Select product shape family label of finished design	Apply voxelization and add labels	New training data: scanned point cloud and sampled point cloud
11		Train all the related classifier again	Updated classifiers with higher accuracy

Upon finish, the user will be asked to give the new design a shape family label within the same generation of the initial product shape family. It could be the same label if only small changes are applied. Or it could be a new category under the same parent product shape family if it differs largely from any of the existing families within the generation.

In a special case of a hybrid created by combination of modules from different parent families, the label could be the major shape category of the design, which if otherwise very hard to tell, then two labels will be assigned to the new design at the parent family level, and the family link information in the family library will be updated.

6.3.5 Improving and Expanding the Framework

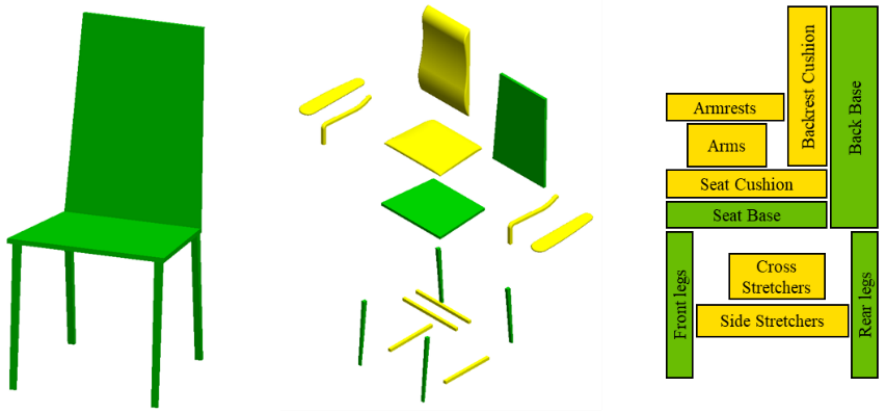
Through each usage of the design tool, labels of newly reconstructed point clouds are confirmed by the user, and newly generated designs are sampled to new point clouds. These labeled data are fed back into the training process of each classifier that is related to the current shape family and thus can further improve the accuracy of classification. Also, users are contributing new shape categories for their designs when new iconic shapes are created. And for the hybrid case, a criterion can be setup for the system to decide to create a new shape family if multiple such hybrids sharing the same parents are designed and stored in the library. As a result, the shape family library will keep expanding, and the accuracy of classification can keep improving.

6.4 Design Examples

We use two examples to demonstrate the design process of our framework. The chair family example focuses more on the overall design process from scanning to modularization. And the second Coke bottle family example focuses on the hierarchical structure of the framework.

6.4.1 Design of a Chair Family

Following the definition of product shape family and its product shape platform, we define the shared iconic 3D shape of a chair family with a combination of a (near) vertical back, a (near) horizontal seat, and a couple of legs. And these three internal modules form the chair shape platform, i.e. for any product to be identified as a chair in the system, these three components are necessary. Any simpler version will be of another shape family or sub-shape-family of the chair family, e.g. stool family (without back), floor chair family (without legs), or lean chair family (without seat). External modules for chair family include backrest cushion, seat cushion, arm, armrest, side stretcher, and cross stretcher. The module configuration is shown in Figure 6-7.



(a) (b) (c)

Figure 6-7 (a) Product shape platform of a Chair family (b) Modularized model (green for internal modules and yellow for external modules) and (c) Module configuration shown in block diagram

As we mentioned in Section 6.3.4, the design process started from taking photos of a design target, a certain chair product. In our example, we took photos around a wooden armchair covered with cushion on backrest and seat. Figure 6-8 shows the images and the reconstructed dense point cloud along with estimated camera motion. Floor points can be taken out by a plane estimation with RANSAC or with the similar classification method described in Section 5.4.2 (Figure 5-15 and Figure 5-16). In total the reconstructed chair has 106,222 points and clearly represents the shape of most of the original chair (cross stretchers were occluded in the photos).

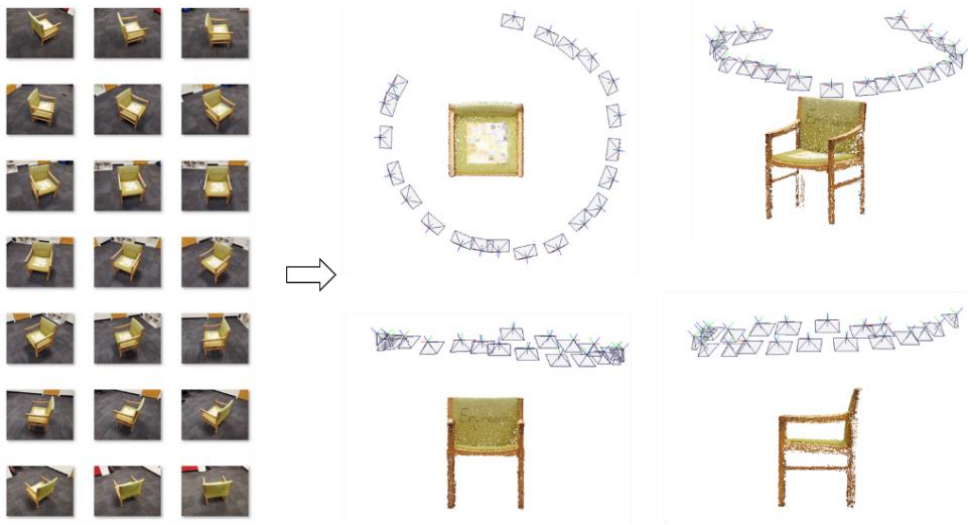
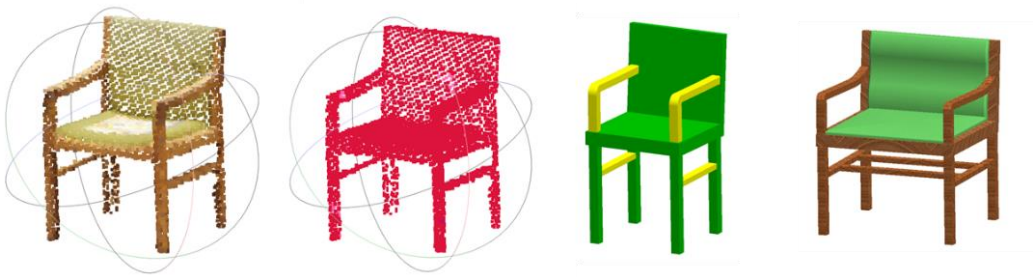


Figure 6-8 Photos around a chair and reconstructed camera motion and dense point cloud viewed from different angles.

In the next step, the dense point cloud is normalized to fit into a unit sphere with geometric center moved to origin, and then down sampled with voxel size of 0.02, resulting in 3,133 uniformly distributed samples. After voxelization process around each point with $10\times10\times10$ voxel grids and $0.2\times0.2\times0.2$ cell size, the data are fed into our trained classification model to identify furniture including chairs, shelves, and tables, as described in Section 5.4.2. The classification result gives 98.95% possibility for the point cloud to be a chair, and 0.03% to be a shelf, and 1.02% to be a table. The normalized and down-sampled point cloud and the visualized classification results are shown in Figure 6-9 (a) and (b).



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

Figure 6-9 (a) Sampled point cloud (3133 points) (b) classification result: Chair 98.95%, Shelf 0.03%, Table 1.02%. (c) Generated model after point distribution analysis. (d) Edited and rendered model to be closer to original chair shape.

Since the point cloud is uniformly sampled, a point distribution analysis on the point cloud can help locating structures and generating a closer generic model. As show in Figure 6-10, number of points are counted along height direction (assumed to be normal to floor plane) and highest peak suggests seat area, which separates a chair into leg side and backrest side. Further analysis on small peaks on leg side gives location of stretchers, and similarly on backrest side gives location of arms.

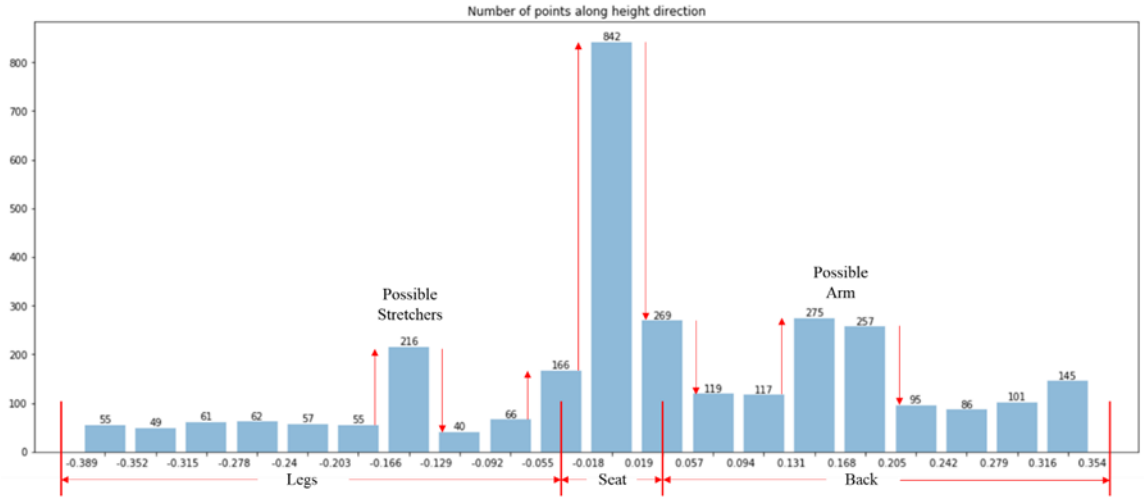


Figure 6-10 Bin plot of number of points along height direction and analysis of module sections.

With the size ratio calculated and some of the external modules detected, the user is then provided with a chair that has generic arms and stretches that looks closer to the scanned chair shape, as shown in Figure 6-9 (c). And Figure 6-9 (d) shows a shape model closer to the original scanned chair after individual module shape editing and material rendering.

The editing steps on the model are illustrated in Table 6-1 from step 4 to step 8. Figure 6-11 shows the editing results for every step with special focus on the pushing and pulling of control points on individual modules to change the length and thickness of each module, the shape of seat and back bases, the curve of the backrest cushion and seat cushion, the location of stretchers, angle of bases and legs, and the curve of arms. All the results shown here are created and rendered by Solidworks [9] and are only demonstration of the concept of our design and modeling framework.

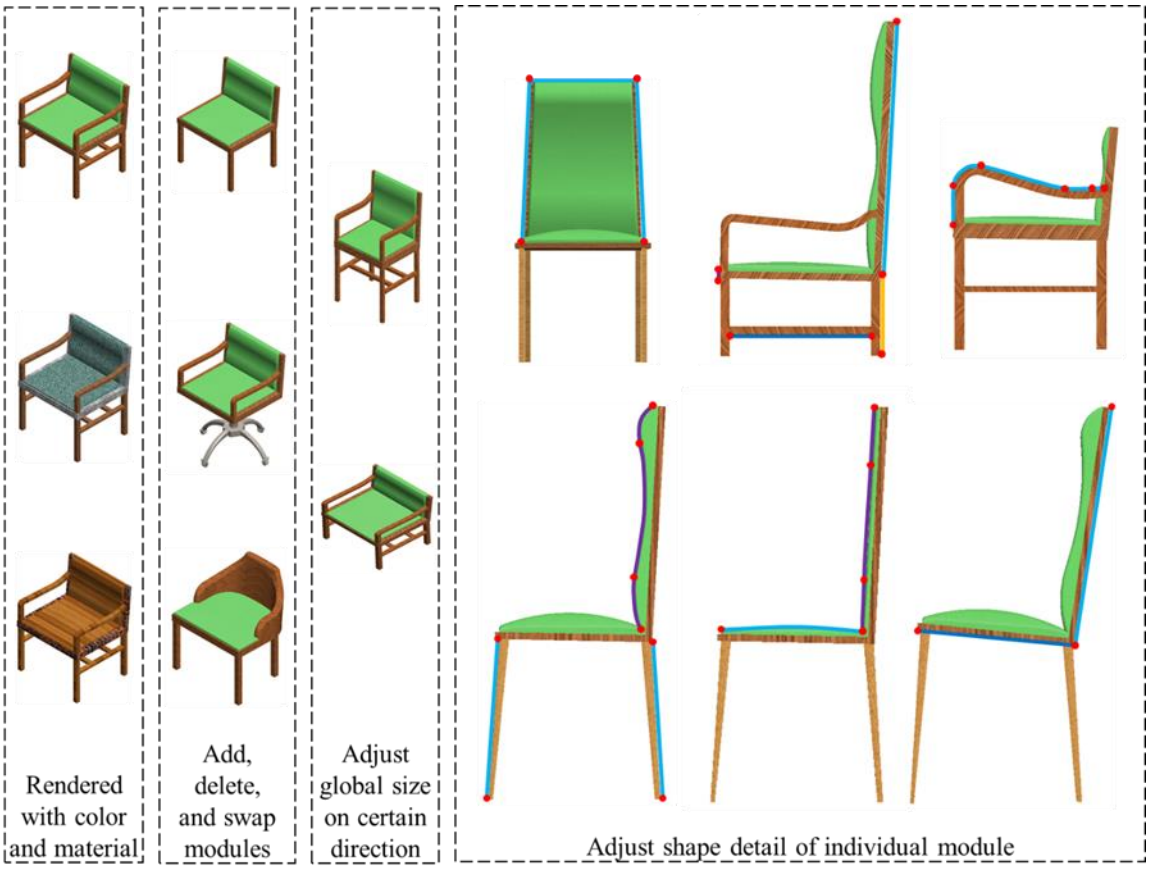


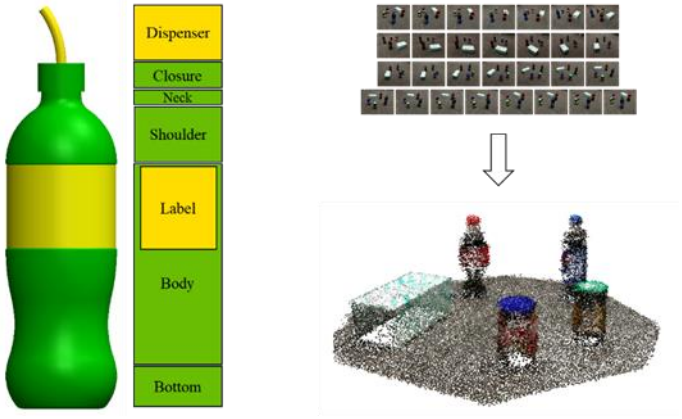
Figure 6-11 Editing of a modularized model of chair shape family

6.4.2 Design of a Coke Bottle Family

Similarly, we define the product shape platform of Coke bottle family with iconic closure, neck, shoulder, body, and bottom. External modules include dispensers and label. In a hierarchical product shape family tree, Coke bottle family is a subfamily of the bottle family, which would have a broader definition of modules, so that cosmetic bottles and liquor bottles could be included. Figure 6-12 (a) shows the coke bottle shape model and module configuration with block diagram.

A scene containing a Coke bottle, a Pepsi bottle, two similar jars, and a box is reconstructed from 29 photos with our SfM and MVS systems. The result was shown in Figure 5-15. In our design framework, after hierarchical classification is done and confirmed, separated labeled clusters will

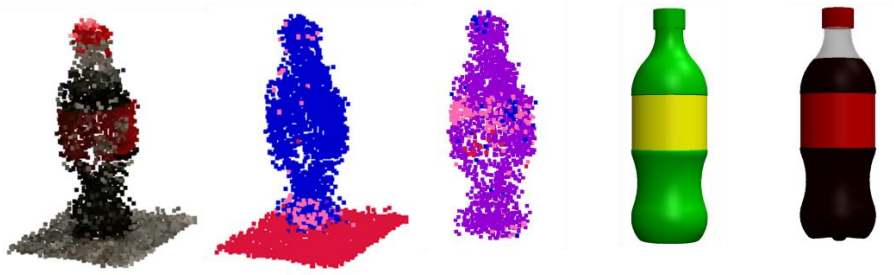
be linked to different shape families and contribute to the training of better classification models. Here we focus specially on the Coke bottle design.



(a) (b)

Figure 6-12 (a) Modularized model (green for internal modules and yellow for external modules) and module configuration shown in block diagram (b) Photos of a scene containing two bottles, two jars, and a box, and the reconstructed dense point cloud.

As discussed in Section 5.4.1 and shown in Figure 5-12, a bottle identified from a scene can be further identify to be a Coke bottle, a Pepsi bottle, or a jar. We apply the next level classification model on the Coke bottle cluster, and it results in the following probabilities: Coke bottle 83.48%, Pepsi bottle 12.09%, Jar 3.48%, and Floor 0.96%. The visualized result is shown in Figure 6-13. From here, a Coke bottle shape family will be given to the user as one of the 5 most possible guesses.



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

Figure 6-13 (a) Coke bottle from the scan (b) Results from Figure 5-15 in Section 5.4.2 (c) Next level classification for Coke, Pepsi, Jar, or floor, predicted probability: Coke bottle 83.48%, Pepsi bottle 12.09%, Jar 3.48%, and Floor 0.96%. (d) Generated model after point distribution analysis. (d) Edited and rendered model to be closer to original shape

A similar analysis can be applied on the point cloud, but for bottle families it requires a different manner to estimate the shape information. For a Coke bottle, we may identify the label area through following steps: (1) Get furthest 25% of points away from vertical center axis, which is a line going through averaged coordinate and orientated long floor normal. (2) Bin plot the height distribution of these points, and a large sudden increase and a large sudden drop mark the label area, with which a closer model can be generated, as shown in Figure 6-13 (d). And Figure 6-13 (e) shows a closer model after module editing and color and material rendering.

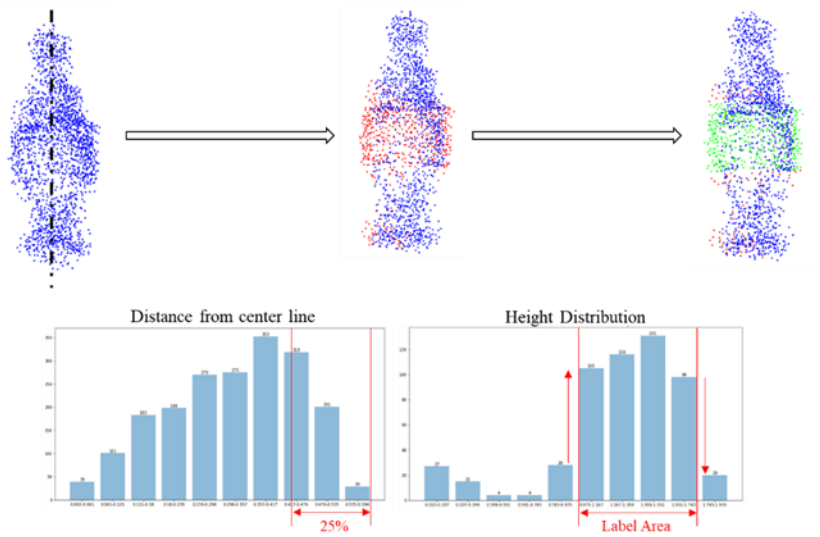


Figure 6-14 Point distribution analysis to locate label area of a scanned Coke bottle

Some demonstration of editing the Coke bottle shape model are shown in Figure 6-15, including adjusting width of bottle, location of label area, height of label, shoulder curve, body curve, as well as swapping bottom to be the more real concave shape. The new design can be rendered with materials, colors, and labels for more realistic looks.

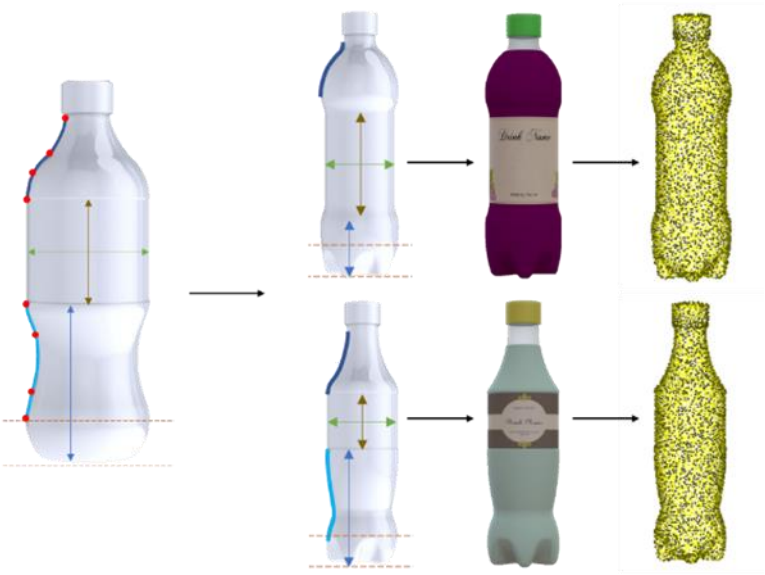


Figure 6-15 Examples of editing individual modules on a Coke bottle shape family model and sampled new designs.

The design results in the example shown above actually does not belong to Coke bottle family anymore. The top design can be assigned to Pepsi bottle family, and bottom design to Sprite bottle family. These new family classes can be confirmed by the user and then sampled into a dense point cloud and be linked to corresponding families to further contribute to the training process.

Another design scenario is when a certain shape is spotted by the user, but the library may not have any documented shape family. The reconstructed point cloud will still generate 5 most possible guesses that has close shape to what has been scanned. The advantage for the framework is to provide some shape family with close geometry so that the user can still obtain a close shape through modularized editing of a selected shape model. We use the following example to demonstrate this design situation.

Consider from an inspiration of a hand-bell, a designer wants to design a bell-shaped bottle. The bottle family model can be obtained and edited in the same manner as above discussion. But there might be no bell model registered in the shape family library. Hierarchical classification of the reconstructed bell could result in a long neck pot, which has very similar shape to a bell. The shoulder, body, and bottom module can be selected and edited to resemble a bell shape, and then

merged with the closure, neck, and shoulder modules of the bottle shape model. The merged hybrid can then be rendered with color and material. The process is shown below in Figure 6-16. And now since the bell-shaped bottle does not belong to any shape family at the Coke bottle generation and the design is unique, no new subfamily will be created. The design will be assigned to bottle shape family and a newly created bell family, where further sampled point cloud will be linked to.

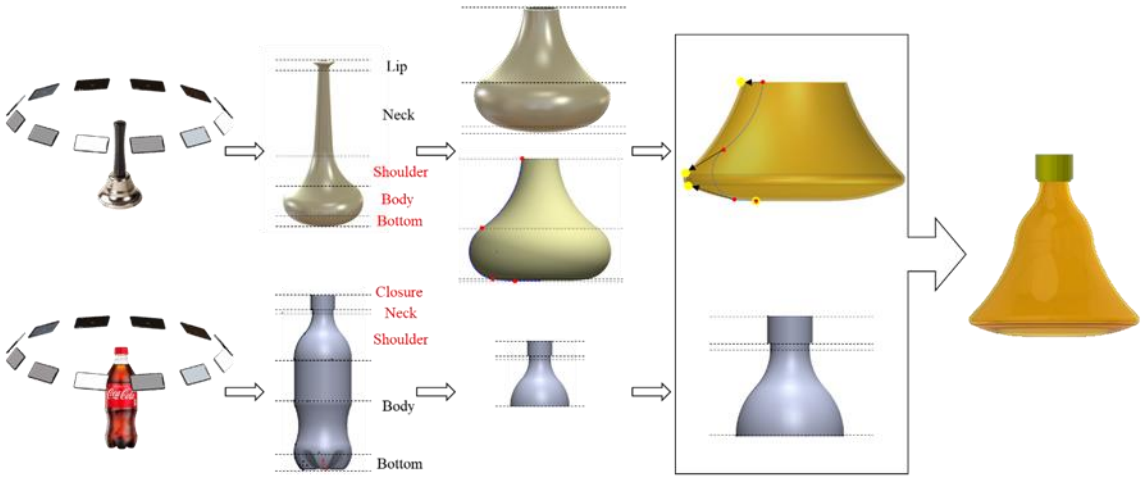


Figure 6-16 Example of creating a hybrid when only similar model exists in the library.

With these two examples, we demonstrated the convenience of our design and modeling framework which increases efficiency for conceptual design and provides abundant possibilities for model variation with easy operations. It is also shown how the design library can be expanded, and classification accuracy can be improved through the use of the design tool.

6.5 Remarks

In this chapter, we illustrated how our product shape family concept is derived from the idea of product family and the motivation to develop a design and modeling framework. The framework integrated the image-based reconstruction system and the 3D CNN shape classification system and utilized a hierarchical shape family tree structure to improve the efficiency of product conceptual design but still provide abundant room for design flexibility. The whole design process

does not require users to have any background knowledge of CAD operations or modeling logic. A 3D model is presented all the way throughout the design process. And the design and modeling process is simplified to photo taking, model selection, module swapping, and pushing-and-pulling operations. The success of the framework depends heavily on the function of image-based reconstruction, accuracy of classification, and diversity of the product shape families. But we have seen that the reconstruction and classification is proven to have good performance, the accuracy and capability of the classifiers improves with the usage of the design tool, and the diversity of the product shape families can be expanded both through usage or by manually updating.

CHAPTER 7

CONLCUSIONS AND RECOMMENDATIONS

In this dissertation, a design and modeling framework has been developed and demonstrated to facilitate product shape conceptual design. In this final chapter, we start with a review of the target issue and our framework solution in Section 7.1, followed by a summary of contribution from the research in section 7.2. Limitations of the research are discussed in Section 7.3, and finally recommendation of possible future work are described in Section 7.4.

7.1 Research Review

As stated in Chapter 1, one of the causes for low conceptual design efficiency is that the format of design at different stages needs to be converted to the next stage, which always involves constant editing of design and back-and-forth cross-professional communication. Designing in 3D would provide a more holistic view of the design and much easier transition between stages. However current powerful feature-based CAD tools are slow, error-prone, and requires a lot of training. To solve the problem, we developed a new design framework that simplifies the modeling process but provides abundant possibilities for 3D shape variation and thus improve conceptual design efficiency.

The framework integrated an image-based reconstruction system as the starting point of design and modeling, which uses the idea of reverse engineering to generate a 3D point cloud model from analyzing photos of a product. The first step is to detect strong features with SIFT descriptor

on every photo and then match them into pairs. From analysis of epipolar geometry between every two images, the relative camera motion can be extracted and thus triangulate 3D points from the matched 2D strong feature pairs. A good pair is identified with lowest sum of reprojection errors, and from the 3D structure obtained, a third camera motion can be estimated by projecting 3D points onto shared 2D feature matches on the additional photo. An incremental scheme can start from here repeatedly, but the accumulation of estimation error will cause drifting problem and result in wrong further camera motion estimation. The issue is improved with introduction of RANSAC-based estimation and bundle adjustment, but the optimization processes rely on a good initial condition. With our proposed extrinsic camera matrix correction method, the result shows that the initial camera locations are closer to true solution and thus improve the robustness of SfM system. After all the photos are process incrementally, the estimated camera motion are then used in the next system, Multi-View-Stereo, where epipolar geometry between photos can be reversely estimated from camera motion and then used to rectify the images so that epipolar lines are parallel to each other. The point correspondence can now be located from a line search, and thus denser matching features can be obtained easily. Then incrementally, dense 3D points are triangulated and transformed into the same coordinate system, and shared points are tracked and reprojected to provide a measure of excluding wrong matches.

The system needs to understand what is in the scan to assist the design and modeling process. A 3D convolutional neural network model is constructed to identify 3D shapes. The model is trained by labeled pre-processed point cloud data obtained from both reconstruction and CAD file sampling. The preprocessing includes normalization, down-sampling, manually labeling, and voxelization. The voxel representation made it possible to use convolutional-styled learning method. And the two-layered configuration of convolution-ReLU-max pooling proved to have good performance in classifying 3D shapes.

Then we define the concept of product shape family and proposed a shape family tree library using hierarchical structure so that classification can be done at different level with a smaller

number of candidate classes. Chain rule is used to calculate the possibility at certain shape family and this way the user can be provided with multiple best guesses and select from them at different family generations. A modularized model is defined within every product shape family including internal modules that form the product shape platform and external modules that are optional. The editing process of the selected shape family model is simply selecting desired module and edit shape with pushing and pulling operations on predefined control points. The new design could form a new family branch in the library, or it can be sampled and preprocessed in the same manner as training data and fed back to training process along with newly scanned point clouds. A summary of the research contributions is presented in the next section.

7.2 Contributions

The contributions of this dissertation are introduced in Section 1.4 and are substantiated throughout the discussion, derivation, experiments, and conceptual demonstrations in the previous chapters. The primary contribution is a product shape family design and modeling methodology integrating image-based reconstruction, hierarchical shape learning with 3D CNNs, and modularized shape family modeling into a design and modeling framework. Additional contributions from the dissertation include the following:

- Fully implemented Structure from Motion system, Multi-View Stereo system, and 3D shape learning system with 3D CNNs: Section 3.4, 4.2, and 5.3.
- Initial pair selection strategy and extrinsic matrix correction method for improved accuracy and robustness of SfM: Section 3.5.2, and 3.5.3.
- Solution for reconstructed data handling for training and testing, Section 5.3.1, 5.3.2, and 5.3.3.
- A dataset of manually-labeled pre-processed 3D point clouds that is ready-to-use for 3D geometry learning tasks: Section 5.3.3 and 6.3.2.

- A structure of 3D shape classification model using 3D CNNs and a few trained classifiers: Section 5.3.4 and 5.4.
- Shape dataset expanding method using corrupted samples from CAD models and datasets of manually labeled 3D point clouds of different containers and furniture: Section 5.3.3.
- Concept of product shape family and product shape platform: Section 6.1.2.
- A hierarchical-structured modularized shape family library: Section 6.1.3.
- A new design modeling process using the framework: Section 6.3.4.

Every individual item listed above contributed to its corresponding research field. For structure from motion, a very important if not the most crucial issue is the reconstruction robustness. Various solutions exist to deal with the error accumulation, like by using RANSAC-based optimization and bundle adjustment, or working around incremental method with global or hybrid estimation instead. But our practice of first pair selection and camera motion correction provides a better initial camera motion for any further global adjustment optimization. For learning on the reconstructed data, or in fact it could be on any 3D point cloud data sampled from real objects or virtual models, we provide detailed processing steps to prepare the data for training, and to label new input point cloud for 3D geometric classification and segmentation purpose. The method is very intuitive because it imitates human learning of iconic product shapes, and thus our architecture can actually be used on any 3D shape classification of any size with different parameters of voxelization. However, it is very difficult to have a universal classifier to identify every shape category. In our research the hierarchical product shape family library structure provided not just different levels of design flexibility but also an access for hierarchical classification. This requires large number of training data at each level of the family tree, especially at leaf nodes. So, our practice of using Gaussian noise-corrupted point clouds sampled from CAD-generated models can enlarge the training set exponentially. The modularized modeling method can actually be

independent from other parts of the framework and is a new approach of 3D modeling by itself. But altogether these components form a new design methodology that is new, reliable, efficient, and non-CAD-user friendly.

7.3 Limitations

In this section we discuss the limitations of our research through a critical evaluation on the design and modeling framework and the components of it.

For the product shape family design and modeling framework, the design flexibility is sacrificed to some extent to improve modeling efficiency. The 3D model of any module in any shape family is still created from feature-based modeling method. Some of the entities like end points or center points of lines, arcs, or spline curves are left under-defined to allow user to edit 3D shape from dragging and pulling these points. However, the definition of these entities and their related features confine the 3D geometry of the module and thus variation of it is limited. Although swapping of module helps provide more flexibility, any module for swapping is still a model with predefined geometry. Consequently, some complex surface shapes may not be able to be created. Indeed, many more generic modules with different shapes could be registered into the library to cover a broader range of complex geometries, but it will in return lower design efficiency due to too many choices presented to users. The same issue applies to the hierarchical structure of the shape family library. In the dissertation, we only use the Coke bottle family example to demonstrate the feasibility of the hierarchical shape family design concept. The mentioned container hierarchy also only gives an example of how a product hierarchy can possibly be defined. No evaluation was done on how different hierarchy setup could affect design flexibility and efficiency.

Definition of the interface between modules is another important detail that is not covered in this dissertation. The objective is certainly to maintain defined boundary condition for continuity and smoothness between modules. But the task could be very complicated in cases like two

modules having different geometries at the connecting interface, as shown in Figure 7-1, or different modules having different connecting mechanisms with a same module. Also, for each module, the shape geometry is predefined and only possible changes allowed on the module are positions of control points related to feature profiles or guide curves. This indeed limits the room for designers to improvise on the topology of the product.

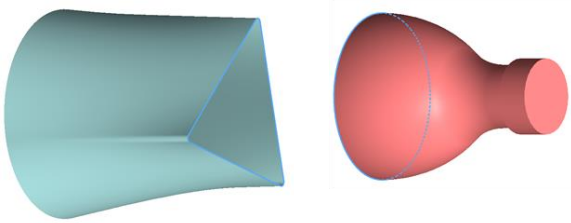


Figure 7-1 Example of modules having different shape at connecting interface.

Also, the performance of the framework relies heavily on the performance of each component, so the limitations coming from 3D reconstruction and classification are also considered shortages of the framework.

The image-based 3D reconstruction obtains all the estimations based on analysis of feature matches between images, which requires abundant number of features in the first place. Thus, reconstruction could become very challenging and most of the time generating wrong results when facing texture-lacking surfaces, transparent surfaces, too dark or too shiny scanning environment, thin/fuzzy structure, non-Lambertian surfaces, patterned texture, or occlusion. That means for products like glass bottles or alloy steel tools, the very first step in the design framework could fail. Also, although our reconstruction framework is automated as much as possible, there are still parameters that needs to be tuned, like the reprojection error rejecting line during bundle adjustment process, because too strict threshold would result in too few points for estimation, whereas too loose threshold keeps too many wrong points. Our current setup successfully reconstruct most tasks, but in some failing cases it still needs to be adjusted manually.

For 3D shape learning with CNNs, the first issue came from the training source. The point clouds sampled from CAD-generated models are closer to scanned data especially after applying Gaussian noise on them. But real scans could have areas of holes due to locally reconstruction failure. And most cases scans only captures visible surfaces of a product. These situations were not taken into consideration. Although we saw that the chair example where all training data is obtained from CAD model and it still give good prediction for our scanned chair, but as the families getting more diverse, the consistency between details of training data and new inputs would be more important. Also, the hierarchy setup of the shape family library is related to the performance of classifiers, which has not been investigated in this dissertation.

7.4 Recommendations

The limitations of our research set up the basis for possible future works, which includes validating, improving, and implementing current design and modeling framework, and exploring new design and modeling concept on top of the framework.

For the components of the framework, solutions for passive reconstruction on reflective or feature-lacking surfaces need to be explored, which is challenging but featuring a large group of modern products. A possible path is to use 2D learning on images to detect these non-Lambertian or feature-lacking surfaces and exclude features from around the areas. Better 3D CNN architecture can be developed using deeper layers or different activation function or pooling strategy. 3D CNN-based segmentation could also be applied on a single point cloud cluster to analyze the geometric combination of the scanned product.

A product shape family hierarchy exploring method needs to be investigated to come up with a more systematic approach to define and update product hierarchies. And the hierarchical classification concept needs to be further implemented on a designed product shape family tree and evaluation method needs to be explored to measure the performance of branch suggestion. The

measure can also provide information like which nodes of the tree are probably in need of better classification models by increasing training data size or by redefining a family node into multiple families or the other way around by merging families into one.

For every product shape family, a shape model modularization strategy is imperative so that design flexibility can be maintained with less need of user operations. Special focus should be put on defining interface connecting constraints between modules. Also, a product shape platform exploring method needs to be investigated to analyze geometric commonality, possibly from 3D shape classification aspect. For every module, other simple but intuitive operations needs to be identified and developed, like volume cutting with selected contour, bending, twisting, and patterning. These operations can also be presented to users as modularizes module-editing options and allow applying these edits with pushing or pulling operations.

In cases when no models or even close models are registered in the library, the framework should be expanded to handle point cloud data modeling directly. Systematic methods need to be developed to divide a point cloud data into modules and to reshape it with pushing and pulling of control points, possibly using the idea of warping. For hybrid generation cases, a point cloud may be reconstructed into a mesh surface model and then used as an editable module to connect with other shape model modules.

Finally, a user interface needs to be developed implementing all the functions of modularized product shape family design and modeling. Especially with today's technological development in virtual reality (or mixed reality), hand gesture recognition, and natural language understanding, the operations on 3D design and modeling can go far beyond mouse clicks on a 2D monitor, where our design and modeling framework could show better potential.

REFERENCES

1. ENVARY. https://www.envary.com/portfolio/diva-prosecco-spirits-bottle-design-branding-product/.
2. Adobe, Adobe Illustrator (Version 23.0). 2018, Retrieved from https://www.adobe.com/products/illustrator.html?promoid=PGRQQLFS&mv=other.
3. Adobe, Adobe Photoshop (Version 20.0). 2018, Retrieved from https://www.adobe.com/products/photoshop.html.
4. Weisberg, D.E., The engineering design revolution: The people, companies and computer systems that changed forever the practice of engineering. Cyon Research Corporation. 2008. 1-26.
5. Hoffmann, C.M. and R. Joan-Arinyo, Parametric modeling. Handbook of CAGD. 2002. 519-541.
6. X3D. Extensible 3D (X3D) Part 1: Architecture and base components, 27 NURBS component.
7. Technavio. Top 10 Vendors in the Global CAD Market in the Industrial Machinery Industry. 2016; Available from: https://www.technavio.com/blog/top-10-vendors-global-cad-market-industrial-machinery-industry.
8. Mäntylä, M., D. Nau, and J. Shah, Challenges in feature-based manufacturing research. Communications of the ACM, 1996. 39(2): p. 77-85.
9. Dassault-Systemes, Solidworks 2016. http://www.solidworks.com/.
10. Várady, T., R.R. Martin, and J. Cox. Reverse engineering of geometric models. in An Introduction, ComputerAided Design. 1997. Citeseer.
11. Jarvis, R.A., A perspective on range finding techniques for computer vision. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1983(2): p. 122-139.
12. Beheim, G. and K. Fritsch, Range finding using frequency-modulated laser diode. Applied Optics, 1986. 25(9): p. 1439-1442.
13. Moring, I., et al., Acquisition of three-dimensional image data by a scanning laser range finder. Optical engineering, 1989. 28(8): p. 288897.
14. Brooks, M.J. and B.K. Horn, Shape and source from shading. 1985.
15. Kender, J.R. Shape from texture: An aggregation transform that maps a class of textures into surface orientation. in Proceedings of the 6th international joint conference on Artificial intelligence-Volume 1. 1979. Morgan Kaufmann Publishers Inc.
16. Healey, G. and T.O. Binford, Local shape from specularity. Computer Vision, Graphics, and Image Processing, 1988. 42(1): p. 62-86.
17. Brady, M. and A. Yuille, An extremum principle for shape from contour. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1984(3): p. 288-301.
18. Winkelbach, S. and F.M. Wahl. Shape from 2D edge gradients. in Joint Pattern Recognition Symposium. 2001. Springer.
19. Van den Heuvel, F.A., 3D reconstruction from a single image using geometric constraints. ISPRS Journal of Photogrammetry and Remote Sensing, 1998. 53(6): p. 354-368.

20. Zhang, L., et al., Single-view modelling of free-form scenes. Computer Animation and Virtual Worlds, 2002. 13(4): p. 225-235.
21. Remondino, F. and A. Roditakis. Human figure reconstruction and modeling from single image or monocular video sequence. in 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on. 2003. IEEE.
22. Kemelmacher-Shlizerman, I. and R. Basri, 3D face reconstruction from a single image using a single reference face shape. IEEE transactions on pattern analysis and machine intelligence, 2011. 33(2): p. 394-405.
23. Han, F. and S.-C. Zhu. Bayesian reconstruction of 3d shapes and scenes from a single image. in Higher-Level Knowledge in 3D Modeling and Motion Analysis, 2003. HLK 2003. First IEEE International Workshop on. 2003. IEEE.
24. Kutulakos, K.N. and S.M. Seitz, A theory of shape by space carving. International journal of computer vision, 2000. 38(3): p. 199-218.
25. Girardeau-Montaut, D., Cloud compare—3d point cloud and mesh processing software. Open Source Project, 2015.
26. Cignoni, P., et al. Meshlab: an open-source mesh processing tool. in Eurographics Italian chapter conference. 2008.
27. VirtualGrid, VRMesh. 2018.
28. Systems, B., Bentley Pointools. 2018.
29. Pomerleau, F., F. Colas, and R. Siegwart, A review of point cloud registration algorithms for mobile robotics. Foundations and Trends® in Robotics, 2015. 4(1): p. 1-104.
30. Muraki, Y., et al., An automatic hole filling method of point cloud for 3D scanning. 2017.
31. Qi, C.R., et al., Pointnet: Deep learning on point sets for 3d classification and segmentation. Proc. Computer Vision and Pattern Recognition (CVPR), IEEE, 2017. 1(2): p. 4.
32. Qi, C.R., et al. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. in Advances in Neural Information Processing Systems. 2017.
33. Maturana, D. and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. in Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on. 2015. IEEE.
34. Wu, Z., et al. 3d shapenets: A deep representation for volumetric shapes. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.
35. Wang, Y., et al., Dynamic graph CNN for learning on point clouds. arXiv preprint arXiv:1801.07829, 2018.
36. Demir, I., D.G. Aliaga, and B. Benes. Procedural editing of 3d building point clouds. in Proceedings of the IEEE International Conference on Computer Vision. 2015.
37. Piegl, L. and W. Tiller, The NURBS book. 2012: Springer Science & Business Media.
38. Coons, S.A., Surfaces for computer-aided design of space forms. 1967, MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC.
39. Kazhdan, M., M. Bolitho, and H. Hoppe, Poisson Surface Reconstruction, in Eurographics Symposium on Geometry Processing. 2006. p. 61-70.
40. Carr, J.C., et al. Reconstruction and representation of 3D objects with radial basis functions. in Proceedings of the 28th annual conference on Computer graphics and interactive techniques. 2001. ACM.
41. Ohtake, Y., A. Belyaev, and M. Alexa. Sparse low-degree implicit surfaces with applications to high quality rendering, feature extraction, and smoothing. in Proc. Symp. Geometry Processing. 2005. Citeseer.
42. Boissonnat, J.-D., Geometric structures for three-dimensional shape representation. ACM Transactions on Graphics (TOG), 1984. 3(4): p. 266-286.
43. Kolluri, R., J.R. Shewchuk, and J.F. O'Brien. Spectral surface reconstruction from noisy point clouds. in Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing. 2004. ACM.

44. Edelsbrunner, H. and E.P. Mücke, Three-dimensional alpha shapes. ACM Transactions on Graphics (TOG), 1994. 13(1): p. 43-72.
45. Amenta, N., S. Choi, and R.K. Kolluri, The power crust, unions of balls, and the medial axis transform. Computational Geometry, 2001. 19(2-3): p. 127-153.
46. Dey, T.K. and J. Giesen, Detecting undersampling in surface reconstruction, in Discrete and Computational Geometry. 2003, Springer. p. 329-345.
47. Muraki, S., Volumetric shape description of range data using "blobby model". ACM SIGGRAPH computer graphics, 1991. 25(4): p. 227-235.
48. Alexa, M., et al., Computing and rendering point set surfaces. IEEE Transactions on visualization and computer graphics, 2003. 9(1): p. 3-15.
49. Jenke, P., et al. Bayesian point cloud reconstruction. in Computer Graphics Forum. 2006. Wiley Online Library.
50. Ye, X., et al., Reverse innovative design—an integrated product design methodology. Computer-aided design, 2008. 40(7): p. 812-827.
51. Ohtake, Y., et al. Multi-level partition of unity implicits. in ACM Transactions on Graphics (TOG). 2003. ACM.
52. Lipman, Y., et al., Parameterization-free projection for geometry reconstruction. ACM Transactions on Graphics (TOG), 2007. 26(3): p. 22.
53. Kazhdan, M. and H. Hoppe, Screened poisson surface reconstruction. ACM Transactions on Graphics (ToG), 2013. 32(3): p. 29.
54. Hornung, A. and L. Kobbelt. Robust reconstruction of watertight 3 d models from non-uniformly sampled point clouds without normal information. in Symposium on geometry processing. 2006. Citeseer.
55. Fleishman, S., D. Cohen-Or, and C.T. Silva, Robust moving least-squares fitting with sharp features. ACM transactions on graphics (TOG), 2005. 24(3): p. 544-552.
56. Avron, H., et al., ℓ 1-Sparse reconstruction of sharp point set surfaces. ACM Transactions on Graphics (TOG), 2010. 29(5): p. 135.
57. Huang, H., et al., Edge-aware point set resampling. ACM Transactions on Graphics (TOG), 2013. 32(1): p. 9.
58. Li, G., et al., Analysis, reconstruction and manipulation using arterial snakes. ACM Trans. Graph., 2010. 29(6): p. 152:1-152:10.
59. Cao, J., et al. Point cloud skeletons via laplacian based contraction. in Shape Modeling International (SMI 2010). 2010. IEEE.
60. Tagliasacchi, A., et al. Vase: Volume-aware surface evolution for surface reconstruction from incomplete point clouds. in Computer Graphics Forum. 2011. Wiley Online Library.
61. Oesau, S., F. Lafarge, and P. Alliez. Indoor scene reconstruction using primitive-driven space partitioning and graph-cut. in Eurographics workshop on urban data modelling and visualisation. 2013.
62. Schnabel, R., R. Wahl, and R. Klein. Efficient RANSAC for point-cloud shape detection. in Computer graphics forum. 2007. Wiley Online Library.
63. Schnabel, R., P. Degener, and R. Klein. Completion and reconstruction with primitive shapes. in Computer Graphics Forum. 2009. Wiley Online Library.
64. Lafarge, F. and P. Alliez. Surface reconstruction through point set structuring. in Computer Graphics Forum. 2013. Wiley Online Library.
65. Van Kreveld, M., T. Van Lankveld, and R.C. Veltkamp. Watertight scenes from urban lidar and planar surfaces. in Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing. 2013. Eurographics Association.
66. Berner, A., et al. Shape analysis with subspace symmetries. in Computer Graphics Forum. 2011. Wiley Online Library.
67. !!! INVALID CITATION !!! {}.

68. Li, Y., et al. Globfit: Consistently fitting primitives by discovering global relations. in ACM Transactions on Graphics (TOG). 2011. ACM.
69. Yingze Bao, S., et al. Dense object reconstruction with semantic priors. in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2013.
70. Shao, T., et al., An interactive approach to semantic modeling of indoor scenes with an rgbd camera. ACM Transactions on Graphics (TOG), 2012. 31(6): p. 136.
71. Li, Y., et al. 2D-3D fusion for layer decomposition of urban facades. in Computer Vision (ICCV), 2011 IEEE International Conference on. 2011. IEEE.
72. Sharf, A., et al. Interactive topology-aware surface reconstruction. in ACM Transactions on Graphics (TOG). 2007. ACM.
73. Nan, L., et al., Smartboxes for interactive urban reconstruction. ACM Transactions on Graphics (TOG), 2010. 29(4): p. 93.
74. Arikan, M., et al., O-snap: Optimization-based snapping for modeling architecture. ACM Transactions on Graphics (TOG), 2013. 32(1): p. 6.
75. Cohen-Steiner, D. and F. Da, A greedy Delaunay-based surface reconstruction algorithm. The visual computer, 2004. 20(1): p. 4-16.
76. Amenta, N. and M. Bern, Surface reconstruction by Voronoi filtering. Discrete & Computational Geometry, 1999. 22(4): p. 481-504.
77. Amenta, N., et al. A simple algorithm for homeomorphic surface reconstruction. in Proceedings of the sixteenth annual symposium on Computational geometry. 2000. ACM.
78. Boissonnat, J.-D. and F. Cazals, Smooth surface reconstruction via natural neighbour interpolation of distance functions. Computational Geometry, 2002. 22(1-3): p. 185-203.
79. Edelsbrunner, H., Surface reconstruction by wrapping finite sets in space, in Discrete and computational geometry. 2003, Springer. p. 379-404.
80. Chaine, R., A geometric-based convection approach of 3-d reconstruction. 2002, INRIA.
81. Bernardini, F., et al., The ball-pivoting algorithm for surface reconstruction. IEEE transactions on visualization and computer graphics, 1999. 5(4): p. 349-359.
82. Petitjean, S. and E. Boyer, Regular and non-regular point sets: Properties and reconstruction. Computational Geometry, 2001. 19(2-3): p. 101--126.
83. Berger, M., et al. State of the art in surface reconstruction from point clouds. in EUROGRAPHICS star reports. 2014.
84. Autodesk, Alias 2019. https://www.autodesk.com/products/alias-products/overview.
85. McNeel, R., Rhinoceros. 2018: NURBS modleing for Windows: http://www.rhino3d.com/.
86. Catmull, E. and J. Clark, Recursively generated B-spline surfaces on arbitrary topological meshes. Computer-aided design, 1978. 10(6): p. 350-355.
87. Autodesk, MAYA. 2018: https://www.autodesk.com/products/maya/overview.
88. Blender, Blender. 2018: https://www.blender.org/.
89. Autodesk, 3DS MAX. 2018: https://www.autodesk.com/products/3ds-max/overview.
90. Autodesk, AUTOCAD. 2018: https://www.autodesk.com/products/autocad/overview.
91. Trimble, SketchUp Pro 2019.
92. E-Chalk. Jack the Hare. Swivel Heads; Available from: https://www.echalk.co.uk/amusements/OpticalIllusions/SwivelHeads/SwivelHeads.html.
93. Longuet-Higgins, H.C., A computer algorithm for reconstructing a scene from two projections. Nature, 1981. 293(5828): p. 133.
94. Canny, J., A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence, 1986(6): p. 679-698.
95. Harris, C. and M. Stephens. A combined corner and edge detector. in Alvey vision conference. 1988. Citeseer.
96. Mikolajczyk, K. and C. Schmid. Indexing based on scale invariant interest points. in Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on. 2001. IEEE.

97. Lowe, D.G. Object recognition from local scale-invariant features. in Computer vision, 1999. The proceedings of the seventh IEEE international conference on. 1999. Ieee.
98. Bay, H., T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. in European conference on computer vision. 2006. Springer.
99. Freeman, W.T. and E.H. Adelson, The design and use of steerable filters. IEEE Transactions on Pattern Analysis & Machine Intelligence, 1991(9): p. 891-906.
100. Schaffalitzky, F. and A. Zisserman. Multi-view matching for unordered image sets, or "How do I organize my holiday snaps?". in European conference on computer vision. 2002. Springer.
101. Carneiro, G. and A.D. Jepson. Multi-scale phase-based local features. in Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on. 2003. IEEE.
102. Lowe, D.G., Distinctive image features from scale-invariant keypoints. International journal of computer vision, 2004. 60(2): p. 91-110.
103. Ke, Y. and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. in Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on. 2004. IEEE.
104. Mikolajczyk, K. and C. Schmid, A performance evaluation of local descriptors. IEEE transactions on pattern analysis and machine intelligence, 2005. 27(10): p. 1615-1630.
105. Winder, S., G. Hua, and M.A. Brown, Picking the best daisy. 2009.
106. Winder, S.A. and M. Brown. Learning local image descriptors. in Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on. 2007. IEEE.
107. Lepetit, V. and P. Fua, Keypoint recognition using randomized trees. IEEE transactions on pattern analysis and machine intelligence, 2006. 28(9): p. 1465-1479.
108. Ozuysal, M., P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. in Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on. 2007. Ieee.
109. Hartley, R. and A. Zisserman, Multiple view geometry in computer vision. 2003: Cambridge university press.
110. Weng, J., P. Cohen, and M. Herniou, Camera calibration with distortion models and accuracy evaluation. IEEE Transactions on Pattern Analysis & Machine Intelligence, 1992(10): p. 965-980.
111. Faugeras, O. and O.A. FAUGERAS, Three-dimensional computer vision: a geometric viewpoint. 1993: MIT press.
112. Tsai, R., A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. IEEE Journal on Robotics and Automation, 1987. 3(4): p. 323-344.
113. Zhang, Z., Camera calibration with one-dimensional objects. IEEE transactions on pattern analysis and machine intelligence, 2004. 26(7): p. 892-899.
114. Caprile, B. and V. Torre, Using vanishing points for camera calibration. International journal of computer vision, 1990. 4(2): p. 127-139.
115. Zhang, Z., A flexible new technique for camera calibration. IEEE Transactions on pattern analysis and machine intelligence, 2000. 22.
116. Maybank, S.J. and O.D. Faugeras, A theory of self-calibration of a moving camera. International Journal of Computer Vision, 1992. 8(2): p. 123-151.
117. Hartley, R.I. Euclidean reconstruction from uncalibrated views. in Joint European-US workshop on applications of invariance in computer vision. 1993. Springer.
118. Mohr, R., L. Quan, and F. Veillon, Relative 3D reconstruction using multiple uncalibrated images. The International Journal of Robotics Research, 1995. 14(6): p. 619-632.

119. Triggs, B. Autocalibration and the absolute quadric. in Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on. 1997. IEEE.
120. Zisserman, A., P.A. Beardsley, and I.D. Reid. Metric calibration of a stereo rig. in Representation of Visual Scenes, 1995.(In Conjuction with ICCV'95), Proceedings IEEE Workshop on. 1995. IEEE.
121. Pollefeys, M., The modulus constraint: a new constraint for self-calibration Marc Pollefeys*, Luc Van Gool and Andrée Oosterlinck Katholieke Universiteit Leuven, ESAT/MI2 Marc. Pollefeys@ esat. kuleuven. ac. be.
122. Hartley, R.I. In defence of the 8-point algorithm. in Computer Vision, 1995. Proceedings., Fifth International Conference on. 1995. IEEE.
123. Snavely, N., S.M. Seitz, and R. Szeliski, Modeling the world from internet photo collections. International journal of computer vision, 2008. 80(2): p. 189-210.
124. Bolles, R.C. and M.A. Fischler. A RANSAC-Based Approach to Model Fitting and Its Application to Finding Cylinders in Range Data. in IJCAI. 1981.
125. Agarwal, S., et al. Bundle adjustment in the large. in European conference on computer vision. 2010. Springer.
126. Snavely, N., S.M. Seitz, and R. Szeliski. Skeletal graphs for efficient structure from motion. in CVPR. 2008.
127. Havlena, M., A. Torii, and T. Pajdla. Efficient structure from motion by graph optimization. in European Conference on Computer Vision. 2010. Springer.
128. Carlone, L., et al. Initialization techniques for 3D SLAM: a survey on rotation estimation and its use in pose graph optimization. in Robotics and Automation (ICRA), 2015 IEEE International Conference on. 2015. IEEE.
129. Ozyesil, O. and A. Singer. Robust camera location estimation by convex programming. in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.
130. Hand, P., C. Lee, and V. Voroninski, ShapeFit: Exact location recovery from corrupted pairwise directions. Communications on Pure and Applied Mathematics, 2018. 71(1): p. 3-50.
131. Triggs, B., et al. Bundle adjustment—a modern synthesis. in International workshop on vision algorithms. 1999. Springer.
132. Crandall, D., et al. Discrete-continuous optimization for large-scale structure from motion. in Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. 2011. IEEE.
133. Bar-Itzhack, I.Y., New method for extracting the quaternion from a rotation matrix. Journal of guidance, control, and dynamics, 2000. 23(6): p. 1085-1087.
134. Loop, C. and Z. Zhang. Computing rectifying homographies for stereo vision. in Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on. 1999. IEEE.
135. Dall'Asta, E. and R. Roncella, A COMPARISON OF SEMIGLOBAL AND LOCAL DENSE MATCHING ALGORITHMS FOR SURFACE RECONSTRUCTION. International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences, 2014. 45.
136. Hartley, R.I., Theory and practice of projective rectification. International Journal of Computer Vision, 1999. 35(2): p. 115-127.
137. Seitz, S.M. and C.R. Dyer, Photorealistic scene reconstruction by voxel coloring. International Journal of Computer Vision, 1999. 35(2): p. 151-173.
138. Eisert, P., E. Steinbach, and B. Girod. Multi-hypothesis, volumetric reconstruction of 3-D objects from multiple calibrated camera views. in Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on. 1999. IEEE.
139. Isidoro, J. and S. Sclaroff. Stochastic Refinement of the Visual Hull to Satisfy Photometric and Silhouette Consistency Constraints. in ICCV. 2003.

140. Gargallo, P. and P. Sturm. Bayesian 3D modeling from images using multiple depth maps. in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. 2005. IEEE.
141. Habbecke, M. and L. Kobbelt. Iterative multi-view plane fitting. in Int. Fall Workshop of Vision, Modeling, and Visualization. 2006.
142. Pollefeys, M., et al., Visual modeling with a hand-held camera. International Journal of Computer Vision, 2004. 59(3): p. 207-232.
143. Bradski, G., The OpenCV Library. Dr. Dobb's Journal of Software Tools, 2000.
144. Hirschmuller, H., Stereo processing by semiglobal matching and mutual information. IEEE Transactions on pattern analysis and machine intelligence, 2008. 30(2): p. 328-341.
145. McCulloch, W.S. and W. Pitts, A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 1943. 5(4): p. 115-133.
146. Fukushima, K., Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological cybernetics, 1980. 36(4): p. 193-202.
147. Krizhevsky, A., I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks. in Advances in neural information processing systems. 2012.
148. Simonyan, K. and A. Zisserman, Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.
149. Szegedy, C., et al. Going deeper with convolutions. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.
150. Girshick, R., et al. Rich feature hierarchies for accurate object detection and semantic segmentation. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.
151. Frome, A., et al. Recognizing objects in range data using regional point descriptors. in European conference on computer vision. 2004. Springer.
152. Golovinskiy, A., V.G. Kim, and T. Funkhouser. Shape-based recognition of 3D point clouds in urban environments. in 2009 IEEE 12th International Conference on Computer Vision. 2009. IEEE.
153. Koppula, H.S., et al. Semantic labeling of 3d point clouds for indoor scenes. in Advances in neural information processing systems. 2011.
154. Gupta, S., et al. Learning rich features from RGB-D images for object detection and segmentation. in European Conference on Computer Vision. 2014. Springer.
155. Ren, X., L. Bo, and D. Fox. Rgb-(d) scene labeling: Features and algorithms. in 2012 IEEE Conference on Computer Vision and Pattern Recognition. 2012. IEEE.
156. Prokhorov, D., A convolutional learning system for object classification in 3-D LIDAR data. IEEE Transactions on neural networks, 2010. 21(5): p. 858-863.
157. Simpson, T.W., J.R. Maier, and F. Mistree, Product platform design: method and application. Research in engineering Design, 2001. 13(1): p. 2-22.
158. Dai, Z. and M.J. Scott, Effective product family design using preference aggregation. Journal of Mechanical Design, 2006. 128(4): p. 659-667.
159. Akundi, S.V., T.W. Simpson, and P.M. Reed. Multi-objective design optimization for product platform and product family design using genetic algorithms. in ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. 2005. American Society of Mechanical Engineers.
160. Kumar, D., W. Chen, and T.W. Simpson, A market-driven approach to product family design. International Journal of Production Research, 2009. 47(1): p. 71-104.
161. Kim, S. and S.K. Moon, Sustainable platform identification for product family design. Journal of cleaner production, 2017. 143: p. 567-581.

162. Krause, D. and S. Eilmus. A Methodical Approach for Developing Modular Product Families. in DS 68-4: Proceedings of the 18th International Conference on Engineering Design (ICED 11), Impacting Society through Engineering Design, Vol. 4: Product and Systems Design, Lyngby/Copenhagen, Denmark, 15.-19.08. 2011. 2011.
163. Bonvoisin, J., et al., A systematic literature review on modular product design. Journal of Engineering Design, 2016. 27(7): p. 488-514.
164. Yang, Z., et al. Hierarchical attention networks for document classification. in Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2016.
165. Partalas, I., et al., Lshtc: A benchmark for large-scale text classification. arXiv preprint arXiv:1503.08581, 2015.
166. Du, Y., W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.
167. Deng, J., et al. Large-scale object classification using label relation graphs. in European conference on computer vision. 2014. Springer.