Vision Aided Inspection in Industries

BTech Project

Prof. Sankha Deb and Prof. Debashis Sen

Shwetha Krishnamurthy

Roll No.: 14ME10072

Acknowledgement

I express my deep sense of gratitude to my guide Prof. Sankha Deb and Prof. Debashis Sen for their valuable guidance and inspiration throughout the course of this work. I am thankful to them for their kind help, assistance and support and their motivation toward independent thinking. It has been a great experience working under them in the cordial environment.

BTECH PROJECT, MECHANICAL ENGINEERING DEPARTMENT, INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR

This research was done under the supervision of Prof.Sankha Deb and Prof. Debashis Sen, from July 2017 to April 2018.
All the code for this project can be found here:
`https://github.com/shwetha-krishnamurthy/reconstruction`

Certificate

This is to certify that the project entitled *Vision Aided Inspection in Industries* submitted by *Shwetha Krishnamurthy, 14ME10072*, to Indian Institute of Technology, Kharagpur, for the award of the Bachelor of Technology, is a record of research work carried out by her under our supervision and guidance. The thesis, in our opinion, is worthy of consideration for the award of the degree of Bachelor of Technology in Engineering in accordance with the rules and regulations of the Institute.

PROFESSOR SANKHA DEB
Department of Mechanical Engineering
Indian Institute of Technology
Kharagpur, 721302

PROFESSOR DEBASHIS SEN
Department of Electronics
and Electrical Communications Engineering
Indian Institute of Technology
Kharagpur, 721302

Certificate of Examination

This is to certify that we have examined the thesis *Vision Aided Inspection in Industries* submitted by *Shwetha Krishnamurthy, 14ME10072*, Undergraduate student of the department of Mechanical Engineering. We hereby accord our approval of it as a study carried out and presented in a manner required for its acceptance in partial fulfillment for the Undergraduate Degree for which it has been submitted. This approval does not necessarily endorse or accept every statement made, opinion expressed or conclusion drawn as recorded in this thesis.

EXAMINERS: SUPERVISORS:

Contents

1. Abstract

This project reviews the various vision-aided inspection techniques presently in use in manufacturing industries and proposes a a technique for inspection of produced goods, on a moving conveyor belt through a video feed through 2D and 3D Computer Vision, and Machine Learning. Different feature extraction techniques for feature matching with the ideal object features using Deep Learning and Computer Vision are studied.

The in-line inspection technique requires a sample image of the object to be inspected for defects. Scale Invariant Feature Transform (SIFT) algorithm is used to identify the object from a given scene image, in any scale or rotation.

Once the object is identified in the scene (here, the conveyor belt), it is compared with the sample image of the ideal object and features are matched. This will serve as a stepping stone for in-line inspection using a video feed (a series of images of the object moving on the conveyor belt from different viewpoints). The series of images thus obtained will be put-together to produce a point-cloud (3D geometry is found this way) and, hence, the point-cloud is now compared to the CAD model of the ideal sample of the object to be inspected. This method provides a way to detect dimensional defects in 3-dimensional space, without any requirement of depth sensing cameras.

2. Introduction

2.1 Motivation

Optical inspection for detecting and classifying defects on objects is essential to many manufacturing industries. In most industrial sectors, this task is done mostly by human workers, who are highly susceptible to fatigue over a period of time. Therefore, there is an increasing demand for automating such inspection tasks.

It is important that this inspection takes place through non-contact methods that do not interfere with the existing manufacturing process. Hence in many manufacturing plants, the inspection is performed on freely moving objects in a disturbed environment or it is performed in a separate inspection station.

There are different types of features that need to be inspected in a manufacturing plant, such as; 1) dimensional accuracy, 2) surface defects (scratches, cracks, etc.), 3) presence of foreign elements on the surface, 4) internal defects.

In order to describe the applications of machine vision systems, four categories can be broadly described:

1. Automated Visual Inspection
2. Parts Identification
3. Process Control
4. Robot Guidance and Control

2.2 Vision Aided Inspection in Industries

Beginning in the early 1980's, initial machine vision applications found ready markets where 100% inspection was required, and human inspection was too slow or erroneous. Examples include,automated optical inspection systems for inspection of multi-layer circuits, and 2-D and 3-D online gauging systems for automated coordinate measurement.

The automated vision system can be used for the purpose of measurements, gauging, integrity checking, and quality control. In the area of measurements and gauging, the gauging of small gaps, measurements of the object dimension, alignment of the components, and the analysis of crack formation are common applications. Integrity checking in automotive plants, food industry and other production lines is performed by using such a vision system.

The medical and pharmacological products can be inspected by the vision aided systems. Using such an inspection method in the production line has increased the speed and reliability of the inspections. For example, during the automotive assembly, a vision guided robot recognizes the orientation of the engine heads and picks and places them correctly on the engine blocks. In another case, a system examines the fibre optics assembly line. A PC-based imaging system integrates hardware and software to analyse the captured images for the possible fibre blemishes, chips, and cracks. As an another example, in aerospace industry, a vision-based robot using the self-calibrating and self-teaching techniques has been reported that punches rivets into the airplane metal sheets.

Parts identification and classification are one of the most important applications of a vision system. Sorting of the automotive castings, parts, and identifying and unloading of parts from pallets are important applications. Sorting and grading of the food and other products are another example of such identification applications.

2.3 Computer Vision and Machine Learning

Computer vision is an interdisciplinary field that deals with how computers can be made for gaining high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do.

Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, e.g., in the forms of decisions. Understanding in this context means the transformation of visual images (the input of the retina) into descriptions of the world that can interface with other thought processes and elicit appropriate action. This image understanding can be seen as the disentangling of symbolic information from image data using models constructed with the aid of geometry, physics, statistics, and learning theory.

Examples of applications of computer vision include systems for:

- Automatic inspection, e.g., in manufacturing applications;
- Assisting humans in identification tasks, e.g., a species identification system;
- Controlling processes, e.g., an industrial robot;
- Detecting events, e.g., for visual surveillance or people counting;
- Interaction, e.g., as the input to a device for computer-human interaction;
- Modeling objects or environments, e.g., medical image analysis or topographical modeling;
- Navigation, e.g., by an autonomous vehicle or mobile robot; and Organizing information, e.g., for indexing databases of images and image sequences.

There are a lot of libraries and frameworks available for implementing Computer Vision techniques. I aim to be using Python, C++ along with the open source library OpenCV for 2D computer vision and Point Cloud Library for 3D reconstruction.

3. Literature Review

3.1 CNN for automated feature extraction in industrial inspection

In the paper *Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection*, authors Weimer et.al. aim to segment possible defective area from the background and classify it in pre-defined defect categories using Convolutional Neural Networks.

A dataset from Weakly Supervised Learning for Industrial Optical Inspection (it is a corpus of defect detection on statistically textured surfaces) was used.

The architecture overview of the neural network is as follows:

- ReLU non-linearity used for fast computation.

- Network was divided into auto-encoder stages and stacked together to solve vanishing gradients problem (led to poor adjustments of weights in the initial layers of the network).

- Stochastic Gradient Descent used as the optimization algorithm.

- Dropout used to prevent over-fitting and for more effective training and especially prediction process, which is important for real world applications in dynamic and fast manufacturing processes

Results with different width (number of neurons) and depth (number of layers) were investigated. It is observed that adding layers increases accuracy by 2.35% and adding neurons increases accuracy by 1.34%.

Classification report in the end states that the deep CNN outperforms all existing techniques with respect to the overall detection accuracy, although no prior knowledge or manual input was feed into the CNN.

In conclusion, it is stated, ”In case the generation of training data is expensive, deep learning might not be an appropriate technology. Nevertheless, we believe, that this data driven approach of deep learning algorithms will be a key technique in big data analysis in different application domains in manufacturing.”

3.2 Deep Learning in the Automotive Industry: Applications and Tools

Deep Learning in the Automotive Industry: Applications and Tools, authors Luckow et.al. describe different automotive use cases for deep learning with special focus on computer vision. This paper surveys the current state-of-the-art techniques, creates an automotive dataset that allows to learn automatically and recognize different vehicle properties, and evaluates different architectures(AlexNet, Googlenet, Inception Module) on top of different frameworks(Tensorflow, Theano, Caffe, CNTK). It is showed that existing model architectures, and transfer learning can be applied to solve computer vision problems in the automotive domain. An accuracy of 85% was recorded in real world use using a mobile app and Amazon-cloud based storage and back-end.

3.3 Transfer Learning for Automated Optical Inspection

Kim et.al. in *Transfer Learning for Automated Optical Inspection* evaluate how Transfer Learning can be applied using image data from an entirely different domain, since the main problem with using of CNN is the lack of training data.

Transfer Learning is performed by using pre-trained weights from a source network to set the weights of target network. Now, this target network is used to extract features from the images and obtain significant performance improvements in object recognition tasks. Transfer Learning works well only for similar source and target domains, enough to fix the transferred weights to obtain meaningful features of the target data.

Fine tuning of the target network is done as follows:

- Transfer the source network weights up to the convolutional layers, i.e all convolutional layers of the target network are initialized by the transferred weights, while the remaining fully-connected layers are randomly initialized.

- Train the entire network using Stochastic Gradient Descent.

A constraint on using a pre-trained network is that the size of the input image should be equal to the size of the input layer of the network. Solution to this problem is to extract patches from training images in the target dataset. To augment the training set, 3 patches from each non-defective image are randomly extracted. To balance the number of defective and non-defective images, extract 20 patches around the defect region from each defective image. Additional rotation and flipping is also added to augment training sets.

At test time, windows of input size are chosen. Extract 9 slightly overlapped patches by the window size. It is important that these patches cover the whole image. Each patch forward propagated to yield prediction result. The most frequently occurring pattern is selected as the pattern. If there exists at least one patch predicted as defective, then it is marked as defective.

In conclusion, it states that,through fine-tuning, most of the unnecessary variation encoded in the weights of the source network are deactivated. At the same time, meaningful variations of the target dataset are amplified to capture the new simple variations of the target domain.

3.4 Defects Detection based on Deep Learning and Transfer Learning

In this paper, Ri-Xian et.al. propose a method which first establishes a Deep Belief Network and trains it according to the source domain sample features, in order to obtain weights according to the source domain samples.

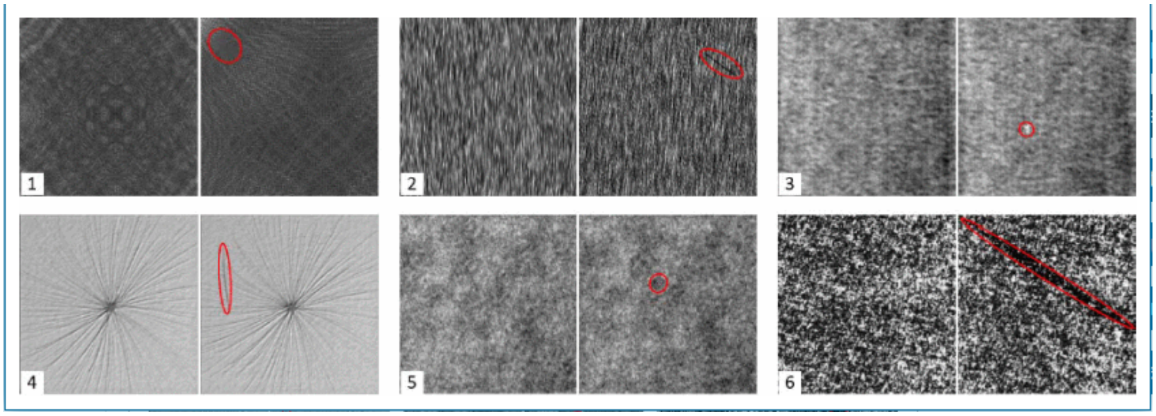


Figure 3.1: Sample images from the DAGM industrial optical inspection dataset. The dataset has 6 different texture patterns, and each pattern has a subset of data having defects in a particular location, which is marked with red ellipsoids in the figures. The classification is a task of differentiating these 12 different patterns whether the texture has defective parts or not as well as the data belong to which texture pattern.

The structure and parameters of the source domain DBN are transferred to the target domain and those samples are used to fine-tune the network to obtain mapping relationship between the target domain training samples and the defect-free samples.

This method has several shortcomings such as, it can only detect the product defects at the same locations and also it doesn't have translational and rotational invariance.

3.5 Transfer Learning Applied to a Building Quality Assessment Robot

Liu et.a. in this paper propose an automated post-construction quality assessment robot system for crack, hollowness, and finishing defects.

In this, a region proposal network is used to find the region of interest. R-CNN is used for feature extraction and a linear classifier is used with supervised learning as an object classifier. Moreover, active learning of top-N ranking region of interest is undertaken for fine-tuning of the transfer learning on convolutional activation feature network.

Excellent results were obtained by fine-tuning pre-trained models. Support Vector Machine used as the linear classifier.Detection is performed by an established regression threshold.

3.6 Deep Multi-Task Learning for Railway Track Inspection

Though, this paper and the previous one are not pertaining to inspection in manufacturing plants, the techniques shown can be applied to any scenario.

Gibert et.al. propose a railway track inspection technique using multi-task learning.

Multi task learning (MTL) is an inductive transfer learning technique in which two or more learning machines are trained co-operatively. MTL has a mechanism in which knowledge learned from one task is transferred to other tasks.

In this paper, a bottom-up approach is taken first to locate the fasteners and then to determine whether it is good or bad.

For best possible generation at test time, the detector is based on the maximum margin principle of the Support Vector Machine.

Instead of training a multi-class SVM, they have used the one-vs-rest strategy, and also, instead of treating the background class as just another class, they have treated it as a special case and used a pair of SVMs per object class. For instance, if they had used a single learning machine, they would be forcing the classifier to perform two different tasks:

- Reject the image that doesn't contain random texture.
- Reject the image that doesn't belong to the given category.

Therefore, given a set of object classes, they have trained 2 detectors for each object category. Asking the linear classifier to perform both tasks at the same time would result in a narrower margin than training separate classifiers for each individual task.

3.7 Automatic In-Line Inspection of Shape (Photogrammetry)

This paper by Bergstrom et.al. describes a fully automatic in-line shape inspection system for controlling the shape of moving objects on a conveyor belt. The shapes of the objects are measured using a full-field optical shape measurement method based on photogrammetry.

The photogrammetry system consists of four cameras, a flash, and a triggering device. When an object to be measured arrives at a given position relative to the system, the flash and cameras are synchronously triggered to capture images of the moving object. From the captured images a point-cloud representing the measured shape is created. The point-cloud is then aligned to a CAD-model, which defines the nominal shape of the measured object, using a best-fit method and a feature-based alignment method. Deviations between the point-cloud and the CAD-model are computed giving the output of the inspection process.

The steps involved are:

- Shape measurement of moving arbitrarily oriented objects.
- Alignment of the representation of the measured shape to the CAD model.
- Comparison with the CAD model.

In phtogrammetry, the natural pattern of the object, i.e. the surface structure, scratches, features, etc., are used to determine the 3D shape. Since the measured objects are arbitrarily oriented, the point-cloud must first be aligned to the CAD model before deviations can be found. Hence, a registration problem must be solved.

The comparison points are: 1) Surface points, 2) Trimmed edge points, and 3) Center points of circular holes.

In conclusion, it is found that it is possible to measure the shape of moving objects on a conveyor belt using phtogrammetry and do an automatic shape comparison with a nominal shape defined by a CAD model.

3.8 Some Existing Industrial Applications

3.8.1 Line-Scan Cameras to Check The Laser Cut Parts

Machine builders need metal parts of highest quality. Companies that supply these metal parts use laser cutting tools guided by computer numerical control to guide a laser beam to cut these parts from large sheets of flat metal.

Stemmer Imaging recently designed a vision-system for inspection for the laser cut metal parts manufacturer Strum-Gruppe Industries.

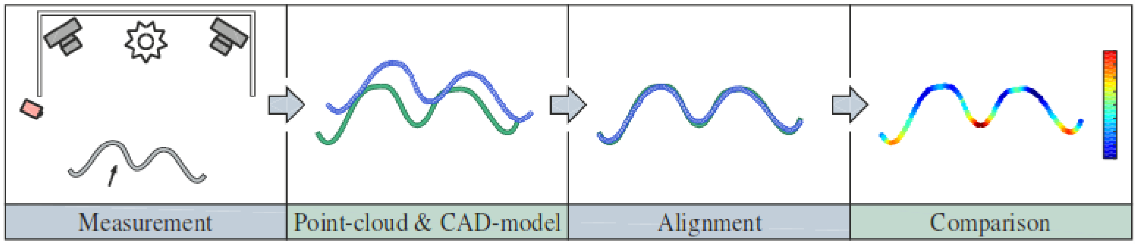


Figure 3.2: Illustration of the process for each measured part. 1) Measurement using photogrammetry. The system consists of cameras, a flash, and a triggering device. 2) CAD-model and resulting point-cloud in the same coordinate system. 3) Alignment of point-cloud to the CAD-model. 4) Comparison with the CAD-model. Computing deviations

In this machine, the conveyor belt carrying the finished parts moves at a speed of 20m/min. To check the type of each part, the alphanumeric code on each part must be checked and compared with the part number in the database.

After each part passes inspection, it must be measured to ensure that its characteristics meet that of the CAD data stored in the systems database. To do so, parts moving along the 170cm broad conveyor are first transferred into an imaging station. As the parts move through the system, they are imaged by eight line-scan cameras (three of which are shown) fitted with 150mm lenses that are mounted approximately 1.5-2m above the conveyor belt (3.3).

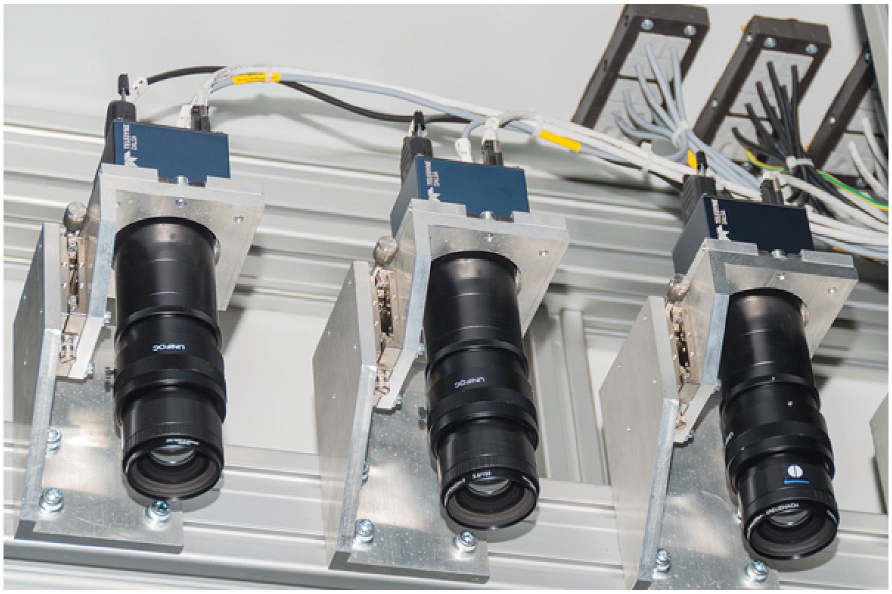


Figure 3.3: As the parts move through the system, they are imaged by eight line-scan cameras (three of which are shown) fitted with 150mm lenses that are mounted approximately 1.5-2m above the conveyor belt

To calibrate the system developed by Sturm-Gruppe, a calibration target is placed on conveyor belt and the operator then enters specific target values.Images captured by the camera must then be analyzed and compared with the known good data in the systems CAD database. Images are first thresholded to remove the image of the background image conveyor. Using shape finding and blob analysis tools, the contours and edges of objects of the metallic parts are then computed and stored in a database.

Comparisons between image data and the CAD model are displayed on the operator's monitor as green (complete match), yellow (partially good) or red (wrong). Parts that are not deemed sufficiently good are then removed by an operator after it emerges from the vision station.

3.8.2 Vibratory Feeder with Integrated Vision System Orients Parts

This vibratory feeder was designed to feed a metal washer into a machine assembling a consumer product. While flat, round parts like washers are normally easy to feed, this part had a small indentation on one side that needed to be oriented facing down.

A vision system was mounted over the vibratory bowl tooling right before the discharge to check the top to bottom orientation. If an improperly oriented part was detected, the PLC that controlled the vision system activated an air jet which blew the part off the track and back into the bowl feeder for recirculation.

The PLC that controlled the vision system was mounted in the junction box to the left of the camera, with the camera's monitor mounted on the front of the box.

3.8.3 Vision System inspects Hydraulic Components

A custom-built mechanical fixture and off-the-shelf machine-vision components combine to automatically inspect hydraulic subassemblies.

Kawasaki Precision Machinery, a leading manufacturer of hydraulic motors and pumps, approached Industrial Vision Systems to develop a machine-vision system that could inspect pumps as part of a semi-automated assembly process.

The vision system requires that each component and all its variants be presented repeatedly in the same position in a controlled lighting environment.As the image-processing system inspects and categorizes the devices, and only allows good products to continue down the production line, it also collects statistical data during the inspection process. After the machine has checked several batches of product, the data are used to determine how any changes in acceptable tolerances will affect the yield of the product.

In addition, the system provides ongoing statistical process control of the products being manufactured for later analysis.

3.8.4 Machine Vision for the Inspection of Mass Products

This article talks about inspection of 600 metal or injection-moulded components such as screws, disks or other mass products in fastener technology, mainly employed in the automobile and aerospace industries, by GEFRA GmbH in Friedewald. With STEMMER IMAGING machine vision playing a major role.

The objects to be inspected are placed loosely on the glass plates and are now passed past six stations where they are inspected with the aid of twelve cameras.

The current plant is used for inspecting screw joints for the brake lines of vehicles. This requires strict adherence to the tolerances of all geometrical sizes including the quality of the threads to ensure 100% functionality. This is where the special 360 degrees inspection with four cameras per pass, developed by GEFRA to detect faulty threads during a pass, comes into its own. It is also important to be able to detect chips or burrs resulting from previous work steps and to reliably sort out components with these faults.

The screws are delivered in bulk, separated out via a vibrating bowl selected and mechanically adapted by us, and fed into the actual sorting system in two possible positions. They are then transferred to the rotating glass ring and passed by the six inspection stations.

The first of these six stations measures the geometric dimensions, such as the thread diameter, the drill holes and the hex of the screws via edge pursuit in order to ensure the correct spanner width. In addition, possible burrs on the inspection objects are detected here

A further partial system is located in the same space of the station which measures a side view of the nominal and core diameters of the thread, the phase, height and helix.

The next station is equipped with LED dark field illumination and allows further contour inspection for chips. This is followed by surface inspection of the screws which detects damage such as impressions, scratches, deformation or coating faults, i.e. incomplete coating

The next inspection station, which is illuminated from top to bottom, serves to detect differences in coating as well as deviations in brightness and colour.

At the end of the completed inspection process, a check is conducted as to whether the threads follow the right direction, are continuous, and whether the thread tips and the thread roots meet requirements. Four cameras are installed at this station for this purpose, arranged at 90 degree intervals. Each of these cameras covers an angle of 110 to 120 degrees, so that the objects can be inspected all-round and overlapping. Only the good parts proceed and others are explicitly rejected.

4. Problem Description

4.1 Objectives and Scope of the present work

Defect detection, as an important aspect of industrial production, provides an important guarantee for the quality of products. Currently, many Chinese companies adopt the traditional manual approach for inspection of surface quality. Manual inspection is a type of contact inspection, which not only has low degree of automation, high labor intensity, low efficiency and high missed detection probability1, but also features greatly varying standards, which thus cannot meet the requirements of modern industry. With the development of computer vision technology, machine vision has gained application in the industrial product inspection. Machine vision-based defect detection system detects the color, shape and texture of products by machine vision technology, which is a non-contact inspection that does not require direct contact with products. Since the products inspected are not subject to secondary pollution, it is a truly environmentally-friendly inspection system.

Hence, my aim is to design and implement a fully-functional vision aided inspection system for mechanical parts using computer vision. The present methods are custom made to match the requirements of a particular industry and it is difficult to deploy a single algorithm for inspection to various manufacturing plants. Hence I aim to devise an algorithm as generalizable as possible so that numerous plants can inspected. Also, I aim to design the system to be capable of carrying out in-line inspection so that the production process need not stop just for the purpose of quality control.

5. Mathematical Formulation

5.1 Computer Vision Algorithms Used

Various basic Computer Vision algorithms have been implemented by me with OpenCV on C++. The details are outlined in the Appendix 1. Here, I shall elaborate on two main complex techniques:

1. *Scale Invariant Feature Transform* - It is basically used for matching features across different images. When all images are similar in nature (same scale, orientation, etc) simple corner detectors can work. But when we have images of different scales and rotations, we need to use the Scale Invariant Feature Transform.

2. *Structure From Motion* - It is a technique for obtaining information about the geometry of 3D scenes from 2D images. This task is challenging because the image formation process is not generally invertible: from its projected position in a camera image plane, a scene point can only be recovered up to a one-parameter ambiguity corresponding to its distance from the camera. Hence, additional information is needed to solve the reconstruction problem. This might be of tremendous use when there is need for inspection of objects while they are moving on the conveyor belt of the production line.

5.1.1 Scale Invariant Feature Transform

As the name suggest, SIFT feature matching is scale invariant, but in addition to that, we can get good result even when we change the following in the images:

1. Scale
2. Rotation
3. Illumination
4. Viewpoint

For example, let’s say we need to find an object in a given image of a scene, like shown in 5.1. The results of applying SIFT and identifying the object are shown in 5.2.

SIFT is quite an involved algorithm. It has a lot going on and can become confusing, So I’ve split up the entire algorithm into multiple parts. Here’s an outline of what happens in SIFT.

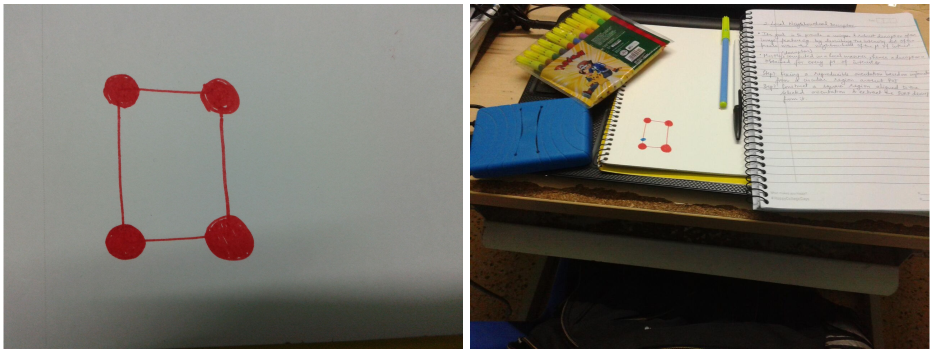


Figure 5.1: Given 2D object (left) to be identified in the scene (right)

Constructing a Scale Space

This is the initial preparation. You create internal representations of the original image to ensure scale invariance. This is done by generating a "scale space".

In the first step of SIFT, we generate several octaves of the original image. Each octave's image size is half the previous one. Within an octave, images are progressively blurred using the Gaussian Blur operator as seen in 5.3.

Mathematically, "blurring" is referred to as the convolution of the gaussian operator and the image. Gaussian blur has a particular expression or "operator" that is applied to each pixel. What results is the blurred image.

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y)$$

The symbols:

L is a blurred image

G is the Gaussian Blur operator

I is an image

x,y are the location coordinates

σ is the "scale" parameter. This is the amount of blur. Greater the value, greater the blur. The * is the convolution operation in x and y. It "applies" gaussian blur G onto the image I.

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

This is the actual Gaussian Blur operator.

Assume the amount of blur in a particular image is σ. Then, the amount of blur in the next image will be k*σ. Here k is whatever constant we choose.

LoG Approximation The Laplacian of Gaussian

This is great for finding interesting points (or key points) in an image. But it's computationally expensive. So we cheat and approximate it using the representation created earlier.

We take an image, and blur it a little. And then, we calculate second order derivatives on it (or, the "Laplacian"). This locates edges and corners on the image. These edges and corners are good for finding keypoints.

But the second order derivative is extremely sensitive to noise. The blur smoothes it out the noise and stabilizes the second order derivative.

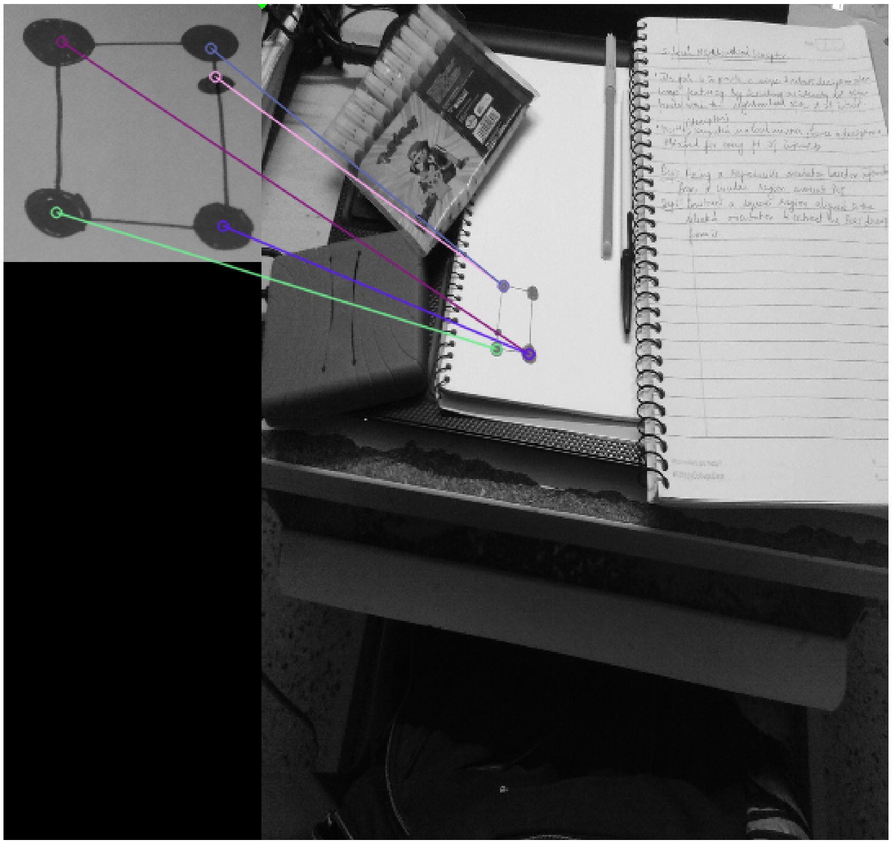


Figure 5.2: Keypoints matching using SIFT

The problem is, calculating all those second order derivatives is computationally intensive. So we cheat a bit by directly calculating the difference between two consecutive scales. Or, the Difference of Gaussians (6.6).

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

The σ in the denominator of the Gaussian function is the scale. If we get rid of it, we'll have true scale independence. So, if the laplacian of a gaussian is represented like this:

$$\nabla^2 G$$

Then the scale invariant laplacian of gaussian would look like this:

$$\sigma^2 \nabla^2 G$$

But all these complexities are taken care of by the Difference of Gaussian operation. The resultant images after the DoG operation are already multiplied by the σ^2.

Finding keypoints

Finding keypoints with the super fast approximation, we now try to find key points. These are maxima and minima in the Difference of Gaussian image we calculate in step 2.

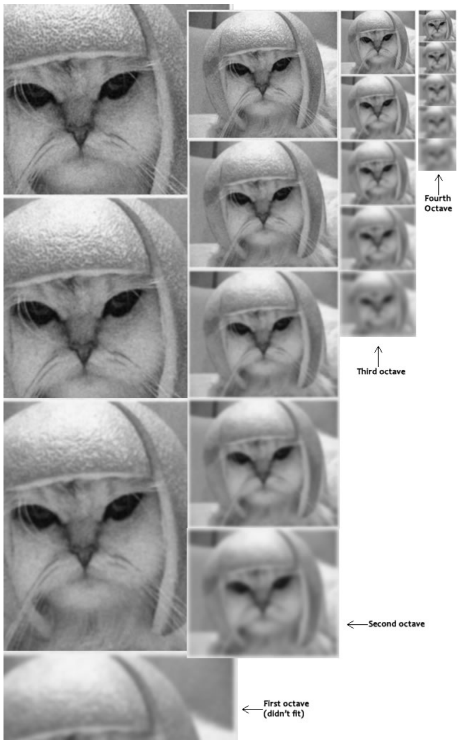


Figure 5.3: Images of the same size (vertical) form an octave. Above are four octaves. Each octave has 5 images. The individual images are formed because of the increasing "scale" (the amount of blur).

Finding key points is a two part process

1. *Locate maxima/minima in DoG images*: We iterate through each pixel and check all it's neighbours. The check is done within the current image, and also the one above and below it.

This way, a total of 26 checks are made. X is marked as a "key point" if it is the greatest or least of all 26 neighbours. Note that keypoints are not detected in the lowermost and topmost scales. There simply aren't enough neighbours to do the comparison.

2. *Find subpixel maxima/minima*: Using the available pixel data, subpixel values are generated. This is done by the Taylor expansion of the image around the approximate key point.

Mathematically, it's like this:

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}$$

We can easily find the extreme points of this equation (differentiate and equate to zero). On solving, we'll get subpixel key point locations. These subpixel values increase chances of matching and stability of the algorithm.

The author of SIFT recommends generating two such extrema images. So, we need exactly 4 DoG images. To generate 4 DoG images, we need 5 Gaussian blurred images. Hence the 5 level of blurs in each octave.

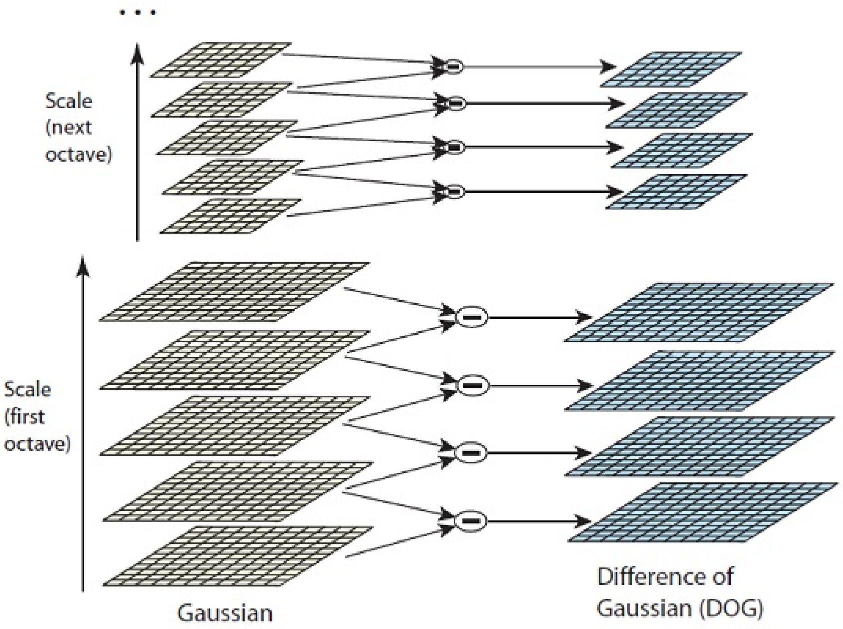


Figure 5.4: These Difference of Gaussian images are approximately equivalent to the Laplacian of Gaussian. And we've replaced a computationally intensive process with a simple subtraction (fast and efficient).

Get rid of Edges and low contrast regions

Key points on edges and low contrast regions are bad keypoints. Eliminating these makes the algorithm efficient and robust. A technique similar to the Harris Corner Detector is used here. [1]

1. *Removing low contrast features* is simple. If the magnitude of the intensity (i.e., without sign) at the current pixel in the DoG image (that is being checked for minima/maxima) is less than a certain value, it is rejected.

Because we have subpixel keypoints (we used the Taylor expansion to refine keypoints), we again need to use the taylor expansion to get the intensity value at subpixel locations. If it's magnitude is less than a certain value, we reject the keypoint.

2. *Removing edges* and retaining the corners because corners are great keypoints.We calculate two gradients at the keypoint. Both perpendicular to each other. If both of them are large we let it pass as a key point. Otherwise, it is rejected. Mathematically, this is achieved by the Hessian Matrix. Using this matrix, you can easily check if a point is a corner or not. See Appendix 1 for details.

Keypoint Orientation

Assigning an orientation to the keypoints An orientation is calculated for each key point. Any further calculations are done relative to this orientation. This effectively cancels out the effect of orientation, making it rotation invariant.

The idea is to collect gradient directions and magnitudes around each keypoint. Then we figure out the most prominent orientation(s) in that region. And we assign this orientation(s) to the keypoint.

Any later calculations are done relative to this orientation. This ensures rotation invariance.

[1] A COMBINED CORNER AND EDGE DETECTOR, *Chris Harris & Mike Stephens*

Gradient magnitudes and orientations are calculated using these formulae:

$$m(x,y) = \sqrt[2]{(L(x+1,y)-L(x-1,y))^2+(L(x,y+1)-L(x,y-1))^2}$$

$$\theta(x,y) = tan^{-}1((L(x,y+1)-L(x,y-1))/(L(x+1,y)-L(x-1,y)))$$

The magnitude and orientation is calculated for all pixels around the keypoint. Then, A histogram is created for this. Using the histogram, the most prominent gradient orientation(s) are identified. If there is only one peak, it is assigned to the keypoint. If there are multiple peaks above the 80% mark, they are all converted into a new keypoint (with their respective orientations).

Generating SIFT Features

Finally, with scale and rotation invariance in place, one more representation is generated. This helps uniquely identify features. Lets say we have 50,000 features. With this representation, we can easily identify the feature we're looking for (say, a particular eye, or a sign board).

We take a 16x16 window of "in-between" pixels around the keypoint. We split that window into sixteen 4x4 windows. From each 4x4 window you generate a histogram of 8 bins. Each bin corresponding to 0-44 degrees, 45-89 degrees, etc. Gradient orientations from the 4x4 are put into these bins. This is done for all 4x4 blocks. Finally, we normalize the 128 values we get.

To solve a few problems, we subtract the keypoint's orientation and also threshold the value of each element of the feature vector to 0.2 (and normalize again).

5.1.2 Structure From Motion

Structure from motion is a technique for obtaining information about the geometry of 3D scenes from 2D images.

This task is challenging because the image formation process is not generally invertible: from its projected position in a camera image plane, a scene point can only be recovered up to a one-parameter ambiguity corresponding to its distance from the camera. Hence, additional information is needed to solve the reconstruction problem.

One possibility is to exploit prior knowledge about the scene to reduce the number of degrees of freedom. For example parallelism and coplanarity constraints can be used to reconstruct simple geometric shapes such as line segments and planar polygons from their projected positions in individual views.

Another possibility is to use corresponding image points in multiple views. Given its image in two or more views, a 3D point can be reconstructed by triangulation. An important prerequisite is the determination of camera calibration and pose, which may be expressed by a projection matrix. The geometrical theory of structure from motion allows projection matrices and 3D points to be computed simultaneously using only corresponding points in each view. Structure from motion techniques are used in a wide range of applications including photogrammetric survey, the automatic reconstruction of virtual reality models from video sequences, and for the determination of camera motion (e.g. so that computer-generated objects can be inserted into video footage of real-world scenes).

I propose that this technique can be put to use in inspection of objects moving on the conveyor belt in a production process.

Available SFM Packages

As part of the research into previous work, a survey of the existing open-source software that has been developed by various researchers was conducted. Based on this research, it appears that the majority of the current toolkits are based on the Bundler package, a Structure from Motion system for unordered image collections developed by N. Snavely(Ref S1). It was released as an outcome of the Photo Tourism project (Ref S1).

Bundler generates a sparse 3D reconstruction of the scene. For dense 3D reconstruction, the preferred approach seems to be to use the multi view stereo packages CMVS and PMVS, developed by Y. Furukawa (Ref S2).

Bundler, CMVS and PMVS are all command line tools. As a result, a number of other projects have developed integrated toolkits and visualization packages based on these tools.

Of note are the following, which were evaluated as part of this project:

• OSM Bundler (Ref S3) - a project to integrate Bundler, CMVS and PMVS into Open Street Map

• Python Photogrammetry Toolbox (PPT) (Ref S4) - a project to integrate Bundler, CMVS and PMVS into an open-source photogrammetry toolbox by the archeological community.

• Visual SFM (Ref S5) - a highly optimized and well integrated implementation of Bundler, PMVS and CMVS. Of particular note are the inclusion of a GPU based SIFT algorithm (SiftGPU) and a multi-core implementation of the Bundle Adjustment algorithm. The use of these packages allows VisualSFM to perform incremental Structure from Motion in near linear time.

Several packages are available for visualization of point clouds, notably MeshLab, CloudCompare and the Point Cloud Library (PCL) which integrates nicely with OpenCV.

For this project, first Visual SFM was deployed to obtain a dense Point Cloud. Once the point cloud was created, Poisson's Surface Reconstruction was used to create a mesh of the object. In order to perform the comparison with the CAD model to inspect the fitness of the produced good, I have used Iterative Closest Point Registration technique for alignment of the CAD model with the generated point cloud, and hence generation of a rotation and translation matrices, along with the fitness factor, which is just the average vector difference between two corresponding points on the point cloud and the template.

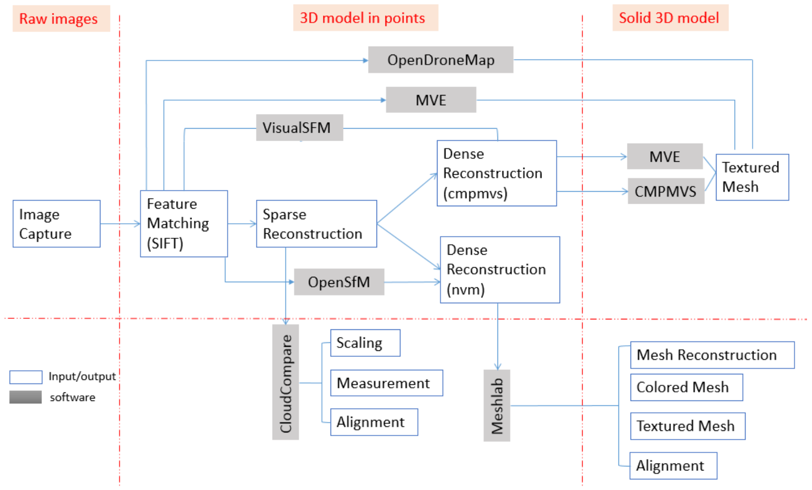


Figure 5.5: Workflow required for different applications

Visual SFM

The time complexity of incremental structure from motion (SfM) is often known as O(n4) with respect to the number of cameras. As bundle adjustment (BA) being significantly improved recently by preconditioned conjugate gradient (PCG), it is worth revisiting how fast incremental SfM is. Changchang Wu's technique introduces a novel BA strategy that provides good balance between speed and accuracy. Through algorithm analysis and extensive experiments, it was shown that incremental SfM requires only O(n) time on many major steps including BA.

This essentially gives a dense point cloud from a series of unordered images by using PMVS and CMVS.

The detailed algorithm is mentioned in Appendix 2.

Poisson Surface Reconstruction

After scanning an object into a point cloud, we want to create a mesh from this points. There are different approaches for creating a mesh, for example:

- Poisson Surface Reconstruction
- Ball Pivoting
- Power Crust

Because we want to get a watertight and smooth surface I decided to use the "Poisson Surface Reconstruction". This approach was introduced by Michael Kazhdan and is used for example in the mesh processing software "MeshLab" as well as in the CGAL (Computational Geometry Algorithms Library). Since there is also an implementation from Kazhdan available (which is by the way also used in Meshlab), PCL used his code in their framework via a wrapper class, and I've used the function in PCL.

The detailed algorithm is mentioned in Appendix 3.

Point Cloud Alignment using Iterative Closest Point Registration

Now that the mesh is created, now we need to compare it with a CAD model and provide a fitness score on the basis of which, it will be decided if the product is good or not.

For this Iterative Closest Points (ICP) algorithm, the mainstream algorithm used in the process of accurate registration of 3D point cloud data, is used. The algorithm requires a proper initial value and the approximate registration of two point clouds to prevent the algorithm from falling into local extremes, but in the actual point cloud matching process, it is difficult to ensure compliance with this requirement. Therefor, this method uses the geometrical features of the point cloud to be registered, such as curvature, surface normal and point cloud density, to search for the correspondence relationships between two point clouds and introduces the geometric features into the error function to realize the accurate registration of two point clouds.

This error function is used as a fitness score. The detailed algorithm is mentioned in Appendix 4.

6. Solution strategy or methodology

6.1 Implementation of SIFT

In order to implement the SIFT algorithm on mechanical parts, I clicked a couple of pictures of the object to be identified and the scene in which it needs to be identified, like shown in 6.1.

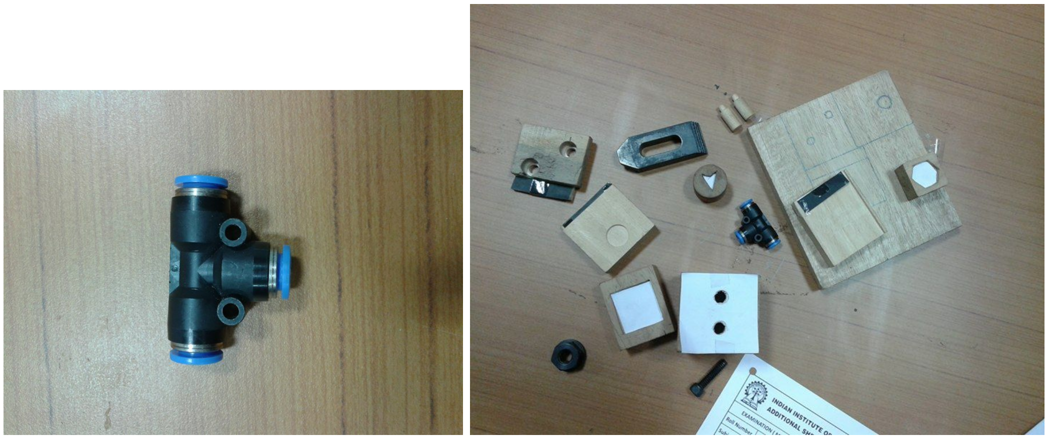


Figure 6.1: Given 3D object (left) to be identified in the scene (right)

In order to obtain better results, I pre-processed the images by removing the shadows from both the images. This was done by converting the image into HSV (Hue, Saturation, and Value) scale, and setting the value parameter to a fixed integer (between 0 and 255) throughout the image. The result of shadow removal is shown in 6.2.

The idea is to identify the object in the given scene and determine its orientation so that dimensional inspection can be carried out.

Hence, we use SIFT algorithm for identification and determination of the orientation, as discussed in Chapter 5. Direct usage of SIFT gives too many keypoints. Therefore we find a bounding box of the various components in the scene image, and take into account only those keypoints inside the

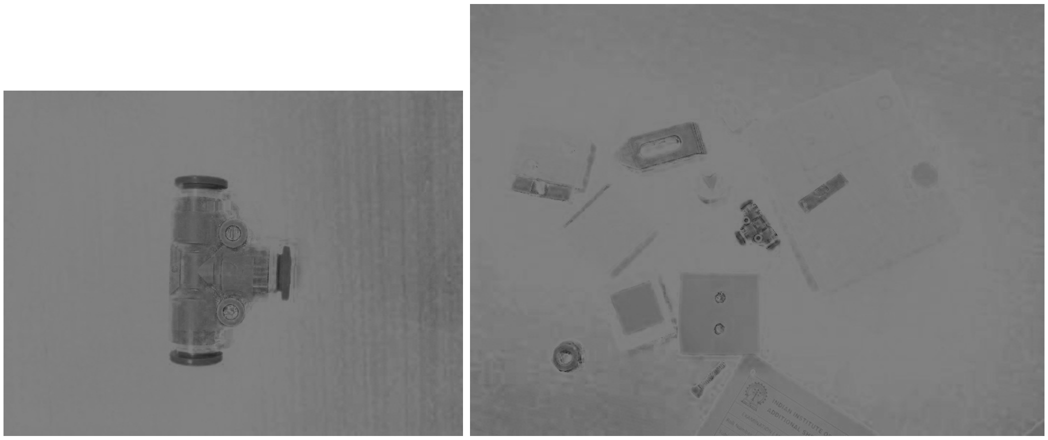


Figure 6.2: The object to be identified (left) and the scene in which it needs to be identified (right) with the shadows removed and converted to grayscale.

contour where there is maximum concentration of keypoints (which is the object identified in the scene).

The results of bounding box generation are shown in 6.3.

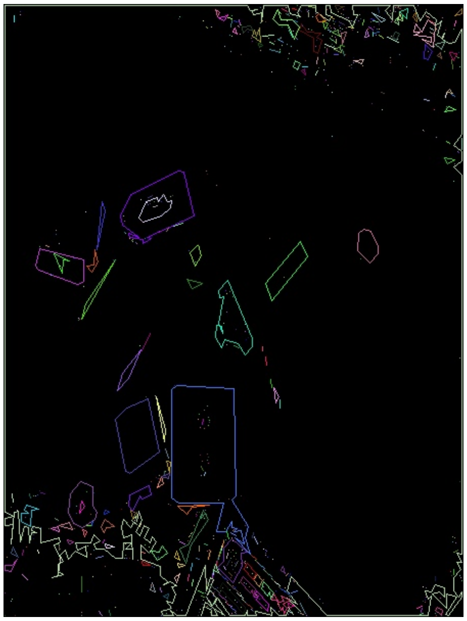


Figure 6.3: Contours generated; the bounding box of the object can be seen in green

The 6.4 shows the final results after applying SIFT.

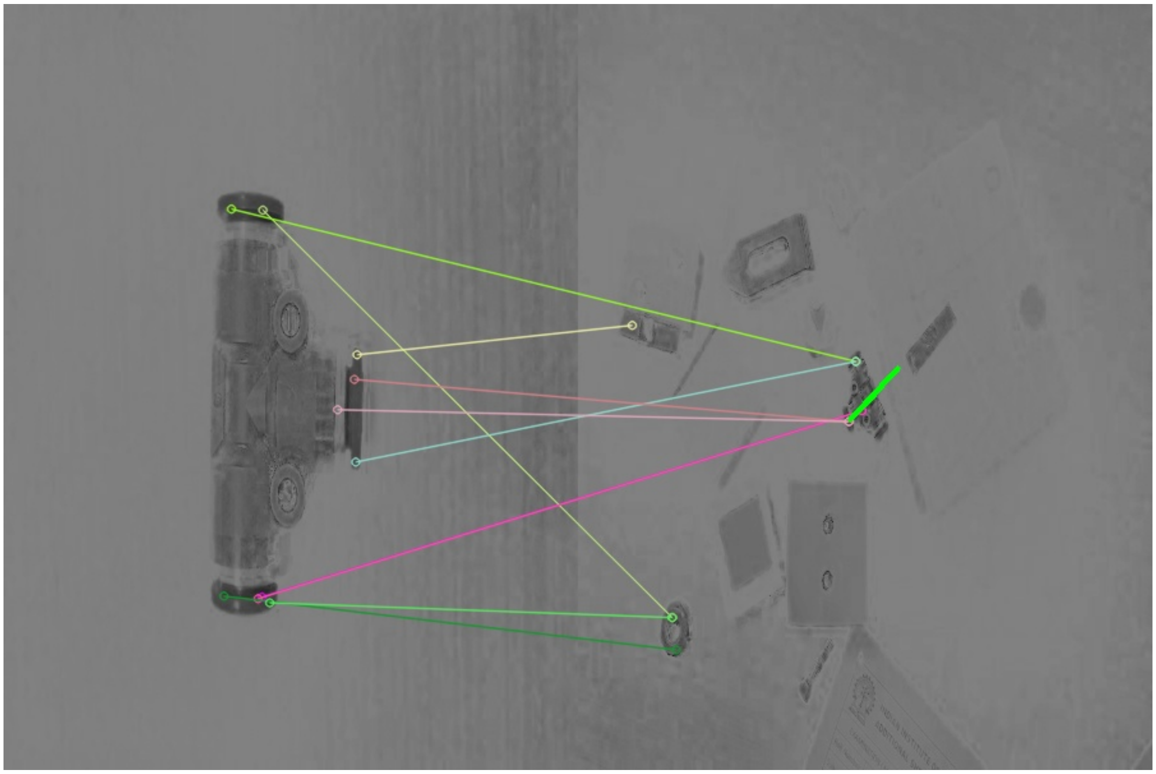


Figure 6.4: Keypoints matches; the axis (orientation) of the object detected in the can be seen in bright green

6.2 Structure From Motion

Visual SFM

The first step is to capture snaps from a video of the object from different angles. We can exploit the motion of the conveyor belt to obtain a few number of viewpoints, but still, we need some motion of the camera too to be able to capture the object from all possible angles.

For experimentation purposes, I have used an external hard drive as the object. I made a video of the object from all angles, and captured every 25th frame, and stored them. Example snaps are shown in 6.5.

These snaps obtained from the video are then loaded in the Visual SFM software.The snaps after being loaded on the platform is shown in Fig:6.6.

Once the snaps are loaded, SIFT Feature Extractor and Matcher is run by using pairwise SIFT matching to compute missing matches. Once the missing matches are computed, the sparse point cloud from the images is generated. Then, the dense reconstruction of the point cloud is performed using CMVS. The final point cloud after dense reconstruction is shown in 6.7.

Poisson Reconstruction

Once the dense point cloud is obtained, poisson's surface reconstruction is applied and a mesh is generated. This has been coded using Point Cloud Library in C++. The mesh generated is visualized using Meshlab as shown in Fig:6.8

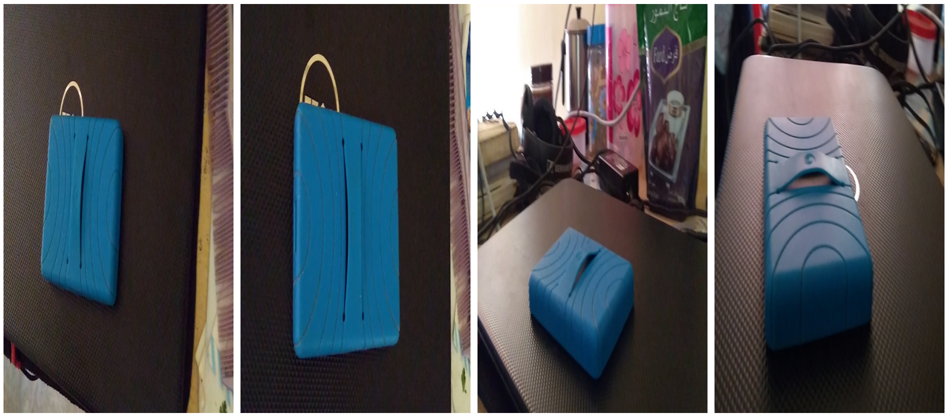
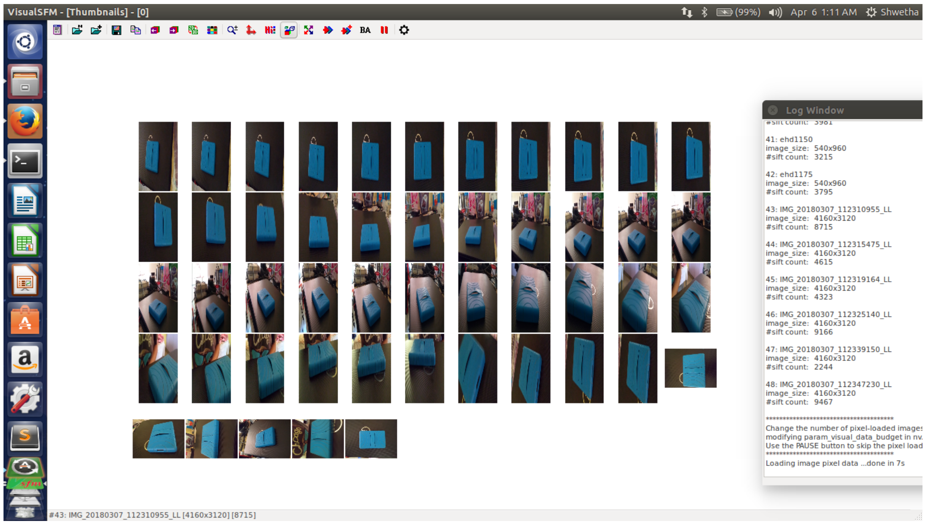


Figure 6.5: Example images of the external hard drive amidst the clutter



```
#sift count:  3981

41: ehd1150
image_size:  540x960
#sift count:  3215

42: ehd1175
image_size:  540x960
#sift count:  3795

43: IMG_20180307_112310955_LL
image_size:  4160x3120
#sift count:  8715

44: IMG_20180307_112315475_LL
image_size:  4160x3120
#sift count:  4615

45: IMG_20180307_112319164_LL
image_size:  4160x3120
#sift count:  4323

46: IMG_20180307_112325140_LL
image_size:  4160x3120
#sift count:  9166

47: IMG_20180307_112339150_LL
image_size:  4160x3120
#sift count:  2244

48: IMG_20180307_112347230_LL
image_size:  4160x3120
#sift count:  9467

****************************************
Change the number of pixel-loaded images
modifying param_visual_data_budget in nv.
Use the PAUSE button to skip the pixel load
****************************************
Loading image pixel data ...done in 7s
```

```
#43: IMG_20180307_112310955_LL [4160x3120] [8715]
```

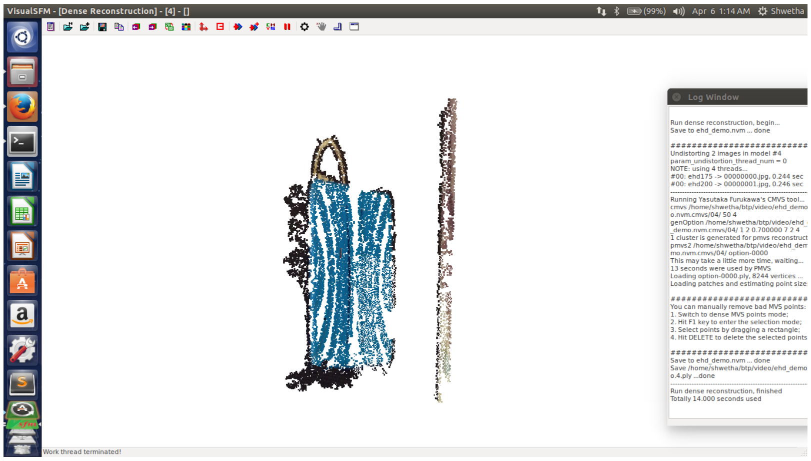
Figure 6.6: Loading the snaps in VisualSFM

Iterative Closest Point Registration and Comparison

Once the mesh is obtained, we need to compare this with the CAD model of the product and give a fitness score. This is done, as was discussed in Chapter 5, through Iterative Closest Point registration process. This part was also coded in C++ using the Point Cloud Library. The output of this step is shown in Fig:6.9. The fitness score, the rotation matrix, and translation vector obtained is to be observed.

The Future Role of Socket Programming

Till now, all the code for this project has been written in C++ using OpenCV and Point Cloud Libraries. The only step that needs a GUI software is creation of dense point-cloud using VisualSFM. Therefor, socket programming in C++ can help us access the software and operate through a C++ server. Socket Programming is essentially used in establishing a connection between a server and a client. Here, the VisualSFM software will serve as the client and a C++ program will serve as the server. I aim to integrate this part too in the future.


VisualSFM - [Dense Reconstruction] - [4] - []
(99%) Apr 6 1:14 AM Shwetha
Log Window
Work thread terminated!

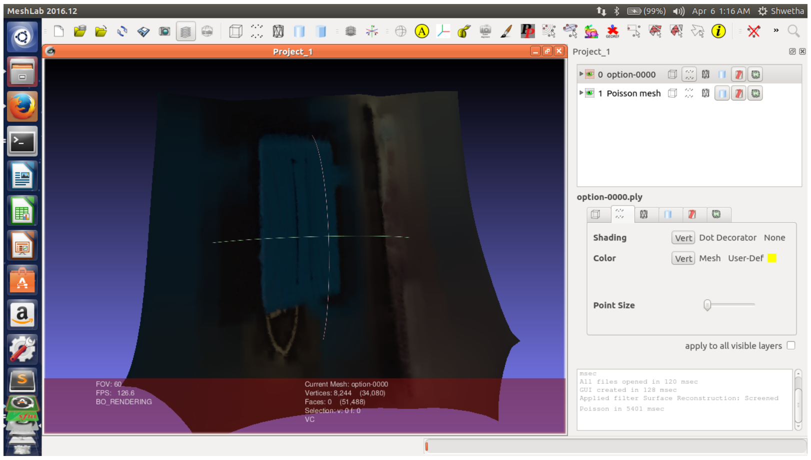
```
Run dense reconstruction, begin...
Save to ehd_demo.nvm ... done

###########################
Undistorting 2 images in model #4
param_undistortion_thread_num = 0
NOTE: using 4 threads...
#00: ehd175 -> 00000000.jpg, 0.244 sec
#00: ehd200 -> 00000001.jpg, 0.246 sec
------------------------------------------
Running Yasutaka Furukawa's CMVS tool...
cmvs /home/shwetha/btp/video/ehd_demo
o.nvm.cmvs/04/ 50 4
genOption /home/shwetha/btp/video/ehd_
_demo.nvm.cmvs/04/ 1 2 0.700000 7 2 4
1 cluster is generated for pmvs reconstruct
pmvs2 /home/shwetha/btp/video/ehd_dem
mo.nvm.cmvs/04/ option-0000
This may take a little more time, waiting...
13 seconds were used by PMVS
Loading option-0000.ply, 8244 vertices ...
Loading patches and estimating point size

###########################
You can manually remove bad MVS points:
1. Switch to dense MVS points mode;
2. Hit F1 key to enter the selection mode;
3. Select points by dragging a rectangle;
4. Hit DELETE to delete the selected points

###########################
Save to ehd_demo.nvm ... done
Save /home/shwetha/btp/video/ehd_demo
o.4.ply ...done
------------------------------------------
Run dense reconstruction, finished
Totally 14.000 seconds used
```

Figure 6.7: Dense Point Cloud Generated from Visual SFM



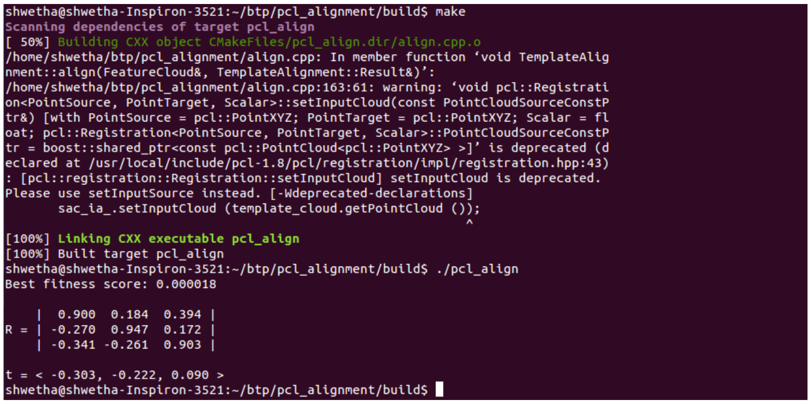
Shading Vert Dot Decorator None
Color Vert Mesh User-Def

FOV: 60
FPS: 126.6
BO_RENDERING

Current Mesh: option-0000
Vertices: 8,244 (34,080)
Faces: 0 (51,488)
Selection: v: 0 f: 0
VC

```
msec
All files opened in 120 msec
GUI created in 128 msec
Applied filter Surface Reconstruction: Screened
Poisson in 5401 msec
```

Figure 6.8: Mesh Generation after Poisson Surface Reconstruction



```
shwetha@shwetha-Inspiron-3521:~/btp/pcl_alignment/build$ make
Scanning dependencies of target pcl_align
[ 50%] Building CXX object CMakeFiles/pcl_align.dir/align.cpp.o
/home/shwetha/btp/pcl_alignment/align.cpp: In member function ‘void TemplateAlig
nment::align(FeatureCloud&, TemplateAlignment::Result&)’:
/home/shwetha/btp/pcl_alignment/align.cpp:163:61: warning: ‘void pcl::Registrati
on<PointSource, PointTarget, Scalar>::setInputCloud(const PointCloudSourceConstP
tr&) [with PointSource = pcl::PointXYZ; PointTarget = pcl::PointXYZ; Scalar = fl
oat; pcl::Registration<PointSource, PointTarget, Scalar>::PointCloudSourceConstP
tr = boost::shared_ptr<const pcl::PointCloud<pcl::PointXYZ> >]’ is deprecated (d
eclared at /usr/local/include/pcl-1.8/pcl/registration/impl/registration.hpp:43)
: [pcl::registration::Registration::setInputCloud] setInputCloud is deprecated.
Please use setInputSource instead. [-Wdeprecated-declarations]
       sac_ia_.setInputCloud (template_cloud.getPointCloud ());
                                                             ^

[100%] Linking CXX executable pcl_align
[100%] Built target pcl_align
shwetha@shwetha-Inspiron-3521:~/btp/pcl_alignment/build$ ./pcl_align
Best fitness score: 0.000018

    |  0.900  0.184  0.394 |
R = | -0.270  0.947  0.172 |
    | -0.341 -0.261  0.903 |

t = < -0.303, -0.222, 0.090 >
shwetha@shwetha-Inspiron-3521:~/btp/pcl_alignment/build$ █
```

Figure 6.9: Output After Alignment and Comparison

7. Results, Discussions, and Conclusions

The results of implementation of SIFT are outlined in the previous chapter. It is observed that there is a lot of scope for improvement of results using tuning of hyper-parameters.

As with Machine Learning methods, we need a lot of training data to be able to carry out any classification, which is difficult to procure. Additionally, a trained ML model, that is trained to perform a particular type of classification (or any other task) for a particular object for inspection is not portable to other objects, as it would require training of the model once again altogether. It is important to notice that most mechanical parts don't show much intra-class variation, hence, using an ML algorithm is somewhat unnecessary.

Scale Invariant Feature Transform (or SURF - Speeded Up Robust Features, which is a sped up version of SIFT with even better approximations) requires just one image of the object to be able to locate it any scene, in any orientation, and is also portable, as you don't require any training to be able to implement it. Also, it is a fast algorithm, which can fetch results in the small period of time between image capturing (on the conveyor belt) and decision making (a robotic arm that might push a defective piece from the assembly line.

The pipeline proposed for obtaining the point cloud by SIFT feature matching without the use of a depth sensor, or stoppage of the conveyor belt, makes this a novel technique which can be deployed in the industries with minimal changes. The future work for this project involves making a Graphics User Interface integrated with the C++ program as the back-end.

Bibliography

[1] Daniel Weimerac, *Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection*, CIRP Annals Volume 65, Issue 1, 2016, Pages 417-420.

[2] Andre Luckow, *Deep Learning in the Automotive Industry: Applications and Tools*, 2016 IEEE International Conference on Big Data (Big Data) 978-1-4673-9005-7/16/.

[3] Seunghyeon Kim, *Transfer Learning for Automated Optical Inspection*, 2017 International Joint Conference on Neural Networks (IJCNN).

[4] L. Ri-Xian, *Defects Detection Based on Deep Learning and Transfer Learning*, 2015 Metal Journal No. 7.

[5] Lili Liu, *Transfer learning on convolutional activation feature as applied to a building quality assessment robot*, International Journal of Advanced Robotic Systems, May-June 2017: 1–12.

[6] Xavier Gibert, *Deep Multi-task Learning for Railway Track Inspection*, IEEE Trans. on Pattern Analysis and Machine Intelligence, 2015.

[7] Bergström, Per , *Automatic in-line inspection of shape based on photogrammetry*, The 7th International Swedish Production Symposium, SPS16, Conference Proceedings: 25th – 27th of October 2016, Lund: Swedish Production Academy , 2016, 1-9 p.

[8] *Line-Scan Cameras to Check The Laser Cut Parts*,

`http://www.vision-systems.com/articles/print/volume-21/issue-8/features/line-scan-cameras-check-the-quality-of-laser-cut-parts.html`

[9] *Vibratory Feeder with Integrated Vision System Orients Parts*

`http://performancefeeders.com/custom-applications/vibratory-feeder-with-integrated-vision`

[10] *Vision System inspects Hydraulic Components*

http://www.vision-systems.com/articles/print/volume-18/issue-5/features/vision-system-inspects-hydraulic-components.html

[11] *Machine Vision for the Inspection of Mass Products*

https://www.stemmer-imaging.de/media/uploads/websites/documents/applications/en_DE-ApplicationStory-Gefra_Fastener_Inspection-201407.pdf

[12] David Lowe, *Distinctive Image Features from Scale-In variant Keypoints*, International Journal of Computer Vision, 2004.

[13] *SIFT: Theory and Practice* http://www.aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction/

[14] Linda G. Shapiro; George C. Stockman (2001). *Computer Vision*. Prentice Hall. ISBN 0-13-030796-3.

[15] K. Häming & G. Peters (2010). *The structure-from-motion reconstruction pipeline – a survey with focus on short image sequences*. Kybernetika.

[16] H. Bay; T. Tuytelaars & L. Van Gool (2006). *Surf: Speeded up robust features*, 9th European Conference on Computer Vision.

[17] F. Dellaert; S. Seitz; C. Thorpe & S. Thrun (2000).*Structure from Motion without Correspondence*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

[18] *Structure From Motion*

http://mi.eng.cam.ac.uk/~cipolla/publications/contributionToEditedBook/2008-SFM-chapters.pdf

[19] *OpenCV Documentation*

http://docs.opencv.org/2.4/doc/tutorials/tutorials.html

[20] K.S. Fu, R.C. Gonzalez, C.S.G. Lee, *Robotics - Control, Sensing, Vision, and Intelligence* (Book form).

[21] Ying He, Bin Liang, Jun Yang, Shunzhi Li, and Jin He, *An Iterative Closest Points Algorithm for Registration of 3D Laser Scanner Point Clouds with Geometric Features* (Sensors (Basel)v.17(8); 2017 Aug).

[22] Changchang Wu, *Towards Linear-time Incremental Structure from Motion* (3D Vision-3DV 2013, 2013 International Conference on, pages 127–134. IEEE, 2013).

[23] Y.Furukawa, B.Curless, S.Seitz, R.Szeliski, *Towards Internet-scale Multi-view Stereo* (CVPR (2010))

[24] *Bundler*

https://www.cs.cornell.edu/~snavely/bundler/

[25] *CMVS/PMVS*

http://www.di.ens.fr/cmvs/

[26] *OSM Bundler*

https://code.google.com/p/osm-bundler/

[27] *VisualSFM*

http://ccwu.me/vsfm/

[28] Hartley and Zisserman, *Multiple View Geometry* (Book Form)

[29] Shawn McCann, *3D Reconstruction from Multiple Images* (Term Paper of a student of CS231a course, 2015 at Stanford)

[30] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe *Poisson Surface Reconstruction*(Eurographics Symposium on Geometry Processing (2006))