Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Iwer Petersen

Using object tracking for dynamic video projection mapping

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Iwer Petersen

Using object tracking for dynamic video projection mapping

Bachelorthesis submitted within the scope of the Bachelor examination

in the degree programme Bachelor of Science Technical Computer Science
at the Department of Computer Science
of the Faculty of Engineering and Computer Science
of the Hamburg University of Applied Science

Mentoring examiner: Prof. Dr. Ing. Birgit Wendholt
Second expert: Prof. Dr.-Ing. Andreas Meisel

Submitted at: January 31, 2013

Iwer Petersen

Title of the paper

Using object tracking for dynamic video projection mapping

Keywords

video projection mapping, object tracking, point cloud, 3D

Abstract

This document presents a way to realize video projection mapping onto moving objects. Therefore a visual 3D tracking method is used to determine the position and orientation of a known object. Via a calibrated projector-camera system, the real object then is augmented with a virtual texture.

Iwer Petersen

Thema der Arbeit

Objekttracking für dynamisches Videoprojektions Mapping

Stichworte

Video Projektions Mapping, Objektverfolgung, Punktwolke, 3D

Kurzzusammenfassung

Dieses Dokument präsentiert einen Weg um Video Projektions Mapping auf sich bewegende Objekte zu realisieren. Dafür wird ein visuelles 3D Trackingverfahren verwendet um die Position und Lage eines bekannten Objekts zu bestimmen. Über ein kalibriertes Projektor-Kamera System wird das reale Objekt dann mit einer virtuellen Textur erweitert.

Contents

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Structure . . . 3

2 Related Work **4**
2.1 Corresponding projects . . . 4
2.1.1 Spatial Augmented Reality . . . 4
2.1.2 2D Mapping onto people . . . 5
2.1.3 Marker based tracking of known projection surfaces . . . 6
2.1.4 Mapping on servo actuated Objects . . . 7
2.1.5 Mapinect . . . 8
2.1.6 KinectFusion . . . 10
2.2 Technology to consider . . . 10
2.2.1 Calibration . . . 10
2.2.2 Reconstruction . . . 11
2.2.3 Tracking . . . 12
2.3 Discussion . . . 13

3 Analysis **15**
3.1 Subproblem Analysis . . . 15
3.1.1 3D sensor . . . 15
3.1.2 Projector camera calibration . . . 16
3.1.3 Dynamic model creation / Object tracking . . . 16
3.1.4 Virtual Scene Construction . . . 17
3.2 Summary . . . 18

4 Realization **20**
4.1 Assumptions . . . 21
4.2 Tools and Libraries . . . 21
4.2.1 Point Cloud Capturing Sensor . . . 21
4.2.2 Projector Camera Calibration . . . 22
4.2.3 Libraries . . . 23
4.2.4 Data structures and Algorithms . . . 24
4.3 Solution . . . 30
4.3.1 Calibration . . . 31
4.3.2 Model acquisition . . . 32

4.3.3 Object tracking 35
4.3.4 Virtual Scene Construction 35
4.4 Evaluation 36
4.4.1 The calibration parameters 37
4.4.2 Model acquisition 38
4.4.3 Runtime 39
4.4.4 Measurements 40
4.5 Summary 44

5 Conclusion **45**

1 Introduction

1.1 Motivation

Computer science is more and more incorporated by artists for inspiration or for concrete realisations by using programming languages as design medium (see Bohnacker et al. [7]). Several environments like the Java based Processing language ([35]) or the openFrameworks toolkit ([28]) which is written in C++ provide a convenient starting point for creative coding. It is not uncommon that software created to solve a technical task is being used for creative applications. Artists use for example software like OpenCV for image processing as well as micro-controllers like Arduino to realize artistic installations.

Video projection mapping is part-discipline where virtual textures are projected to real physical objects with high lumen video projectors to create an immersive illusion. For example like in figure 1.1 a white surface consisting of tetrahedrons was constructed and all sides visible to the projector are augmented with slowly fading color gradient textures. In reality even much more complex surfaces like buildings are used as projection surfaces. Traditionally video projection mapping is performed on static scenes.

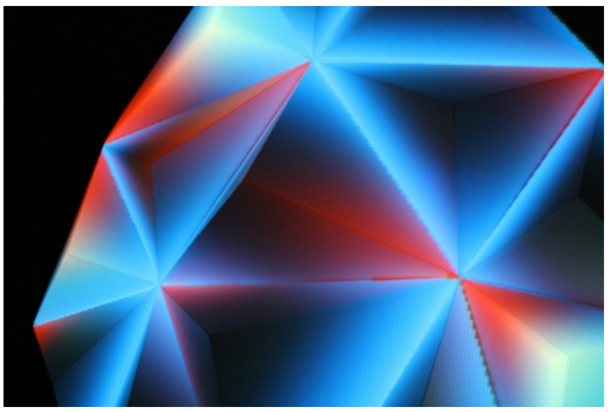


Figure 1.1: 3D Video Projection Mapping: CRYSTALCHOIR by Andrea Sztojánovits, Gábor Borosi

To accomplish such a projection it is necessary to have the virtual world, where the textures are created, aligned to the physical world where those textures shall fit exactly onto a limited surface. That is mostly done by manually align binary mask images to the physical scene

via the projector and apply those mask to the video content to be displayed or by rendering textures onto aligned 2D polygons (*2D mapping*).
A more advanced technique reconstructs the whole scene that should be augmented as a virtual 3D model and chooses a viewpoint for the virtual scene that matches the projectors viewpoint onto the real, physical scene (*3D mapping*). With this technique illusionary 3D deformation or lighting effects are much easier to achieve.

Both methods have in common that the process of aligning the virtual scene to the physical world (commonly called *mapping*) is a protracted process. They also have in common that if the aligning is done, nothing in the physical scene or in the projectors position can be changed without compromising the tedious mapping process.

This thesis is driven by a quite futuristic vision originating from the field of fashion design. The idea is to augment white clothing with virtual textures that are projected onto the clothes. This can be seen as a dynamic form of 3D video projection mapping, where, opposed to traditional static projection mapping, the augmented object is moving and deforming within the scene.

To enhance video projection mapping for dynamic scenes with moving or non-rigid projection surfaces, the virtual scene has to be generated in real-time according to the changes in the physical scene. For 2D Mapping this can be achieved by automatically adjusting binary masks using a camera based object recognition method. For 3D mapping a virtual 3D scene has to be recreated dynamically. The shape, position and orientation of the object is assumed to be variable and has to be remodelled constantly in the virtual scene.
In both cases sensors are required to observe the changing scene. For 2D projection mapping this can be cameras whose images can be processed with computer vision methods (see Szeliski [42]) to reveal necessary data to reconstruct the scene.
For 3D mapping a stereo camera system or modern range sensors can be used to measure the geometric changes of the object. Those sensors mostly provide data as clouds of 3D surface points that can be used to re-construct the virtual scene (see Bernardini and Rushmeier [3]).

This thesis presents a method to perform dynamic projection mapping by using detection algorithms on a 3D point cloud captured with a 3D sensor. The problem to be solved consists of several sub problems which will be further discussed in the following:

- **Projector-camera-calibration:** One of the most obvious issue is that a projector cannot be in the same location as a camera and therefore a viewpoint transformation has to be applied to the scene. This problem is widely known as projector-camera calibration and is not present in traditional video projection mapping as the alignment takes place in projector space and usually no cameras are needed.

- **Dynamic model creation / Object tracking:** Further the shape and pose of the object has to be known to overlay its surfaces with virtual textures. This can be achieved in two ways. Either one can generate a model of the object for every video frame which still is a big challenge on the data processing side (see Bi and Wang [5]), or one can track a known object in the scene to obtain its pose. This problem is not present in traditional video projection mapping as this happens on static scenes so that the model can be created in advance.

- **Virtual scene creation:** Finally the video animation has to be applied to the virtual scene as a texture. In 2D mapping this is done by masking the texture so that it only appears on the object. In 3D mapping, where a 3D model of the object is used for rendering the virtual scene, a standard 3D rendering technology like OpenGL or DirectX can be used. In addition to a 3D mesh representation of the model a 2D texture is created that contains all outer surfaces of the model in an unwrapped form. In the 3D model for every point (or vertex) in space a 2D texture coordinate is computed that defines which point on the texture corresponds to which point on the model.

1.2 Structure

The remaining work is structured as follows. To get an overview of existing solutions some work that projects video on moving objects and technical solutions for the different problem fields is examined (see 2). An analysis of the problem fields reveals the tasks that have to be solved and discusses different approaches. That leads to a decision whether to use a tracking or a reconstruction approach (see 3). Starting with the used tools the solution for the different task are explained in detail and the achieved result is presented (see 4). Finally a conclusion sums up the results of this thesis and shows possibilities for further work (see 5).

2 Related Work

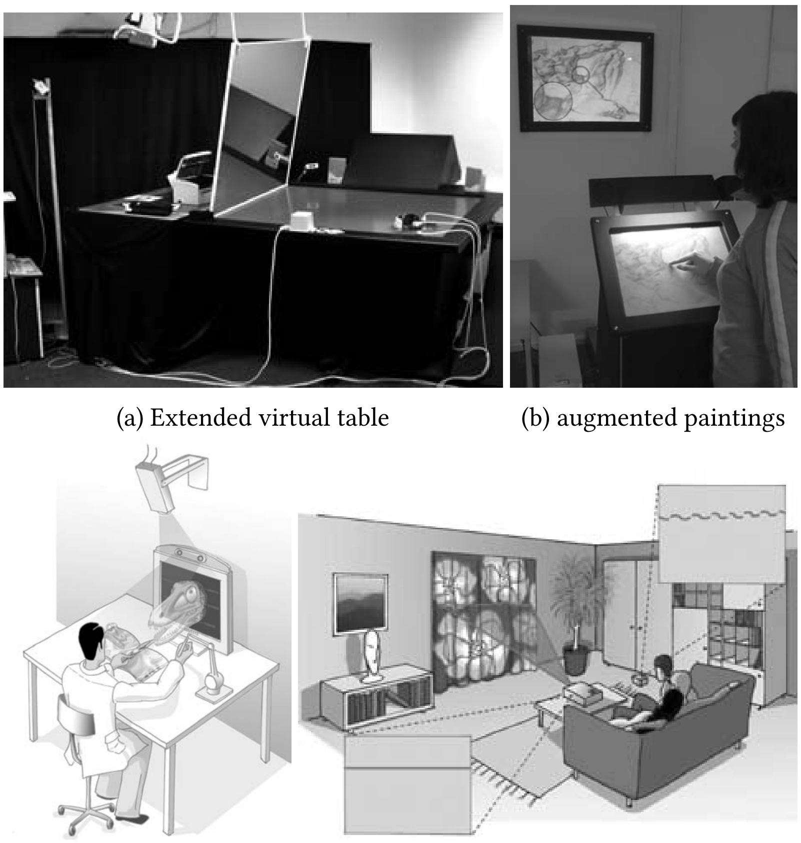
In this chapter existing solutions for mapping video onto moving objects are examined in (2.1). The operating principles of those solutions will be analysed and described. In (2.2) some existing technology is taken into consideration that could help finding a solution for the different problem fields.

2.1 Corresponding projects

The following subsections will give a rough overview of different approaches to projection mapping introducing projects which pursue a similar goal with different key aspects and increasing demands. With (2.1.1) the technical origin of projection mapping will be explained. With (2.1.2) a realization of dynamic 2D mapping will be examined. In (2.1.3) dynamic 3D mapping is realized using a marker-based tracking approach while in (2.1.4) it is achieved with a system that moves virtual and physical models synchronously. A polygonal reconstruction method to achieve dynamic 3D projection mapping is presented in (2.1.5). In (2.1.6) a real-time reconstruction method that produces a dense 3D environment model is explained.

2.1.1 Spatial Augmented Reality

The fields of virtual reality and augmented reality usually take place in a virtual environment using for example head-mounted displays or mobile device screens to overlay virtual textures on camera images. Spatial augmented reality (SAR) makes use of novel display technology like projectors or holographic displays to transport this virtual overlay back into the real world (see Bimber et al. [6, chapter 3,6,7]). Several examples for spatial augmented reality displays have been realized (see [6, chapter 8]). Figure 2.1 depicts four significantly different display variants. The *extended Virtual Table* (2.1a) as well as the *Holostation* (2.1c) use half-reflective mirrors to visually combine virtual and physical worlds. Projector-based illumination is used to realize *augmented paintings* (2.1b). Using scanning techniques, *Smart projectors* (2.1d) can use arbitrary projection surfaces. Traditional static projection mapping has its technical roots



(a) Extended virtual table (b) augmented paintings

(c) Holostation (d) Smart projector

Figure 2.1: Examples for spatial AR displays (Source: [6, chapter 8])

in the field of spatial augmented reality using projector-based illumination and augmentation. Tasks like projector-camera calibration and scene construction as well as all the considerations concerning the projector like pixel-resolution, luminosity and black-level also apply to projection mapping.

2.1.2 2D Mapping onto people

One early realization of the idea to project onto moving objects was presented by (Obermaier [26]) in his interactive dance and media performance *Apparition* at ARS Electronica 2004. In this performance the silhouettes of dancers on stage where extracted using a camera based motion tracking system, and are used for masking textures that are projected onto the dancers using a calibrated projector-camera system. In figure 2.2 half of the dancers silhouette is augmented with a striped texture. The limitation to the dancer is clearly visible. This work reflects an idea that is very attractive for textile designers, to augment clothes with virtual textures for a virtual enhanced fashion show.

Calibration The calibration problem is similar to the one this thesis has to solve. Since this method uses 2D camera images for silhouette extraction and also overlays 2D Textures, it has to perform a transformation from camera to projector space. Further it is likely that more than one camera and projector was used in this project. Therefore all cameras have to be calibrated to a 3D reference coordinate system and for every projector the corresponding transformation from 3D to 2D has to be estimated (see Hartley et al. [18]).

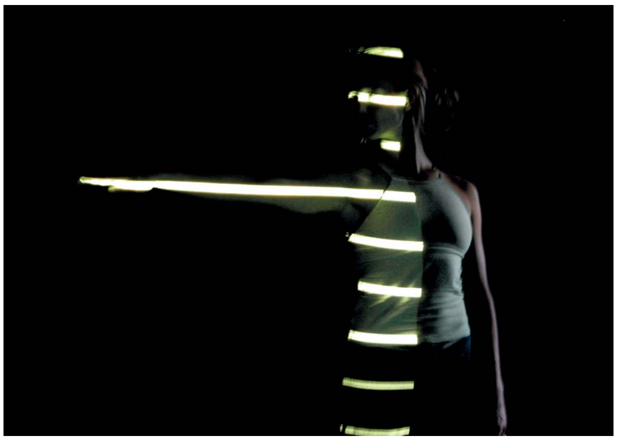


Figure 2.2: 2D projection mapping onto a dancers silhouette (Source: [26])

Dynamic model creation / Tracking of known object: The model in this method is the 2D silhouette of the dancers and is generated in real-time by some sort of background subtraction. The silhouette is then used to separate background textures from textures to be projected on the dancers. This is commonly done with binary masking which is a well known technique from the field of computer graphics.

Virtual scene creation After transforming the captured silhouette-mask from camera to projector space it can be applied as binary mask to any 2D texture to restrict it to the real objects, which are the people on stage in this scenario.

2.1.3 Marker based tracking of known projection surfaces

More for the purpose of creating an augmented stage design, (Yapo et al. [48]) used infra-red LEDs as markers to track moving scenery walls with defined known shapes on stage (figure 2.3). In this work vertical flat walls are equipped with infra-red LEDs on top so that each wall forms a unique pattern of its LEDs. A top view camera is tracking those LEDs as points in

2D. By matching angular and distance features of the LED points against the known patterns, the position and orientation of the screens is obtained and the scene to be projected can be constructed. Related marker-based tracking methods like AR-Toolkit (see Kato and Billinghurst [21]) use visible tracking markers.

Calibration The tracked 2D points acquired from the camera have to be transformed to a virtual 3D space where the models of the walls can be positioned. For every projector a 2D view is rendered with the projectors parameters. Therefore a full projector-camera calibration has to be performed.

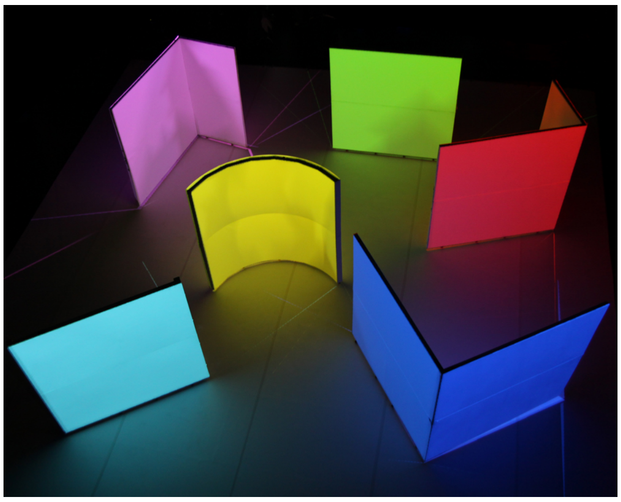


Figure 2.3: Tracked screens on stage lit by calibrated projectors (Source: [48])

Dynamic model creation / Object Tracking: This method works with 3D models of the walls which can translate and rotate on stage. On top of the walls infra-red LEDs are placed in a unique pattern that allows a robust identification against other walls patterns. A camera equipped with visible light filter is observing the scene from above, providing an image which reveals the positions of the LEDs. By matching the led patterns against the known wall shapes, the position and orientation of each screen can be determined.

Virtual scene creation With known wall shapes and known poses it is trivial to construct a virtual 3D scene that can be rendered from the viewpoint of one or more projectors.

2.1.4 Mapping on servo actuated Objects

Another solution for mapping onto moving objects takes advantage of software controlled movements. This technique involves stepper motors to synchronize the movement of a 3D object with a virtual 3D scene. The object is therefore mounted onto a servo actuated table or on a turning axis like in figure 2.4 where a polystyrene object is mounted on a vertical axis. In the grey box on the floor the servo motor and the control circuit are located. The movement of the object is then controllable by software defined time-variable functions or by an input device like a tablet computer running a customized application. This technique is used for example by (White Kanga [45]) a polish interactive media artist and by the (panGenerator collective [31]) which is related to the MediaLab in Chrzelice in Poland.

Calibration The calibration problem is reduced to find the projectors intrinsics and extrinsics with respect to the object, as there is no camera tracking involved. This can be done once before run-time. White Kanga implemented a stripe boundary code calibration for which a camera is needed prior to the performance.

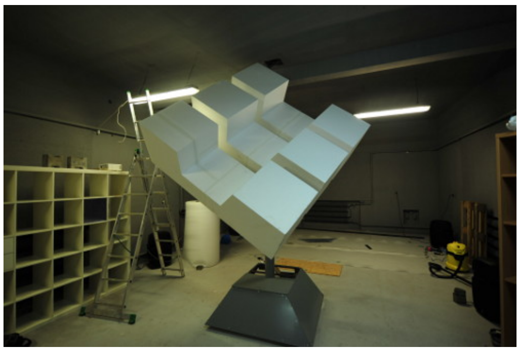


Figure 2.4: Physical object mounted on a servo-driven axis (Source: http://whitekanga.pl/pl/technologia)

Dynamic model creation / Object Tracking: The initial alignment between virtual 3D space and real-world is established initially through the calibration process. A software that controls the models movement maintains the synchronicity between virtual and real model, provided that an exact steering of the actuators can be guaranteed.

Virtual scene creation To create the virtual scene a virtual model, that is true to scale to the real model can be textured using common 3D rendering techniques. By synchronously rotating or translating the virtual scene according to the stepper motors movement, the perspective is updated constantly.

2.1.5 Mapinect

Mapinect ([33]) is a research project of students from the Universidad de la Republica in Uruguay, which aim to develop a simple robotic arm carrying a pocket projector and a Kinect sensor, that will be able to detect surrounding cuboids and project virtual textures onto them. Mapinect uses a pure visual method to locate cuboids for mapping textures onto them. They detect change in the scene and reconstruct the cuboids using planar polygons extracted with

RANSAC based plane extraction methods implemented in the PCL Library. Existing virtual cuboids are compared to the sensor data in each frame and locally corrected using an iterative closest point algorithm (see 4.2.4). If the measurement differs too much from the last position, a new virtual cuboid is created. With the calibrated projector-camera system and the poses of the detected cuboids a virtual scene is rendered that is projected onto the real world. Figure 2.5 shows an example of a scene augmented using Mapinect. Small white boxes on a table are augmented with single color textures carrying letters. The table is augmented with a grid pattern. The boxes can be moved manually and the projected scene is adjusted according to the procedure above.
The limitation of this method is connected with the objects geometrical complexity. Curved surfaces for example would seriously slow down the search for planar surfaces in the scene, because they need to be approximated by many planar surfaces.

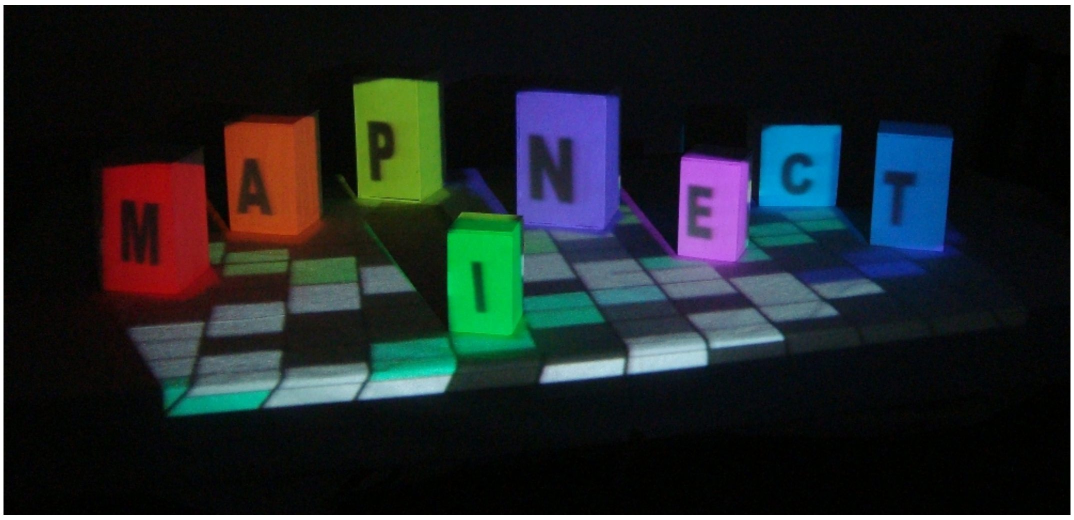


Figure 2.5: Mapinect System litting small boxes on a table (Source: `http://www.fing.edu.uy/grupos/medialab/projects/mapinect/images/mapinect6.jpg`, 9.1.2013)

Calibration To correct the viewpoint for the projector output, only extrinsics and intrinsics of the projector have to be obtained since the Kinect comes with a factory calibration that allows to acquire points in 3D camera view coordinates. This also applies to this thesis as it is intended to use this sensor.

Dynamic model creation / Object Tracking: The 3D model is dynamically built using 3D planar polygons that are extracted using a RANSAC based extraction algorithm. The algorithm iteratively extracts the biggest planar surface candidate from the remaining point cloud (see 4.2.4). From the detected planar polygons they construct 3D models and observe their 3D centroid for change. If the change is within a certain threshold the object gets transformed and redrawn due to the underlying tracking algorithm (ICP, see 4.2.4). Otherwise it is supposed they have disappeared from the scene and a new virtual object is constructed.

Virtual scene creation Using the projectors extrinsics and intrinsics and the 3D polygonal model, the projectors view can be rendered with for example OpenGL. It is much easier to create virtual textured objects from the geometrically relatively simple objects in real-time than from more complex, arbitrary shaped objects.

2.1.6 KinectFusion

(Izadi et al. [19]) present a system to recreate large scenes in real time by using a Microsoft Kinect Sensor. The image and depth data for each frame is successive fused by registering the nth point cloud with the $n+1$th and applying the image data as texture. Using the depth-map provided by the Kinect sensor they compute vertex data for every frame, which is then aligned to the previous frame using iterative closest point (see 4.2.4). The aligned data then is integrated into a volumetric truncated signed distance function model (see Curless and Levoy [12] for details). With a GPU-oriented implementation they achieve interactive real-time reconstruction rates for building a volumetric model of a real scene. Although there is no re-projection into the real world this project is interesting for this thesis with regard to the real-time reconstruction approach.

2.2 Technology to consider

While section 2.1 dealt with projects whose objectives are similar to this thesis, this section discusses technical solutions for the subtasks projector-camera calibration, 3D reconstruction and object tracking that will contribute to the overall solution. Here some techniques are examined if they are applicable to this thesis. This examination is structured by the the problem fields stated in the motivation.

2.2.1 Calibration

A very popular method for projector-camera calibration is the method introduced by (Zhang [50]) which finds a homography between known points of a chessboard pattern of defined dimensions and the corresponding detected points on the camera image. On this basis (Jones

[20, chapter 4.3]) explains projector calibration. (Burrus [9]) revisited this technique to build a calibration work flow for a camera - projector system with a Kinect camera. (Lee [23]) proposes a method to locate projection surfaces by projecting gray-codes onto a surface which are picked up by small light sensors that are fixed to the surfaces. The light sensors measure if the projected pixel is white or black so that the position of those passive markers in projector space can then be calculated by the binary code resulting from the picked up patterns.
Many different methods prove that projector-camera calibration is a well understood field, that provides existing, applicable solutions.

2.2.2 Reconstruction

The reconstruction subtask will be examined for both variants stated in the introduction. Online reconstruction, that is creating a textured 3D model of the moving and deforming object in real-time, would be the preferred method but is believed to be harder to achieve. The alternative, an offline reconstructing method of the 3D model combined with online rigid object tracking, would restrain the objects degrees of freedom though it would certainly reduce the computational costs in the online phase.

Offline modelling of 3D objects (Xu and Aliaga [46]) describe a photometric method to model non-rigid surfaces of camera monitored objects from a stream of camera images. It produced very high detailed 3D models but cannot run in real-time which is required for this thesis purpose. An interesting algorithm which uses a method called landmarking to refine 3D models is presented by (Tevs et al. [43]). This method can fill holes in dynamic meshes and reconstruct temporary lost points. The paper also makes clear that real-time reconstruction of shape and motion is still hard to achieve.
(Bi and Wang [5]) give a good overview of offline 3d data acquisition methods for industrial purposes. They distinguish passive methods like shape-from-X or passive stereo vision from active systems like time-of-flight or triangulation based systems. Further an overview of different 3D sensor data processing methods for data filtering, registration of data from multiple frames and 3D reconstruction, simplification and segmentation is given. According to this work active scanner hardware has become quite powerful for real-time applications but the real-time capability on the software processing side is still unsatisfactory.
A fast model acquisition method is presented by (Pan et al. [30]), which acquires a 3D model of textured objects by a video stream in about 1 minute. Using 2D features, a camera pose relative to the object is calculated in real-time. When the cameras rotation around the object (or the objects rotation in front of the camera) is large enough, model landmarks are calculated

from image features that are then used to cumulatively compute 3D points of the model. With a modified approach of space carving they finally compute the surface representation of the object. Though this methods is really fast with 1 minute of reconstruction time, it is still too slow for the real-time reconstruction approach in this thesis.

Online modelling of 3D objects (Zhang [49]) presents recent advances on 3D shape acquisition by fringe image projection. This method, generally known as structured-light scanning, projects a structured pattern onto the object. The object, that distorts the pattern through its geometry, is observed by a camera which is picking up the distorted pattern. From the camera image and the projected pattern, the 3D shape of the object can be recovered. With recent advances, this method is capable of real-time reconstruction at 30 frames per second. Although this method is fast enough to be a candidate for the real-time reconstruction approach, it relies on the projection of a pattern onto the object which would pre-empt the projection of textures onto the object.

2.2.3 Tracking

Object detection and pose estimation in 2D (Glasner et al. [16], Mei et al. [25]) perform detection and pose estimation for cars by trained viewpoint appearances from 2D image streams. A drawback is that a lot of different poses have to be trained in advance to ensure the reliability of this method.
(Godec et al. [17]) present a new method for tracking deformable objects from 2D image streams based on the generalized Hough-transformation. This method seems to be very reliable for extracting the moving and deforming outlines of an object, but does not reveal 3D position and orientation information.

Model-based object tracking in 3D (Shaheen et al. [41]) compares 3D model-based tracking methods for the purpose of tracking a skeletal model of a human. The base principles of most of the available algorithms can be categorized into bayesian filters, local and global optimizations which can be further differentiated into single hypothesis and multiple hypothesis optimizations. Those base principles still offer a wide range of implementation details (see Poppe [34]). Lots of different approaches using bayesian filters have been made. There are kalman filter approaches (see f.e. Yang and Welch [47]) which are only suitable for linear motion ([34]) and particle filter approaches (see f.e. Azad et al. [2], Brown and Capson

[8], Choi and Christensen [10]) which can track non-linear motion but are computational more expensive.

2.3 Discussion

The technical basics for projections onto 3D objects have been well researched in the field of spatial augmented reality (see 2.1.1) and have led to projection mapping as art form. The relation of projection mapping to this field offers numerous research topics that have the potential to help solving projection mapping problems.

Video projection mapping has been done in 2D for some time with dynamic model reconstruction by calculating the mask image for every video frame (see 2.1.2) or by marker based 2D image tracking methods that allow to reconstruct the virtual scene in 3D (see 2.1.3). While marker-based tracking methods proved to be real-time capable and reliable, it is yet unwanted for the purpose of video projection mapping to put markers onto the object.

Also the basic 3D video projection mapping technique has evolved to a semi-dynamic form by controlling the movement of the object with mechanical actuators (see 2.1.4), which are synchronized with the movement of the virtual model. While this method provides seamless alignment and thus a truly striking immersive effect, the movement of the object is very restricted and determined. The degree of freedom could be increased by using multiple actuators but that would not overcome the fundamental restrictions.

With Mapinect, a project was introduced, that uses a markerless visual tracking method to reconstruct and video map 3D objects with the restriction to objects with low polygon counts (see 2.1.5). The iterative approach of extracting the respective biggest planar surface from the remaining point cloud frame will certainly be too expensive for more detailed models. Nevertheless a closer look into the capabilities of the used library reveals promising features that will allow different approaches.

Kinect Fusion (see 2.1.6) shows that real-time 3D reconstruction using low cost depth sensors is feasible. Though the methods shows interactive rates in integrating successive frames to a full 3D model, the purpose of this project is more to reconstruct room scenes than to augment a moving and deforming object.

All problem fields stated in the introduction are still research topics in the context of computer vision, robotics and augmented reality. While working solutions for a projector-camera calibration with a Kinect sensor are already available (see 2.2.1), it turns out that the reconstruction problem is still the harder part of the work. Although offline reconstruction methods are available and well known, online reconstruction solutions are subject to current research. The only reconstruction methods that meets the performance requirements is based on projected patterns which is not applicable to this thesis (see 2.2.2). Object tracking based on 2D image streams is intensively researched but has a low chance of providing good 3D poses (see 2.2.3).

3 Analysis

As stated previously, a very attractive idea for textile designers is to create a virtual augmented fashion show, where clothes are dynamically textured with video content or generative graphics. Also for artistic purposes like augmented sculptural installations or dance performances video mapping is a promising field that is more and more explored using techniques originating from computer science.

The objective of this thesis is to create a solution for mapping a 3D video projection onto a defined object that freely moves in a defined area. The object may have an almost arbitrary geometrical complexity.

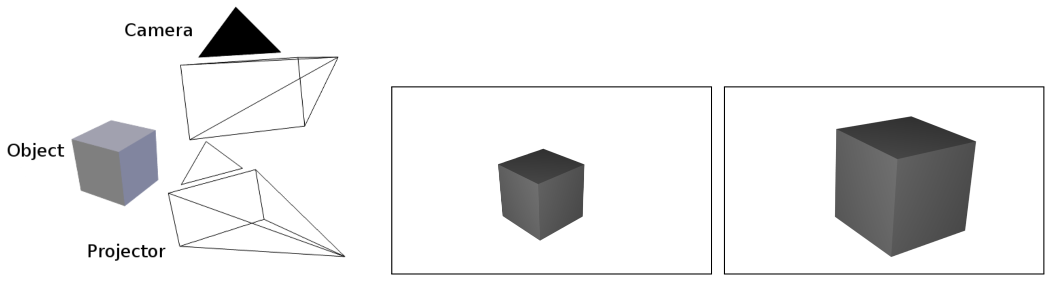
In the remaining of this chapter requirements for a solution of the different problem fields will be analysed in detail. The choice of the sensor to be used will be discussed (3.1.1), as well as the according implications for the projector-camera calibration (3.1.2). As both approaches, real-time reconstruction and tracking of a 3D object can be assumed to be the computational most expensive part, here the analysis will evaluate both online and offline approaches. This results in an assignment of expensive sub tasks to an offline computing phase. The goal is to offload computation from the online phase as less as possible, while maintaining interactive frame rates in the online phase (3.1.3). Finally the construction of the projectors output is discussed (3.1.4).

3.1 Subproblem Analysis

3.1.1 3D sensor

Aiming at projecting textures onto objects, visible light based sensors will not be very suitable for this thesis because it aims to project potentially changing textures onto the object. Sensors that operate with infra-red light will produce more consistent data under the given circumstances. Therefore the Microsoft Kinect was chosen as 3D sensor due to cost considerations and personal experience with the sensor and its related OpenNI drivers.

3.1.2 Projector camera calibration



(a) Overview of the conceptual setup (b) Camera view (c) Projector "view"

Figure 3.1: Projector-camera setup

Seeing the physical scene in 3D, the camera and the projector are observing the object from two different view points and will therefore "see" different perspectives of the object. Figure 3.1 gives an overview of the calibration problem. The resulting viewpoint images show the different appearances for camera and projector. For a correct re-projection the camera image (3.1b) has to be distorted to get the projector "view" (3.1c). In 3D this can be done by translating and rotating the 3D scene so that it is viewed at from the projectors perspective and render it with a virtual camera setting, that resembles the projectors perspective projection. Therefore the intrinsic parameters of the projector, describing the way light gets projected, and the extrinsic parameters with respect to the camera, describing the location and orientation in relation to the camera, have to be obtained. This calibration process is intended to be part of the setup, and is to be performed after projector and camera have found their fixed place. As there are established calibration solutions for projector-camera systems with a Kinect (see 2.2.1) it is not intended to develop a new solution.

3.1.3 Dynamic model creation / Object tracking

Real-time 3D scene reconstruction is a task that has not really been solved (see Bi and Wang [5]). Methods that use structured-light (see 2.2.2), reconstruct static scenes (see 2.1.6) or have restrictions with respect to the geometrical complexity of objects (see 2.1.5) are not really applicable to this thesis. Since real-time reconstruction has to recover shape and pose for each frame only a few tasks are candidates for offline execution. The alternative approach of tracking a known model of the object in a scene seems to be more feasible (see 2.2.3), but implies that the object must have a rigid shape that does not change over time in the online phase. Provided, that the object has a rigid non-trivial shape, allows to prepare the model in an offline phase. Thus the problem turns into an offline model reconstruction combined with an online object tracking problem for virtual scene construction.

Model Acquisition

The principal behind 3D scene reconstruction is to create a virtual 3D model of real objects by using optical measurement rather than modelling it manually. The method of combining point clouds presented in (2.1.6) is known as point registration (see Bernardini and Rushmeier [3]) and can produce models of complex objects by registering several partial, overlapping views of the object to acquire a full 360° model. To obtain partial views of the object out of the bulk of points captured with the 3D sensor, all points that do not belong to the object have to be disposed. Therefore a suiting separation method has to be found. Offline model acquisition can take place in a staged environment, which can help separating the objects points from the environments points. The object can be placed in front of a black background so that the color can be used to separate the object. Another method would be to use an environment that is staged in a geometrical manner to separate the object from the background.

Object Tracking

To recreate a virtual counterpart of the physical scene, the position and orientation of the object has to be obtained in every frame. (Lepetit and Fua [24]) discuss several different tracking methods in 2D and 3D. As they state there are almost as many tracking methods as possible applications. Popular examples are marker-based tracking as in (2.1.2) and (2.3) and the so called bayesian tracking methods which are divided into Kalman (see [24, chapter 2.6.1]) and Particle filters (see [24, chapter 2.6.2]). As stated above marker-based tracking does not suit very well for video projection mapping as it is undesirable to put markers onto the object. Bayesian tracking methods estimate a probability density of successive states in the state space of possible object poses and mostly choose the hypothesis with the maximum-likelihood as the resulting object pose. As stated by (Poppe [34]) Kalman filters can only track linear movement, while particle filters are able to track arbitrary movement.

3.1.4 Virtual Scene Construction

While for tracking purposes a dense point representation of the model is well suited, for visualization purposes a mesh representation is needed to apply a texture to the 3D model. There are several algorithms that calculate a mesh representation from a set of 3D points like

Poisson reconstruction (see Kazhdan et al. [22])or Moving least-squares (see Cuccuru et al. [11]).
For rendering a textured 3D object, every mesh-point has to be provided with a texture coordinate. Those coordinates define from which part of a texture a triangle gets its texture. Most 3D modelling software provides some sort of unwrapping function for that purpose that generates an unwrapped appearance of the 3D object on a 2D texture and stores corresponding texture coordinates with every point. There is a lot of research around the topic of mesh generation, but that goes beyond the scope of this paper. For visualization purposes the point cloud representation of the model will be used in this thesis. To produce the projectors output image, common 3D rendering techniques such as OpenGL or DirectX can be used. With the prepared model, the projector-camera calibration parameters and the constantly obtained pose of the object, a 3D scene can be constructed that shows the textured object. The 3D scene is then rendered from the projectors viewpoint with a virtual camera that has to be configured with the intrinsic parameters of the projector.

3.2 Summary

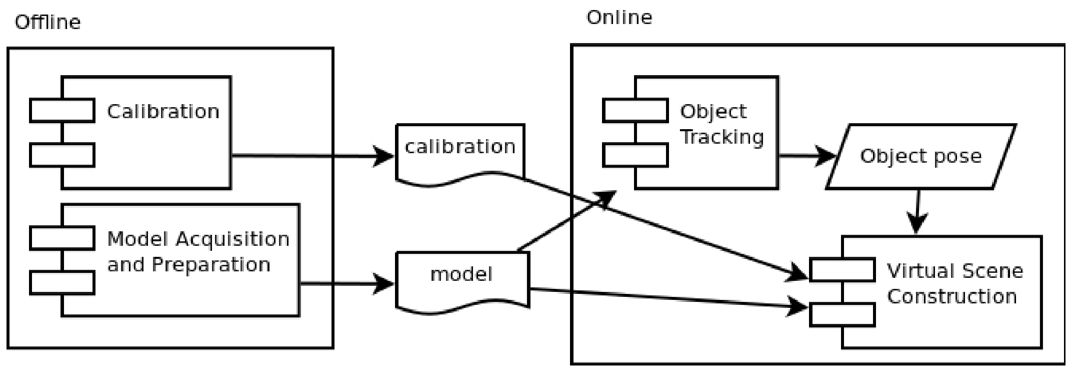


Figure 3.2: Overview of the working phases and their respective problem fields

Due to performance considerations the solutions for the partial problems will be grouped into an online phase and an offline phase. The calibration part is planned to be done with external tools and belongs to the offline phase of the application. Figure 3.2 gives an overview of the partial problems and their assignment to the respective phase. The calibration task and the virtual scene creation can now be assumed as solvable. While the calibration is performed with external tools, the creation of the virtual scene will be realized using basic 3D rendering software. What is left are the processing pipelines for model acquisition and object tracking which need concrete implementations of the above stated algorithms in order to be realized. Suitable libraries, that provide tools that can contribute to this solution, have to be selected and tested.

4 Realization

From the rough outline of the solution specified in chapter 3, this chapter breaks down the detailed solutions of the problem fields and composes the processing pipelines for the model acquisition task and the tracking task. From (2.2.1) it is known that a working calibration solution for a projector-camera system with a Kinect camera is available with *RGBDemo* (see Burrus [9]). In (2.1.5) a promising library candidate was discovered with *PCL* ([32]), that provides a comprehensive set of tools for 3D point data processing. A thorough examination reveals that it provides implementations of the algorithms needed for the model acquisition and tracking tasks. Figure 4.1 breaks down the four subtasks and specifies the incorporated third party software. In the offline phase the calibration is performed using the RGBDemo tools (see 4.3.1). With a set of filters and extraction methods partial model point clouds are obtained, which are then registered to a 360° model by using feature based alignment methods. Finally the resulting model gets refined by some post processing filters. For analytical visualization PCLs visualisation tool is used.

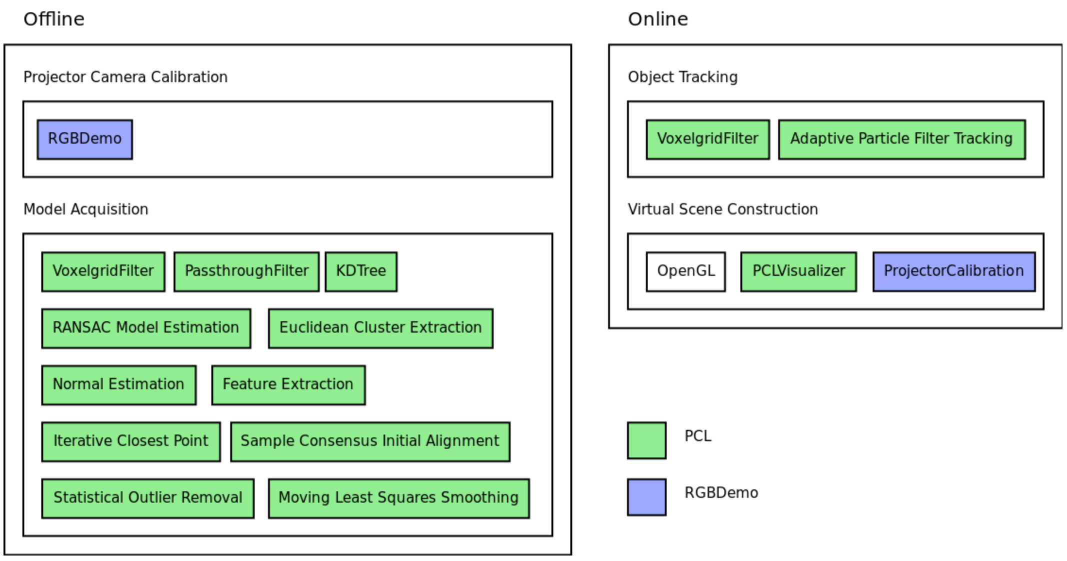


Figure 4.1: Overview of the architecture

The object tracking pipeline in the online phase pre-filters the model and the scene point cloud before the tracking algorithm is applied that produces the objects 3D pose. For the virtual scene construction an OpenGL window is set up with the projector calibration parameters that renders the model point cloud, that constantly gets transformed by the 3D pose.

In the following the assumptions for the solution are defined (see 4.1) before the used tools and libraries are examined in detail (see 4.2). Then the developed solution will be explained (see 4.3) and the results are discussed (see 4.4).

4.1 Assumptions

Projector and camera placement Projector and camera should be placed as close together as possible, so that they are sharing as much of their view-frustums as possible. In that case the calibration method is more likely to produce a small pixel re-projection error and the movement range of the object is maximized.

Object Placement for Model Acquisition For model acquisition the object is to be placed on a flat surface like a table, so that the segmentation of the model from the scene can be simplified.

4.2 Tools and Libraries

4.2.1 Point Cloud Capturing Sensor

The Microsoft Kinect sensor was chosen as point cloud capturing device. Because it is working on infra-red basis it is expected that the interference with light emitted from the projector would be minimal. The raw depth map obtained from the sensor contains $320 * 240 = 76800$ points and is delivered with a rate of 30 frames per second. A suitable precision of measurement is provided in a distance range of 0.5 to 2m from the sensor. Farther away the precision reduces significantly. In figure 4.3 the decreasing precision can be seen from left to right as increasing

Figure 4.2: Microsoft Kinect Camera (Source: http://assets-1.microsoftstoreassets.com/CatalogImages/media_content/Kinect/preview/Kinect_06.jpg, 20.09.2012)

distance between the lines. The depth information is coded into 13 bits which results into 8192 different depth values. These values steps can be seen in the figure below as lines of points. From every pixel in the depth map a 3D point is generated.

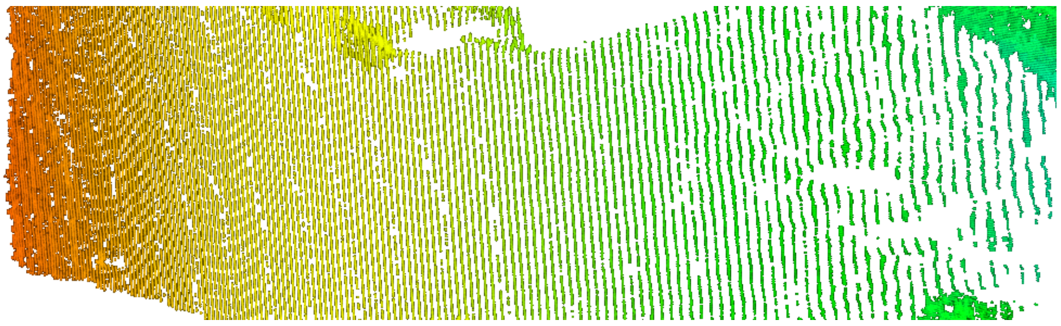


Figure 4.3: Typical planar Point Cloud captured by a Kinect sensor in top-down view. Depth is color coded from red (near) to green (far).

The (OpenNI [29]) driver framework is used as interface to the Kinect sensor. It is supported by the PCL library that gets introduced in section 4.2.3 and provides a hard-coded pre-calibration, implicating that the intrinsic parameters of the sensor are quite the same on all devices. As a result, it is possible to obtain calibrated 3D points from the sensors depth map.

Other point cloud generating device like laser scanners (LIDAR), time-of-flight-cameras or the generation of point clouds from stereo cameras should work with this method. The only constraint is that a frame rate of at least 30 FPS should be provided to leave enough time for processing.

4.2.2 Projector Camera Calibration

As described in (Hartley et al. [18, chapter 2]) a camera roughly follows the perspective projection or pinhole camera model. According to (Falcao et al. [13]) the key to projector-camera calibration is to see the projector as an 'inverse' camera, so that multiple view geometry (Hartley et al. [18]) provides the solution for the calibration problem. The projector is therefore also defined with intrinsic and extrinsic parameters.

Considering the Kinect sensor (4.2.1), the captured point cloud data is represented in calibrated 3D camera space. The coordinate spaces origin is the principal point of the camera with the z-axis facing away from the sensor in view direction. For point cloud processing this coordinate space is suitable. Only for visualization the 3D scene has to be projected to 2D projector space. The projection to 2D is done by the graphics hardware in the rendering pipeline while the respective un-projection is done physically by the projector. Therefore the application of the calibration parameters is reduced to a translation and rotation of the point cloud data, to transform to the viewpoint of the projector, and a calibration of the virtual scene camera according to the projectors intrinsic parameters (e.g. focal-length).

4.2.3 Libraries

RGBDemo

RGBdemo is a simple toolkit to start playing with Kinect data which was initially developed by (Burrus [9]). It provides the library nestk and some demo applications. As driver backends both freenect as well as OpenNI are supported. For this thesis the most interesting feature is the integration with OpenCV and in particular the integrated solution for projector camera calibration. The applications *rgbd-viewer* and *calibrate-projector* are used to obtain the calibration parameters explained in (4.3.1).

Point Cloud Library

The Point Cloud Library ([32]) was presented at the ICRA 2011 (see Rusu and Cousins [37]). The development started in march 2010 at Willow Garage with the goal to provide support for processing of 3D point cloud data with main focus on robotic applications. PCL is a comprehensive C++ library for 2D/3D image and point cloud data processing. It contains modules for filtering, search, feature estimation, surface reconstruction, registration, model fitting, segmentation and visualization of point cloud data. In figure 4.5 a complete overview of PCL's modules and their interdependencies is shown. The C++ library is templated in terms of point representation and defines several types of

Figure 4.4: PCL Logo (Source: http://www.pointclouds.org/downloads/ 9.1.2013)

points.

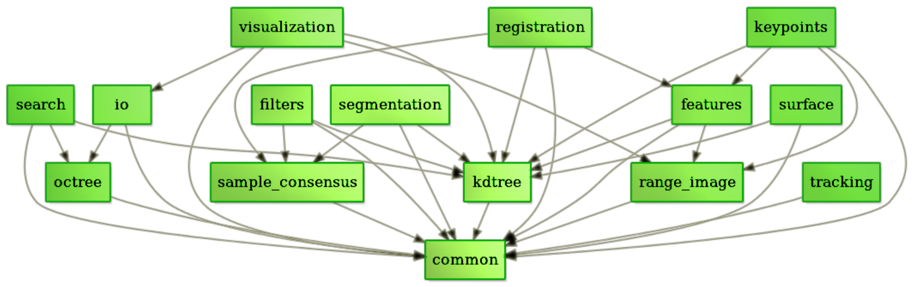


Figure 4.5: PCL module dependencies (Source: http://www.pointclouds.org/about/, 9.1.2013)

PCL is used throughout this thesis for numerous purposes starting with the *PointCloud* data types used with different point representations. The point cloud data is obtained with help of the io module and gets processed using the filters and segmentation module. Partial captured model views are then joined using the registration module and post-processed using the surface module. In the runtime part of this solution the tracking module is used to obtain the location and orientation of the model. To visualize point clouds at any stage the visualization module of PCL provides suitable tools.

In the following the data types and algorithms defined by PCL will be explained further since the principles of operation are essential for understanding the overall solution.

4.2.4 Data structures and Algorithms

Point

The most basic Point that is defined in PCL is *PointXYZ* which represents a 3D Position. Throughout this thesis the *PointXYZRGBA* which contains color information in red, blue, green and alpha components is used as default point type. For feature comparison the point type *FPFHSignature33* defines the 33-bin *Fast Point Feature Histogram*. As particle representation in the tracking algorithm the point type *PointXYZRPY* is used.

Point Cloud

The *PointCloud* data type defines collections of points and is templated by point types. Pointclouds contain a set of points and cloud structure information. *Projectable* point clouds also contain information about sensor position and orientation according to the pinhole camera model. A point cloud can be *organized* which means it can be split into rows and columns like an image. In that case width and height of the point cloud resemble the row and column count. If the point cloud is unorganized, the height is 1 and the width equals the total number of points. Organized point clouds allow for example much faster neighbor operations than unorganized ones. A point cloud is defined as *dense* when it does not contain any infinity or NaN data in the points fields.

K-D Tree

In contrast to the vector-like structure of a point cloud, a k-d tree (k-dimensional tree) is a tree data structure that divides space in a hierarchical structure. Each Node represents a point in space and a split of the room it belongs to. Here the split is always oriented along one of the dimensional axis and all points with a value bigger as the splitting points value of this dimension gets sorted to the right side of the sub-tree and all points with smaller values to the left side. The splitting axis switches with each tree layer. In figure 4.6 the red lines indicate the first split, the green lines the two second splits and the blue lines the third splits. This data structure is most effective for spatial search operation around points such as neighborhood search algorithms and is used in the surface normal estimation.

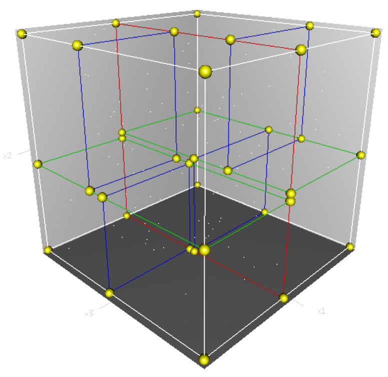


Figure 4.6: KD Tree space partitioning with $K = 3$ source by: http://commons.wikimedia.org/wiki/File:3dtree.png, 9.1.2013

Passthrough Filter

A pass-through filter restricts the viewing volume to a fixed size. Constraints for each axis X, Y and Z can be set, and points that are outside these constraints are removed from the cloud. Effectively the origin space is cut to the given minimum and maximum dimensions in the given directions. This filter is very useful to reduce points by shrinking the viewing volume. It reduces the amount of point in a trivial way and increases the processing speed of following operations on the remaining cloud. The computing time of this filter is nearly constant. In figure 4.7 a pass-through filter is applied to the Z axis with the constraints 0.5 to 1.5 meters. The pass-through filter is used for model acquisition to focus on the volume of interest so that the planar surface on which the object is placed is guaranteed to be the biggest planar surface in the remaining scene.

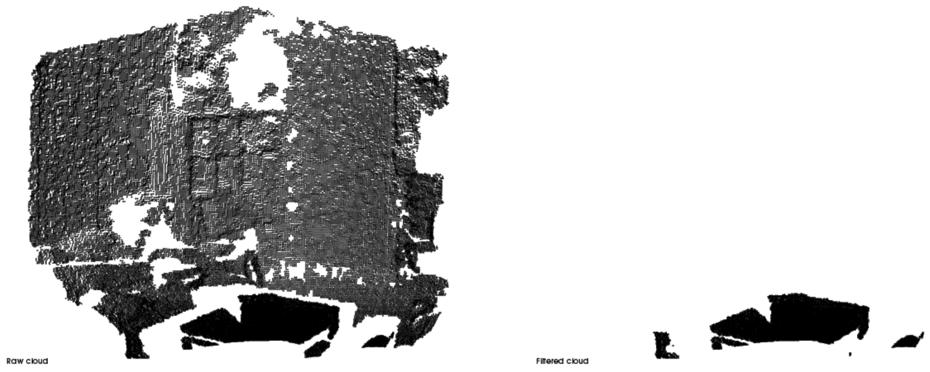


Figure 4.7: Scene before *(left)* and after pass-through filtering *(right)*

Voxelgrid Filter

In a voxelgrid filter the space is divided into voxels of a fixed size. Voxels are tiny 3D boxes in space. Points in each voxel are then approximated to the centroid off all voxel points, so that for each voxel one averaged point is remaining. This approach gives a more accurate result than approximate each voxel by its center. Effectively the whole point cloud gets downsampled to a smaller resolution. The voxelgrid filter is used to down-sample scene and model point cloud to the same resolution for feature calculation. In figure 4.8 a voxelgrid filter with a voxel size of 1 cm is applied to the whole scene.

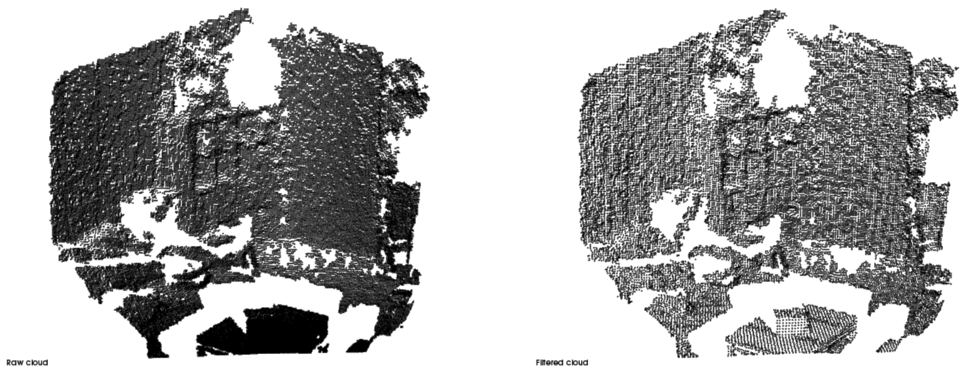


Figure 4.8: Scene before *(left)* and after voxel-grid filtering *(right)*

RANSAC Plane Model Estimation

RANSAC is an iterative method to find values that support a given hypothesis in a noisy set of values. Therefore a random set of values is picked as hypothetical inliers and tested against the hypothesis, producing a qualification score of the model that is defined by the hypothetical inliers. This is repeated for a fixed number of times and the set with the best qualification score is chosen as best fitting for the hypothesis.
In PCL several basic 3D shape models like lines, planes, spheres or cylinders are supported to be searched with different sample consensus estimators like random sample consensus, least median of squares or other more complicated estimators. (Rusu [36, chapter 6.1]) describes the implementation details for PCL.
In this thesis RANSAC is used to effectively separate the object from its surroundings. The trick is to place the object on a flat surface for model acquisition which then can be removed by finding the inliers of the planar surfaces with this method. Therefore the scene has to be pre-filtered with a pass-through filter to make sure that the surface, the model is placed on is the biggest plane in the scene.

Euclidean Cluster Extraction

With this algorithm a point cloud is clustered based on a maximum euclidean point distance. In a tree representation of the point cloud which provide the fastest search method for spatial searches a random leaf is chosen and from there all leafs within a given search radius in 3D space are added to a cluster. If the cluster does not meet the given minimal and maximal point number constraints, it is discarded. This is repeated for every found leaf until the cluster cannot be expanded further and a new cluster is started from another random leaf. The algorithms returns all clusters as point index vectors that can be used to extract the points from the input point cloud. (Rusu [36, chapter 6.2]) describes this method as one of the basic clustering techniques. Here the euclidean cluster extraction is used for model acquisition to extract the model cluster out of the point cloud that is left after plane removal with RANSAC.

Normal Estimation

The surface normal estimation method proposed by the PCL Library is based on a least-square plane fitting estimation as described by (Rusu [36, chapter 4.3]). An analysis of the eigenvectors and eigenvalues of a co-variance matrix built from the k nearest neighbors of the point reveals the normal vector. Surface normals are required for example to compute point feature histogram descriptors.

Point Feature Histograms

For the registration procedure„ corresponding points in two point clouds have to be found. Therefore a robust feature descriptor is needed. Point Feature Histograms are 3D feature descriptors described by (Rusu [36, chapter 4.4]), that specify the neighborhood of a query point P_q. First every k points P_k in a radius r are selected. For every pair of the selected points and their respective surface normals a set of angular features is computed and binned into a histogram. Figure 4.9 depicts the influence radius r of the descriptors as stippled line. The query point P_q and its selected neighbour points P_{k1} to P_{k5} as graph vertices and all pairs of them as graph edges. This feature descriptor is consistent to the six degrees of freedom and is said to be relatively insensitive to noise. The general computation complexity for n points of $\mathcal{O}(nk^2)$ could be reduced to $\mathcal{O}(nk)$ by the Fast Point Feature Histogram Algorithm (described by Rusu et al. [39]), that is not fully interconnecting all pairs so that real-time feature acquisition is possible. The feature estimation is also available in a parallelized form using the OpenMP standard (see [39, 40]).

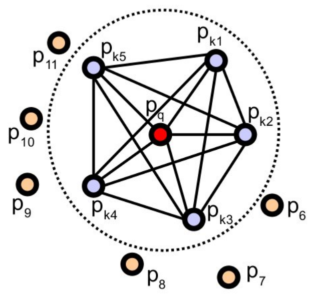


Figure 4.9: Influence region of the point feature histogram descriptor (Source: [39])

The Fast Point Feature Histogram descriptor (*FPFHSignature33*) is used for point cloud registration in the sample consensus initial alignment as well as in the ICP algorithm.

Sample Consensus Initial Alignment

The sample consensus initial alignment algorithm finds a rough global alignment of two partially overlapping point clouds (see Rusu et al. [39, Section IV]). It randomly selects point correspondences from the two point clouds and reduces their euclidean distances by finding a rigid transformation that reduces this distance.
This algorithm is used in this thesis to pre-align two point clouds before refining that alignment using ICP.

Iterative Closest Point

Iterative closest point *(ICP)* is used to match two point clouds that represent a scene from different view points, so that they can be concatenated after alignment. The algorithm incrementally reduces the euclidean distance between closest points of two point clouds. Based on the work of (Besl and McKay [4]) PCL provides a modified version of the ICP algorithm for point cloud registering which is described by (Rusu [36, chapter 5.1]). The algorithm has the drawback that it only finds the closest local minimum. With an initial alignment which gets close to a global minimum distance, it can be used to fine tune the alignment between two clouds.

Statistical Outlier Removal

Scanned point cloud data often contains sparse outliers due to noise or other measurement errors. To remove those outliers from the point cloud, a statistical analysis is performed around each points neighbourhood to compute the global mean distance to a neighbour point and the standard deviation. All points outside an interval defined by those values are considered to be outliers and removed from the point cloud (see Rusu et al. [38]). In this thesis statistical outlier removal is used on the registered 360° model.

Moving least squares smoothing

Moving least squares is method to reconstruct a continuous function from a set of points. The algorithm computes surface manifolds from the given points and re-samples the acquired points to produce a smoothed surface (for details, see Alexa et al. [1]). This algorithm is used to further refine the registered 360° model.

Adaptive Particle Filter Tracking

Based on (Fox [14, 15]) the PCL library provides an implementation for particle filtering with adaptive particle count and parallel computation of probabilty scores. A particle is defined in this method as a pose with a position vector and roll, pitch and yaw information (*PointXYZRPY*). The algorithm estimates pose candidates of the model in the observed scene, and weights the candidates with a probability score. The probability score is calculated on different coherence factors between model points and scene points. In PCL those coherence factor can be based on point colors, point normals or point distances. The candidate with the highest score is supposed to be the best matching candidate.

According to (Lepetit and Fua [24]), a big drawback of this method is that an accumulation of estimation errors can occur because this method bases its estimation on a sequence of captured frames and so estimation errors in previous frames can sum up. Being the only 3D tracking method implemented in PCL this however is the method of choice to obtain object position and orientation in this thesis.

4.3 Solution

There are four applications involved in the solution which are shown figure 4.10. The applications in the blue box, *rgbd-viewer* and *calibrate-projector* are provided by (Burrus [9]) and generate the calibration file (see 4.3.1). The *dynmap* application includes a point cloud processing pipeline for model acquisition and one for object tracking. The model acquisition pipeline produces point clouds of the separated partial model (see 4.3.2). The tracking pipeline delivers the most likely pose of the tracked object (see 4.3.3). The *pairwise-registration* application is derived from a PCL demo application of the same name and is used to successively register all partial model point clouds to a 360° model (see 4.3.2).

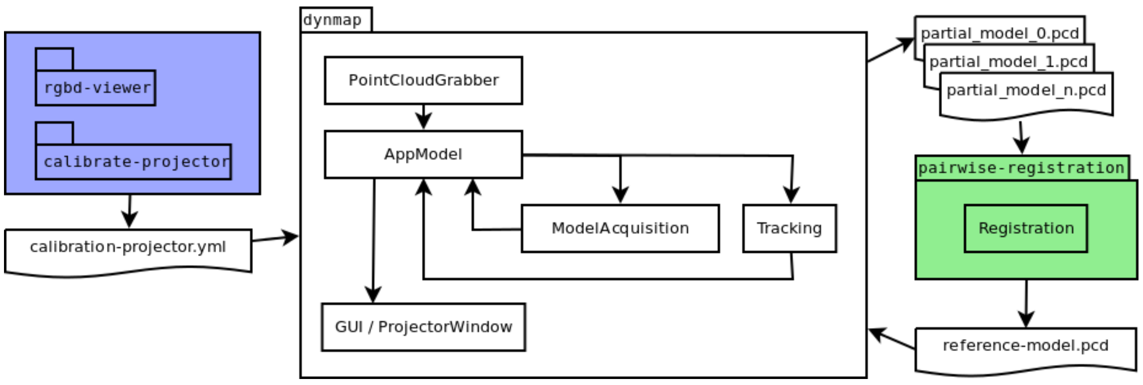


Figure 4.10: Structure of the programs involved in the solution

4.3.1 Calibration

Perform calibration The "Projector-Kinect Calibration Tutorial" by (Burrus [9]) is followed to obtain the projector calibration. Therefore the tools *rgbd-viewer* and *calibrate-projector* from the RGB-Demo project have to be built. The method assumes a projector as a "reverse" camera and performs a stereo camera calibration with methods provided by (OpenCV [27]). A plane board with different chessboard patterns in all corners (see 4.11) is positioned in front off camera and projector, so that another chessboard pattern gets projected on the middle of the board. With the physical chessboard, the world to camera space homography is given while the projected chessboard the projector unveils the projector to world homography. With this information the camera to projector homography can be obtained. With the *rgbd-viewer* about 10 to 30 images with different board poses are captured. The *calibrate-projector* application than takes those images, detects the corners of the real and virtual chessboards and computes a homography between the two coordinate spaces and an estimated pixel re-projection error which should be lower than 1.0 pixel by possibility. If it is much bigger, the calibration possibly failed due to too many images. The resulting parameters contain intrinsics and extrinsics of the projector with respect to the cameras location and are stored in .yml files for later use. Figure 4.12 illustrates the calibration procedure.

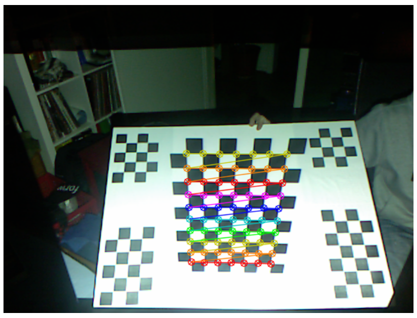


Figure 4.11: Detection of calibration pattern

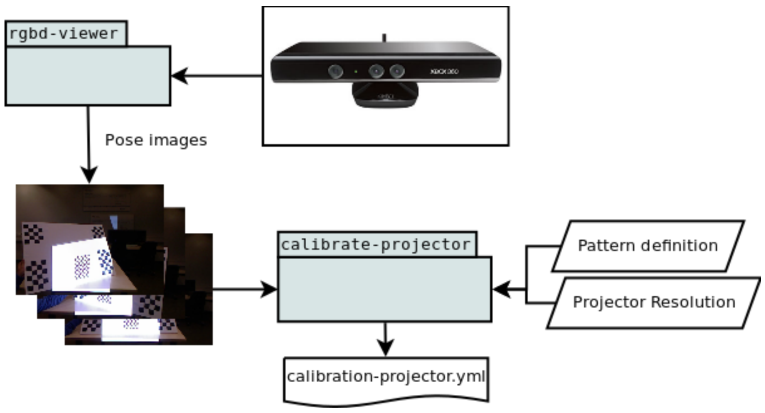


Figure 4.12: Overview of calibration procedure

4.3.2 Model acquisition

Due to the behaviour of ICP to find local minima, it was already considered to use sample consensus initial alignment for rough global alignment in advance. That raises the possibility to find a correct registration transformation for two partial overlapping point clouds but there still can be cases where the alignment fails. Therefore the registration of the partial point clouds can not be automated using this algorithms. Instead first partial model point clouds are saved as files which than can be pairwise registered. Every pairwise registration result has to be manually reviewed for feasible alignment.

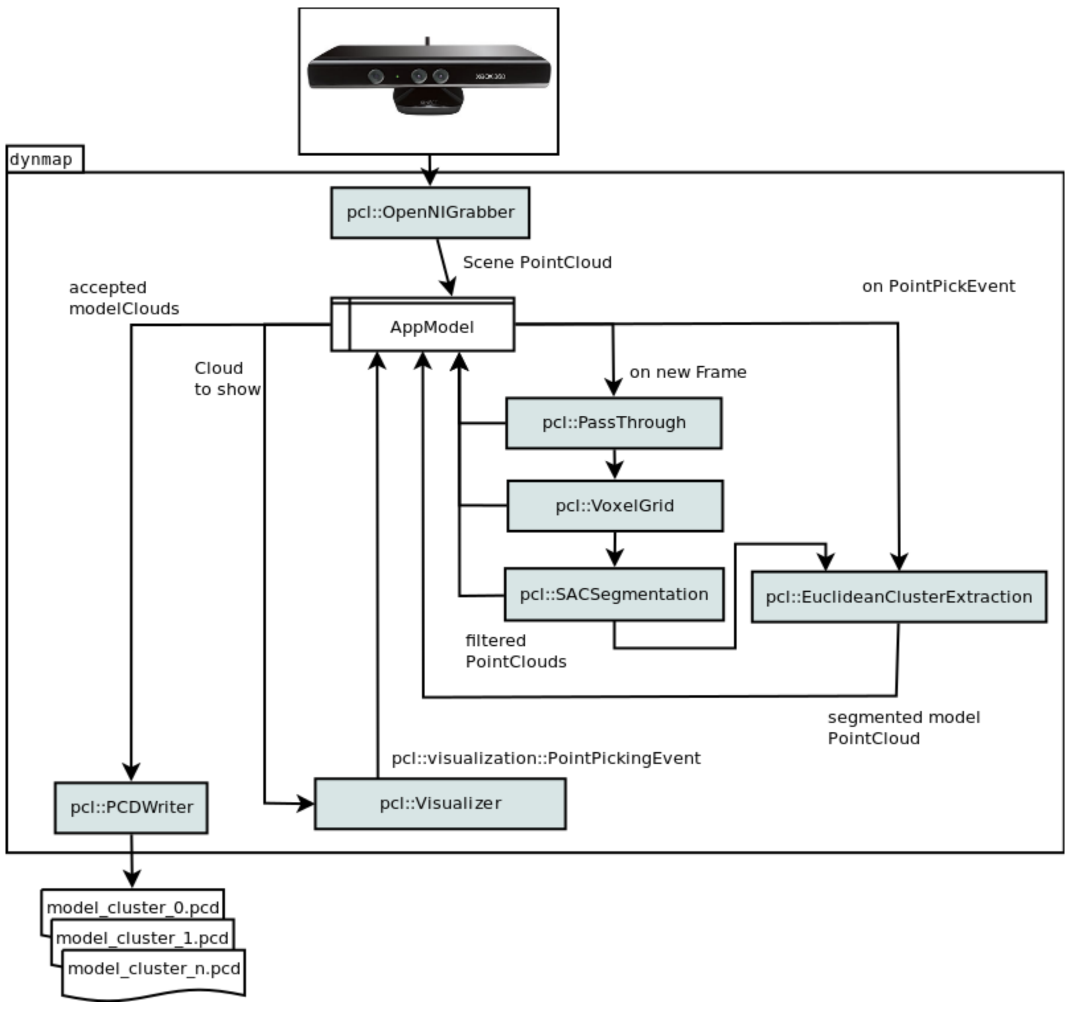
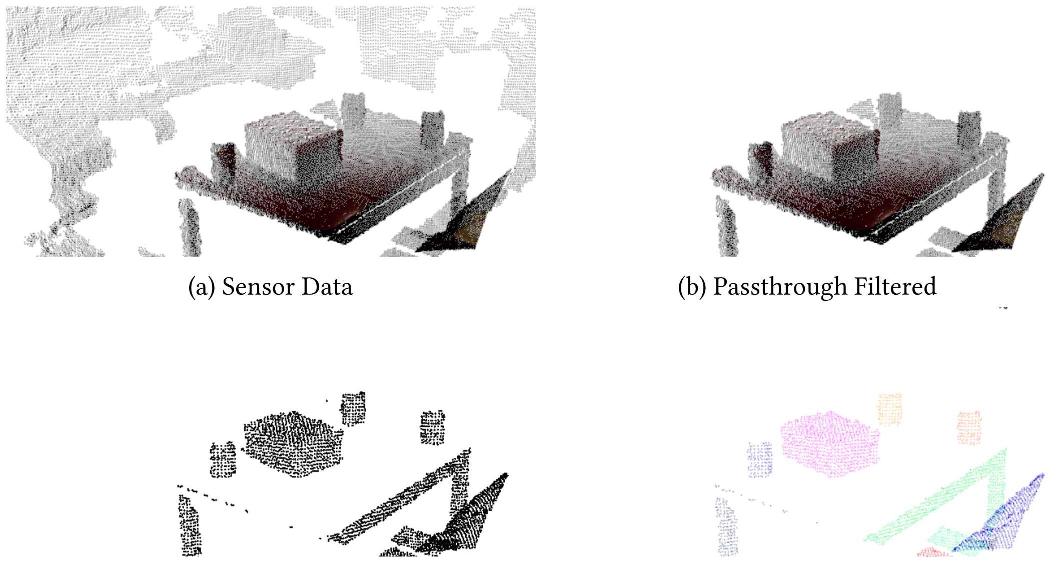


Figure 4.13: Overview of partial model capturing

Partial model acquisition

Figure 4.13 illustrates the function of the model acquisition pipeline. To segment an object out of the point cloud from the sensor, a series of filters are applied to it. First a pass-through filter is applied which only leaves points in a given bounding box. That bounding box is then down-sampled using a voxelgrid filter with a small leaf-size to achieve a uniform resolution while preserving the details of the model. Then the biggest plane in the remaining cloud will be searched with a plane model fitting algorithm that removes all points in this plane, leaving a scene as shown in figure 4.14c. Now the remaining point cloud consist of the main objects without the planar surface and some surrounding clutter. To finally separate the object out of the remaining point cloud it has to be selected in a *PCLVisualizer* by mouse click. The *PCLVisualizer* provides a click point in 3D space from which the closest point in the remaining cloud can be searched. Using the index of this point, an euclidean cluster extraction can collect all points belonging to the object in a separate point cloud that is stored for further processing.



(a) Sensor Data

(b) Passthrough Filtered

(c) Biggest Plane removed

(d) Clustered

Figure 4.14: Model acquisition filter pipeline

Registration with manual inspection

To provide a more complete model, this is done from different views of the object and the clusters are combined to provide a 360° model. The obtained model clusters are registered pairwise by first obtaining a rough alignment transformation via sample consensus initial alignment which is then refined by iterative closest point. To remove sparse outliers and to smooth noisy surfaces the statistical outlier and the moving least squares algorithms are applied. Figure 4.15 illustrates this registration procedure.

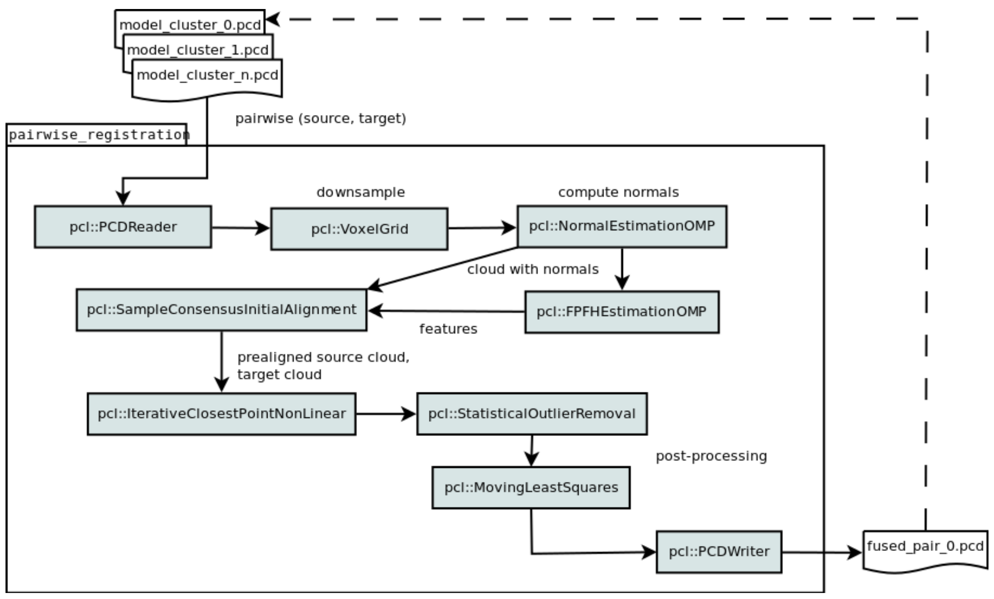


Figure 4.15: Overview of pairwise registration procedure

Figure 4.16 shows two partial views of the model in their original position and after the registration process was successful. The resulting transformation is applied to the corresponding point cloud, before they are concatenated.

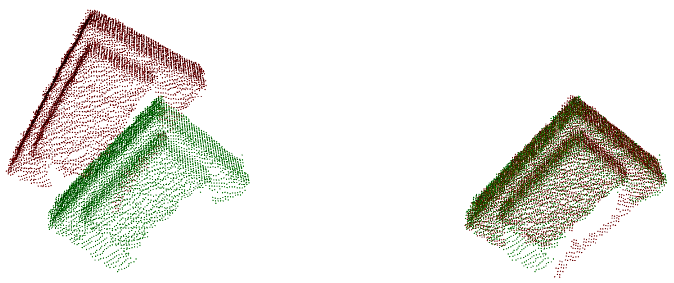


Figure 4.16: Two partial models (red/green) before (left) and after registration (right) using ICP

4.3.3 Object tracking

The tracker gets initialized to use a maximum of 500 particles and to start with 400 particles. As coherence factor point distance was chosen. For the pose estimation the created model and the last captured frame get down-sampled to the same resolution using a voxel-grid filter. The model is set initially as reference point cloud to the tracker while the scene point cloud has to be renewed for every incoming frame. Then the tracker is started to obtain the pose estimation for the object. For every processed frame, the 3D pose is delivered by the Tracker as *PointXYZRPY* which represents the most likely pose candidate of the actual frame. The pose is turned into a transformation matrix and stored for transforming the model point cloud before visualization. Figure 4.17 shows an overview of the tracking pipeline.

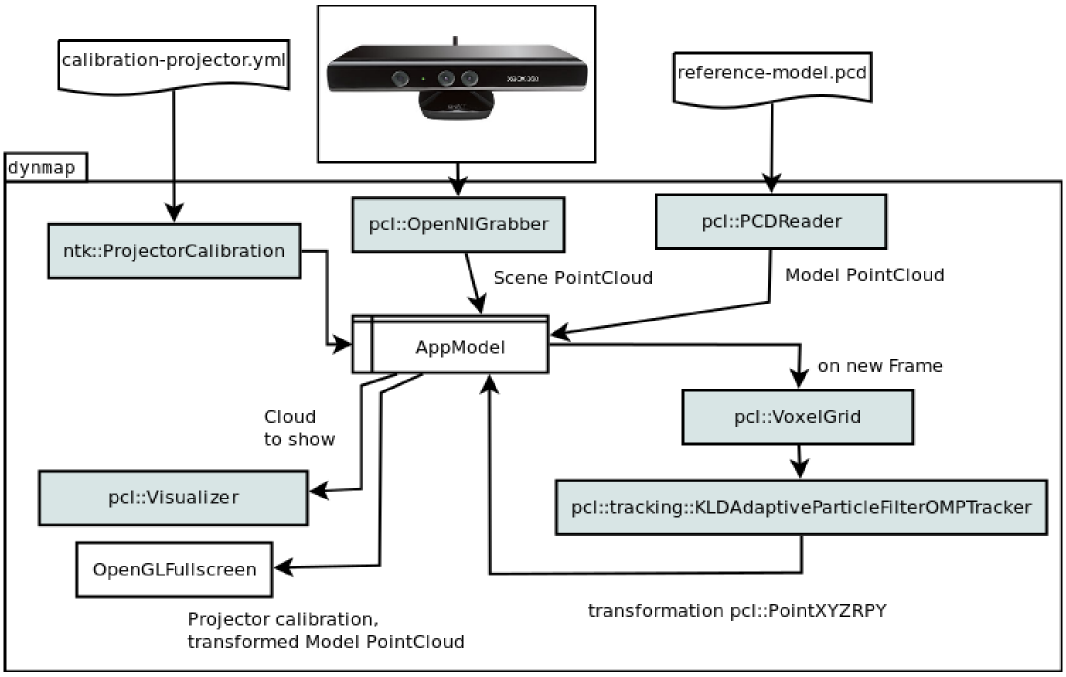

calibration-projector.yml
reference-model.pcd
dynmap
ntk::ProjectorCalibration
pcl::OpenNIGrabber
pcl::PCDReader
Scene PointCloud
Model PointCloud
AppModel
on new Frame
pcl::VoxelGrid
Cloud to show
pcl::Visualizer
pcl::tracking::KLDAdaptiveParticleFilterOMPTracker
OpenGLFullscreen
Projector calibration, transformed Model PointCloud
transformation pcl::PointXYZRPY

Figure 4.17: Overview of tracking procedure

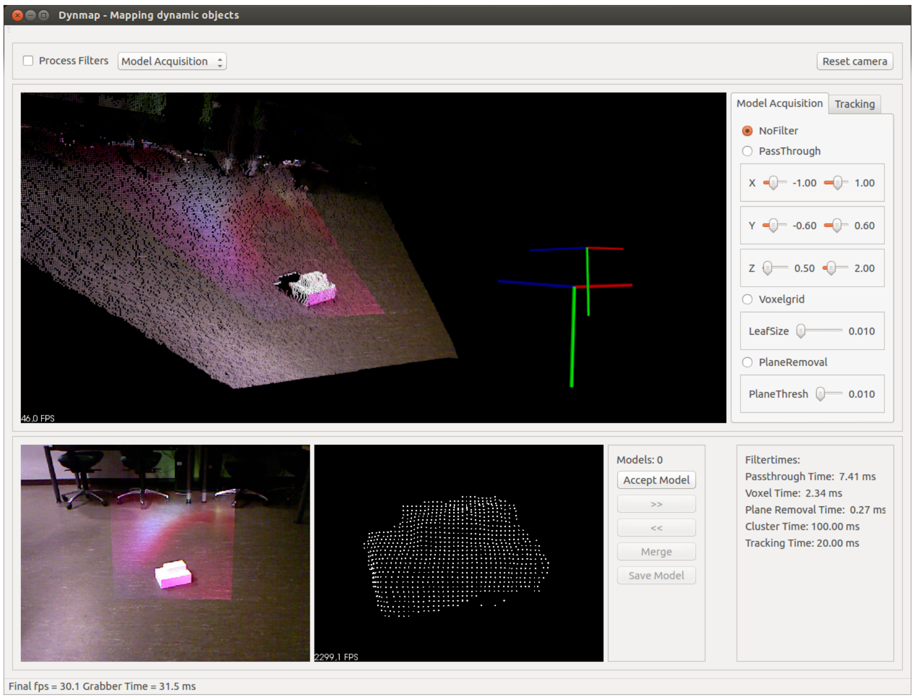
4.3.4 Virtual Scene Construction

To achieve a projector view, the inverse of the extrinsic parameters has to be applied to the scene and the intrinsic parameters have to be used to set up the virtual camera. While the *PCLVisualizer* is a nice tool to provide a technical overview of what is going on in the point cloud, it is not suitable to provide the virtual camera that defines the projector perspective

because the camera of the underlying (VTK Toolkit [44]) does not support off-axis projection yet in the version PCL supports. This is needed, because nearly every video projector projects with an offset angle to the z-axis what means that the lenses principal point is not located in the image center. Therefore an OpenGL based pointcloud visualizer was created which gets configured with an OpenGL version of the intrinsic matrix.
With the pose information from tracking, the model point cloud is transformed and rendered in the projector window.

4.4 Evaluation

In this section the developed solution will be presented and discussed. Figure 4.18 shows the main user interface which displays the point cloud after plane removal at the top, the RGB image from the Kinect at the lower left and the latest segmented model at the lower right. Next to the main visualizer the filter parameters can be controlled and the intermediate point clouds for display can be selected. The rightmost section in the lower panel displays averaged computation times for each filter stage.



Models: 0

NoFilter [✓]
PassThrough
X -1.00 1.00
Y -0.60 0.60
Z 0.50 2.00
Voxelgrid
LeafSize 0.010
PlaneRemoval
PlaneThresh 0.010

46.0 FPS

2299.1 FPS

Filtertimes:
Passthrough Time: 7.41 ms
Voxel Time: 2.34 ms
Plane Removal Time: 0.27 ms
Cluster Time: 100.00 ms
Tracking Time: 20.00 ms

Final fps = 30.1 Grabber Time = 31.5 ms

Figure 4.18: User Interface of the developed solution

4.4.1 The calibration parameters

The calibration is performed as stated in section 4.3.1, several board pose images are captured with the *rgbd-viewer* and after that, the calibration parameters are calculated with *calibrate-projector*. The output depicted in listing 4.1 of the later shows the intrinsic matrix of the projector (line 2-4), the estimated lens distortion coefficients (line 5) and indicates the quality of the calibration (line 7).

```
1 ...
2 [2539.664076205458, 0, 636.4047646278752;
3   0, 2489.958599900968, 555.751728675664;
4   0, 0, 1]
5 [0, 0, 0, 0, 0]
6 Debug: [DBG] "errorInt: 1.77581"
7 Average pixel reprojection error: 0.783524
```

Listing 4.1: Output of *calibrate-projector*

To visualize the positions and orientations of projector and camera in the user interface, their corresponding coordinate systems are drawn. The cameras location in virtual space is at the zero point facing into Z direction while the projectors position and orientation is defined by the acquired extrinsic calibration parameters. Figure 4.19 shows the physical scene with subsequently inserted, manually drawn coordinate systems for Kinect camera and projector.

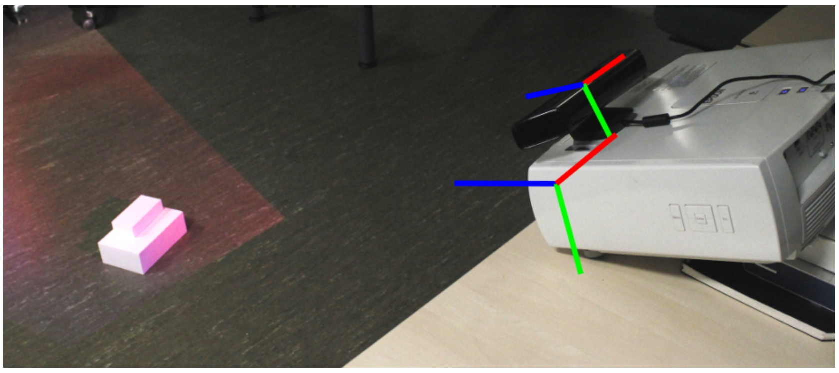


Figure 4.19: Kinect camera and projector placement (coordinate systems drawn manually)

In figure 4.20 the two coordinate systems, which are calculated from the acquired calibration, show up in plausible locations in the virtual scene. This indicates roughly that the calibration is correct.

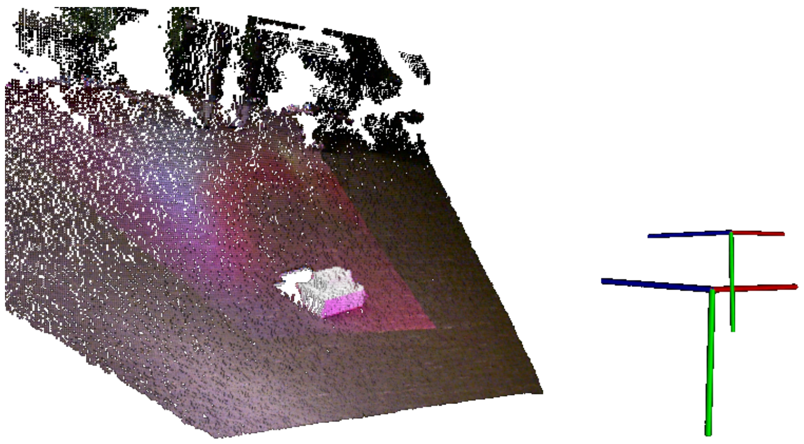


Figure 4.20: Kinect camera and projector placement in virtual space (coordinate systems calculated from calibration)

4.4.2 Model acquisition

For the reference model that will be used in the tracking algorithm, several different views of the physical objects are captured. Figure 4.21 depicts six segmented different views of the object each showing only the surfaces visible to the camera.

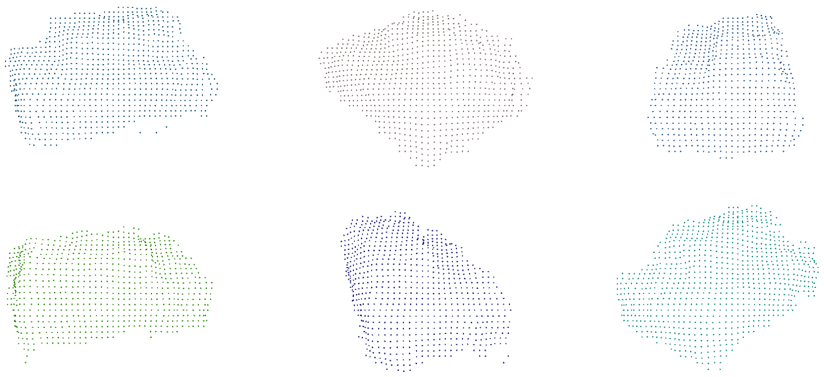


Figure 4.21: Six different segmented views of the object to track

Those partial views are then registered to a full 360° point cloud model of the object by pairwise registering two partial model. Even with a good initial alignment for example through RANSAC initial alignment it happens that the two partial model are fitted in a way that does not reflect the real form of the object. In figure 4.22 the red model would have to be rotated counter-clockwise about 90°to fit the green model. ICP instead found a stronger local minimum by mainly translating the red model to the upper left. Therefore every aligned pair has to be reviewed if the alignment is correct.

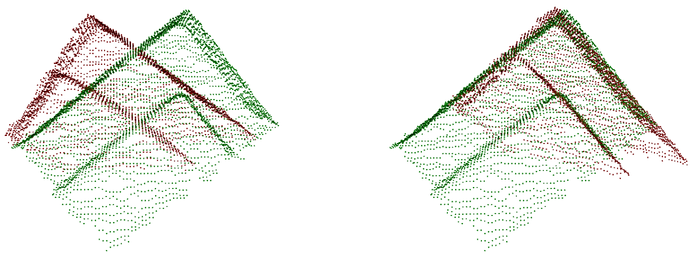


Figure 4.22: Two partial models (red/green) before (left) and after registration (right). ICP has converged to a local minimum

Only successful registered pairs are then used for further pair alignment. Due to the fact that the partial models have overlapping parts, not necessarily all partial models are needed for a full model. In figure 4.23 the registered model is very noisy, but after outlier removal and MLS smoothing the model looks quite clear although the edges also get rounded. The final model can then be used for tracking purposes.

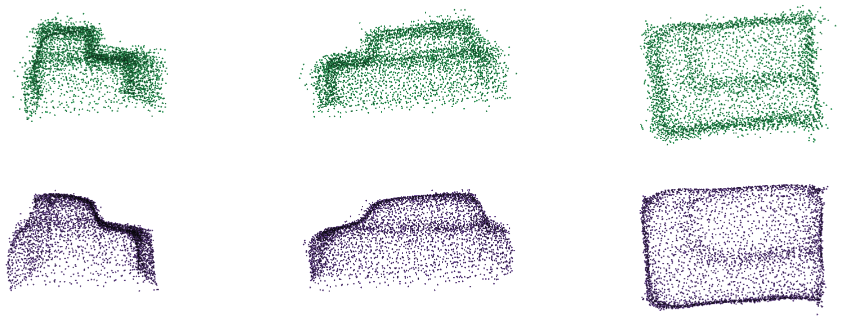


Figure 4.23: Fused model after registration (top row) and smoothed model (bottom row)

4.4.3 Runtime

Below image sequence represents the results of this solutions pretty good and also shows the weaknesses of this approach. The sequence is made with one image per second and the model point cloud is projected in red onto the tracked object. In the first three images of the sequence the alignment is accurate. Then from image 4 to image 12 the projections is misaligned by up to 3cm. After changing movement direction, the projection finds back to the correct pose from image 13 to image 14. Overall the position gets tracked quite reliable so that the projection is following with an expectable delay. Rotation seems to be a little bit more uncertain. If the object is not moving the tracking algorithm does not really comes to rest. For each frame the position and orientation slightly jumps around the real position.

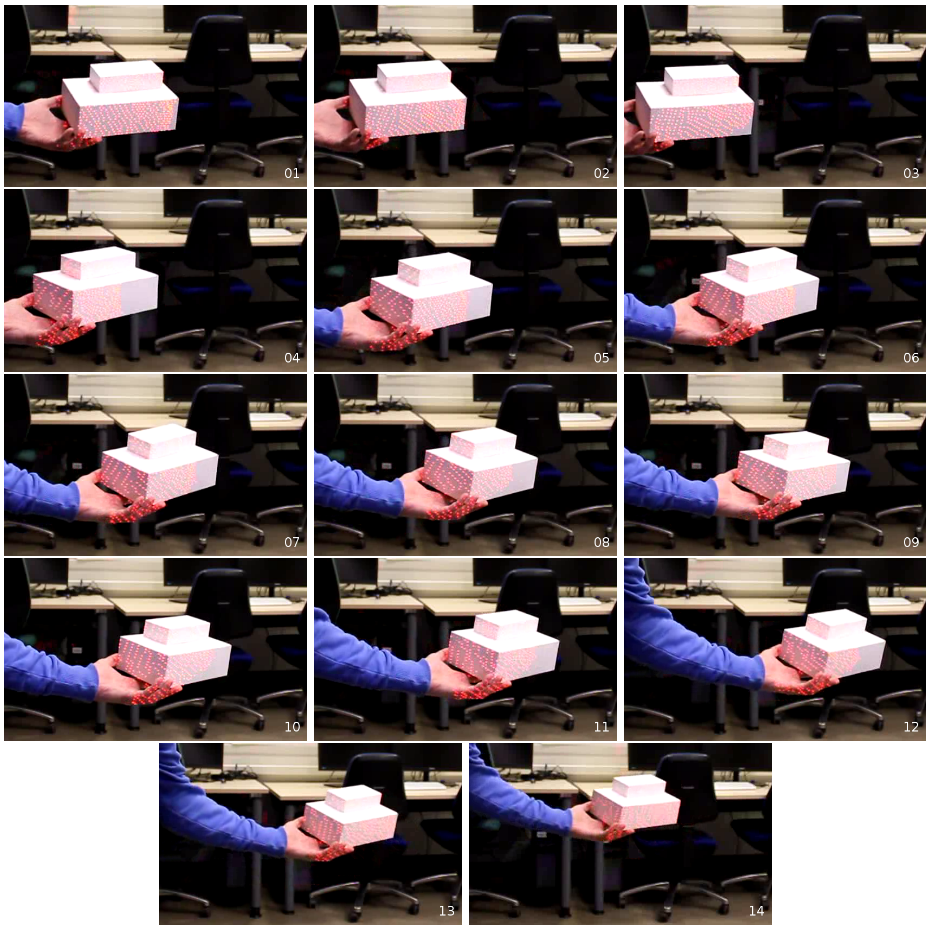


Figure 4.24: Image sequence of tracking with 1 second between images

4.4.4 Measurements

During the experiments average computation times where calculated. Note that the times where calculated with the filter pipelines in affect, so for example the voxelgrid filter time depends on how many points are left after passthrough filtering.

The first filtering stage in the model acquisition pipeline is the passthrough filter. The performance of the following processing stages depends on how many points are left after passthrough filtering. Because of the *organized* nature of the point cloud obtained from the Kinect, there can be only one point at a (x, y) position. Therefore, doubling the edge length of the passtrough filtering volume results maximally in a quadrupling of the point count. The amount of points that the planar surface extraction has to deal with also quadruples at maximum. The input point number for the cluster extraction thus varies due to noise, that results in a varying number of extracted planar surface points.

In figure 4.25 the impact of different passthrough filter volumes to the following processing stages is shown. While the filtering times for passthrough and voxelgrid filter is nearly constant, the time for extracting the biggest planar region grows exponentially. The computation time for the cluster extraction shows the most significant growth, indicating that this algorithms computation time is heavily dependent on the input data size. As a very basic clustering algorithm is used, there probably are more advanced algorithms, that could be used.

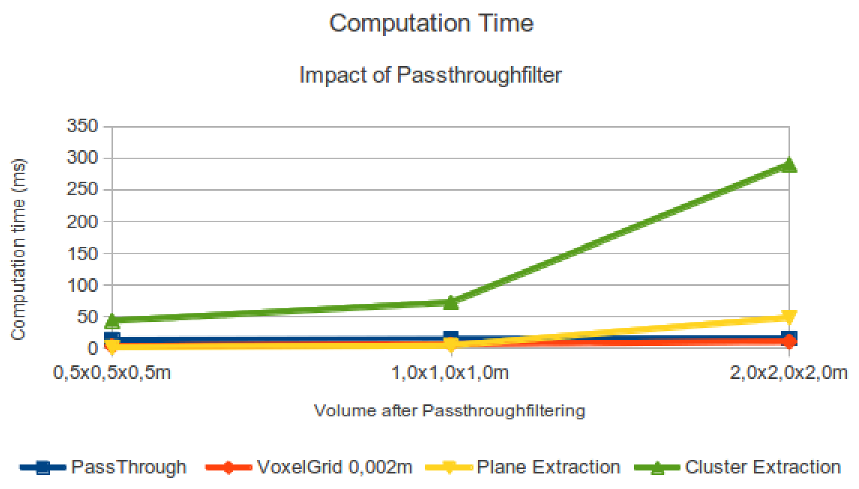


Figure 4.25: Impact of Passthrough filter on following computation

The second suspect of computation time investigation is the voxelgrid filter. For comparison reasons this is done with two different passthrough volumes. The voxelgrid filter leaves one point per leaf, resulting in a maximum of $n = \left(\frac{Xlength_{Passtrough}}{Leafsize_{Voxelgrid}}\right) * \left(\frac{Ylength_{Passtrough}}{Leafsize_{Voxelgrid}}\right)$ points after filtering. Therefore doubling the voxelgrid leaf size results in a quartered point count at maximum after voxelgrid filtering.

Figure 4.26 and 4.27 illustrate the impact of different leaf sizes to the following filters with two different passthrough filter volumes. Of course the passthrough filter is not affected by the changing voxelgrid filter. The voxelgrid filters computation time is only slightly affected by its changing setting. Here again the planar region extraction shows a exponential growth behaviour. The same applies to the computation time of the cluster extraction which shows a different behaviour at the right end in figure 4.26 and on the left end in figure 4.27. This can be explained with the minimum and maximum cluster size setting of the cluster extraction. The denser the point cloud is, the more points are in a cluster and more likely the maximal cluster size is met.

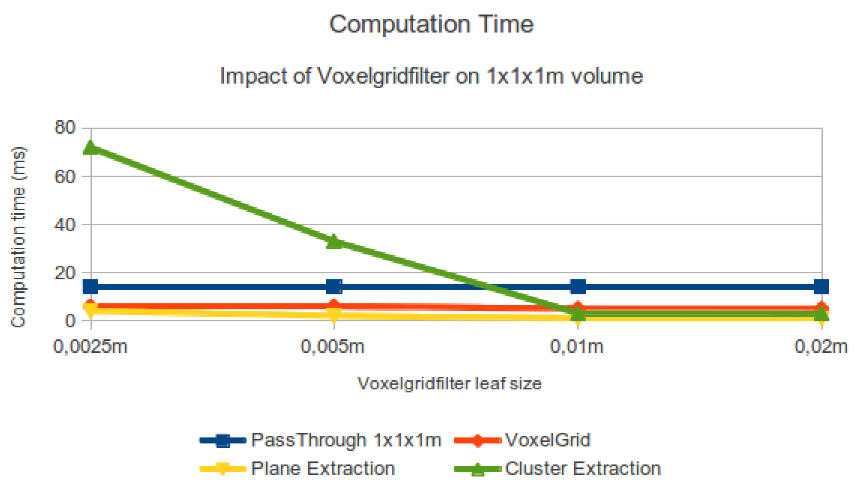


Figure 4.26: Impact of Voxelgrid filter on following computation

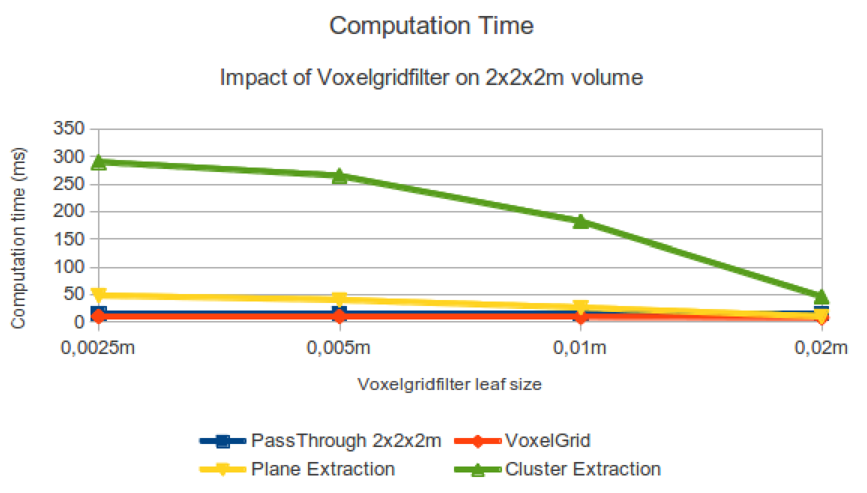


Figure 4.27: Impact of Voxelgrid filter on following computation

In the tracking pipeline is used to down-sample model and scene point clouds to the same resolution. Figure 4.28 shows the impact off different voxelgrid filter leaf sizes on the tracking algorithm. Again the voxelgrid computation time is close to constant. The average tracking time in contrast is very inconsistent, therefore a minimum and maximum value is plotted. It seems that too dense point clouds and also too sparse point clouds cause the tracking algorithm to be more expensive.

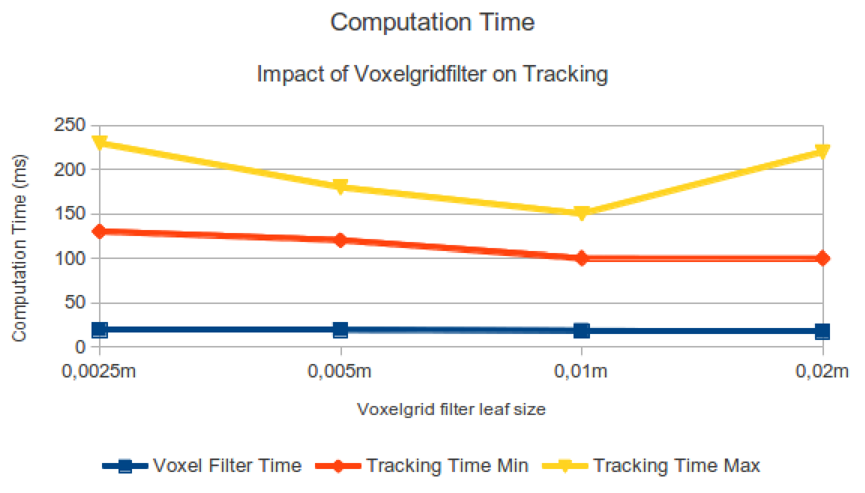


Figure 4.28: Impact of Voxelgrid filter on tracking

Overall the performance of voxelgrid filter, passthrough filter and plane extraction are good enough for real-time application on Kinect data with 30 FPS. The performance of the cluster extraction could surely improved by using more advanced cluster extraction algorithms. The performance of the tracking algorithm would also have to be improved further for real-time tracking. The trackers configuration can certainly be optimized for this purpose, by using different coherence factor combinations with different weights and by tuning the probability parameters.

4.5 Summary

After defining the outlines of the solution and making basic assumptions (see 4.1), a thorough examination of the proposed sensor and library candidates showed the suitability of those tools for this thesis. The algorithms used to develop the solution where explained (see 4.2).

In section (4.3) the structure of the solution is explained for each of the problem fields. Point cloud processing pipelines for model acquisition and tracking have been developed from the given algorithms. An output window that gets configured with the projectors calibration parameters was developed to produce the projectors output.

The developed solution and exemplary results are then presented for each of the problem fields. The implementation of the theoretical solution was proven feasible but shows also that it is still hard, to achieve an optimal result (see 4.4). Measurements have revealed performance bottlenecks in the solution and led to proposals for improvement.

5 Conclusion

Coming from the technical research field of spatial augmented reality, static video projection mapping has been adapted to an art form. Several approaches have been made to video map dynamic objects (see 2.1). From 2D mapping using dynamic masks over marker-based 2D tracking video projection mapping has evolved to 3D mapping approaches using controlled movement or vision-based simple polygonal reconstruction. A real-time reconstruction method based on the Microsoft Kinect sensor introduces the idea of creating a 3D model of successive point clouds. It turned out that the identified problem fields are actual research topics which offer solution approaches (see 2.2).

After defining the objective of this thesis, the problem fields have been analysed. That resulted in a separation of the solution in an offline phase for calibration and model acquisition and an online phase for object tracking and visualization. Also a rough idea of the implementation was developed (see 3).

In (4.2) the used libraries are introduced and thoroughly analysed. The algorithms which contribute to the overall solution are examined and explained (see 4.2.3). With that knowledge the solution was developed. Point cloud processing pipelines for the model acquisition and the object tracking task where implemented and a projector window that resembles the projectors intrinsics from the acquired calibration file was created (see 4.3).

Finally the developed solution is presented which showed that the idea is feasible. The tracking method provided good enough accuracy to project reliably onto the object although the alignment of the projection was not perfectly fitting (see 4.4). Measurement showed bottlenecks regarding the performance (see 4.4.4).

Drawbacks

Frame rate It turned out that the possible movement speed feels much lower than expected because of the delay between successive frames. A small calculation example shows the problem. When the object is moved at a speed of $1m/s$, which is a medium movement speed for a stretched out arm, and the Kinect is producing Point Clouds at 30 frames per second, the object will move $3,3cm$ between two frames, ignoring the calculation time for the re-projection. Therefore for a really seamless projection faster cameras and then certainly faster computation are required.

Tracking Method One problem is that the tracking method fails to estimate a steady pose when the objects is not moving. Possibly this could be circumvented by somehow smooth the acquired poses over time. One other drawback regarding the tracking was already mentioned. Since particle filtering follows an accumulative approach, the tracking fails consequently if it lost track of the object. Therefore the tracker would have to be reset if the tracking fails, to start over without any historical data to base the estimation on.

Global alignment The lack of a reliable global registration methods makes it hard to automatically acquire a 3D model using ICP on partial model point clouds. This problem is widely known and is subject to current research.

Further Work

Further work in this topic is needed with regards to the particle filter configuration, which still offers lots of optimization possibilities. Also the usage of more advanced, faster clustering algorithms could improve this solution. Much work is needed for the vision towards a real-time 3D projection-mapping for moving and deforming objects like clothings. As real-time reconstruction still proves to be a challenge this is an interesting research topic. State off the art research is done in closely related fields that provides lots of inspiration to tackle this idea. Current trends like the continuing efforts in parallelizing algorithms for multi core and GPU computing also represent a favourable development.

Bibliography

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C.T. Silva. Computing and rendering point set surfaces. *Visualization and Computer Graphics, IEEE Transactions on*, 9(1):3–15, 2003.

[2] P. Azad, D. Munch, T. Asfour, and R. Dillmann. 6-dof model-based tracking of arbitrarily shaped 3d objects. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5204–5209. IEEE, 2011.

[3] F. Bernardini and H. Rushmeier. The 3d model acquisition pipeline. In *Computer Graphics Forum*, volume 21, pages 149–172. Wiley Online Library, 2002.

[4] P.J. Besl and N.D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on pattern analysis and machine intelligence*, 14(2):239–256, 1992.

[5] ZM Bi and L. Wang. Advances in 3d data acquisition and processing for industrial applications. *Robotics and Computer-Integrated Manufacturing*, 26(5):403–413, 2010.

[6] O. Bimber, R. Raskar, and M. Inami. *Spatial augmented reality*. AK Peters, 2005.

[7] H. Bohnacker, B. Gross, J. Laub, and C. Lazzeroni. *Generative Gestaltung: entwerfen, programmieren, visualisieren*. Schmidt, 2009.

[8] J.A. Brown and D.W. Capson. A framework for 3d model-based visual tracking using a gpu-accelerated particle filter. *Visualization and Computer Graphics, IEEE Transactions on*, 18(1):68–80, 2012.

[9] N. Burrus. Kinect rgbdemo v0.7.0. URL `http://labs.manctl.com/rgbdemo/index.php/Main/HomePage`. last accessed: 25.09.2012.

[10] C. Choi and H.I. Christensen. Real-time 3d model-based tracking using edge and keypoint features for robotic manipulation. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4048–4055. IEEE, 2010.

[11] G. Cuccuru, E. Gobbetti, F. Marton, R. Pajarola, and R. Pintus. Fast low-memory streaming mls reconstruction of point-sampled surfaces. In *Proceedings of Graphics Interface 2009*, pages 15–22. Canadian Information Processing Society, 2009.

[12] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996.

[13] G. Falcao, N. Hurtos, and J. Massich. Plane-based calibration of a projector-camera system. *VIBOT Master*, 2008.

[14] D. Fox. Kld-sampling: Adaptive particle filters and mobile robot localization. *Advances in Neural Information Processing Systems (NIPS)*, pages 26–32, 2001.

[15] D. Fox. Adapting the sample size in particle filters through kld-sampling. *The international Journal of robotics research*, 22(12):985–1003, 2003.

[16] D. Glasner, M. Galun, S. Alpert, R. Basri, and G. Shakhnarovich. Viewpoint-aware object detection and pose estimation. In *2011 IEEE International Conference on Computer Vision (ICCV)*, pages 1275–1282. IEEE, 2011.

[17] M. Godec, P.M. Roth, and H. Bischof. Hough-based tracking of non-rigid objects. In *2011 IEEE International Conference on Computer Vision (ICCV)*, pages 81–88. IEEE, 2011.

[18] R. Hartley, A. Zisserman, and Inc ebrary. *Multiple view geometry in computer vision*, volume 2. Cambridge Univ Press, 2003.

[19] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 559–568. ACM, 2011. ISBN 978-1-4503-0716-1. doi: 10.1145/2047196.2047270. URL `http://doi.acm.org/10.1145/2047196.2047270`.

[20] B.R. Jones. Augmenting complex surfaces with projector-camera systems. 2011.

[21] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality, 1999.(IWAR'99)*, pages 85–94. IEEE, 1999.

[22] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006.

[23] J.C. Lee. *Projector-based location discovery and tracking*. PhD thesis, Intel, 2008.

[24] V. Lepetit and P. Fua. *Monocular model-based 3D tracking of rigid objects*. Now Publishers Inc, 2005.

[25] L. Mei, J. Liu, A. Hero, and S. Savarese. Robust object pose estimation via statistical manifold modeling. In *2011 IEEE International Conference on Computer Vision (ICCV)*, pages 967–974. IEEE, 2011.

[26] K. Obermaier. Apparition. *Proceedings of Ars Electronica 2004*, pages 314–318, 2004.

[27] OpenCV. Opencv. URL `http://opencv.willowgarage.com/wiki`. last accessed: 08.10.2012.

[28] openFrameworks. openframeworks. URL `http://www.openframeworks.cc`. last accessed: 25.09.2012.

[29] OpenNI. OpenNI. URL `http://openni.org/`. last accessed: 27.11.2012.

[30] Q. Pan, G. Reitmayr, and T. Drummond. Proforma: Probabilistic feature-based on-line rapid model acquisition. In *Proc. 20th British Machine Vision Conference (BMVC)*, 2009.

[31] panGenerator collective. Peacock. URL `http://vimeo.com/49869407`. last accessed: 18.10.2012.

[32] PCL. Point Cloud Library 1.7.0.rv8129. URL `http://pointclouds.org`. last accessed: 25.09.2012.

[33] Guillermo Perez, German Hoffmann, Rodrigo Rivera, and Veronica Manduca. Mapinect. URL `http://mapinect.wordpress.com/`. last accessed: 08.01.2013.

[34] R. Poppe. Vision-based human motion analysis: An overview. *Computer Vision and Image Understanding*, 108(1):4–18, 2007.

[35] Processing. Processing. URL `http://processing.org`. last accessed: 25.09.2012.

[36] Radu Bogdan Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Computer Science department, Technische Universitaet Muenchen, Germany, 10 2009.

[37] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[38] R.B. Rusu, Z.C. Marton, N. Blodow, M. Dolha, and M. Beetz. Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56 (11):927–941, 2008.

[39] R.B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *ICRA'09. IEEE International Conference on Robotics and Automation*, pages 3212–3217. IEEE, 2009.

[40] R.B. Rusu, A. Holzbach, N. Blodow, and M. Beetz. Fast geometric point labeling using conditional random fields. In *International Conference on Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ*, pages 7–12. IEEE, 2009.

[41] M. Shaheen, J. Gall, R. Strzodka, L. Van Gool, and H.P. Seidel. A comparison of 3d model-based tracking approaches for human motion capture in uncontrolled environments. In *Applications of Computer Vision (WACV), 2009 Workshop on*, pages 1–8. IEEE, 2009.

[42] R. Szeliski. *Computer vision: Algorithms and applications*. Springer, 2010.

[43] A. Tevs, A. Berner, M. Wand, I. Ihrke, M. Bokeloh, J. Kerber, and H.P. Seidel. Animation cartography - intrinsic reconstruction of shape and motion. *ACM Transactions on Graphics (TOG)*, 31(2):12, 2012.

[44] VTK Toolkit. Vtk. URL http://www.vtk.org. last accessed: 13.11.2012.

[45] White Kanga. Modeling projection system. URL http://whitekanga.pl/pl/produkty. last accessed: 19.11.2012.

[46] Y. Xu and D.G. Aliaga. High-resolution modeling of moving and deforming objects using sparse geometric and dense photometric measurements. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1237–1244. IEEE, 2010.

[47] H. Yang and G. Welch. Model-based 3d object tracking using an extended-extended kalman filter and graphics rendered measurements. In *Computer Vision for Interactive and Intelligent Environment, 2005*, pages 85–96. IEEE, 2005.

[48] T.C. Yapo, Y. Sheng, J. Nasman, A. Dolce, E. Li, and B. Cutler. Dynamic projection environments for immersive visualization. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1–8. IEEE, 2010.

[49] S. Zhang. Recent progresses on real-time 3d shape measurement using digital fringe projection techniques. *Optics and lasers in engineering*, 48(2):149–158, 2010.

[50] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, January 31, 2013 Iwer Petersen