Simultaneous calibration of a colour camera, depth camera, and projector and 3D object pose recognition

Laura Rosalia Luidolt*†
*Supervised by: Andreas Reichinger**

*VRVis Research Center, Vienna / Austria
†Computer Graphics Research Division, TU Wien, Vienna / Austria

Abstract

In this paper, we present a novel approach to calibrate a 4-component setup, consisting of a movable object beneath a static arrangement of a colour camera, a depth camera, and a projector, which was developed as part of a tactile audio guide to help visually impaired people perceive images. For the static arrangement, we propose a procedure that allows calibrating all three devices simultaneously using a single calibration target with two different calibration patterns – one printed and one projected. The application is easy to use, due to the mostly automated calibration process and the algorithm still achieves reasonable accuracy for the devices and their respective poses. For the movable object, a point cloud matching algorithm is implemented to detect its 3D pose with respect to the three device coordinate systems, when placed beneath the setup. This allows to register interaction gestures to specific areas on the object and to project images onto the object.

Keywords: Camera Calibration, Projector Calibration, Point Cloud Matching

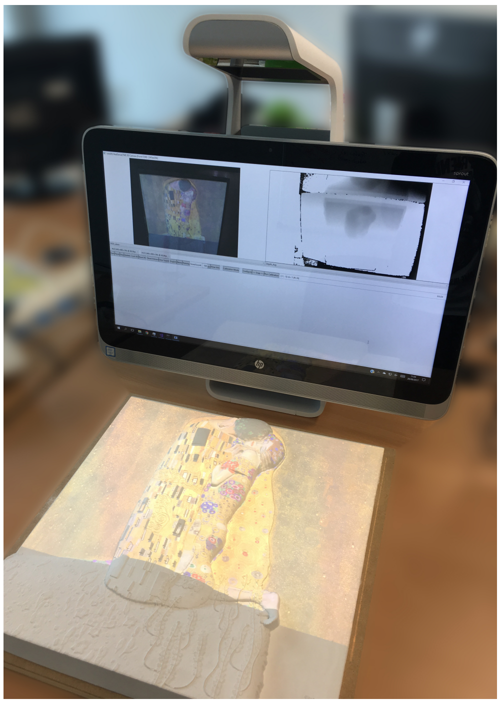

Figure 1: Setup of the interactive audio guide with the HP Sprout.

1 Introduction

This work was developed as part of the ARCHES project[1] for a tactile audio guide to help blind and visually impaired people to perceive images and objects (for more information see [9]). The images, more specifically paintings, are provided as 3D reliefs. While in the previous setup the relative pose of the relief was bound to be static, we present a method to determine its pose automatically. We are using the *Sprout Pro by HP*, an all-in-one desktop computer, with an included colour camera, depth (infrared) camera and projector, which can be seen in Figure 1.

Arbitrary placement of 3D objects in front of a properly calibrated depth camera should lead to immediate recognition at a sufficient accuracy. Additionally, the ability to determine this pose in a colour camera is advantageous in order to register gestures to locations on the relief, which is used as a user interface for the audio guide. Moreover, a calibrated projector can be applied to highlight specific areas for interactive visual feedback, as well as target different types of visual impairments. Therefore, calibration and synchronization of multiple in- and output devices hold various benefits over a single calibrated component.

In this paper, we show a method to calibrate a colour camera, a depth camera and a projector and determine their respective poses using intrinsic and extrinsic camera parameters. The presented calibration procedure can be performed for all three devices with a single dataset of images made with the same calibration target. This enables an easy-to-use, autonomous process that calibrates the whole setup at once, without any other user input needed.

The pose of a given 3D object is extracted from the information provided by the depth camera. By placing an object beneath our setup and finding its pose, the accuracy of the previously performed calibration can easily be evaluated, e.g. by projecting the original image onto the relief (see Figures 1, 11, and 12).

[1] http://www.arches-project.eu/

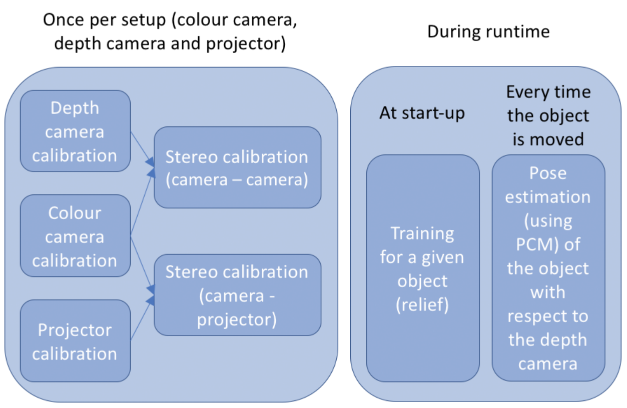
Once per setup (colour camera, depth camera and projector)
During runtime
Depth camera calibration
Colour camera calibration
Projector calibration
Stereo calibration (camera – camera)
Stereo calibration (camera - projector)
At start-up
Every time the object is moved
Training for a given object (relief)
Pose estimation (using PCM) of the object with respect to the depth camera

Figure 2: An application overview.

An application overview is shown in Figure 2. While the main calibration procedure of the colour camera and depth camera (operating on the infrared image) relies on the method proposed by Zhang [12], our contribution lies in the additional simultaneous calibration of a projector using the same input images as for the cameras. Moreover, the pose detection of our reliefs relies on the method of Drost and Ilic [4].

First, a short introduction to previous work is given in Section 2. The main contribution of our paper, the calibration algorithm is described in Section 3. It consists of how different camera colour channels are used (Section 3.1), the calibration algorithms of the colour and the depth camera (Sections 3.2 and 3.3), the algorithm of a stereo calibration of two cameras (Section 3.4) and the calibration algorithm of a projector (Section 3.5). The pose estimation of an object using point cloud matching is described in Section 4. Finally, results are discussed in Section 5 and the conclusion stated in Section 6.

2 Related Work

There is a lot of work focusing on a simple and quick way to calibrate a typical colour camera, for example, using a 1D point system as presented by Zhao et al. [13]. Moreover, self-calibration methods, as presented by Pollefeys et al. [8], can be done completely without specific calibration patterns. Heikkila proposed a new camera model with better accuracy [6], yet this has only been proven on synthetic images. The most common approach was proposed by Zhang [12] and relies on a simple, printed 2D calibration pattern mounted on a planar surface. Multiple images have to be taken and furthermore an algorithm to minimize the projection error for each 2D point is applied. This method is used in our work for the calibration of both the colour camera and the depth camera on the infrared image.

Previous projector calibration procedures, as presented by Zhang et al. [11], usually only operate on fixed calibration planes, so no simple user interaction is possible. A different approach proposed by Audet et al. [1] relies on a prewrap transformation that is gradually improved – this does not allow for static calibration (granting a clear division of the user input by capturing the images and the calibration procedure itself), which is possible in our method. Cassinelli et al. [3] use two different calibration patterns (a printed chessboard pattern and a projected dot pattern) side by side. We are improving this method by allowing an overlapping of the two patterns, which also leads to enabling static input.

Most point cloud matching algorithms rely on very complex features, for example, Rusu et al. [10] present a set of 16D features for each point, which will then be matched. Huang et al. [7] introduce 3D self-similarity vectors. Drost et al. [5] rely on point pair descriptors with simple features, like the angle between two vectors and its length. Due to the simplicity and yet reasonable results, we will be using this approach in our paper.

3 Calibration

When calibrating a camera using a typical algorithm (e.g. Zhang [12]), a calibration pattern, for example, a simple chessboard pattern, has to be captured multiple times from different positions and orientations. Given that the depth camera also provides an infrared image in which the pattern can be detected, the capturing of calibration images can be done without any difficulty for both colour and depth cameras simultaneously. This also enables simple stereo calibration that results in the respective poses (e.g. Bouguet [2]).

A more advanced approach is needed in order to additionally enable a concurrent calibration of a projector. One possible technique is projecting a pattern onto a plane and capturing images with an already calibrated camera. However, the pose of the plane has to be known in camera space and should as well vary in position and orientations in the different images.

So, in order to calibrate all three devices at the same time, both a printed pattern and a projected pattern has to be on a plane simultaneously. Cassinelli and Bergström [3] solved this by using two smaller patterns side by side. Yet this method requires the projected pattern to dynamically change depending on the pose of the calibration pattern.

In our solution, we use different colour mappings for the two different patterns, which allows us to have a static pattern for the projector, as well as being able to cover the whole available camera space for the patterns in the calibration images. The final configuration that worked best for us can be seen in Figure 3. The whole calibration procedure is outlined in Figure 4. In our approach user input is only needed during the capturing of the images, the calibration procedure itself can be done automatically.

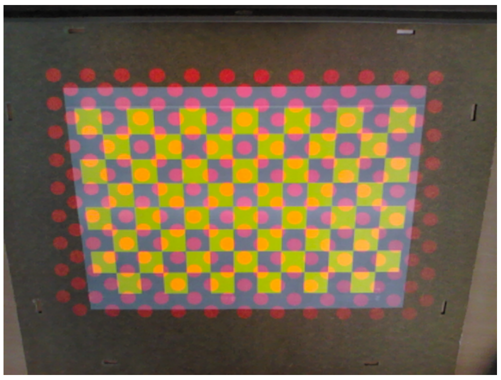

Figure 3: Calibration pattern using different colour mappings: A printed, yellow-grey chessboard pattern and a projected, red dot pattern.

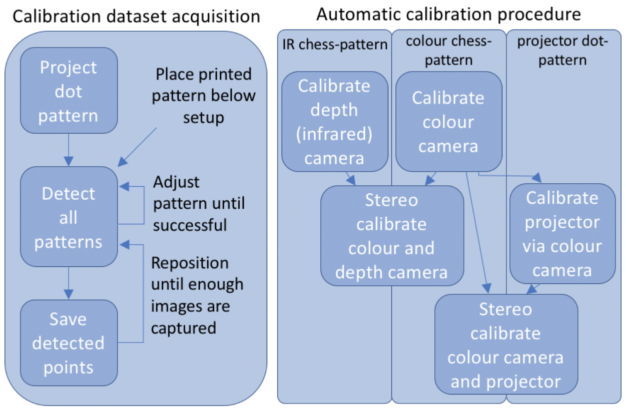
Calibration dataset acquisition
Project dot pattern
Place printed pattern below setup
Detect all patterns
Adjust pattern until successful
Reposition until enough images are captured
Save detected points
Automatic calibration procedure
IR chess-pattern
colour chess-pattern
projector dot-pattern
Calibrate depth (infrared) camera
Calibrate colour camera
Stereo calibrate colour and depth camera
Calibrate projector via colour camera
Stereo calibrate colour camera and projector

Figure 4: Calibration outline. The dataset acquisition can be done with a static setup, independently of the calibration procedure itself.

3.1 Distinguishing between colour channels

A colour image usually consists of three different channels: red, green and blue. We will be using those colour channels to distinguish between two overlapping patterns in one image. To avoid further interferences between the patterns and to allow easy automatic detection, we are using two different patterns: a simple, printed chessboard pattern for the camera calibration (both colour and depth) and a dot pattern for the projector.

In theory, we are setting up the projector pattern in a chosen colour (in our case red), that can solely be seen in the colour camera from the corresponding colour channel (also red). We choose one of the remaining colour channels (in our case blue) and use its complementary colour (yellow) as the printed camera pattern's colour. Thus, in that channel, the printed pigment filters out all blue of the background lighting, resulting in dark squares that can be detected, but still lets the red of the projected pattern in the respective channel pass.

However, the blue squares still influence the "projector channel" (red), since the primaries of the colour camera do not exactly match the filter characteristics of the pigments. Therefore, the background of the pattern has to be evened out by a neutral colour to match the brightness of the printed pattern's colour. We chose a matching grey with dithered black pigments that are used to filter IR light for calibrating the depth camera. Finally, this procedure leads to a clear division of the projected and the printed pattern in two different colour channels.

While this method works quite well under specific circumstances (e.g., even, white background light, no shadows, etc.), the projected pattern usually cannot be detected when the background light is too bright, i.e. when there is too little contrast in the red channel. However, as this procedure only has to be done once per setup, the final usage will not be affected by these restrictions.

Specifically, in our application, the projector uses RGB colour space and therefore directly maps into the camera's RGB colour space. Small divergences in the primaries (e.g. the projector pattern is still slightly visible in other channels) are possible – those will be dealt with later on. The printed pattern relies on the CMYK colour space. Since this is a subtractive colour space, it "removes" one part of the RGB colours, letting the other two colours pass. The black print colour K is the only one detected by the depth camera because all the other pigments do not affect the infrared light and let it pass.

While using the *Sprout*'s built-in camera, an *Intel RealSense F200 3D Camera*, consisting of a depth and a colour camera, another problem occurred: The colour camera only provides its output encoded in the lossy YUY2. This pixel format belongs to the YUV family[2] and has a brightness value for each pixel, but colour information only for a pair of pixels in the form of U and V. Those are the colour-difference signals[3] of blue minus brightness ($U = B - Y$) and red minus brightness ($V = R - Y$). Simply converting this format to the RGB colour space results in mixing up the different signals. For this reason, the undecoded U and V signals are used in our implementation.

The green channel of the colour camera is contaminated by red and blue due to the YUY2 colour model and therefore is not able to properly divide the input colours, so we are operating on the two channels left. The red primaries of the projector and the camera matched better than the blue ones. This means that when projecting a pattern in blue, parts of it are still clearly visible in the red camera signal, while the red projections are hardly detectable in the blue camera signal. Consequently, a red pattern was the best decision for the projector. The projected pattern has to be considerably brighter than the backlight of the scene to enable a proper differentiation between the patterns in the application, see Figure 5 left.

[2]http://www.fourcc.org/pixel-format/yuv-yuy2/
[3]https://www.pcmag.com/encyclopedia/term/55165/yuv

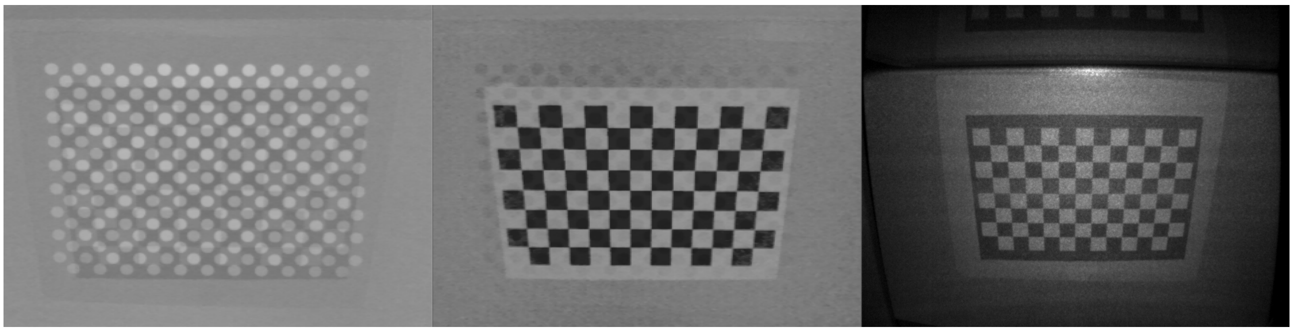

Figure 5: Different patterns, from left to right: The dot-projector pattern (red channel from the colour camera), the chessboard pattern as seen from the colour camera (blue channel) and as seen from the depth (infrared) camera.

By using a yellow pigmented printed pattern and a white background light, the blue from the colour camera is filtered in the yellow areas, while the grey background is detected. Hence the yellow parts appear darker and therefore are easy to distinguish from the background, see Figure 5 middle. Also, the blue camera signal is visibly darker than the other channels, so it produces the least interferences with them. Moreover, the grey colour of the printed pattern is matched with the yellow brightness, to appear approximately the same in the red signal, which also allows a better differentiation for the projected pattern.

In the depth camera, the yellow pigments let the infrared laser pass, while the grey background stands out clearly, due to its black pigments, see Figure 5 right.

3.2 Colour camera

The colour camera calibration operates on the blue channel of its input. A simple algorithm to detect the chessboard pattern's intersections is robust enough, so that no adaptations to the input are necessary, even if there are still some circles visible in the image, see Figure 6.

In order to calibrate a camera, a set of images must be captured by that camera, containing feature points of a pattern. The pattern can easily be printed by a colour laser printer and should be mounted on a planar surface throughout the process. Multiple images of the pattern have to be taken from different views. The points of the used pattern then need to be detected.

Once a pattern has been found in multiple images (preferably at least ten good snapshots) the calibration algorithm by Zhang [12] can be applied to approximate a 3×3 camera matrix and its distortion coefficients.

These are the intrinsic camera parameters, which is a mathematical model, that describes the projection from a 3D point to the camera sensor. They will, later on, be used to define the relief's position in the colour camera space.

3.3 Depth camera

In our application, we will be using the depth camera for the pose registration of a relief and to detect hand gestures. The camera provides a depth image (values of the distance of each pixel to the camera) and an infrared image. We will solely be using the infrared output to calibrate the camera. In this image, no interferences of the projector pattern are visible, because the projector does not transmit infrared light and the printed colour pigments do not filter the infrared light, except for the black pigment used for the grey background. The inverted image is used for the chessboard corner detection. The results can be seen in Figure 7.

All other calibration steps are equal to those of the colour camera, as in Section 3.2. In our use case, it has to be kept in mind that the infrared camera covers a bigger area than the colour camera, so in order to achieve a reasonably good calibration, the outer areas have to be covered by the pattern with additional images that are only used for the depth camera calibration.

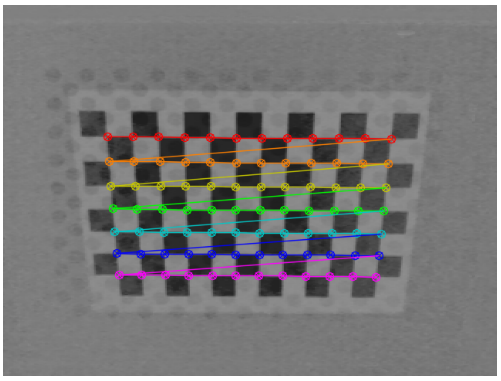

Figure 6: Chessboard pattern detected in the colour camera.

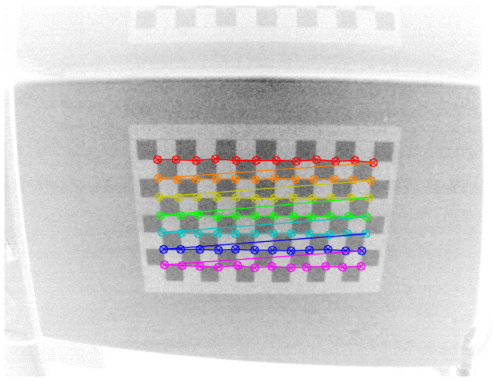

Figure 7: Chessboard pattern detected in the (inverted) depth/infrared camera.

3.4 Stereo calibration

In order to be able to determine the location of an object, detected by the depth camera, in the colour camera space, the transformation, i.e., the extrinsic parameters of one camera relative to the other, must be found. These extrinsic camera parameters consist of a rotation matrix and a translation vector.

Again, a pattern must be detected, but to calibrate two cameras simultaneously the same pattern has to be visible and detected in both camera views at the same time. Here the procedure of Bouguet [2] is used. It estimates the rotation matrix and the translation vector between the first and the second camera coordinate system by minimizing the total re-projection error for all points in all views.

3.5 Projector

In our application, we are using the projector to highlight specific areas of our 3D object (relief). To achieve this, the pose of the object has to be determined in projector space. The projector calibration relies on the red channel of the colour camera. Since the contrast of the red dots with respect to the background can be very low, a few adaptations have to be made to the images before applying a grid circle detection algorithm.

The detection is only successful on very clear input, therefore it has to be filtered beforehand. We implemented a local adaptive contrast enhancement to balance the backlight and the projector. The average local brightness is approximated with a Gaussian filter, which is then subtracted from the original image in order to maximize the contrast. A scaling of min-max to 0-255 increases the contrast as well. The result with the detected points can be seen in Figure 8.

We calibrate the projector using a camera model. This is reasonable since it basically performs the task of a camera "backwards" – while a camera captures an image, a projector displays something. Due to this, the calibration pattern in projector space always stays the same, it only changes on the board. Since we are using an algorithm for cameras, the detected points from the dot pattern need to be transformed from camera space to the calibration board space, more specifically to the plane they are being projected onto. This plane is in the same space as the chessboard pattern, therefore the transformational components of the chessboard can be used. This calibration procedure is based on the work by Cassinelli and Bergström [3]. Before starting the calibration process for the projector, the camera should have been already calibrated, as described in Section 3.2.

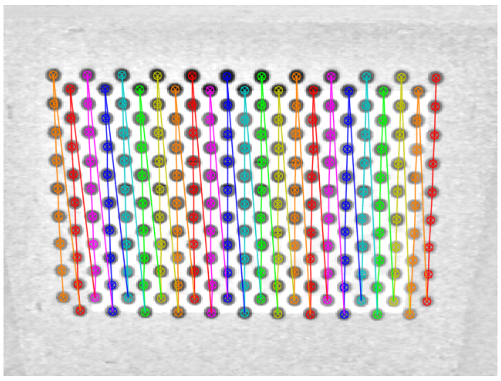
Figure 8: Projector dot pattern detected in the colour camera.

The projector points in board space (in the following referred to as projector board points) can be used as the input to a standard camera calibration procedure [12]. This will find the intrinsic camera parameters for the projector.

To get the pose (the extrinsic parameters) of the projector a stereo calibration procedure [2] can easily be applied to the projector board points and the points of the projector pattern as seen by the camera (simply the detected points in the captured images). This results in the rotation matrix and the translation vector between the two coordinate systems.

To determine the projector board points, the actual projected points in each image must be "back-projected" into board space.

A 2D point on the camera plane can be transformed into a 3D ray in board space in the form of

$$\mathbf{p} = \mathbf{f} + k \cdot \mathbf{d} \tag{1}$$

where $\mathbf{p}$ is a point on that ray, $\mathbf{f}$ its starting point, $\mathbf{d}$ the direction and $k > 0$. In our case, $\mathbf{f}$ is the camera focal point in board-coordinates. The direction $\mathbf{d}$ is given through each projected point's position. We are only looking for a specific point – the intersection of the ray and the planar surface of the calibration board. Therefore, the z-coordinate of the point $\mathbf{p}$ is always 0 in our board coordinate system.

In detail, the projection of a camera point onto the board can be calculated as

$$\begin{aligned} \mathbf{f_b} &= \mathbf{R_b}^{-1} \cdot -\mathbf{t_b} \\ \mathbf{d_p} &= \mathbf{R_b}^{-1} \cdot \mathbf{K}^{-1} \cdot \begin{pmatrix} u_p \\ v_p \\ 1 \end{pmatrix} \\ \begin{pmatrix} x_p \\ y_p \\ 0 \end{pmatrix} &= -\frac{\mathbf{f_b}_z}{\mathbf{d_p}_z} \cdot \mathbf{d_p} + \mathbf{f_b} \end{aligned} \tag{2}$$

where $\mathbf{R_b}$ is the rotation matrix and $\mathbf{t_b}$ the translation vector from the current board b to the camera coordinate system, $\mathbf{K}$ stands for the camera matrix of the camera, u_p and v_p are the coordinates of a single projected point on board b as seen from the camera.

4 Point cloud matching

In order to find the pose of our relief, which changes every time the relief is moved below our setup, we are using only the input of the depth camera. A known 3D object's pose with respect to the depth camera can be found with a point cloud matching algorithm. By using the calibration information we obtained before, see Section 3, that pose can then be determined in every used device's space.

The point cloud matching algorithm used in this implementation is based on the method by Drost et al. [4, 5]. It estimates the position of a known object using 3D features of point pairs. The first and the second point of the pair are denoted as $\mathbf{p_1}$ and $\mathbf{p_2}$ and their respective normal vectors $\mathbf{n_1}$ and $\mathbf{n_2}$. The normal vectors are calculated from the depth image by fitting a plane to all points in a radius r so that the summed squared distance to all points is minimized.

A Point Pair Feature (PPF) consists of the following values:

$\angle(\mathbf{n_1}, \overrightarrow{\mathbf{p_1 p_2}})$	The angle between the normal of $\mathbf{p_1}$ and the vector from $\mathbf{p_1}$ to $\mathbf{p_2}$
$\angle(\mathbf{n_2}, \overrightarrow{\mathbf{p_1 p_2}})$	The angle between the normal of $\mathbf{p_2}$ and the vector from $\mathbf{p_1}$ to $\mathbf{p_2}$
$\angle(\mathbf{n_1}, \mathbf{n_2})$	The angle between the normal of $\mathbf{p_1}$ and the normal of $\mathbf{p_2}$
$\lvert\overrightarrow{\mathbf{p_1 p_2}}\rvert$	The length of the vector from $\mathbf{p_1}$ to $\mathbf{p_2}$

First, a training of the known object (the object that should be detected) must be performed. For every pair of points closer together than a specified maximum radius r_{max} the PPF is calculated. By translating the pair so that $\mathbf{p_1}$ will lie in the origin and rotating $\mathbf{n_1}$ onto the x-axis, the angle α_t can be defined as the angle in the yz-plane. These values are stored in a hash table, with the hash value computed from the four PPF values. If more than 5 entries are added to the same hash, we deem these as too undescriptive and ignore them from further computations.

During the matching phase (which has to be initiated every time the relief is moved) we iterate over the 3D scene from the depth camera. For every pair of points in the same radius r_{max} we again calculate the PPFs, as well as the angle α_m the same way as during the training. We compute the hash value of each current PPF and use this as the key to get all similar PPFs from the hash table which contains the trained values. Then we compare the current α_m to each of the trained pairs' α_t, as in $\alpha_{diff} = \alpha_m - \alpha_t$. The angle α_{diff} describes the rotation around the x-axis from the scene point pair to the trained pair (see Figure 9).

We are basically using this to break down the relative pose between two PPFs consisting of 6 dimensions into a single one, in order to simplify the matching process. So, during the comparison both $\mathbf{p_1}$ points are being moved into the origin, therefore the three translation dimensions are fixed. Moreover, by rotating both $\mathbf{n_1}$ normal vectors onto the x-axis and only considering the angle in the yz-plane, this is the only rotational dimension left. Therefore, α_{diff} describes the rotational component of the object's pose in a single value.

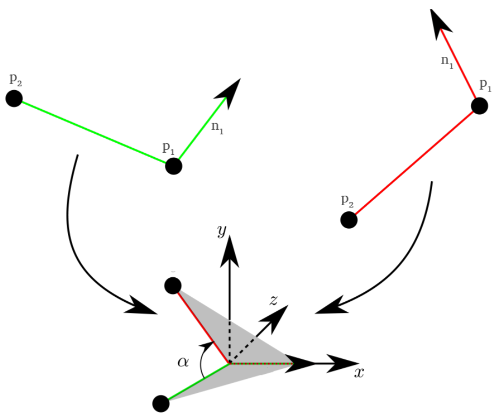

Figure 9: Angle α_{diff}. Taken and adapted from [5]

A Hough-like voting scheme is performed over the first point of every feature pair in the hash table and α_{diff}. For every point the pose with the largest number of votes is taken into account – these are the optimal local coordinates. In the end, a clustering algorithm is applied to all poses in order to find the final pose with the most point pair matches, assuming that this is the right solution.

Once the position of the object is found, the outlines of it will be displayed by our application on the screen on top of the infrared view. This is done by projecting predefined labels, for example as seen in Figure 10, which are loaded from a PNG-file, and rendered into the camera space. Additionally, by transforming the labels into the projector space, the projector can highlight the labels on the object below.

The training step is performed on the 3D object beforehand. Since the created data can get very large, about 500 MB for an approximately 40×40 cm sized relief, it is more efficient to start the training at start-up than to store and load the trained data explicitly. Then during the runtime of our application, the second phase of the algorithm is executed to determine the object's current pose. The depth values provided by the camera are averaged over the last five frames to avoid noise or irregularities and get undistorted with use of its intrinsic parameters. A bigger number of frames to average did not crucially improve the quality, but reduces the likeliness of future real-time usage, so five frames seemed a good trade-off. By using a relatively large radius $r = 4$px for the plane fitting, we generate mostly robust normals, which proved sufficient for our uses.

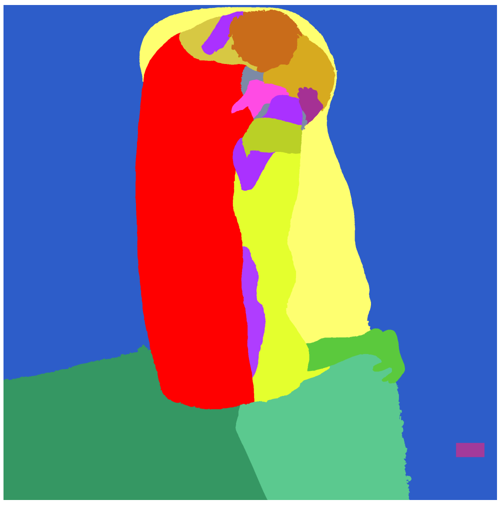
Figure 10: Labels of the painting “Der Kuss”.

5 Results

Our application runs on a *Sprout Pro by HP*[4] with an *Intel Core i7-6700*. The colour and depth camera are combined in the *Intel RealSense 3D Camera F200* and both have been used at a resolution of 640×480 pixels for all results. Moreover, it has a built-in DLP Projector with 1280×800 pixels.

The capturing of the calibration images including pattern detection, as described in Section 3, can be done in real time. The automatic calibration process takes about 10 - 15 seconds for ca. 20 images. The results from the calibration are reported by our system in terms of the RMS re-projection error. While the calibration of the colour camera returns very low errors, usually below 0.5 pixels, higher values are reached with the depth camera, in most cases about 1.0 pixels. This deviation still proved sufficient for our uses. Due to the difficulty of capturing the projected calibration pattern in many varying positions the re-projection error of the projector is worse, approximately around 1.5 pixels. The re-projection errors for the stereo calibrations lie between 1.0 and 1.5 pixels. All in all, this seems to be a sufficiently accurate solution for our use case.

While the training phase can take up to 10 seconds for a 40×40 cm sized relief, the matching usually can be done in less than 5 seconds. This seemed sufficient since it only has to be carried out when the relief is moved. The point cloud matching algorithm leads to an accurate solution at least once in three matching tries. Since the projector will visually mark the object’s pose, errors can easily be detected by the human eye and the matching can be restarted. A correctly detected pose, with projected labels, can be seen in Figure 11. Due to the large, mostly planar regions of the model, most point pair matches are detected in that area – even with the removal of redundant features in the hash table – which can easily outnumber actual significant matches, therefore leading to smaller inaccuracies as in Figure 12.

[4]`http://www8.hp.com/h20195/v2/getpdf.aspx/c04920623.pdf?ver=2`

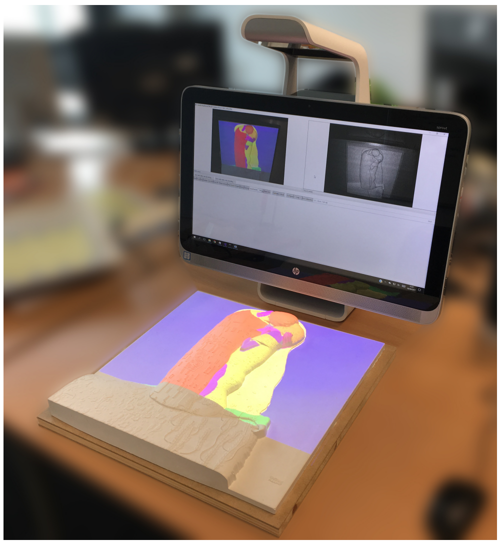
Figure 11: Correctly detected solution and labels projected onto the object.

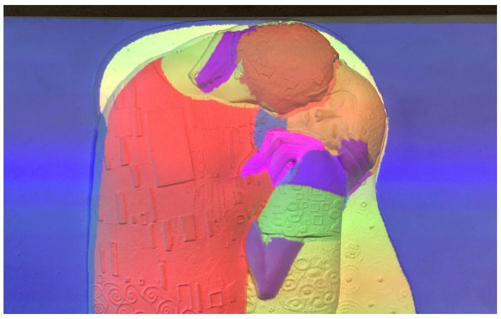
Figure 12: Minor errors during the point cloud matching phase.

6 Conclusions and Future Work

In this paper, a method to calibrate a setup consisting of four components (colour camera, depth camera, projector, 3D object) was introduced. While the depth camera can determine the pose of a given object and is used for hand gesture detection, the projector is set up to project onto the object to illustrate different regions of the object. The calibration procedure is able to simultaneously calibrate all three imaging devices, with the use of different colour patterns, one printed and one projected. While the colour and the depth camera get calibrated with a printed pattern, a projected pattern as seen by the colour camera gets transformed into the space of the printed pattern’s board and can then be used for the calibration of the projector. The point cloud matching algorithm relies on simple features between a pair of points and determines the best position through choosing the locally most likely result and clusters the options at last.

While the calibration for the colour camera and the projector leads to very high accuracy, the process for the depth camera still needs some improvement, due to noise and other irregularities. An option to improve the quality of the depth camera measurements would be to introduce a deviation model to calibrate the depth values of the camera.

To improve all different calibration steps, images with high re-projection error could automatically be detected and ignored in the following computations. This would lead to an improvement of the total error, even though bad images can currently be sorted out by hand.

Moreover, the point cloud matching algorithm relies on plain simple features that can easily lead to incorrect solutions, especially when the quality of the provided depth image is low, which is the case in this application. While changing to a different matching algorithm that is more immune against errors could solve the problem, another option would be to take information from the colour camera into account. Most objects used in this project are single coloured and the background usually is the plate from a desk and therefore has a single colour as well. With the use of colour segmentation, the approximate 2D position on the desk could be estimated and the algorithm can be improved, with the new information obtained.

Acknowledgements

I want to thank Andreas Reichinger from VRVis and Bernhard Steiner from TU Wien for their help and feedback.

This work was supported by VRVis, funded by BMVIT, BMWFW, Styria, SFG and Vienna Business Agency in the scope of COMET - Competence Centers for Excellent Technologies (854174) managed by FFG and by the H2020 project ARCHES which has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 693229.

References

[1] Samuel Audet and Masatoshi Okutomi. A user-friendly method to geometrically calibrate projector-camera systems. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, pages 47–54. IEEE, 2009.

[2] Jean-Yves Bouguet. Camera calibration tool-box for matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/`, 2002. Last visited 06.04.2018.

[3] Alvaro Cassinelli and Niklas Bergström. *Easy "Camera + Projector" Calibration (OF addon).* `https://www.youtube.com/watch?v=pCq7u2TvlxU`, Jul 2012. Last visited 24.10.2017.

[4] Bertram Drost and Slobodan Ilic. 3d object detection and localization using multimodal point pair features. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, pages 9–16. IEEE, 2012.

[5] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 998–1005. Ieee, 2010.

[6] Janne Heikkila. Geometric camera calibration using circular control points. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(10):1066–1077, 2000.

[7] Jing Huang and Suya You. Point cloud matching based on 3d self-similarity. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 41–48. IEEE, 2012.

[8] Marc Pollefeys, Reinhard Koch, and Luc Van Gool. Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters. *Int. J. Comput. Vis.*, 32(1):7–25, 1999.

[9] Andreas Reichinger, Helena Garcia Carrizosa, Joanna Wood, Svenja Schröder, Christian Löw, Laura Rosalia Luidolt, Maria Schimkowitsch, Anton Fuhrmann, Stefan Maierhofer, and Werner Purgathofer. Pictures in your mind: Using interactive gesture-controlled reliefs to explore art. *ACM Trans. Access. Comput.*, pages 2:1–2:39, 03 2018.

[10] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Aligning point cloud views using persistent feature histograms. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3384–3391. IEEE, 2008.

[11] Song Zhang and Peisen S Huang. Novel method for structured light system calibration. *Optical Engineering*, 45(8):083601–083601, 2006.

[12] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1330–1334, 2000.

[13] Zijian Zhao, Yuncai Liu, and Zhengyou Zhang. Camera calibration with three noncollinear points under special motions. *IEEE Trans. Image Process.*, 17(12):2393–2402, 2008.