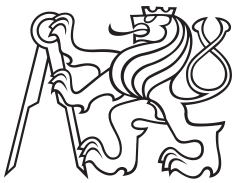
Bachelor Project



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Two-Body Structure from Motion

Petr Hrubý

Supervisor: doc. Ing. Tomáš Pajdla, Ph.D.
Field of study: Open Informatics
Subfield: Computer and Informatic Science
May 2019

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Hrubý Petr** Personal ID number: **466292**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Branch of study: **Computer and Information Science**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Two-Body Structure from Motion

Bachelor's thesis title in Czech:

Trojdimenzionální rekonstrukce dvou pohybujících se těles

Guidelines:

1. Review the state of the art in Structure from Motion (SfM) as well as in Multi-Body SfM [1-7].
2. Propose a method for reconstructing scenes consisting of a single moving object in front of a static background captured by multiple image sequences. Consider the scenario when each image sequence captures the object and the background in a static configuration.
3. Implement the method as an extension of the COLMAP [2] SfM pipeline.
4. Demonstrate and evaluate the method on real data.

Bibliography / sources:

[1] J Schonberger and J-M Frahm. Structure-from-Motion Revisited. CVPR 2016.
[2] COLMAP https://colmap.github.io/
[3] K Schindler and D Suter. Two-view multibody structure-and-motion with outliers through model selection. IEEE Transactions on Pat-tern Analysis and Machine Intelligence, 28(6):983–995, 2006.
[4] K Schindler, D Suter, H Wang. A model-selection framework for multibody structure-and-motion of image sequences. International Journal of Computer Vision, 79(2):159–177, 2008.
[5] X Xu, L-F Cheong, Z Li. Motion Segmentation by Exploiting Complementary Geometric Models. CVPR 2018
[6] L Magri, A Fusiello. Robust multiple model fitting with preference analysis and low-rank approximation. BMVC 2015
[7] E Maset, F Arrigoni, A Fusiello. Practical and efficient multi-view matching. ICCV 2017.

Name and workplace of bachelor's thesis supervisor:

doc. Ing. Tomáš Pajdla, Ph.D., Applied Algebra and Geometry, CIIRC

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **04.10.2018** Deadline for bachelor thesis submission: **24.05.2019**

Assignment valid until: **30.09.2020**

doc. Ing. Tomáš Pajdla, Ph.D.	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Ing. Pavel Ripka, CSc.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

. _____
Date of assignment receipt

_____
Student's signature

Acknowledgements

This research was supported by IMPACT CZ.02.1.01/0.0/0.0/15_003/0000468 EU Structural and Investment Funds, Operational Programe Research, Development and Education. I would like to express my thanks to my supervisor doc. Ing. Tomáš Pajdla, Ph.D. for his guidance and his valuable advice. I would like to thank Ing. Michal Polic, Ing. Stanislav Steidl and Federica Arrigoni, Ph.D. for their advice. I would like to thank my family for their support.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 24. May 2019

Abstract

Multi-Body Structure from Motion (MBSfM) is a problem of 3D reconstruction of objects in a scene, which is not static. The MBSfM problem has not yet been solved in general. We propose a solution to a relaxed version of the problem, where the images depict several static configurations of the scene and the number of the objects is limited to two.

We show in this thesis that this relaxed MBSfM has a practical usage because in some cases the reconstruction of an object can be stabilized, using a background. When the object is placed on the background and multiple sequences of different configurations of the scene are taken, the object can be reconstructed and segmented from the background with the method proposed in this thesis.

Keywords: structure from motion , multi-body, two-body, 3D reconstruction, SfM, MBSfM, dynamic scene

Supervisor: doc. Ing. Tomáš Pajdla, Ph.D.
Aplikovaná algebra a geometrie CIIRC,
Jugoslávských partyzánů 1580/3,
Praha 6

Abstrakt

Trojdimenzionální rekonstrukce více pohybujících se těles (MBSfM) je problém rekonstrukce scény, která není statická. Tento problém zatím nemá obecné řešení. V této práci navrhujeme řešení relaxované verze tohoto problému, kde je scéna zachycena v několika statických konfiguracích a počet objektů je omezen na 2.

Dále v této práci ukážeme, že řešení tohoto relaxovaného problému má praktické využití, protože v některých případech může být trojdimenzionální rekonstrukce objektu stabilizována přítomností pozadí ve scéně. Pokud je objekt umístěn na pozadí a tato scéna je vyfotografována v několika statických konfiguracích, může být tento objekt zrekonstuován metodou popsanou v této práci.

Klíčová slova: struktura z pohybu, dvě tělesa, 3D rekonstrukce, SfM, MBSfM, dynamická scéna

Překlad názvu: Trojdimenzionální rekonstrukce dvou pohybujících se těles

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Motivation

Structure from Motion [1] is a problem of reconstruction of a 3D model of a scene from a set of images which depict this scene. The case, where the scene is static, is well understood and implemented in many pipelines, such as [2] [3] [4].

Multi-Body Structure from Motion (MBSfM) is the case of a dynamic scene, where the objects may move between the images. This problem has on the other hand not yet been solved in general. We have relaxed the problem in such way that the number of the moving objects is limited to 2 and the images depict several static configurations of the scene. The objects may move between the different configurations.

The motivation behind this task is that a hypothesis exists, that the presence of a background in a scene may improve the quality of a 3D reconstruction of an object. In this case, one of the objects is the background and the other one is the object which we want to reconstruct.

The task described in this section would be able to reconstruct the object together with the background and to separate the background from the object. Our intention was to perform a couple of experiments, which would prove the

functionality of the method and show for which objects the method improves the results of the reconstruction.

1.2 Thesis structure

In Chapter 1 the problem is formulated. In Chapter 2 some important concepts, such as representation of rotation in 3D or representation and registering of the cameras, are introduced. In Chapter 3 the State of the Art in the Structure from Motion and in motion segmentation is reviewed. A solution to the problem, which is introduced in Section 1.3, is proposed In Chapter 4. This solution is evaluated on real data in Chapter 5. In the same chapter we compare the models reconstructed by our method with the models reconstructed by the State of the Art single body pipeline.

1.3 Problem formulation

The scene consists of a single object on a static background, and it has been captured by multiple image sequences (takes). Each of the takes captures the scene in a static configuration and the object moves between the takes. The task is to reconstruct the object as well as the background and to segment the points belonging to the object and to the background.

take	a sequence of images of a static configuration of the scene
k	number of takes
$(A_{i,j}, \overrightarrow{b}_{i,j})$	motion of the object between the configurations captured by the takes i, j
X_B^i	a point from the background in the configuration captured by take i, the background is static, therefore $\forall i \forall j : X_B^i = X_B^j$
X_O^i	a point from the object in the configuration captured by take i, the object moves according to $(A_{i,j}, \overrightarrow{b}_{i,j})$, so $\forall j : X_O^j = A_{i,j} X_O^i + \overrightarrow{b}_{i,j}$
$\{I_i\}_{i=1}^n$	a sequence of images of the scene
t_i	a take to which the image I_i belongs

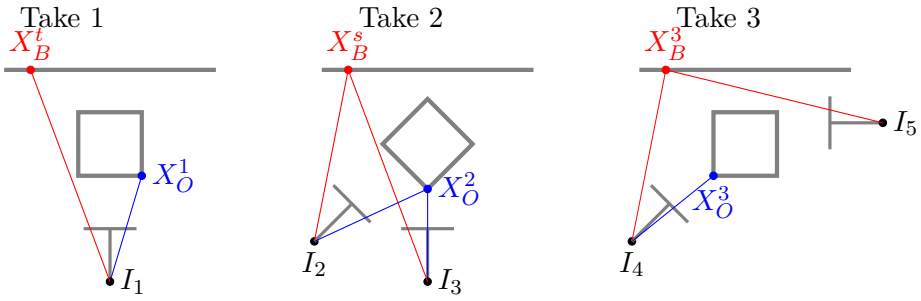


Figure 1.1: An example of the input scene; $k = 3$, $n = 5$, $t_1 = 1, t_2 = 2, t_3 = 2, t_4 = 3, t_5 = 3$, the point is connected with the camera if the camera observes the point

1.4 Contributions

In this thesis, we propose a solution to the problem where two objects are captured in several static configurations and the task is to reconstruct both objects and to distinguish between them. Usually, one of the objects is a background and the second one is the object which we want to reconstruct. To our best knowledge there is no algorithm which solves this particular problem. The concepts introduced in this thesis can be used in a more general version of Multi-Body SfM.

We also demonstrate on real data, that in some cases the reconstruction of an object with a background using our method has better properties than the reconstruction of the same object without the background using a State of the Art method. We also show in the thesis that models of some objects, which the State of the Art method cannot reconstruct, can be reconstructed using our method.

Chapter 2

Frequently used concepts

2.1 Representation of rotation in 3D

Different representations of rotation in space exist, each of which has its advantages and disadvantages. Transformations between the representations can be found in [5].

Rotation matrix. As the rotation of vectors in $\mathbb{R}^3$ is a linear transformation, it can be represented by a matrix $R \in \mathbb{M}^{3,3}$. R is a full rank orthogonal matrix with determinant equal to 1. Representation by the matrix is suitable for operations with the rotations, as the vector $\overrightarrow{x}$ can be rotated easily performing $R\overrightarrow{x}$. Compound rotation can as well be easily obtained by multiplication of the rotation matrices. This representation is however not suitable for clustering, storing and viewing, as 9 numbers are necessary to determine the matrix.

Euler vector. (or angle-axis representation) is a vector $\overrightarrow{e} \in \mathbb{R}^3$. The direction of the vector is the axis of the rotation and the norm of the vector is the angle of the rotation in radians. This is the minimal representation of the rotation and is therefore suitable for clustering and storing of the rotations, as well as for the viewing of the vectors because they belong to the 3D space. It is however not suitable for operations with the rotations as they are more complicated with the Euler vectors than with the rotation matrices.

2.1.1 Norm of the rotation

The norm of the rotation is the angle of the rotation. It can, therefore, be computed as the norm $||\overrightarrow{e}||$ of the Euler vector which represents the rotation.

2.1.2 Distance between two rotations

There are different ways to compute the distance between two rotations R_1, R_2. We compute it as the norm of the rotation $R_3 = R_2^{-1}R_1$. In order to do that the rotation R_3 has to be converted to the Euler vector form.

2.2 Camera representation

A pinhole camera model [1] is a model of a camera consisting of the camera centre $\overrightarrow{c}$ and the projective plane π onto which the points are projected.

f	focal length of the camera; the distance between the centre $\overrightarrow{c}$ and the projective plane π
$(\overrightarrow{o}, \alpha)$	the image coordinate system; $\overrightarrow{o}$ is the origin of the projective plane (therefore $\overrightarrow{o} \in \sigma$), α is the basis of the plane, for standard cameras α is orthonormal but it is not a condition
$(\overrightarrow{c}, \beta)$	the camera coordinate system; the centre of the system is the centre of the camera $\overrightarrow{c}$, the first two vectors from the basis β are identical with the basis α and the last one is the vector $\overrightarrow{o} - \overrightarrow{c}$.
$(\overrightarrow{O}, \delta)$	the world coordinate system; $\overrightarrow{O}$ is a
X_δ	a 3D-point in the world coordinate system
X_δ	the point X_δ in the camera coordinate system
p	a projection ray; a line connecting the point X with the camera centre C
$\overrightarrow{x}$	a 2D point in the image coordinate system which is the projection of the point X_δ

The projection of a point X_δ in the world coordinate system is realized in two steps. At first, the point is transformed to the camera coordinate system, then the projection is found as an intersection of the ray p with the projective plane π. The projection is then represented in the image coordinate system.

Transformation of the point X_δ in the world coordinate system to the point X_β in the camera coordinate system is an affine transformation $\mathbb{R}^3 \to \mathbb{R}^3$, it can, therefore, be performed by multiplication of a matrix $P \in \mathbb{M}^{3,4}$ by homogeneous coordinates of the point X. The matrix P is the camera projection matrix and it can represent the camera.

$$X_\beta = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = P \begin{bmatrix} X_\delta \\ 1 \end{bmatrix} \tag{2.1}$$

The form of the basis of the camera coordinate system β ensures that the 3D-points on the projective plane π have the third coordinate equal to 1 and the first two coordinates are equal to the representation of the point in the image system $(\overrightarrow{o}, \alpha)$, so the coordinates are the homogeneous coordinates of the representation of the 2D-point in the image system.

The intersection of the ray p with the plane π can be obtained by multiplication of the point X_β by a scalar σ such, that the point σX_β is on the projective plane π. Because of the property of the system $(\overrightarrow{c}, \beta)$ is the scalar σ equal to z:

$$\overrightarrow{x} = \begin{bmatrix} \frac{x_1}{x_3} \\ \frac{x_2}{x_3} \end{bmatrix} \tag{2.2}$$

The transformation of the points from the coordinate system can be decomposed into the transformation to the coordinate system with the orthonormal basis whose last vector is orthogonal to the projective plane and with the center in the point $\overrightarrow{c}$, the camera matrix $\overrightarrow{c}$ can, therefore, be decomposed into:

$$P = KR\left[I| - \overrightarrow{c}\right] \tag{2.3}$$

, where K is the camera calibration matrix [1], R is the rotation of the camera and $\overrightarrow{c}$ is the centre of the camera.

2.3 PnP

PnP is a procedure which registers a camera onto a set of 3D points with n correspondences between the 3D points and the 2D features on the image [1].

The simplest type of the PnP algorithm is P6P which requires 6 points to compute the matrix P. The camera matrix P is stacked into a vector $\overrightarrow{P}$ which is used as the vector of unknowns. Each correspondence between a point and a feature generates two linear equations which are derived from equations (2.1), (2.2), one equation for each element of the feature $\overrightarrow{x}$.

Because the matrix P has 12 elements, at least 12 linearly independent equations are required to find it. These equations can be generated from 6 correspondences. More advanced versions of PnP exist which require fewer correspondences. If the camera is calibrated, only 3 correspondences are required to compute the pose.

Chapter 3

State of the art

3.1 Structure from Motion

Reconstruction of static scenes is a well-understood problem. Two main approaches to it exist, namely incremental SfM and global SfM. There are different SfM pipelines, such as Bundler [2], COLMAP [3] or OpenMVG[4]. Among these pipelines, we have chosen COLMAP as the base of our method. The most common approach is the incremental SfM, which consists of the following steps: feature extraction and matching, the initial pose calculation, camera registration, point triangulation, and bundle adjustment.

3.1.1 Feature extraction and matching

At first, features, such as SIFT [6], are found in each of the images. Corresponding features in a pair of images are matched using nearest neighbor search on the descriptors of the features. In the exhaustive matching, the matches are found between all pairs of images, therefore the matching has quadratic complexity. Other variants of matching exist, which have linear complexity and therefore are better suited for large datasets. Transitive matches (tracks) may be used in order to increase the number of the tracks. Maset, Arrigoni and Fussielo [7] use eigendecomposition to maximize consistency of the tracks.

Usually, many mismatches occur among the matches, so two-step filtering is applied. In the first step such matches are rejected, whose Lowe ratio [6] is too high. Lowe ratio is a ratio of the nearest neighbor and the second nearest neighbor in the space of the descriptors. The second step is geometric filtering. Fundamental matrix [1] is fitted to the matches using RANSAC [8] and only the inliers to the model are retained. In COLMAP [3] both the fundamental matrix and the homography are fitted to the data and selects the model which has more inliers.

3.1.2 Initial reconstruction

Among the images, an initial pair is selected. Fundamental matrix is fitted to the matches using RANSAC. The relative pose of the cameras from the initial pair is computed from the fundamental matrix [1]. With the knowledge about the relative pose of the two cameras, the points are triangulated from the matches.

3.1.3 Incremental reconstruction

As the initial reconstruction has been performed, positions of some 3D points are known. Correspondences between the 2D features and the 3D points are obtained from the matches between the features. A new camera is registered towards the existing point cloud using the PnP (Section 2.3). The newly found camera pose enables the triangulation of new points, so the point cloud grows and new cameras can be registered towards it. This procedure is repeated until there are non-registered cameras which can be registered towards the point cloud.

3.1.4 Bundle adjustment

Errors arise during the reconstruction and they accumulate, so they can easily become unsustainable. In order to prevent this, Bundle Adjustment[9], which minimizes the sum of squares of reprojection errors, is performed. Bundle Adjustment is performed after every step of the incremental reconstruction. In COLMAP [3], global bundle adjustment is performed only after the model grows beyond a given threshold, otherwise, a local bundle adjustment, which

improves only the limited number of cameras, is performed, so the time complexity of the reconstruction is better.

3.2 Motion segmentation

In the motion segmentation problem, motion tracks between two or more images are given, and the task is to split the tracks into groups that move together. The problem is strongly connected to the Multi-body Structure from Motion. If all motion groups were correctly determined, the MBSfM problem would be reduced to the static scene reconstruction. On the other hand, if the transformations of the cameras towards each of the motion groups were known, it would be easy to determine which point belongs to which group. The problem is that the segmentation and the models have to be found simultaneously. Different approaches to the problem exist.

3.2.1 Algebraic methods

One group of the algebraic methods is based on matrix factorization. These methods describe the motion segmentation as a subspace clustering problem. Among these methods belong for example [10] and [11], which perform the factorization via SVD.

R. Vidal et. al. [12] have proposed The Multibody Epipolar Geometry, where the multibody fundamental matrix is a sequence of fundamental matrices. A point which is inlier to this model fulfills the epipolar constraint for at least one of the fundamental matrices. The Multibody epipolar constraint leads to a set of polynomial equations, which is solved using GPCA. This solution however cannot be applied if erroneous matches occur. Rao et. al. [13] have proposed an algebraic method which is able to deal with the erroneous matches. The method is based on the merging of the points such that the coding length of the model is minimized using Agglomerative Lossy Compression. Rao et. al. have proposed a Hybrid Perspective Constraint [14], which can describe both planar and perspective motions.

3.2.2 Random sampling methods

These methods are based on drawing random minimal samples and computing models such as homography or fundamental matrix from the samples. The random sampling methods are not as elegant as the algebraic methods but they are more universal and usually perform better, especially if gross outliers are present. The methods can be divided into two groups: consensus analysis and preference analysis.

The simplest method from the consensus analysis group is the sequential RANSAC. [15] The model with the highest support is selected using RANSAC [8] and further motions are found from the outliers to this model. The problem of this method is that it greedily extracts the matches which belong to the object, so in the case of overlapping models, the matches may be assigned to a model to which they do not belong. Multi-RANSAC [16] finds the models simultaneously, rather than sequentially. K. Schindler and D. Suter [17] generate the random samples and after discarding the worst ones they select the actual motions with Taboo search in order to avoid the greedy approach.

Preference analysis is a method dual to the consensus analysis. Preference sets of the points are examined instead of the residuals of the models. W. Zhang and J. Kosecka [18] examine the histograms of the residuals of the points. Peaks in the histogram belong to the individual motions. J-Linkage [19] and T-Linkage [20] are methods which cluster the points in the space of their residuals using the linkage clustering. They differ in the distance which they use for the clustering.

In Random Cluster Model Simulated Annealing (RCMSA) [21] larger than minimal samples are used to compute the hypotheses. Graph-cut approach is used, where the problem is solved using the simulated annealing approach. In [22], L. Magri and A. Fusiello decompose the preference matrix (matrix whose columns are the residual vectors of the points) as the sum of a low-rank matrix and a sparse one (noise) using Robust PCA. The low rank matrix is then factorized using Nonnegative Matrix Factorization. In [23], X. Xu, L-F Cheong and Z. Li. find simultaneously different models such as fundamental matrix, homography and affinity. They use the fact that if two points belong to the same homography group, they belong to the same epipolar group, etc. The motions are clustered iteratively via Subset Constrained Multi-View Spectral Clustering.

Sometimes it is difficult to perform the segmentation, as degenerate situations exist where two independent motions can be explained with a single fundamental matrix.

3.2.3 Other methods

Another motion segmentation methods exist, which belong neither to the algebraic group nor to the random sampling one. These methods are based for example on the optical flow [24] or on an iterative approach inspired by EM [25]. After the rise of the neural networks, semantic segmentation methods have also been developed [26].

Growing homographies is an approach similar to the random sampling. From each match, an affine transformation is computed, which is subsequently refined to the form of a homography using inliers to the transformation. F. Šrajer [27] uses the Planar Homology concept to merge the homographies which have arisen from the same motion.

3.3 Reconstruction of dynamic scenes

The most of the motion segmentation methods concentrate on the segmentation between 2 images. K Schindler, D Suter and H Wang [28] have proposed a method for n-view MBSfM. They draw random samples for each pair of subsequent images and after filtering of the worst models they link the models to create chains, which describe the motion in the whole sequence. The chains are selected with the maximum likelihood criterion. The method is more suitable for videos than for general image sets. Method [13] works with a matrix of tracks, so it is able to work with n views.

Little has been done in the actual reconstruction of the dynamic scenes (MBSfM), so the problem still remains unsolved in general. YASFM [27] is a pipeline, which performs a motion segmentation and is, therefore, able to reconstruct multiple objects from the dynamic scenes. However, it is shown in [29], that YASFM is not able to reconstruct the scenes described in Section 1.3.

COLMAP [3] is not able to reconstruct dynamic scenes, however it contains tools for MBSfM, such as the matching which supports multiple models. The filtering of the matches is then performed with the sequential RANSAC.

Chapter 4

Proposed solution

4.1 Reconstruction of the takes

At first, the matches are found between each pair of images. The option `multiple_objects` is selected, so correct matches between points from both the object and the background are retained during the verification step. Each take is reconstructed individually in a standard Structure from Motion pipeline COLMAP [3]. The take from which the reconstruction has been obtained is the **anchor take** of the reconstruction. After a sparse point cloud is obtained from the anchor take, cameras from other takes are registered onto the point cloud via sequential PnP.

Sequential PnP is a procedure, where the camera is registered onto the points using a RANSAC PnP, and the outliers to the found camera pose are reused for further registration. This procedure is repeated until no pose with satisfactory support is found. Therefore it is possible that multiple camera poses are registered from a single camera.

RANSAC_PnP($\mathcal{M}, r$) is a procedure that performs registration of a camera using a set of 2D to 3D correspondences $\mathcal{M}$, the number of iterations of the RANSAC algorithm is r, the output of the procedure is the registered pose P. Find_Inliers(P, $\mathcal{M}$) outputs a subset $\mathcal{I} \subseteq \mathcal{M}$ of inliers to the pose P.

Algorithm 1: Sequential PnP

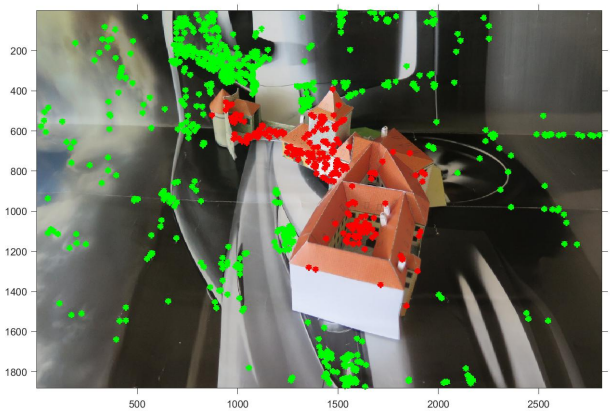
input : $\mathcal{M}$ a finite set of correspondences between 2D features and 3D points
m minimal number of inliers for a pose
r number of the iterations of RANSAC

output : $\mathcal{P}$ a finite set of camera matrices

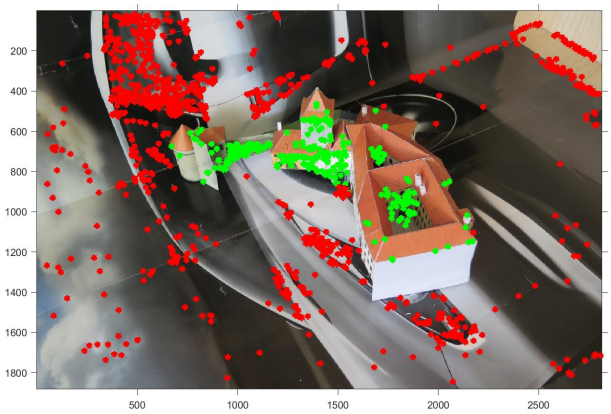
```
P ← ∅ ;
while |M| ≥ m do
    P = RANSAC_PnP(M, r);
    I ← Find_Inliers(P, M);
    if |I| ≥ m then
        P ← P ∪ {P};
        M ← M \ I;
    else
        break;
    end
end
```

The idea behind this is that cameras from takes different from the anchor take have different poses towards the static background and towards the object, therefore in the ideal case the sequential PnP should find one pose towards the background and another one towards the object. The points belonging to the object as well as those belonging to the background can be recognized as the inliers to these poses.

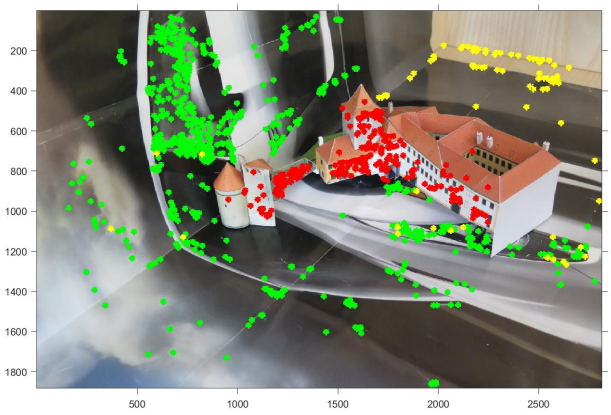
The motion of the object between the anchor take and the take to which the image belongs (the second take) can also be computed from the poses of the camera towards the background and the object which have been found using the sequential PnP.



(a)



(b)



(c)

Figure 4.1: Example of different results of the sequential PnP, points with the same color are observed by the same camera. The cameras have been registered in the order green, red, yellow.

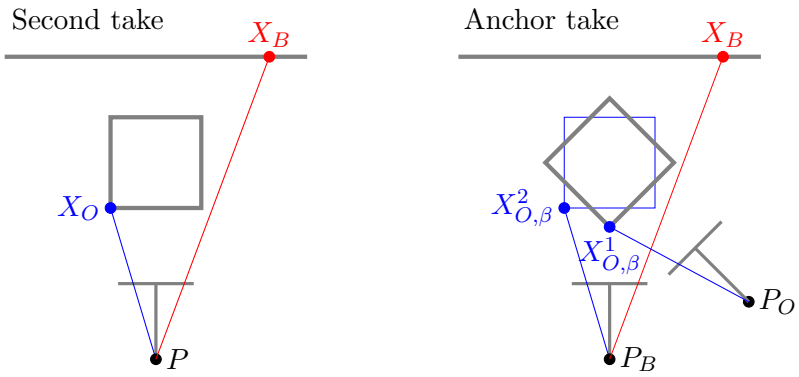


Figure 4.2: Example of a camera after the sequential PnP registration onto the anchor take.

4.2 Object motion calculation

X_O - arbitrary 3D point from the object
P - a camera from the second take which observes the point X_O
$\overrightarrow{x}$ - projection of the point X_O on the camera P
$X^1_{O,\beta}$ - reconstruction of the point P in the anchor take reconstruction
β - basis of the anchor take reconstruction
A - matrix of rotation of the object between the anchor take and the second take
$\overrightarrow{b}$ - translation of the object between the anchor take and the second take
$X^2_{O,\beta}$ - new position of the point $X^1_{O,\beta}$ after movement of the object from the anchor take configuration to the second take configuration; for every point it holds true: $X^2_{O,\beta} = AX^1_{O,\beta} + \overrightarrow{b}$
P_B - camera matrix of the camera P registered on the background points of the anchor take reconstruction
P_O - camera matrix of the camera P registered on the object points of the anchor take reconstruction

If the camera P from the second take is registered on the object points from the reconstruction of the anchor take, it projects the object points from the anchor take onto the same 2D points onto which they were projected in the configuration of the second take. Particularly this means that $\overrightarrow{x}$ is a projection of the point $X^1_{O,\beta}$ onto the camera P_O
If the camera P from the second take is registered on the background points from the reconstruction of the anchor take, its position is the same as it was in the second take. This means that if the object points were transformed to

the position in which they were in the second take, they would be projected onto the same 2D points onto which they were projected in the second take. This implies that $\overrightarrow{x}$ is a projection of the point $X^2_{O,\beta}$ onto the camera P_B.

The following equation must therefore hold true:

$$\sigma_1 P_O \begin{bmatrix} X^1_{O,\beta} \\ 1 \end{bmatrix} = \sigma_2 P_B \begin{bmatrix} X^2_{O,\beta} \\ 1 \end{bmatrix} \tag{4.1}$$

We can introduce new symbols for rotation, center and camera calibration matrix for cameras P_O and P_B and write this equation as

$$\sigma_1 K \begin{bmatrix} R_O & -R_O \overrightarrow{c}_O \end{bmatrix} \begin{bmatrix} X^1_{O,\beta} \\ 1 \end{bmatrix} = \sigma_2 K \begin{bmatrix} R_B & -R_B \overrightarrow{c}_B \end{bmatrix} \begin{bmatrix} AX^1_{O,\beta} + \overrightarrow{b} \\ 1 \end{bmatrix} \tag{4.2}$$

After elimination of K and rewriting the equation into the polynomial form

$$\sigma_1 R_O X^1_{O,\beta} - \sigma_1 R_O \overrightarrow{c}_O = \sigma_2 R_B A X^1_{O,\beta} + \sigma_2 R_B \overrightarrow{b} - \sigma_2 R_B \overrightarrow{c}_B \tag{4.3}$$

Because the equation has to hold true for all $X^1_{O,\beta}$, it holds true also for $X^1_{O,\beta} = 0$, which implies that the following two equations are also valid:

$$-\sigma_1 R_O \overrightarrow{c}_O = \sigma_2 R_B \overrightarrow{b} - \sigma_2 R_B \overrightarrow{c}_B \tag{4.4}$$

$$\sigma_1 R_O = \sigma_2 R_B A \tag{4.5}$$

We can easily find the relative rotation A from the latter equation as

$$A = \tfrac{\sigma_1}{\sigma_2} R_B^{-1} R_O$$

Because A, R_B^{-1} and R_O are all rotation matrices, $\sigma_1 = \sigma_2$ and therefore we can write

$$A = R_B^{-1} R_O \tag{4.6}$$

$$-R_O \overrightarrow{c}_O = R_B \overrightarrow{b} - R_B \overrightarrow{c}_B \tag{4.7}$$

We can find translation $\overrightarrow{b}$ as

$$\overrightarrow{b} = \overrightarrow{c}_B - R_B^{-1} R_O \overrightarrow{c}_O = \overrightarrow{c}_B - A \overrightarrow{c}_O \tag{4.8}$$

In Figure 4.1 it is shown that in the sequential PnP order of the objects towards whose points the poses are calculated is arbitrary. That means that apart from the true motions, the inverted ones (motion of the background if the object is considered static) can be calculated. Also, we can see in Figure

4.1(c), that not all points from the background have to be detected in the first iteration and an additional pose towards the background could be found. As these points belong to the same object, the detected poses will be almost identical and therefore the motion computed from these camera pairs will be near to the zero motion. Later we will show that this can be quite disturbing and we have to deal with these zero motions. The PnP registration can also be completely wrong, which results in relative motions which are wrong, as well.

According to equations (4.6), (4.8) we have computed relative motions between all such camera pairs P_1, P_2 where both cameras have arisen from the same sequential PnP, therefore they share the image and the take, onto which they have been registered. We can see from the equations that one camera pair is enough for finding the relative motion.

Because for every a, b all cameras from the take a registered onto a reconstruction of the take b observe the same motion, relative motions found in this way will cluster around true values of the relative motions from take b to a. In order to make the clustering easier, the computed motions are inverted in the case that $b < a$.

Computed relative rotations and translations are in the coordinate system of the reconstruction of the anchor take, onto which the cameras are registered. For further processing of the relative motions it is necessary to bring them all to the same coordinate system.

4.3 Bringing the motions into the same coordinate system

All calculated motions are in the coordinate system of the reconstruction of the anchor take, onto which the cameras have been registered. In order to cluster and verify the motions, they have to be transformed to the same coordinate system. To do so we have to find transformations between the coordinate systems first.

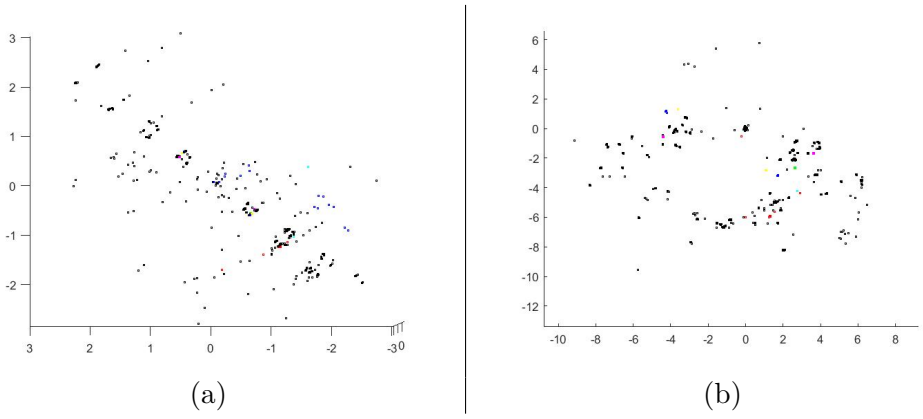


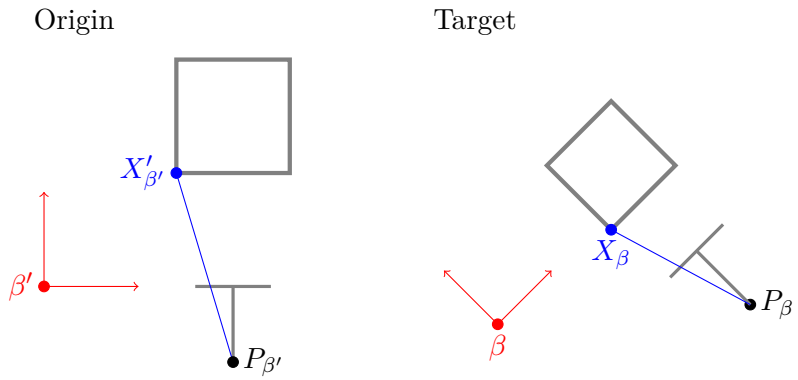
Figure 4.3: Euler vectors (a) and translations (b) calculated from a dataset consisting of 8 takes. The situation is before the transformation into the same coordinate system. Clusters of motions are observable.

4.3.1 Description of the basis transformation

β' - origin coordinate system
β - target coordinate system
$X'_{\beta'}$ - arbitrary 3D point in the origin coordinate system
X_{β} - point $X'_{\beta'}$ in the target coordinate system
$P_{\beta'}$ - camera matrix in the origin coordinate system
P_{β} - camera matrix in the target coordinate system
B - change of basis matrix from β' to β, because both bases are orthonormal, it has form of $B = \sigma R$ where σ is a scalar and R is a rotation matrix.
$\overrightarrow{o'}_{\beta}$ - origin of the β' coordinate system in the β coordinate system
Then every point in the origin coordinate system can be transformed to the target system as follows:

$$X_{\beta} = BX'_{\beta'} + \overrightarrow{o'}_{\beta} = \sigma R X'_{\beta'} + \overrightarrow{o'}_{\beta} \tag{4.9}$$

$\overrightarrow{x'}$ - projection of the 3D point onto the camera



The 3D point is projected onto the same point on the camera in both reconstructions, so the following holds true:

$$\overrightarrow{x} = \sigma_1 P_\beta \begin{bmatrix} X_\beta \\ 1 \end{bmatrix} = \sigma_2 P_{\beta'} \begin{bmatrix} X'_{\beta'} \\ 1 \end{bmatrix} \tag{4.10}$$

For some scalars σ_1, σ_2. We assume that both pictures have been taken with the same camera K. We introduce $\alpha = \frac{\sigma_1}{\sigma_2}$, R_β, $\overrightarrow{c}_\beta$ as rotation and center of matrix P_β and $R_{\beta'}$, $\overrightarrow{c}_{\beta'}$ as rotation and center of matrix $P_{\beta'}$ and write

$$\alpha K \begin{bmatrix} R_\beta & -R_\beta \overrightarrow{c}_\beta \end{bmatrix} \begin{bmatrix} \sigma R X'_{\beta'} + \overrightarrow{o}'_\beta \\ 1 \end{bmatrix} = K \begin{bmatrix} R_{\beta'} & -R_{\beta'} \overrightarrow{c}_{\beta'} \end{bmatrix} \begin{bmatrix} X'_{\beta'} \\ 1 \end{bmatrix} \tag{4.11}$$

$$\alpha(R_\beta(\sigma R X'_{\beta'} + \overrightarrow{o}'_\beta) - R_\beta \overrightarrow{c}_\beta) = R_{\beta'} X'_{\beta'} - R_{\beta'} \overrightarrow{c}_{\beta'} \tag{4.12}$$

$$\alpha\sigma R_\beta R X'_{\beta'} + \alpha R_\beta \overrightarrow{o}'_\beta - \alpha R_\beta \overrightarrow{c}_\beta = R_{\beta'} X'_\beta - R_{\beta'} \overrightarrow{c}_{\beta'} \tag{4.13}$$

This equation has to hold true for every $X'_{\beta'}$, so following two equations must be valid:

$$\alpha\sigma R_\beta R = R_{\beta'} \tag{4.14}$$

$$\alpha R_\beta \overrightarrow{o}'_\beta - \alpha R_\beta \overrightarrow{c}_\beta = -R_{\beta'} \overrightarrow{c}_{\beta'} \tag{4.15}$$

Because R_β, R and $R_{\beta'}$ are all rotation matrices, in order for the first equation to be valid, $\alpha\sigma$ must be equal to 1, so

$$\alpha = \frac{1}{\sigma} \tag{4.16}$$

We can therefore write:

$$R = R_\beta^{-1} R_{\beta'} \tag{4.17}$$

Rotation between coordinate systems can be found with one camera pair.

The latter equation can be multiplied by σ:

$$R_\beta \overrightarrow{o}'_\beta - R_\beta \overrightarrow{c}_\beta = -\sigma R_{\beta'} \overrightarrow{c}_{\beta'} \tag{4.18}$$

We introduce new symbols $\overrightarrow{t}_{\beta} = -R_{\beta}\overrightarrow{c}_{\beta}$ and $\overrightarrow{t}'_{\beta'} = -R_{\beta'}\overrightarrow{c}_{\beta'}$ and write:

$$R_{\beta}\overrightarrow{o}'_{\beta} + \overrightarrow{t}_{\beta} = \sigma\overrightarrow{t}'_{\beta'} \tag{4.19}$$

$$R_{\beta}\overrightarrow{o}'_{\beta} - \sigma\overrightarrow{t}'_{\beta'} = \overrightarrow{t}_{\beta} \tag{4.20}$$

This equation can be rewritten in a element-wise form:

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} o_1 \\ o_2 \\ o_3 \end{bmatrix} - \sigma \begin{bmatrix} t'_1 \\ t'_2 \\ t'_3 \end{bmatrix} = - \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{4.21}$$

If we know more such camera pairs, this can be written as a following system of linear equations:

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} & -t'_1 \\ R_{21} & R_{22} & R_{23} & -t'_2 \\ R_{31} & R_{32} & R_{33} & -t'_3 \\ & \dots & & \end{bmatrix} \begin{bmatrix} o_1 \\ o_2 \\ o_3 \\ \sigma \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ \dots \end{bmatrix} \tag{4.22}$$

Translation and scale between the coordinate systems can be found with at least two camera pairs.

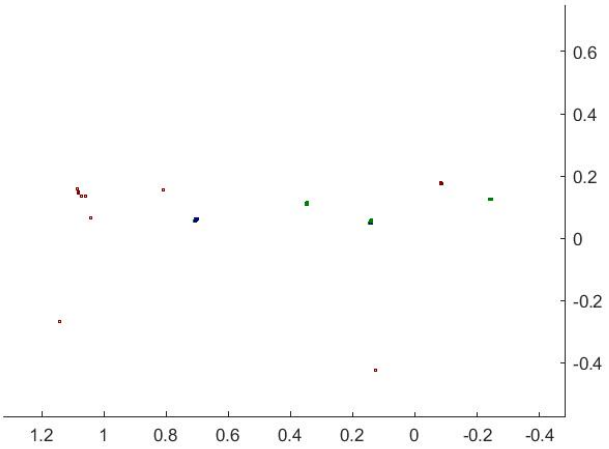


Figure 4.4: Euler vectors of the rotations between coordinate systems of three reconstructions. The clusters of the vectors are observable.

4.3.2 Clustering of the basis transformations

For each pair of takes a, b, $a \neq b$, all such camera pairs (P_a, P_b) are generated, that the camera P_a is a camera from the take a in the reconstruction of the take a, therefore in its anchor take and the camera P_b is the same camera registered onto the take b, or vice versa, the camera P_b is a camera from the take b in the reconstruction of the take b and the camera P_a is the same camera registered onto the take a. These pairs can be found using a function Select_Pairs($\mathcal{M}$,a, b). In order to compute the basis transformation from the reconstruction of the take a to the reconstruction of the take b, we use cameras P_a as $P_{\beta'}$ and P_b as P_β and follow the equations from the previous paragraph.

The problem is that both cameras registered towards the background points and towards the object points exist. Transformations between the bases of reconstructions a, b differ depending on the fact whether the background or the object is considered to be static. Therefore we cannot just take a set of all camera pairs (P_a, P_b) and compute the transformations but the pairs have to be clustered first.

As the transformation can be found with two camera pairs, we have utilized a RANSAC algorithm which randomly generates pairs of the camera pairs and from them calculates the hypothesis for the basis transformation according to the equations (4.17), (4.22) using the least squares method. For each pair of pairs inliers to their hypothesis are found among all camera pairs (P_a, P_b) and the hypothesis with the biggest support is selected while its inliers build a new cluster. Outliers to the selected hypothesis are reused in the following iteration and the RANSAC algorithm is repeated until all camera pairs are in some cluster. Rotation of the cluster is then found as the median of the Euler vectors and translation is found by the equation (4.22) with the least squares.

Select_Pairs($\mathcal{M}$,a, b) selects the pairs of cameras where a is the origin take and b is the target take. Reverse($\mathcal{M}$) reverses the pairs from the set $\mathcal{M}$. Random_Pair($\mathcal{M}$) selects a random pair from the set $\mathcal{M}$. Find_Rotation($(P_{a,1}, P_{b,1})$) returns the rotation calculated according to equation (4.17), Find_TS($(P_{a,1}, P_{b,1})$, $(P_{a,2}, P_{b,2})$, $R_{a,b,1}$) returns the translation and scale calculated according to (4.22). Find_Inliers($\mathcal{S}$, $R_{a,b}$, $\overrightarrow{o}_{a,b}$, $\sigma_{a,b}$)

finds inliers to the transformation $(R_{a,b}, \overrightarrow{o}_{a,b}, \sigma_{a,b})$ from the set $\mathcal{S}$ according to the procedure described in this section.

Algorithm 2: Cluster bases

input : $\mathcal{S}$ a finite set of camera pairs
a the origin take
b the target take
r number of the iterations of RANSAC

output : $\mathcal{C}$ a finite set of finite sets (clusters) of camera pairs

$\mathcal{S} \leftarrow$ Select_Pairs($\mathcal{M}$,a,b);
$\mathcal{C} \leftarrow \emptyset$;
while $|\mathcal{S}| > 1$ **do**
  $\mathcal{I}_{best} = \emptyset$;
  for $i \leftarrow 1$ **to** r **do**
    $(P_{a,1}, P_{b,1}) \leftarrow$ Random_Pair($\mathcal{S}$);
    $(P_{a,2}, P_{b,2}) \leftarrow$ Random_Pair($\mathcal{S}$);
    $R_{a,b} \leftarrow$ Find_Rotation($(P_{a,1}, P_{b,1})$);
    $(\overrightarrow{o}_{a,b}, \sigma_{a,b}) \leftarrow$ Find_TS($(P_{a,1}, P_{b,1})$, $(P_{a,2}, P_{b,2})$, $R_{a,b,1}$);
    $\mathcal{I} \leftarrow$Find_Inliers_Basis($\mathcal{S}$, $R_{a,b}$, $\overrightarrow{o}_{a,b}$, $\sigma_{a,b}$);
    if $|\mathcal{I}| > |\mathcal{I}_{best}|$ **then**
      $\mathcal{I}_{best} \leftarrow \mathcal{I}$;
    end
  end
  $\mathcal{C} = \mathcal{C} \cup \{\mathcal{I}_{best}\}$;
  if $\mathcal{I}_{best} = \emptyset$ **then**
    break;
  end
end

Inlier recognition

In order to use the RANSAC algorithm, a procedure for inlier recognition has to be chosen. The randomly selected hypothesis consists of two camera pairs $(P_{a,1}, P_{b,1})$, $(P_{a,2}, P_{b,2})$. Each camera can be represented according to Section 2.2 as $P_{t,i} = K_{t,i} * [R_{t,i} - R_{t,i}\overrightarrow{c}_{t,i}]$. We want to check whether a pair $(P_{a,3}, P_{b,3})$ is consistent with this hypothesis. In order to be so, both rotation and translation has to be consistent.

We can see from equation (4.17), that rotation can be checked with one camera pair. We therefore calculate $R_{a,b,3} = R_{b,3}^{-1} R_{a,3}$ and then we find

distance D between $R_{a,b,3}$ and $R_{a,b,1} = R_{b,1}^{-1} R_{a,1}$ using a procedure in Section 2.1.2. This distance is in radians and therefore scale-invariant and thus the rotational consistency can be checked by comparison of the distance D with a fixed value, which can be a parameter of the pipeline. If the rotation is inconsistent, the camera pair is rejected. Otherwise, we follow to checking of the translational consistency.

According to the equation (4.22) we compute the origin $\overrightarrow{o}_{a,b}$ and scaling $\sigma_{a,b}$ from camera pairs $(P_{a,1}, P_{b,1})$ and $(P_{a,2}, P_{b,2})$. If the transformation is consistent with the pair $(P_{a,2}, P_{b,2})$, it will transform centre $\overrightarrow{c}_{a,3}$ of the camera $P_{a,3}$ to the centre $\overrightarrow{c}_{b,3}$ of the camera $P_{b,3}$. So we transform the point $\overrightarrow{c}_{a,3}$ according to equation (4.9) as $\overrightarrow{c}'_{a,3} = \sigma_{a,b} R_1 \overrightarrow{c}_{a,3} + \overrightarrow{o}_{a,b}$ and we compare it with $\overrightarrow{c}_{b,3}$. Translation, however, is not scale invariant, so we use the apical angle concept. We compute the center of mass $\overrightarrow{m}$ of the points from the reconstruction of the take b which are visible by the camera $P_{b,3}$ and then we find the angle between vectors $\overrightarrow{c}'_{a,3} - \overrightarrow{m}$ and $\overrightarrow{c}_{b,3} - \overrightarrow{m}$. If the angle is smaller than a fixed value (which can again be a parameter of the pipeline), the pair $(P_{a,3}, P_{b,3})$ is accepted as an inlier, otherwise it is rejected.

Find_Centre(P) returns the centre of the camera P, Angle($\overrightarrow{x}$, $\overrightarrow{y}$) returns the angle between the vectors $\overrightarrow{x}$, $\overrightarrow{y}$. Point_Of_Mass(P) returns the point of mass of the points observed by the camera P.

Algorithm 3: Find inliers basis

input : $\mathcal{S}$ a finite set of camera pairs
$(R, \overrightarrow{o}, \sigma)$ the hypothetical basis transformation
t_{rot} rotational threshold
t_{trans} translational threshold

output : $\mathcal{I}$ a finite set of inliers $\mathcal{I} \subseteq \mathcal{S}$

$\mathcal{I} \leftarrow \emptyset$;
foreach $(P_{a,3}, P_{b,3}) \in \mathcal{S}$ **do**
  $R_3 \leftarrow$ Find_Rotation($(P_{a,3}, P_{b,3})$);
  if *Dist(R, R3)* $\leq t_{rot}$ **then**
    $\overrightarrow{c}_{a,3} \leftarrow$ Find_Centre($P_{a,3}$);
    $\overrightarrow{c}'_{a,3} \leftarrow \sigma R \overrightarrow{c}_{a,3} + \overrightarrow{o}$;
    $\overrightarrow{c}_{b,3} \leftarrow$ Find_Centre($P_{b,3}$);
    $\overrightarrow{m} \leftarrow$ Point_Of_Mass($P_{b,3}$);
    if *Angle*$(\overrightarrow{c}'_{a,3} - \overrightarrow{m}, \overrightarrow{c}_{b,3} - \overrightarrow{m}) \leq t_{trans}$ **then**
      $\mathcal{I} \leftarrow \mathcal{I} \cup \{(P_{a,3}, P_{b,3})\}$
    end
  end
end

4.3.3 Verification of the basis transformations

We have found clusters of the camera pairs and their corresponding transformations. But we still need to group the transformations between different coordinate systems, as we want to transform all motions to the same coordinate system and we need that the transformations are either all towards the background or all towards the object. In order to do so, we use the verification via cycles of the transformations of length three.

We have three takes a, b, c together with their basis transformations and the corresponding clusters of camera pairs. If a vector in the first coordinate system is transformed to the second coordinate system, then to the third coordinate system, and then back to the first coordinate system, the resulting vector is equal to the original vector. This forms a constraint for coordinate change matrices which change the coordinates in a cycle.

$R_{a,b}$ - rotation between bases of takes a, b
$\sigma_{a,b}$ - scale between first and second basis
$\overrightarrow{o}_{a,b}$ - origin of the first basis in the second coordinate system
$R_{b,c}$ - rotation between second and third basis
$\sigma_{b,c}$ - scale between second and third basis
$\overrightarrow{o}_{b,c}$ - origin of the second basis in the third coordinate system
$R_{c,a}$ - rotation between third and first basis
$\sigma_{c,a}$ - scale between third and first basis
$\overrightarrow{o}_{c,a}$ - origin of the third basis in the first coordinate system

x_a - arbitrary point in the first coordinate system
x_b - point x_a in the second coordinate system
x_c - point x_a in the third coordinate system

Transformations between the points:

$$x_b = \sigma_{a,b} R_{a,b} x_a + \overrightarrow{o}_{a,b} \tag{4.23}$$

$$x_c = \sigma_{b,c} R_{b,c} x_b + \overrightarrow{o}_{b,c} \tag{4.24}$$

$$x_a = \sigma_{c,a} R_{c,a} x_c + \overrightarrow{o}_{c,a} = \sigma_{c,a} R_{c,a} (\sigma_{b,c} R_{b,c} x_b + \overrightarrow{o}_{b,c}) + \overrightarrow{o}_{c,a}$$
$$x_a = \sigma_{c,a} R_{c,a} (\sigma_{b,c} R_{b,c} (\sigma_{a,b} R_{a,b} x_a + \overrightarrow{o}_{a,b}) + \overrightarrow{o}_{b,c}) + \overrightarrow{o}_{c,a}$$
$$x_a = \sigma_{c,a} R_{c,a} \sigma b, c R_{b,c} (\sigma_{a,b} R_{a,b} x_a + \overrightarrow{o}_{a,b}) + \sigma_{c,a} R_{c,a} \overrightarrow{o}_{b,c} + \overrightarrow{o}_{c,a}$$

$$x_a = \sigma_{c,a}R_{c,a}\sigma_{b,c}R_{b,c}\sigma_{a,b}R_{a,b}x_a + \sigma_{c,a}R_{c,a}\sigma_{b,c}R_{b,c}\overrightarrow{o}_{a,b} + \sigma_{c,a}R_{c,a}\overrightarrow{o}_{b,c} + \overrightarrow{o}_{c,a} \tag{4.25}$$

In order for the last equation to be valid, $R_{c,a}R_{b,c}R_{a,b}$ must be equal to the identity matrix, $\sigma_{c,a}\sigma_{b,c}\sigma_{a,b}$ must be equal to 1 and $\sigma_{c,a}R_{c,a}\sigma_{b,c}R_{b,c}\overrightarrow{o}_{a,b} + \sigma_{c,a}R_{c,a}\overrightarrow{o}_{b,c} + \overrightarrow{o}_{c,a}$ must be equal to 0.

From all existing cycles of the clusters we want to select those which satisfy this condition (the consistent cycles) and reject the other cycles. To do so, we compute values
$R_{cy} = R_{c,a}R_{b,c}R_{a,b}$
$\sigma_{cy} = \sigma_{c,a}\sigma_{b,c}\sigma_{a,b}$
$t_{cy} = \sigma_{c,a}R_{c,a}\sigma_{b,c}R_{b,c}\overrightarrow{o}_{a,b} + \sigma_{c,a}R_{c,a}\overrightarrow{o}_{b,c} + \overrightarrow{o}_{c,a}$
The cycle of the transformations is then marked as consistent, if it satisfies following conditions:

1. Norm of the rotation R_{cy} (Section 2.1.1) has to be smaller than a threshold value.

2. Distance of σ_{cy} from 1 has to be smaller than another threshold.

A condition for t_{cy} cannot be a single threshold because the translation is not scale invariant, so apical angle is again used. For each camera in the cluster which belongs to the reconstruction c the apical angle between the average point of the camera m and a point $m + t_{cy}$ has to be lower than yet another threshold. All these thresholds can be parameters of the pipeline.

After the consistent cycles are found, a graph is built whose vertices are the clusters of the camera pairs. There is an edge between the clusters if there exists a consistent cycle which contains both clusters. This graph splits into connected components, one of the components should contain basis transformations towards the background, another one should contain basis transformations towards the object.

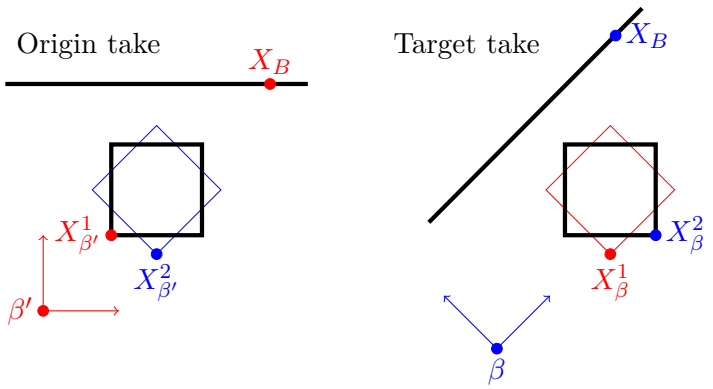
4.3.4 Transformation of the motions

We have verified the clusters via cycle consistency and divided them into connected components. For each connected component we create another graph but this time the vertices represent the takes and there is an edge between the vertices a, b if a cluster exists in the connected component which relates bases of reconstructions of takes a, b. The edge has a weight of $\frac{1}{R_{cy}}$ where R_{cy} is the rotational consistency of the most consistent cycle among those which contain a cluster relating takes a, b. Such cluster and its corresponding basis transformation is assigned to the edge.

To obtain transformations between all takes, it is sufficient to have a spanning tree of the second graph and transformations between takes which are not connected can be computed transitively. In this case, the minimum spanning tree of the second graph is used.

We have selected the central take as the one with the highest degree in the spanning tree of the second graph and we want to transform all computed motions to the coordinate system of the central take. In order to do so, we need to know how the computed rotation and translation of the object behaves under the change of the basis.

β' - origin basis
β - target basis
B - change of basis matrix from β' to β, because both bases are orthonormal, it has form of $B = \sigma R$ where σ is a scalar and R is a rotation matrix.
$\overrightarrow{o'_\beta}$ - origin of the β' coordinate system in the β coordinate system
$X^1_{\beta'}$ - arbitrary 3D point from the object in the origin coordinate system
$X^2_{\beta'}$ - new position of the point $X^1_{\beta'}$ after moving the object to a position in the second take; point is in the origin coordinate system
$A_{\beta'}$, $\overrightarrow{b}_{\beta'}$ - rotation and translation of the object in the origin coordinate system
X^1_β - point $X^1_{\beta'}$ in the target coordinate system
X^2_β - point $X^2_{\beta'}$ in the target coordinate system
A_β, $\overrightarrow{b}_\beta$ - rotation and translation of the object in the target coordinate system



It holds true:

$$X^2_{\beta'} = A_{\beta'} X^1_{\beta'} + \overrightarrow{b}_{\beta'} \tag{4.26}$$

$$X^2_{\beta} = A_{\beta} X^1_{\beta} + \overrightarrow{b}_{\beta} \tag{4.27}$$

$$X^1_{\beta} = \sigma R X^1_{\beta'} + \overrightarrow{o}'_{\beta} \tag{4.28}$$

$$X^2_{\beta} = \sigma R X^2_{\beta'} + \overrightarrow{o}'_{\beta} \tag{4.29}$$

We can see that both second and fourth equations are equal to X^2_β, we can therefore write:

$$\begin{gathered} A_\beta X^1_\beta + \overrightarrow{b}_\beta = \sigma R X^2_{\beta'} + \overrightarrow{o}'_\beta \\ A_\beta(\sigma R X^1_{\beta'} + \overrightarrow{o}'_\beta) + \overrightarrow{b}_\beta = \sigma R(A_{\beta'} X^1_{\beta'} + \overrightarrow{b}_{\beta'}) + \overrightarrow{o}'_\beta \\ A_\beta(\sigma R) X^1_{\beta'} + A_\beta \overrightarrow{o}'_\beta + \overrightarrow{b}_\beta = \sigma R A_{\beta'} X^1_{\beta'} + \sigma R \overrightarrow{b}_{\beta'} + \overrightarrow{o}'_\beta \end{gathered}$$

The equation must be valid for every $X^1_{\beta'}$, which means:

$$A_\beta(\sigma R) = \sigma R A_{\beta'} \tag{4.30}$$

$$A_\beta \overrightarrow{o}'_\beta + \overrightarrow{b}_\beta = \sigma R \overrightarrow{b}_{\beta'} + \overrightarrow{o}'_\beta \tag{4.31}$$

We can express relative rotation in the target basis from the first equation as:

$$A_\beta = R A_{\beta'} R^{-1} \tag{4.32}$$

We can see that transformed relative rotation can be obtained only with relative rotation between the coordinate systems, we do not need to find the translation nor the scale between the coordinate systems.
We can express relative translation in the target basis from the second equation as:

$$\overrightarrow{b}_\beta = \sigma R \overrightarrow{b}_{\beta'} + \overrightarrow{o}'_\beta - R A_{\beta'} R^{-1} \overrightarrow{o}'_\beta \tag{4.33}$$

If there is a direct connection between the origin and the central takes in the spanning tree, we can directly transform the translation and the rotation according to the equations (4.32), (4.33) using the transformation assigned to the edge connecting the takes. If there is no such connection, we can follow the path from the origin take to the central take in the spanning tree and transform the rotation and the translation sequentially.

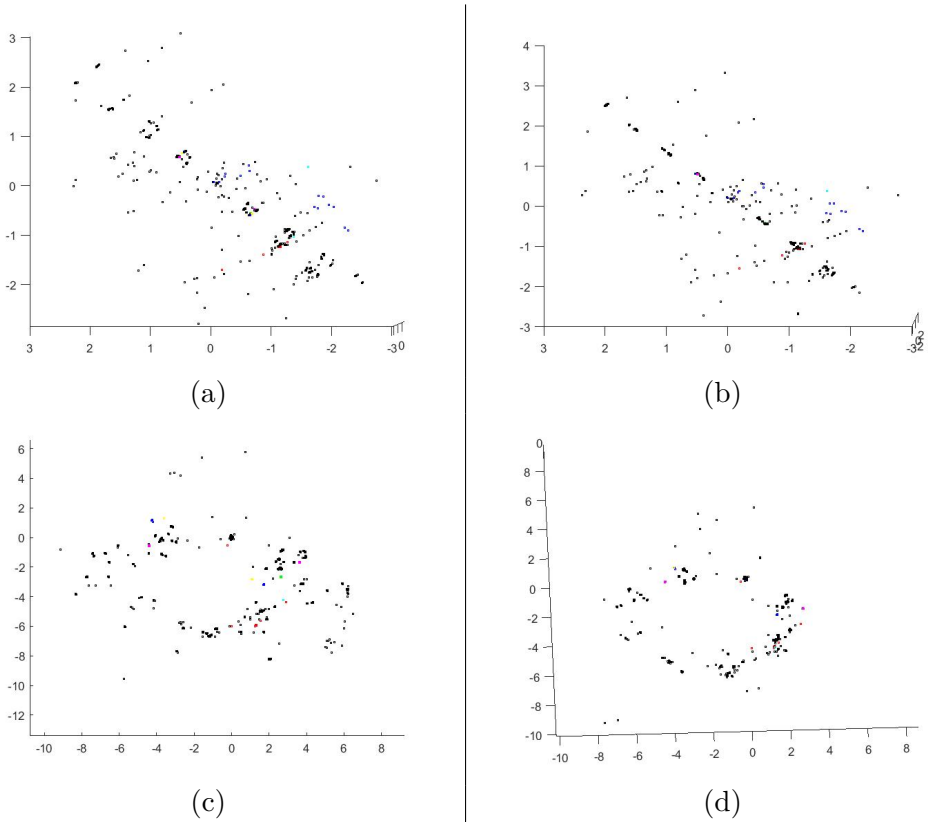


Figure 4.5: Euler vectors (a), (b) and translations (c), (d) calculated from a dataset consisting of 8 takes. Images (a), (c) depict the situation before the transformation, images (b), (d) depict the situation after the transformation. The object was always rotated around the vertical axis which is observable in (b).

4.3.5 Zero motions removing

Due to the multiple registration of a camera towards the points (Figure 4.1) of the same object, zero motions arise among the calculated motions of the object which is observable in Figures 4.7(b), 4.7(d). However these motions can make a problem in a following cyclic verification of the motions (Section 4.5.1), so we need to dispose of them. In order to do that we use spectral clustering to discover clusters of the zero motions.

We use multi-view spectral clustering [30], [23] by the kernel addition to get clusters of points which are distinguished by both rotation and translation. The clustering using the RANSAC would not be sufficient because in some cases the cluster of the zero motions appears to have too high extent. Figure 4.7(b) The rotational distance between two motions is computed according to Section 2.1.2 and the translational distance between the points is computed using the Euclidean distance between the translations. The distances D are converted into affinities A using a transformation $A = \frac{1}{D}$. The normalized Laplacian [30] is used in order to eliminate the scaling of the translations. The final kernel is computed as a sum of both Laplacians. Because the number of all motions is too high to perform eigendecomposition effectively, the motions are divided according to the origin and final takes of the motions. The number of clusters k is equal to the number of eigenvalues which are lower or equal to 1 while the maximum eigenvalue is 2 because of the normalization. The k-means algorithm is used to obtain the clusters from the first k eigenvectors.

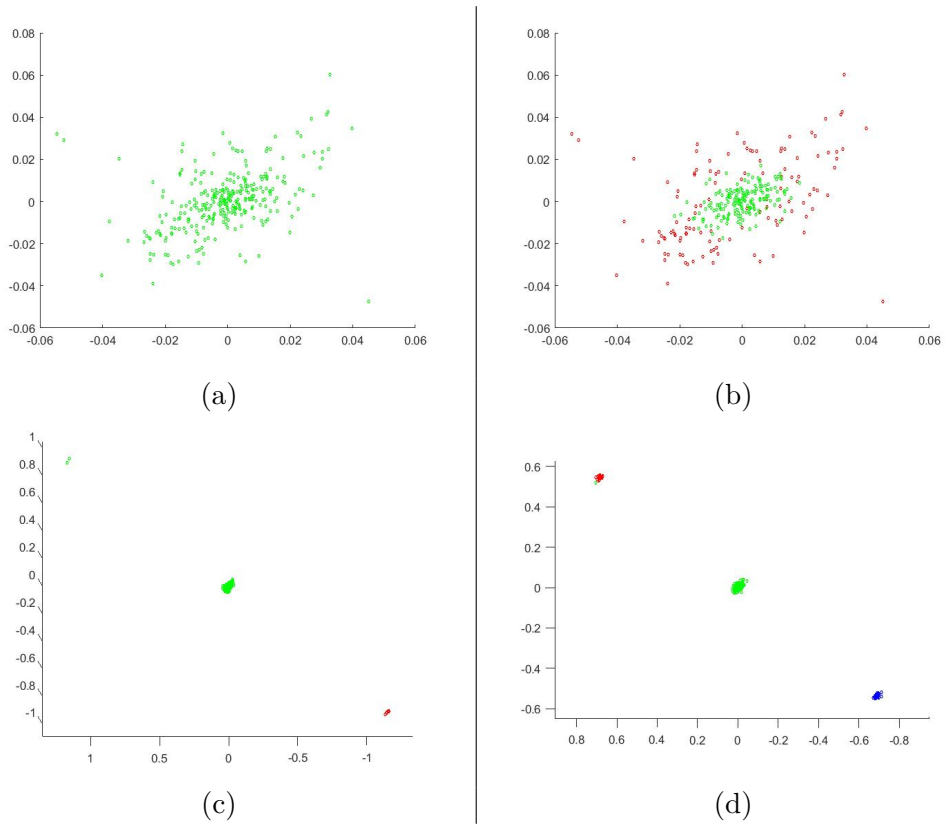


Figure 4.6: Examples of results of the spectral clustering of the clusters. Only rotations are depicted.

All clusters are then tested on the distance from zero. For rotation, the norm (Section 2.1.1) of the median Euler vector among the motions in the cluster is

used and the median translation has to be lower than a fixed fraction of the median distance from cameras from the central take to the points which they observe. If both rotation and translation are near enough to zero, the whole cluster is removed. According to Figure 4.7 this procedure is able to remove the cluster of the zero motions. If the zero cluster is split like in Figure 4.6(b), both zero clusters are removed.

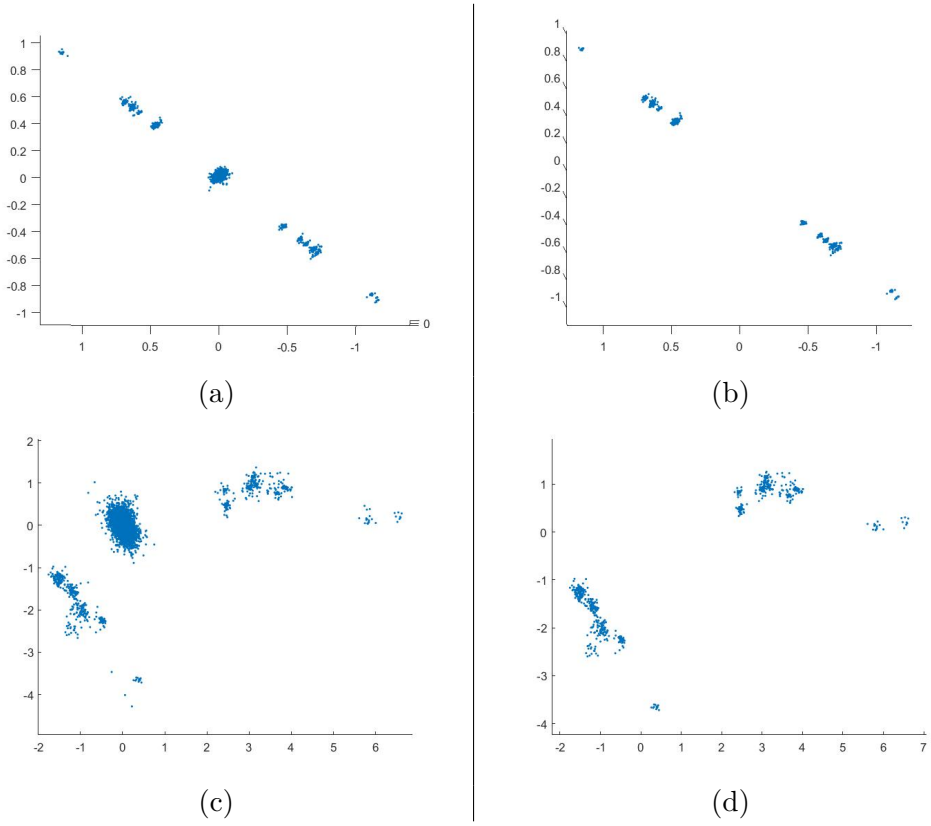


Figure 4.7: The Euler vectors (a), (b) and the translations (c), (d) of the object motion. Images (a), (c) depict the situation before the removing, images (b), (d) depict the situation after the removing.

4.4 Motion clustering

We have transformed the motions into the same coordinate system and removed the zero motions. We can, therefore, follow to clustering the motions. The main purpose of the clustering is that although the zero motions have been eliminated, there are motions, which have been computed from pairs where the first camera has been registered towards the background points and the second one has been registered towards the object points, as well as motions which have been computed from pairs where the sequential PnP

performed in the reversed order and pairs where one or two cameras have been registered wrongly. We want to distinguish these groups of cameras from each other in order to distinguish the points which the cameras observe.

Similarly to the clustering of the basis transformations, we use RANSAC but unlike that, one camera pair is sufficient to compute the motion according to equations (4.6), (4.8), so this time the hypothesis consists of one motion only. Because of a small total number of hypotheses, they do not have to be drawn randomly but all hypotheses can be checked.

The motions are divided into subsets $S_{a,b}$ according to their original and target takes $a, b, a < b$ and they are clustered separately. For each motion $A_1, \overrightarrow{b}_1$ in the subset $S_{a,b}$, its support is computed as the number of inliers to this motion from the set $S_{a,b}$. The motion with the highest support is selected and together with its inliers it builds a new cluster. Outliers to the motion are reused in the next iteration and the procedure is repeated until all motions are in some track.

The clusters are then represented by a single Euler vector and a single translation vector. Both are medians of the motions in the cluster.

Algorithm 4: Cluster motions

input : $\mathcal{S}$ a finite set of transformed camera pairs
a the origin take
b the target take
output : $\mathcal{C}$ a finite set of finite sets (clusters) of camera pairs

```
S ← Select_Pairs(M,a,b);
C ← ∅;
while |S| > 0 do
    I_best = ∅;
    foreach (P_a,1, P_b,1) ∈ S do
        R_a,b ← Find_Rotation((P_a,1, P_b,1));
        t⃗_a,b ← Find_Translation((P_a,1, P_b,1));
        I ←Find_Inliers_Motions(S, R_a,b, t⃗_a,b);
        if |I| > |I_best| then
            I_best ← I;
        end
    end
    C = C ∪ {I_best};
    if I_best = ∅ then
        break;
    end
end
```

4.4.1 Inlier recognition

As well as at the clustering of the transformations of the coordinate systems, a procedure for inlier recognition has to be chosen. The hypothesis consists of one motion (A_1, b_1) which has been computed from a camera pair $(P_{1,1}, P_{2,1})$. We want to check whether a motion (A_2, b_2) is consistent with this hypothesis. In order to be so, both the rotation A and the translation b have to be consistent.

We find distance D_r between the rotations A_1 and A_2 using a procedure in Section 2.1.2. This distance is in radians and therefore scale-invariant and thus the rotational consistency can be checked by comparison of the distance D_r with a fixed value, which can be a parameter of the pipeline. If the rotation is inconsistent, the camera pair is rejected. Otherwise, we follow to the checking of the translational consistency.

In order to check the translation, we use the apical angle once more. We find distance D_t between the translations t_1 and t_2 as the Euclidean distance. If the distance D_t is smaller than a fixed fraction of the distance from the center of the camera $P_{1,1}$ to the median $\overrightarrow{c}$ of the points which are observer by this camera. Both points were transformed to the coordinate system of the central take before the checking of the translation.

4.5 Motion verification

We have found clusters of the motions between the pairs of takes. We assume that all computed motions which belong to the same cluster have arisen from the same physical motion, which can be either a motion of the object on the static background if the first camera was registered towards the points from the background and the second one towards the points from the object, or it can be the inverse motion if the cameras were registered in the reversed order (the motion would correspond to the motion of the background towards the static object).

In order to detect the points which belong to the background and to the object, we need to group the clusters of the motions throughout all pairs of takes. We need the motions from the camera pairs where the first camera has been registered towards the background to be in one group and the reversed motions to be in the reversed group. The erroneous motions should belong to none of these groups.

Like at the verification of the basis transformations we use the verification via cycles of the transformations. We have discovered that unlike the verification of the basis transformations, the verification via 3-cycles has not shown to be sufficient in this case because between some takes no cameras could be registered towards the object points, especially if the angle of the rotation of the object between these takes is too large and so there are not enough points from the object which are observed by cameras from both takes. This was not a problem at the verification of the basis transformations as the whole operation could be performed only with cameras registered towards the background and with the cameras from the anchor take.

4.5.1 Condition for consistent cycles

If we move the object from the original position in the first take to a position of the second take, then from a position in the second take to a position in the third take and then from the position in the third take back to the original position in the first take, a position of an arbitrary point on the object will be equal to its original position. From this a condition for cycles of length 3 can be derived.
X_1 - Original position of a point.
X_2 - Point X_1 after its movement from the first to the second take
X_3 - Point X_2 after its movement from the second to the third take
A_{12} - Rotation from the first to the second take
b_{12} - Translation from the first to the second take
A_{23} - Rotation from the second to the third take
b_{23} - Translation from the second to the third take
A_{31} - Rotation from the third to the first take
b_{31} - Translation from the third to the first take
We assume that all points, rotations and translations are in the same coordinate system. Relationships between the points are as follows:

$$\begin{aligned} X_2 &= A_{12}X_1 + b_{12} \\ X_3 &= A_{23}X_2 + b_{23} \\ X_1 &= A_{31}X_3 + b_{31} \end{aligned}$$

It can therefore be written

$$X_1 = A_{31}(A_{23}X_2 + b_{23}) + b_{31} = A_{31}(A_{23}(A_{12}X_1 + b_{12}) + b_{23}) + b_{31} = A_{31}A_{23}(A_{12}X_1 + b_{12}) + A_{31}b_{23} + b_{31} = A_{31}A_{23}A_{12}X_1 + A_{31}A_{23}b_{12} + A_{31}b_{23} + b_{31}$$

From this equation the constraint for consistency of the cycles of the length 3 can be derived as:

$$A_{31}A_{23}A_{12} = I \tag{4.34}$$

$$A_{31}A_{23}b_{12} + A_{31}b_{23} + b_{31} = 0 \tag{4.35}$$

If the length of the cycle is n, we have a sequence of takes $\{t_i\}_{i=1}^{n+1}$ where $t_{n+1} = t_1$ in order for the takes to form a cycle. The corresponding sequence of translations and rotations is $\{A_{t_i,t_{i+1}}, b_{t_i,t_{i+1}}\}_{i=1}^{n}$, where $A_{a,b}$ is a rotation from take a to take b and $b_{a,b}$ is a translation from take a to take b. The point X_{t_1} is gradually transformed as follows:

$$X_{t_{i+1}} = A_{t_i,t_{i+1}}X_{t_i} + b_{t_i,t_{i+1}} \tag{4.36}$$

Because after the object is moved back to the first take, its position has to be equal to the original position, the following has to hold true:

$$X_{t_{n+1}} = X_{t_1} \tag{4.37}$$

This equation is used as a translational condition for cycle consistency. A point is transformed gradually by all motions in the cycle and the final position $X_{t_{n+1}}$ has to be near enough to the original one X_{t_1}. Before checking this condition, an easier rotational condition stemming from the same equation is applied:

$$A_{t_i,t_{i+1}} \cdot ... \cdot A_{t_2,t_3} \cdot A_{t_1,t_2} = I \tag{4.38}$$

This product of matrices is obtained and converted into an Euler vector. The norm of this vector has to be small enough for the cycle to be consistent. If it is so, a condition from equation (4.37) is used to accept or discard the cycle.

This is the place where the occurrence of the zero motions would be the most problematic. Not only would an arbitrary sequence of zero motions be accepted as a consistent cycle but a sequence of one forward motion (first camera registered towards the background points, second camera towards the object points), one backward motion (cameras registered in the reversed order) and one zero motion would be accepted, too. This would group together the forward and the backward motions, which is an unacceptable result.

4.5.2 Chordal completion

We need to verify cycles of the motions whose length can be longer than three. Trying all possible cycles would however not be a task with polynomial complexity. We, therefore, use an algorithm based on the chordal completion of a graph to discover the chordless cycles which form the cycle basis of the graph in order to check their consistency. If two consistent cycles from the cycle basis share an edge, the sum of these cycles is consistent, as well.

A graph is built where the vertices represent the clusters of the motions. Each of the clusters has assigned two takes, the initial one and the final one. There is an edge between those clusters which share exactly one of the takes and therefore the motions they represent can be combined into one compound motion. These edges are the original edges and they do not have any predecessor. The graph without any added edges is the original

graph. We want to find all chordless cycles in this original graph. In order to do that we fill these cycles with additional chords until they are fully connected and then we can discover the original cycle and check its consistency.

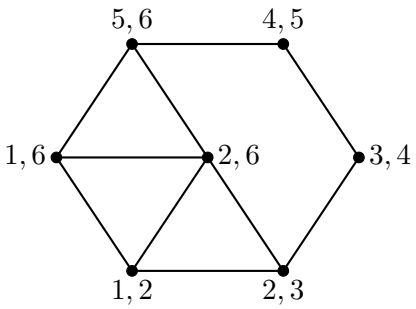


Figure 4.8: An example of a graph of the clusters. The labels of the vertices are the numbers of the initial and the final takes of the clusters they represent. The valid cycles are $((1,2),(2,3),(3,4),(4,5),(5,6),(1,6))$, $((1,2),(2,6),(1,6))$. The invalid cycles are $((1,2),(2,3),(2,6))$, $((1,6),(2,6),(5,6))$

A triplet of vertices can be a subset of a cycle if it does not contain multiple vertices corresponding to the same pair of takes, as in the cyclic transformation each take can be visited only once except for the first one which is the last one as well. The triplets satisfying this condition are inserted in a queue. Then until the queue is empty or no new chord can be added and no cycle can be verified remove the triplets from the queue.

Once a triplet is removed, the count of the existing edges between the vertices in the triplet is detected. The only possible values are 0, 1, 2, 3. If the count is 0 or 1, the triplet is returned back to the queue, as nothing can be done with it yet. If the count is 2, the missing chord is added to the graph and the triplet is assigned to the chord as its predecessor. The two other edges are assigned to the triplet as its predecessors. If the count is 3, the original cycle can be discovered and verified, all three edges are predecessors of the triplet.

The procedure is not optimal in the terms of number of operations as some cycles can be discovered more times, as the chordal completion algorithm has a greedy nature and is not optimal.

Add_Edges(V, $\mathcal{E}$) adds edges from $\mathcal{E}$ to a graph represented by an adjacency matrix V, Valid($\mathcal{E}$) returns true if edges from a set $\mathcal{E}$ can build a valid cycle.

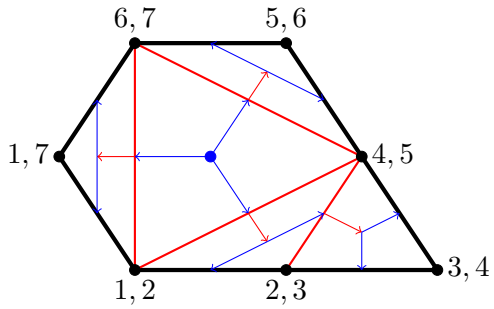


Figure 4.9: An example of a cycle after the chordal completion. Red edges are the chords. Red arrows lead from the chord to the 3-cycle which is its predecessor. Blue arrows lead from the 3-cycle to the edges which are its predecessors.

Check_3((R_i, t_i), (R_j, t_j), (R_k, t_k)) checks the cycle of length 3 according to equations (4.34), (4.35). Check_N(C) checks the cycle C of length N according to equations (4.37), (4.38).

Algorithm 5: Init graph

input : n number of the motion clusters
$\{a_i, b_i\}_{i=1}^{n}$ a sequence of takes of the clusters, $a_i < b_i$
output : G adjacency matrix of the initial graph $V \in M^{n,n}$

$G \leftarrow 0^{n,n}$;
for $i \leftarrow 1$ **to** n **do**
  for $j \leftarrow i + 1$ **to** n **do**
    if $a_i = a_j$ *and* $b_i = b_j$ **then**
      $G_{i,j} \leftarrow -2$;
      $G_{j,i} \leftarrow -2$;
    else if $a_i = a_j$ *or* $a_i = b_j$ *or* $b_i = a_j$ *or* $b_i = b_j$ **then**
      $G_{i,j} \leftarrow 0$;
      $G_{j,i} \leftarrow 0$;
    else
      $G_{i,j} \leftarrow -1$;
      $G_{j,i} \leftarrow -1$;

Algorithm 6: Chordal completion

input : n number of the motion clusters
$\{R_i, t_i\}_{i=1}^{n}$ a sequence of motions belonging to the clusters
$\{a_i, b_i\}_{i=1}^{n}$ a sequence of takes of the clusters, $a_i < b_i$
output : V adjacency matrix of the verified graph $V \in M^{n,n}$

```
V ← 0^{n,n};
G ← Init_Graph(n, {a_i, b_i}_{i=1}^n);
pos ← 1;
T ← ();
Q ← empty queue;
for i ← 1 do
    for j ← i + 1 do
        for k ← j + 1 do
            if G_{i,j} ≠ −2 and G_{j,k} ≠ −2 and G_{i,k} ≠ −2 then
                T_pos ← (i, j, k);
                Q.enqueue(pos);
                pos ← pos + 1;
            end
        end
    end
end
while Q not empty do
    pos ← Q.dequeue;
    (i, j, k) ← T_pos if G_{i,j} ≥ 0 and G_{j,k} ≥ 0 and G_{i,k} ≥ 0 then
        E ← Check_Cycle((i, j, k), G, {R_i, t_i}_{i=1}^n, T);
        V ← Add_Edges(V, E);
    else if G_{i,j} ≥ 0 and G_{j,k} ≥ 0 then
        G_{i,k} ← pos;
    else if G_{i,j} ≥ 0 and G_{i,k} ≥ 0 then
        G_{j,k} ← pos;
    else if G_{j,k} ≥ 0 and G_{i,k} ≥ 0 then
        G_{i,j} ← pos;
    else
        Q.enqueue(pos);
    end
end
```

Cycle discovering

If the 3-cycle consists only of original edges, it can be directly checked on the consistency. Otherwise, the original edges are discovered via DFS. The edges from the cycle are inserted into the stack. Once an edge is removed

from the stack, it is controlled whether it is an original edge or an additional chord. If it is an original edge, it is saved to a set of edges. If it is an additional chord, it has a predecessor cycle. The other two edges from this predecessor cycle are also inserted to the stack. The search finishes when the stack is empty and the set of edges contains all edges which belong to the cycle.

Algorithm 7: Check cycles

input : (i, j, k) ids of the clusters in the cycle
G adjacency matrix of a graph
$\{R_i, t_i\}_{i=1}^{n}$ a sequence of motions of the clusters
T a sequence of triplets of clusters

output : $\mathcal{E}$ set of the verified edges

$\mathcal{E} \leftarrow \emptyset$;
if $G_{i,j} + G_{j,k} + G_{i,k} = 0$ **then**
  if *Check_3*$((R_i, t_i), (R_j, t_j), (R_k, t_k))$ **then**
    $\mathcal{E} \leftarrow \{ (i, j), (j, k), (i, k) \}$;
else
  $Q \leftarrow$ empty queue;
  Q.enqueue$((i, j))$;
  Q.enqueue$((j, k))$;
  Q.enqueue$((i, k))$;
  while Q *not empty* **do**
    $(i, j) \leftarrow Q$.dequeue;
    if $G_{i,j} = 0$ **then**
      $\mathcal{E} \leftarrow \mathcal{E} \cup \{(i, j)\}$;
    else
      $pred \leftarrow G_{i,j}$;
      $(a, b, c) \leftarrow T_{pred}$;
      if $i \neq a$ *or* $j \neq b$ **then**
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{(a, b)\}$;
      if $i \neq a$ *or* $j \neq c$ **then**
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{(a, c)\}$;
      if $i \neq b$ *or* $j \neq c$ **then**
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{(b, c)\}$;
  if *not Valid*$(\mathcal{E})$ **then**
    $\mathcal{E} \leftarrow \emptyset$;
  else
    $C \leftarrow$ Build_Cycle$(\mathcal{E}, \{R_i, t_i\}_{i=1}^{n})$;
    if *not Check_N*(C) **then**
      $\mathcal{E} \leftarrow \emptyset$;

These edges have to be tested whether they can build a valid cycle and eventually sorted to build one. A valid cycle is a sequence of motions $\{A_{t_i,t_{i+1}}, b_{t_i,t_{i+1}}\}_{i=1}^{n}$ where $t_1 = t_{i+1}$ and no other takes in the sequence $\{t_i\}_{i=1}^{n+1}$ repeat. The reason for discarding the cycles with repeating takes is that a sequence of forward motions $(A_{x,y}, b_{x,y}), (A_{y,z}, b_{y,z})$ followed by a sequence of backward motions $(A'_{z,y}, b'_{z,y}), (A'_{y,x}, b'_{y,x})$ could be a discovered cycle. This cycle would pass the condition from equation (4.37), but if it did, the forward and backward motions would be connected, which should not happen. The cycle is however not valid, as the take y repeats in it. An example of a cycle which is not valid is the cycle $(1, 2), (2, 3), (2, 6)$ in Figure 4.8.

We have a set of edges. Each edge has two adjacent vertices, an initial one and a final one. The task is to build a valid cycle from these edges. At first, an arbitrary edge is selected and added to the sequence. Among the remaining edges, such edge is selected that one of its adjacent vertices is the final edge of the sequence. If the vertex is the final vertex of the edge, the edge has to be reversed and the motions belonging to the vertices are reversed, too. After all the edges have been inserted into the sequence, we test whether it is a valid cycle and if so, the conditions (4.38), (4.37) are used to check the consistency of the cycle. This time however the thresholds are not fixed but they depend linearly on the length of the cycle because longer cycles have higher uncertainty.

Algorithm 8: Build cycle

input : $\mathcal{E}$ set of edges
$\{R_i, t_i\}_{i=1}^{n}$ a sequence of motions of the clusters
output : C sequence of motions which builds a valid cycle

```
C ← ();
last ← −1;
pos ← 1;
while E ≠ ∅ do
    foreach (a, b) ∈ E do
        if last = −1 or last = a then
            C_pos ← (R_a, t_a);
            pos ← pos + 1;
            last ← b;
            E ← E \ (a, b);
        else if last = b then
            C_pos ← (R_a^{-1}, −R_a^{-1} t_a) pos ← pos + 1;
            last ← a;
            E ← E \ (a, b);
```

4.5.3 Clustering of the motion clusters

A subgraph of the original graph is created only with those edges which belong to some consistent cycle. This results from the fact that if the pairs of the cameras in all clusters have been registered in the same order, the motions assigned to the clusters should build a consistent cycle. Potential clusters of clusters correspond to each connected component of the graph. There is however a possibility that a cycle is verified as consistent, although the motions belong to different objects. This can happen for example due to erroneous motions.

If the connected component contains clusters $(A_{i,j}, b_{i,j}), (A'_{i,j}, b'_{i,j})$ of motions whose initial and final takes i, j are both equal and the distance of the clusters is too high, the component is considered to be inconsistent. If it contains an inconsistent 3-cycle, it is inconsistent, too. The distance between the clusters is detected by the same procedure which is applied for inlier recognition at the clustering of the motions but the thresholds are higher. An inconsistent component is split into components which are consistent via multi-view spectral clustering. The affinities in the first Laplacian are calculated from the rotational consistency condition (4.38) and the affinities in the second Laplacian are calculated from the translational consistency condition (4.37). The final kernel is the sum of these Laplacians. The spectral clustering is performed without normalization because it gives a better splitting. The initial number of clusters is 2 and it is incremented until the clustering gives only consistent clusters. Among the found clusters of clusters, the one is selected, which contains the highest number of camera pairs. The cameras in the clusters in the selected cluster of clusters are used to distinguish the 3D points.

4.6 Track building and segmentation

We know which cameras observe the points belonging to the background and which observe the points belonging to the object. Before we classify these points, tracks are built from the points so the points can be classified in all reconstructions, not just in those reconstructions where they are directly observed by a camera from the cluster. A track contains points from different reconstructions which have been reconstructed from the same observations and therefore represent the same real point.

4.6.1 Track building

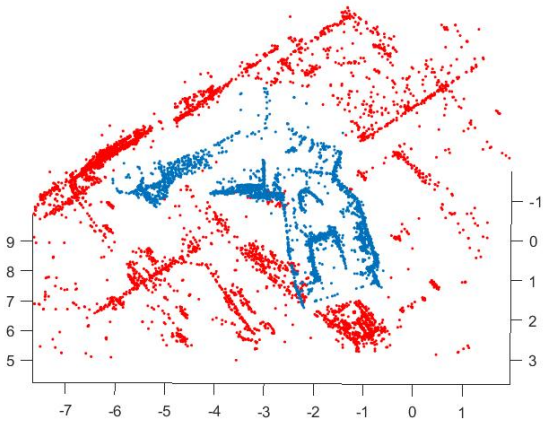
A graph is built whose vertices are the 3D points from all reconstructions. Two 3D points X, Y from different reconstructions a, b are connected by an edge if there is a 2D feature in some image, which observes the 3D point X in the reconstruction a and the 3D point Y in the reconstruction b. The weight of this edge is equal to the number of such 2D features. Each connected component of the graph is a tentative track. It is however possible due to mismatches, that points reconstructed from different real points are connected in the graph and therefore should occur in the same track.

A track is inconsistent if it contains multiple 3D points from the same reconstruction. If a track is inconsistent, it is split via the spectral clustering. The Laplacian of the graph is used directly without changes. The initial number of clusters is 2 and it is incremented until the k-means clustering of the eigenvectors of the Laplacian gives only consistent tracks.

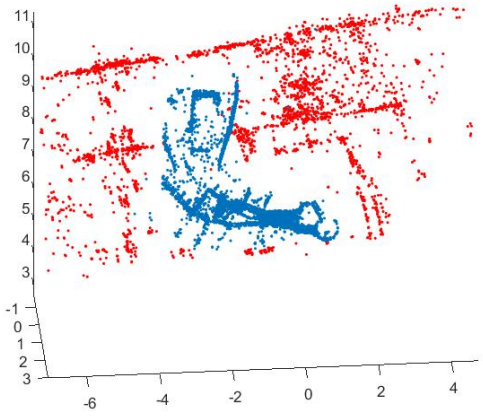
4.6.2 Track segmentation

The motions in the clusters verified according to equations (4.37), (4.38) have been calculated from two cameras. The first camera was registered towards the background points and the second one towards the object points. After the cameras were registered, no further points were triangulated. Therefore for each camera pair (P_a, P_b) in the cluster the points observed by the first camera P_a should belong to the background and the points observed by the second camera P_b should belong to the object.

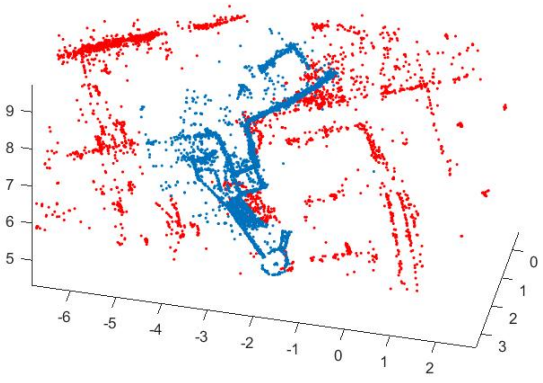
The cameras which are on the first position some pair in the cluster build the group G_a, the cameras on the second position build the group G_b. For each track, a score is introduced with 0 as the initial value. If the track is observed by a camera from the group G_a, its score is increased. If the track is observed by a camera from the group G_b, its score is decreased. The tracks with a positive score are assigned to the background, the tracks with a negative score are assigned to the object and the tracks whose score is zero are ignored.



(a)



(b)



(c)

Figure 4.10: Reconstructions of a single take after the segmentation of the points. Points which do not belong to background nor to the object are not shown.

4.7 Merging of the reconstructions

4.7.1 Order of the merging

The number of common points from the object between the reconstructions of takes i, j is $n^o_{i,j}$, the number of common points from the background between the reconstructions i, j is $n^b_{i,j}$. A graph is built whose vertices are the takes. The weight of the edge between the vertices i, j is $w_{i,j} = \min(n^o_{i,j}, n^b_{i,j})$. If the weight should be zero, the edge does not exist. The first two takes in the order are the takes a, b which are adjacent to the edge with the highest weight. One of these takes is selected as the reference take. These two vertices are collapsed afterward. Weight of the edges between this collapsed vertex $\{a, b\}$ and an another vertex c is $w_{\{a,b\},c} = w_{a,c} + w_{b,c}$.

The take which is connected to the collapsed vertex with the edge with the highest weight is selected. This take is added as the next one in the order and it is merged with the collapsed vertex in the same way as the vertex was originally collapsed. This is repeated until all vertices are collapsed or there is no vertex connected to the collapsed vertex.

The order is determined greedily, however it is not necessary for each reconstruction to have common points with the reconstruction of the reference take.

4.7.2 Merging of the points

The final coordinate system is the system of the reference take r. The other reconstructions are transformed to this system in the given order. For each take two transformations are necessary, one for the object points and another one for the background points. $T^O_{r,s}$ is a set of all tracks from the object which contain points in both the reference take and in the take s which is to be transformed. P^O_s is a sequence of points from the take s which are contained in a track from $T^O_{r,s}$. $\{p^s_i, p^r_i\}^n_{i=1}$ is a sequence of pairs of corresponding points from $T_{r,s}$. The first point is from the take s, the second point is from the take r. The task is to find the scale, rotation and translation which transforms each p^s_i to its corresponding p^r_i as $p^r_i = \sigma R p^s_i + \overrightarrow{t}$. The transformation is

found using the least squares algorithm [31].

The centroids C_s of the points from $\{p_i^s\}_{i=1}^n$ and C_r of the points from $\{p_i^r\}_{i=1}^n$ are found and sequences $\{p_i'^s\}_{i=1}^n$, $\{p_i'^r\}_{i=1}^n$ are obtained by subtracting the centroids from the points from the sequences. For each i the scale σ_i is obtained as:

$$\sigma_i = \frac{||p_i'^r||}{||p_i'^s||} \tag{4.39}$$

The estimated scale transformation $\sigma_{s,r}$ from the take s to the reference take is the median from the sequence $\{\sigma_i\}_{i=1}^n$.

The sequence $\{p_i''^s\}_{i=1}^n$ is obtained from the sequence $\{p_i'^s\}_{i=1}^n$ by multiplication of the vectors p_i' by the scale $\sigma_{s,r}$. P_s is a matrix whose rows are the vectors from the sequence $\{p_i''^s\}_{i=1}^n$, P_r is a matrix whose rows are the vectors from the sequence $\{p_i'^r\}_{i=1}^n$. The task is to find a rotation R which transforms the rows from P_s to the rows from P_r such that the square of the errors $\sum_{i=1}^n ||Rp_i''^s - p_i'^r||^2$ is minimized. The least squares estimate of such matrix is obtained by:

$$H = P_s^T P_R \tag{4.40}$$

The resulting matrix H however may not be a rotation. In order to get the nearest rotation, SVD is used to make the eigenvalues equal to 1.

$$USV^T = H \tag{4.41}$$

$$R_{s,r}^O = UV^T \tag{4.42}$$

For each i the translation $\overrightarrow{o}_i$ is obtained as:

$$\overrightarrow{o}_i = \overrightarrow{p}_i^r - \sigma_{s,r} R_{s,r}^O \overrightarrow{p}_i^s \tag{4.43}$$

The estimated translation $\overrightarrow{o}_{s,r}^O$ is the median of the vectors $\{\overrightarrow{o}_i\}_{i=1}^n$.

The transformation of the background $\sigma_{s,r}, R_{s,r}^B, \overrightarrow{o}_{s,r}^B$ is computed analogously from the set $T_{r,s}^B$ of all tracks from the object which contain points in both the reference take and in the take s. All points from the take s are transformed according to the estimated transformations and they are added to the reference take r. The points from the following takes are transformed using the references to the original points as well as to the points which have been added to the reference take during the merging of one of the previous takes.

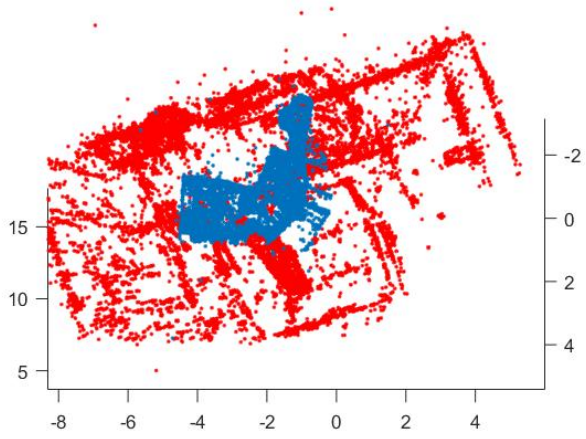


Figure 4.11: Points merged from 8 reconstructions of takes.

4.7.3 Calculation of the object motion from the transformations

s the take from whose coordinate system we transform

r the reference take to which we transform

$(\sigma_{s,r}, R^B_{s,r}, \overrightarrow{o}^B_{s,r})$ transformation of points of the background from the take s to the reference take r

$(\sigma_{s,r}, R^O_{s,r}, \overrightarrow{o}^O_{s,r})$ transformation of points of the object from the take s to the reference take r

X^s_O a point from the reconstruction s from the object

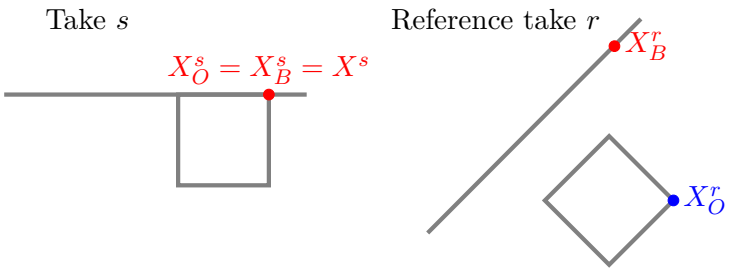
X^s_B a point from the reconstruction s from the background which is on the same position as the point X^s_O: $X^s_O = X^s_B$

X^r_O the point X^s_O transformed by $(\sigma_{s,r}, R^O_{s,r}, \overrightarrow{o}^O_{s,r})$

X^r_B the point X^s_B transformed by $(\sigma_{s,r}, R^B_{s,r}, \overrightarrow{o}^B_{s,r})$

$(A_{r,s}, \overrightarrow{b}_{r,s})$ motion of the object from the take r to the take s

$(A_{s,r}, \overrightarrow{b}_{s,r})$ motion of the object from the take s to the take r



The points X_O^s, X_B^s are on the same position, they can be replaced by a single point X^s, therefore we can write:

$$X_B^r = \sigma_{s,r} R_{s,r}^B X^s + \overrightarrow{o}_{s,r}^B \tag{4.44}$$

$$X_O^r = \sigma_{s,r} R_{s,r}^O X^s + \overrightarrow{o}_{s,r}^O \tag{4.45}$$

If the object was transformed from the position in the take r to the position in the take s, it would be on the same position where X_B^r is.

$$X_B^r = A_{r,s} X_O^r + \overrightarrow{b}_{r,s} \tag{4.46}$$

$$A_{r,s}(\sigma_{s,r} R_{s,r}^O X + \overrightarrow{o}_{s,r}^O) + \overrightarrow{b}_{r,s} = \sigma_{s,r} R_{s,r}^B X + \overrightarrow{o}_{s,r}^B \tag{4.47}$$

This equation has to hold true for all X, therefore also for the zero vector, therefore:

$$A_{r,s} \overrightarrow{o}_{s,r}^O + \overrightarrow{b}_{r,s} = \overrightarrow{o}_{s,r}^B \tag{4.48}$$

$$A_{r,s} \sigma_{s,r} R_{s,r}^O = \sigma_{s,r} R_{s,r}^B \tag{4.49}$$

As the scaling $\sigma_{s,r}$ is the same for the transformation of the object points and the background points, $A_{r,s}$ can be found from equation (4.49) as:

$$A_{r,s} = R_{s,r}^B (R_{s,r}^O)^{-1} \tag{4.50}$$

$\overrightarrow{b}_{r,s}$ can be found from equations (4.48) and (4.50) as:

$$\overrightarrow{b}_{r,s} = \overrightarrow{o}_{s,r}^B - R_{s,r}^B (R_{s,r}^O)^{-1} \overrightarrow{o}_{s,r}^O \tag{4.51}$$

The inverse motion $(A_{s,r}, \overrightarrow{b}_{s,r})$ transforms each point back to the original position:

$$Y = A_{s,r}(A_{r,s} Y + \overrightarrow{b}_{r,s}) + \overrightarrow{b}_{s,r} \tag{4.52}$$

This has to hold for every Y, so:

$$0 = A_{s,r} \overrightarrow{b}_{r,s}) + \overrightarrow{b}_{s,r} \tag{4.53}$$

$$I = A_{s,r}A_{r,s} \tag{4.54}$$

From equations (4.53), (4.54) the inverse motion $(A_{s,r}, \overrightarrow{b_{s,r}})$ can be found as:

$$A_{s,r} = A_{r,s}^{-1} = R_{s,r}^{O}(R_{s,r}^{B})^{-1} \tag{4.55}$$

$$\overrightarrow{b}_{s,r} = -A_{s,r}\overrightarrow{b}_{r,s} = -R_{s,r}^{O}(R_{s,r}^{B})^{-1}(\overrightarrow{o}_{s,r}^{B} - R_{s,r}^{B}(R_{s,r}^{O})^{-1}\overrightarrow{o}_{s,r}^{O}) \tag{4.56}$$

$$\overrightarrow{b}_{s,r} = \overrightarrow{o}_{s,r}^{O} - R_{s,r}^{O}(R_{s,r}^{B})^{-1}\overrightarrow{o}_{s,r}^{B} \tag{4.57}$$

4.7.4 Merging of the cameras

P_s a camera in the reconstruction of the take s

P_r^B the camera after the transformation to the reference take towards the background points

P_r^O the camera after the transformation to the reference take towards the object points

I image from which the camera P_s arises

t take from which the camera P_s and therefore also the image I arises

s the reconstruction onto which the camera P has been registered

d descent of the camera P; if $s = t$, the camera is in the anchor take, so $d = A$, otherwise the camera has been registered using the sequential PnP. If it was registered towards the background, then $d = B$, if it was registered towards the background, then $d = O$

R_s rotation of the camera P_s in the original position

$\overrightarrow{c}_s$ centre of the camera P_s in the original position

R_r^B rotation of the camera P_r^B in the coordinate system of the reference take r towards the background points

$\overrightarrow{c}_r^B$ centre of the camera P_r^B in the coordinate system of the reference take r towards the background points

R_r^O rotation of the camera P_r^O in the coordinate system of the reference take r towards the object points

$\overrightarrow{c}_r^O$ centre of the camera P_r^O in the coordinate system of the reference take r towards the object points

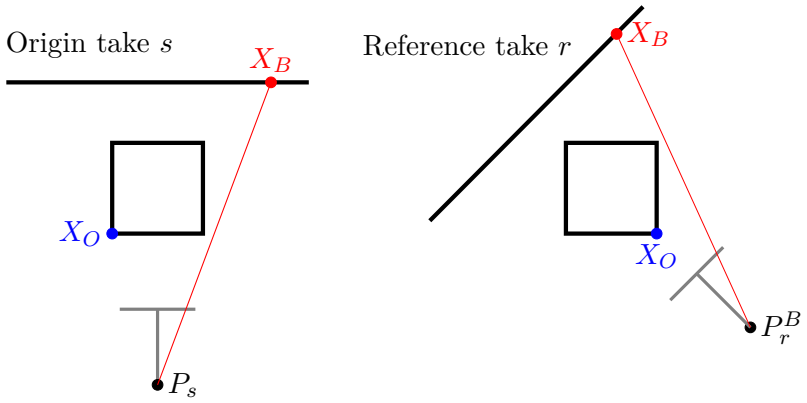
At first, the cameras are transformed to the reference take r to the position towards the background. The procedure of the transformation of the camera P_s to P_r^B however differs depending on t, s and o.

$s = r; d \in \{A, B\}$

The camera is already in the desired position, no transformation is necessary.

$s \neq r; d \in \{A, B\}$

The camera is in another coordinate system.



The transformation to the reference take is performed according to the equations (4.17), (4.18) as:

$$R_r^B = R_s(R_{s,r}^B)^{-1} \tag{4.58}$$

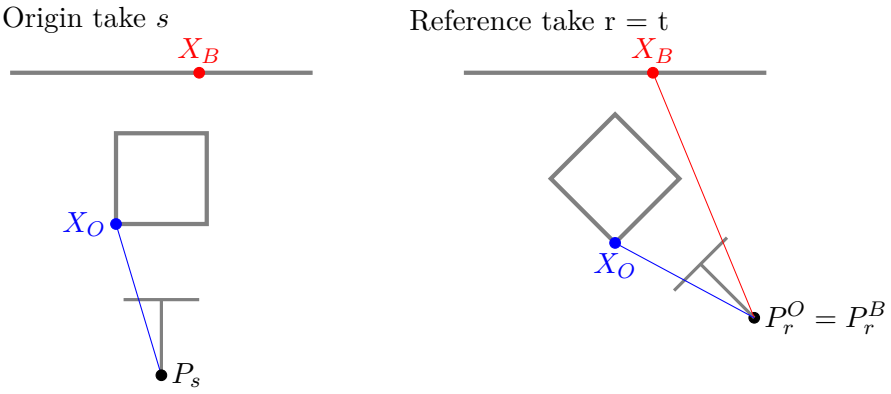
$$R_r^B \overrightarrow{c}_r^B = R_r^B \overrightarrow{o}_{s,r}^B + \sigma_{s,r} R_s \overrightarrow{c}_s \tag{4.59}$$

$$\overrightarrow{c}_r^B = \overrightarrow{o}_{s,r}^B + \sigma_{s,r}(R_r^B)^{-1} R_s \overrightarrow{c}_s \tag{4.60}$$

The transformation $(\sigma_{s,r}, R_{s,r}^B, \overrightarrow{o}_{s,r}^B)$ transforms the points from the background in the coordinate system s to the points in the reference take r. The camera in the transformed pose $(R_r^B, \overrightarrow{c}_r^B)$ therefore observes the points from the background in the reference take, which is the desired result.

$s \neq r; o = O; t = r$

The camera comes from the reference take r but it has been registered onto another reconstruction towards the object points.



The pose of the camera P_r^O observing the object points in the reference take transformed by $(\sigma_{s,r}, R_{s,r}^O, \overrightarrow{c}_{s,r}^O)$ can be computed according to the equations (4.17), (4.18):

$$R_r^O = R_s (R_{s,r}^O)^{-1} \tag{4.61}$$

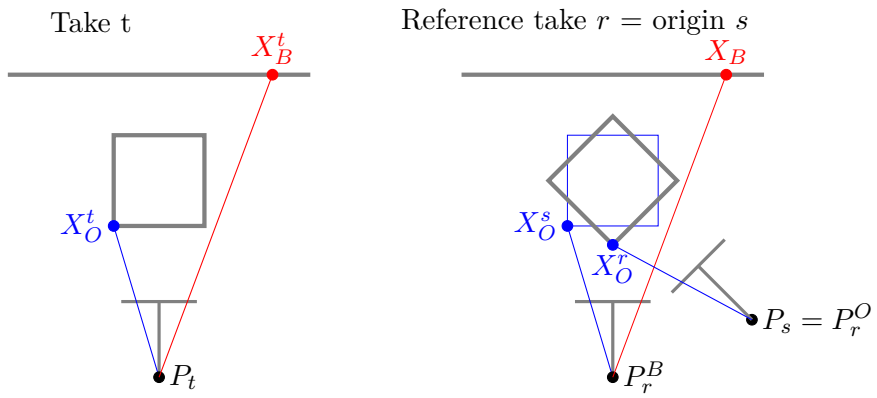
$$R_r^O \overrightarrow{c}_r^O = R_r^O \overrightarrow{o}_{s,r}^O + \sigma_{s,r} R_s \overrightarrow{c}_s \tag{4.62}$$

$$\overrightarrow{c}_r^O = \overrightarrow{o}_{s,r}^O + \sigma_{s,r} (R_r^O)^{-1} R_s \overrightarrow{c}_s \tag{4.63}$$

The pose of the transformed camera is towards the object points. But because the camera arises from the reference take, the pose towards the object and the background in the reference take is the same, so $R_r^B = R_r^O$, $\overrightarrow{c}_r^O = \overrightarrow{c}_r^B$.

$s = r; o = O; t \neq r$

The camera is already in the target coordinate system, but it registered towards the object. It has to be moved to the position where it would observe the background points.



The camera P_s observes a point X on the position X_O^r where it is in the reference take. If the camera was registered towards the background, it would observe the point on its original position X_O^s.

$$X_O^s = A_{r,t} X_O^r + \overrightarrow{b}_{r,t} \tag{4.64}$$

Where $A_{r,t}$, $\overrightarrow{b}_{r,t}$ are calculated according to equations (4.55), (4.57). Rotation and centre of the camera registered towards the background points can be found according to equations (4.6), (4.7) as:

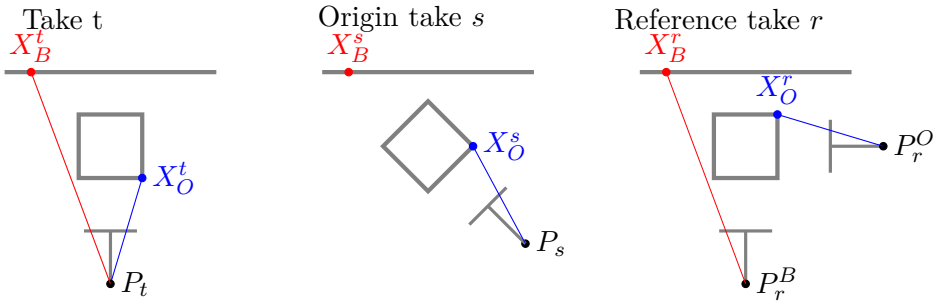
$$R_r^B = R_s A_{r,t}^{-1} = R_s A_{t,r} \tag{4.65}$$

$$R_r^B \overrightarrow{c}_r^B = R_r^B \overrightarrow{b}_{r,t} + R_s \overrightarrow{c}_s \tag{4.66}$$

$$\overrightarrow{c}_r^B = \overrightarrow{b}_{r,t} + (R_r^B)^{-1} R_s \overrightarrow{c}_s \tag{4.67}$$

$s \neq r; o = O; t \neq r$

The camera does not come from the reference take, so the transformation $(\sigma_{s,r}, R_{s,r}^O, \overrightarrow{c}_{s,r}^O)$ does not bring the desired result. This case can however be solved as a combination of the two previous cases.



If the camera P_s is transformed with $(\sigma_{s,r}, R^O_{s,r}, \overrightarrow{c}^O_{s,r})$ according to equations (4.61), (4.63) to the reference take r, the transformed camera P^O_r observes the object points of the reference take on the same features where the original camera P_s observed the object points in the take s.

$$R^O_r = R_s(R^O_{s,r})^{-1} \tag{4.68}$$

$$\overrightarrow{c}^O_r = \overrightarrow{o}^O_{s,r} + \sigma_{s,r}(R^O_r)^{-1}R_s\overrightarrow{c}_s \tag{4.69}$$

This converts the problem to the previous one where the task is to transform the camera P^O_r observing the object to the camera P^B_r which observes the background. This can be done according to the equations (4.65), (4.67) where $A_{r,t}$, $\overrightarrow{b}_{r,t}$ are calculated according to equations (4.55), (4.57):

$$R^B_r = R^O_r A^{-1}_{r,t} = R^O_r A_{t,r} \tag{4.70}$$

$$\overrightarrow{c}^B_r = \overrightarrow{b}_{r,t} + (R^B_r)^{-1}R_s\overrightarrow{c}_s \tag{4.71}$$

Camera averaging

After all the cameras are transformed to the reference take r to the pose towards the points from the background, all cameras which arise from the same image I should have the same pose.

A_I is a set of all transformed cameras P^B_r which arise from the image I. R_I is rotation calculated as the median of the Euler vectors which represent the rotations R^B_r of the cameras from A_I, $\overrightarrow{c}_I$ is camera center calculated as the median of the centers $\overrightarrow{c}^B_r$ of the cameras from A_I. All cameras from A_I can therefore be merged to one camera P_I with pose $(R_I, \overrightarrow{c}_I)$ and whose observations is a union of all observations from all cameras from A_I

If the take t from which the image I arises is the reference take r, the camera P_I observes points from both the background and the object, therefore the merging is finished. But if $t \neq r$, the camera P_I observes only the points from the background. In this case the camera needs to be split into two cameras P^B_I, P^O_I, where P^B_I observes the background and P^O_I observes the object.

The pose of the camera P_I^B is the same as the pose of the camera P_I. The pose of the camera P_I^O is the pose towards the object. It can therefore be calculated from the pose of the camera towards the background P_I as the inverse of the equations (4.65), (4.67) which compute the pose towards the background from the pose towards the object:

$$R_I^O = R_I^B A_{r,t} \tag{4.72}$$

$$R_I^O \overrightarrow{c}_I^O = R_I^B \overrightarrow{c}_I^B - R_I^B \overrightarrow{b}_{r,t} \tag{4.73}$$

$$\overrightarrow{c}_I^O = A_{r,t}^{-1}(\overrightarrow{c}_I^B - \overrightarrow{b}_{r,t}) \tag{4.74}$$

The observations of the camera P_I are split in such way, that the camera P_I^B observes the points which belong to the background and the camera P_I^O observes the points which belong to the object.

4.8 Bundle adjustment

The merged reconstruction is used as an input for the final bundle adjustment so it can be further improved. The bundle adjustment is not a standard one which is used in the single body pipelines, because there are specific constraints between some cameras.

k number of takes from which the reconstruction has been merged

r the reference take

$\{I_i\}_{i=1}^n$ sequence of the images from which the reconstruction has been created

t_i take from which the image I_i arises

$\{X_i^B\}_{i=1}^m$ sequence of the 3D points

d_i descent of the 3D point; $d_i = O$ if the point is from the object and $d_i = B$ if the point is from the background

$(A_{r,t}, \overrightarrow{b}_{r,t})$ motion of the object from its position in the take r to its position in the take t

P_i^B camera which arises from the image I_i and observes the points from the background

$(R_i^B, \overrightarrow{c}_i^B)$ pose of the camera P_i^B

P_i^O camera which arises from the image I_i and observes the points from the object; if $t_i = r$, then $P_i^O = P_i^B$, otherwise must according to equations (4.72), (4.74) hold true $R_i^O = R_i^B A_{r,t_i}$, $\overrightarrow{c}_i^O = A_{r,t_i}^{-1}(\overrightarrow{c}_i^B - \overrightarrow{b}_{r,t_i})$

$\{O_i\}_{i=1}^l$ sequence of the observations; an observation O_i is a triplet $(x_i, p_i, \overrightarrow{f}_i)$ where x_i is index of the 3D point which is observed, p_i is index of the camera which observes the point and $\overrightarrow{f}_i$ is a 2D feature onto which the point is projected;

Parameters of the BA are all 3D points $\{X_i\}_{i=1}^m$, the poses of the cameras $\{P_i^B\}_{i=1}^n$ which observe the background points and the motions $\{(A_{r,t}, \overrightarrow{b}_{r,t})\}_{t=1,t\neq r}^k$. Poses of the cameras $\{P_i^O\}_{i=1}^n$ are not parameters as they can be computed from the cameras P_i^B and from the motions $(A_{r,t}, \overrightarrow{b}_{r,t})$ and because we want to ensure that the motion of the object from the take t to r computed using equations (4.6), (4.8) is the same if computed from any camera pair arising from the take t.

The task is to find such parameters which minimize the sum of squares of the reprojection errors over all observations

$$argmin(\sum_{i=1}^{d} ||\overrightarrow{f}_i - q_i||^2) \tag{4.75}$$

Where q_i is the projection of the point X_{x_i} onto the camera $P_{p_i}^B$ if $d_{x_i} = B$ and the projection of the point X_{x_i} onto the camera $P_{p_i}^O$ if $d_{x_i} = O$. The projections are found according to Section 2.2.

Two different iterative approaches to the BA have been tried. The first one is the classical gradient descent which improves all parameters in each iteration. The other one is alternating minimization, where in the first iteration the cameras are fixed and the points are improved and in the next one the points are fixed and the cameras are improved; this is repeated until convergence. These approaches are evaluated in Section 5.4.

4.9 Filtering of the reconstruction

This is an arbitrary step, which may improve the quality of the reconstruction in the case of a high number of misclassified points in the reconstruction.

If a point is misclassified (arises from the background but is assigned to the object or vice versa), it will probably be observed by some of the cameras with a high reprojection error. This method requires a threshold t and the points, whose reprojection error after the projection onto an arbitrary camera is greater than t, are removed from the reconstruction.

This method works correctly only if the cameras are estimated correctly, otherwise it would remove the correctly classified points, as well, which would make the quality of the reconstruction worse. From the same reason a high threshold t is used, which still removes the most of the misclassified points. The filtering may be performed before or after the BA.

Chapter 5

Experiments

5.1 Overview of used objects and backgrounds

In order to prove that our method works well, we have selected several objects and we have created a dataset of images for each of the objects. We have selected appropriate backgrounds for the objects according to [29]. Each dataset is divided into 4 to 8 takes, each of which depicts a static configuration of the object towards the background, as it is described in 1.3.

Object	Background	Takes	Images	Special	Figure
Daliborka	Nissan GTR	8	338	NONE	5.1(a)
Lycan	Nissan GTR	8	140	NONE	5.1(b)
Colosseum	Peugeot 908	8	294	NONE	5.1(c)
Vatican	Peugeot 908	8	270	NONE	5.1(d)
Ganesha	Peugeot 908	4	160	Only translation	5.1(e)
Salt lamp	Seat Ibiza	8	330	Translucent	5.1(f)
Buddha	Peugeot 908	8	300	Shiny	5.1(g)
Pillow	Seat Ibiza	8	278	Repetitive	5.1(h)
Transformer	Audi S8	8	329	NONE	5.2(a)
Ship	Audi S8	8	277	NONE	5.2(b)
Catalog	Seat Ibiza	5	150	Planar	5.2(c)
Lego	Seat Ibiza	4	264	Repetitive, Planar	5.2(d)

Table 5.1: Properties of the datasets.

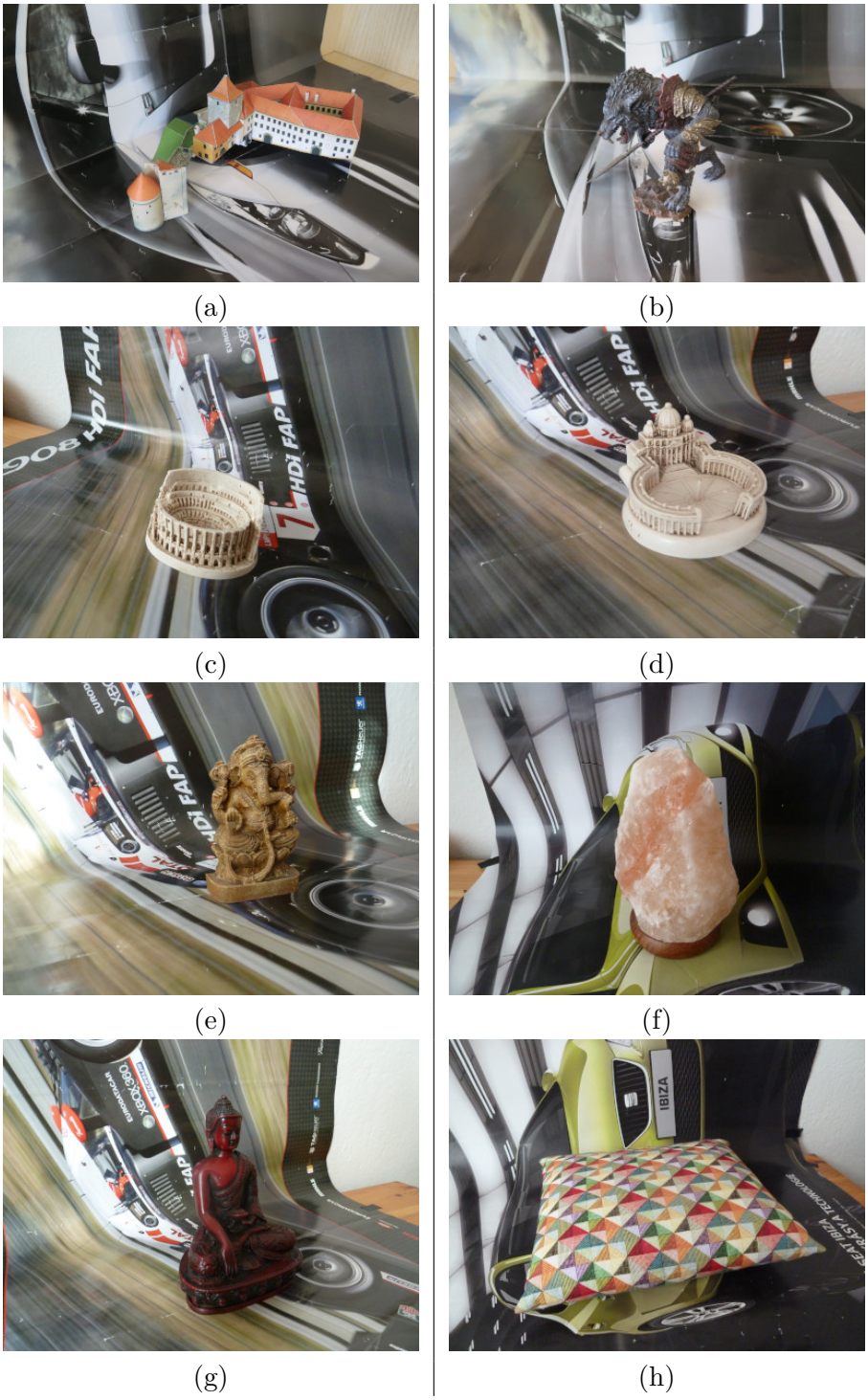


Figure 5.1: Images taken from the datasets, which are used for reconstruction. Part 1.

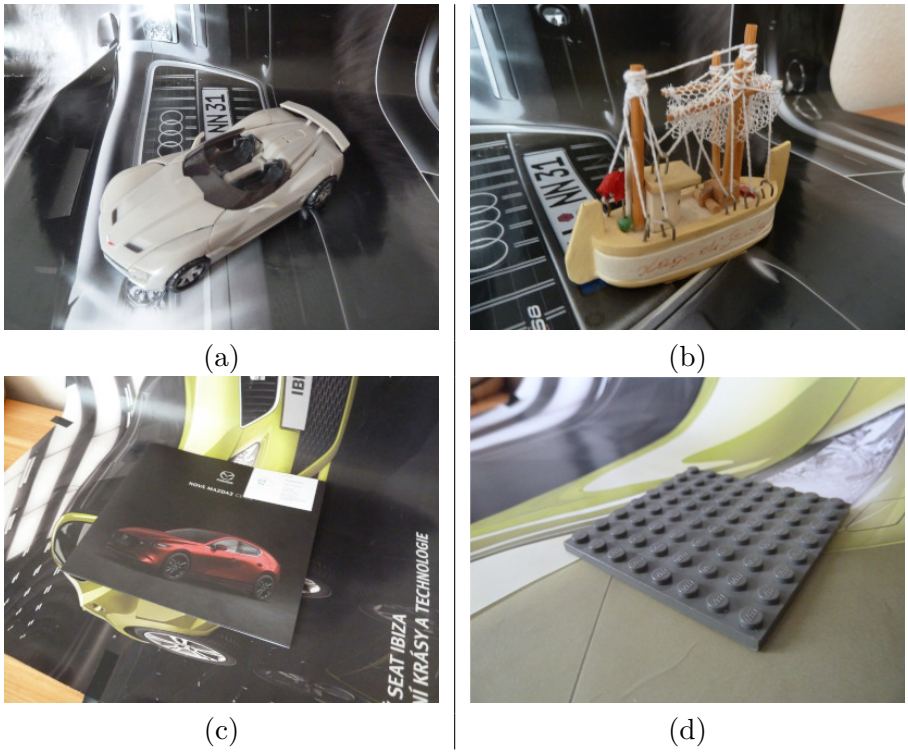


Figure 5.2: Images taken from the datasets, which are used for reconstruction. Part 2.

5.2 Qualitative results of the method

Except for the Buddha dataset, the datasets have been successfully reconstructed by our method. Table 5.2 and figures 5.3, 5.4 depict the results of the reconstructions. The blue points belong to the object and the red ones to the background. According to Section 4.6.2 the only criterion to label the background and the object is the order of the sequential PnP registration. The registration can be arbitrary, so the labels of the background and the object can be swapped, which actually happened at the datasets 5.4(a), 5.4(c), 5.4(f).

The table shows whether the background and the object have been swapped as well as numbers of the points from the object and from the background. If the labels of the background and the object have been swapped, Points Obj. is the number of points from the true object, which has been labeled as the background.

The reconstruction of the Buddha dataset has failed probably due to the lack of features on one side of the object so the sequential PnP could not register towards the object and the cyclic check could not be performed.

Dataset	Figure	Points Obj.	Points Bck.	Swapped
Daliborka	5.3(b)	15904	16553	NO
Lycan	5.3(d)	10286	8138	NO
Colosseum	5.4(b)	23535	25990	NO
Vatican	5.4(d)	17175	15282	NO
Ganesha	5.4(f)	19199	10528	YES
Salt lamp	5.4(h)	2322	24235	NO
Pillow	5.5(b)	92166	18335	YES
Transformer	5.5(d)	2070	12606	NO
Ship	5.5(f)	4665	9879	NO
Catalog	5.5(h)	5047	8421	YES
Lego	5.6(b)	354	7765	NO

Table 5.2: Results of the reconstructions.

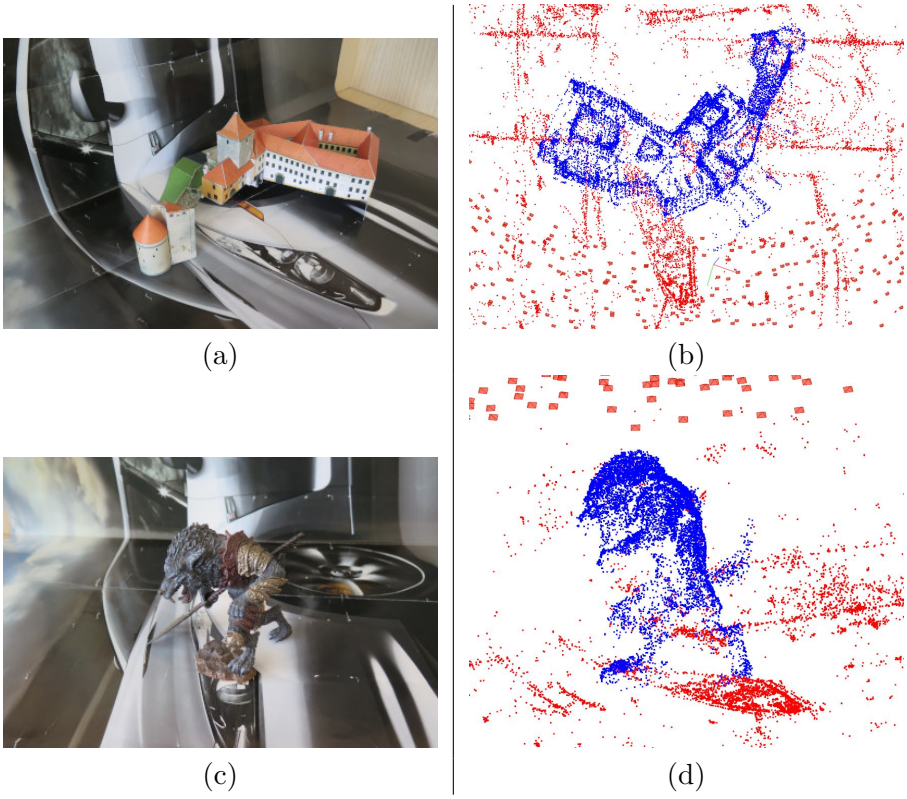


Figure 5.3: Reconstructed datasets, Part 1.

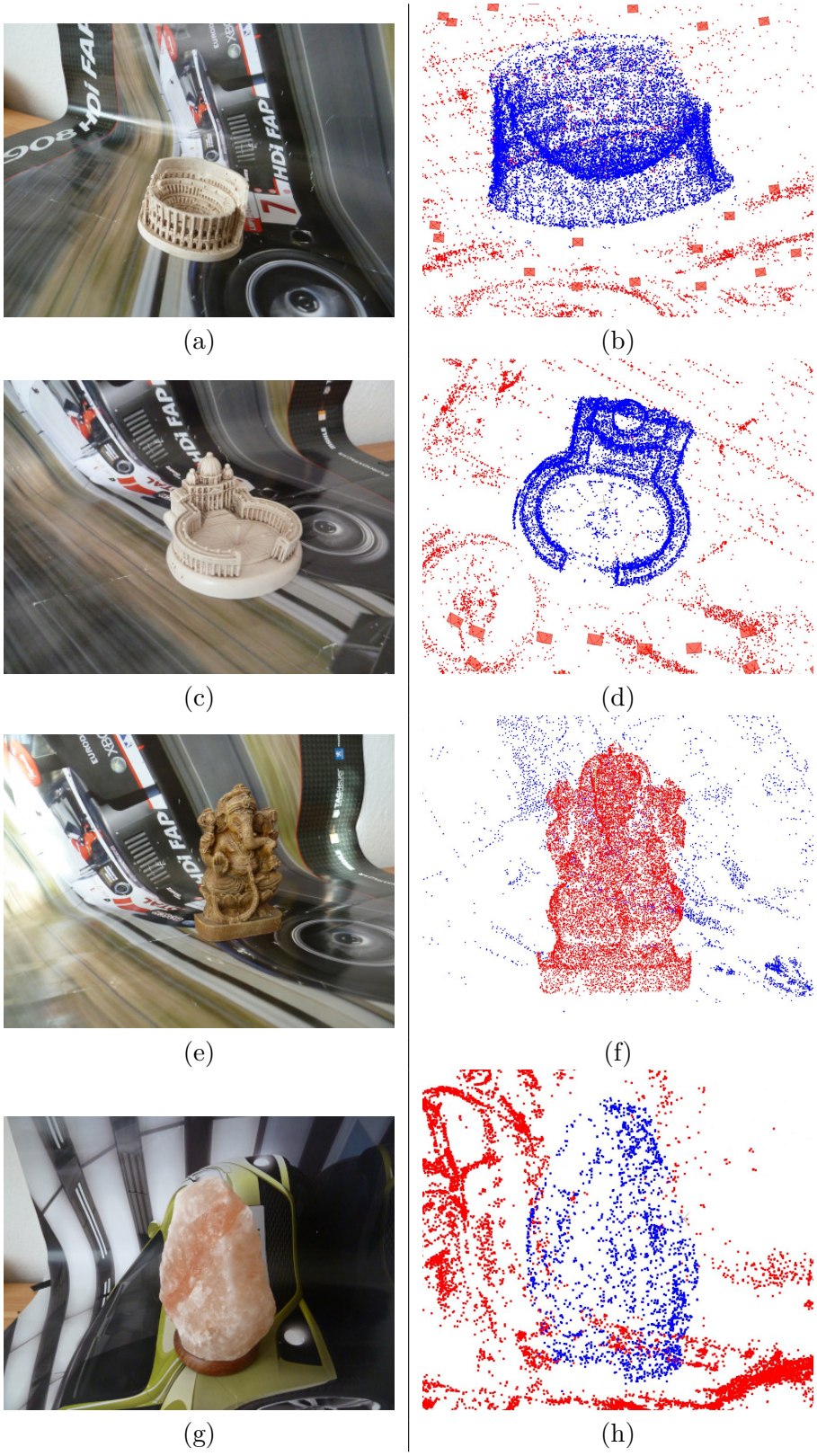
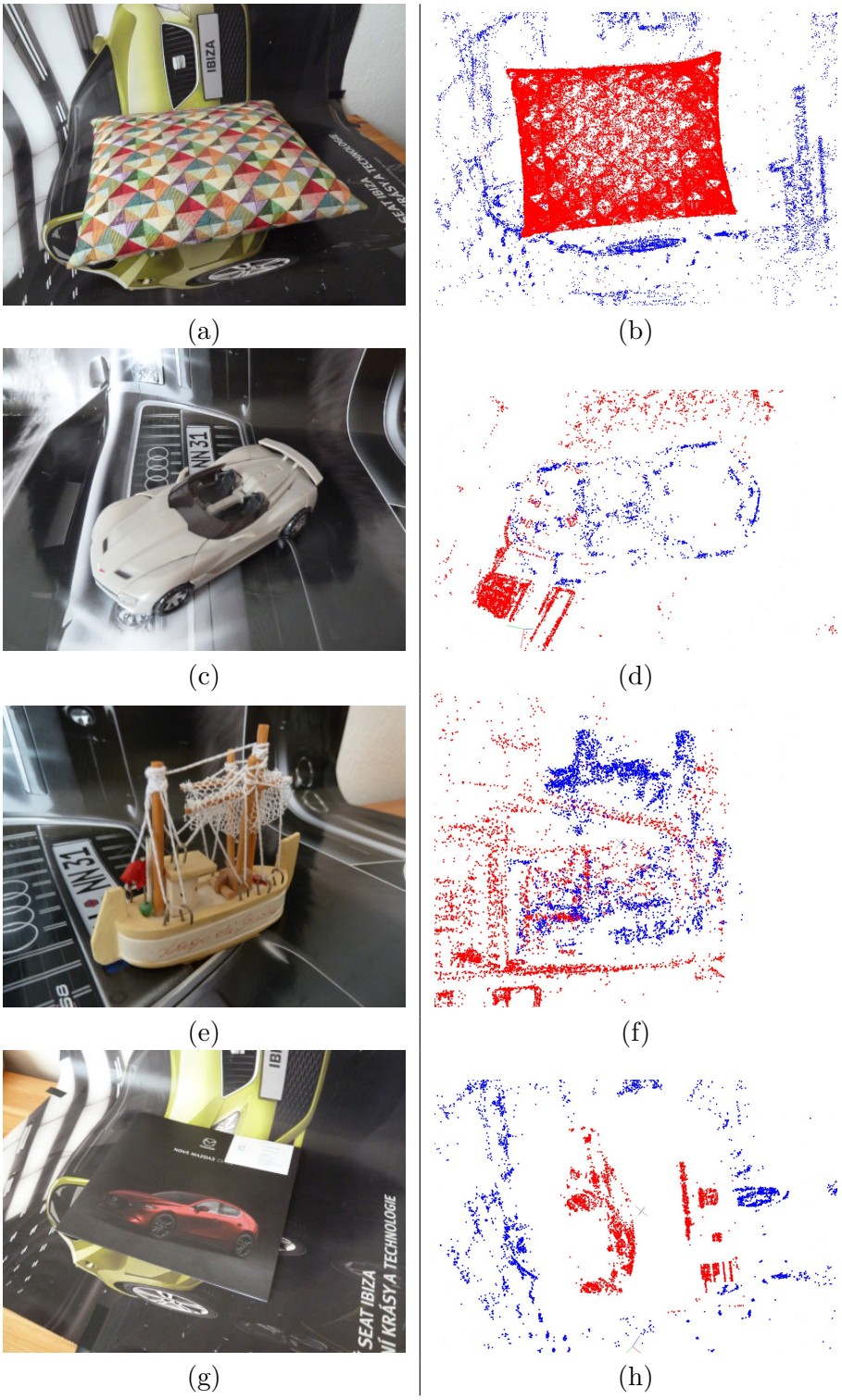
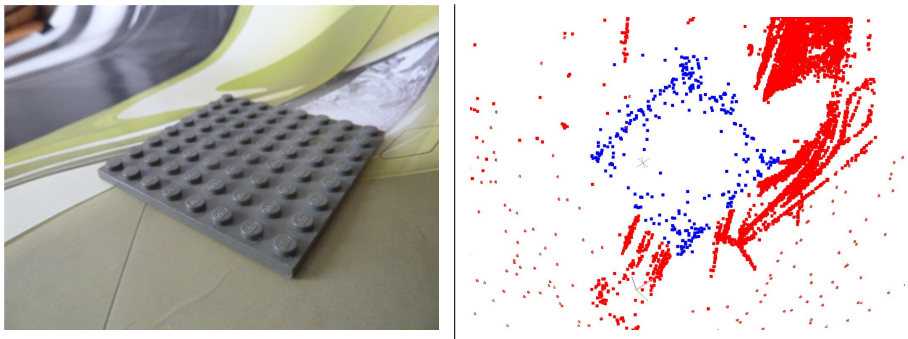


Figure 5.4: Reconstructed datasets, Part 2.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 5.5: Reconstructed datasets, Part 3.



(a) (b)

Figure 5.6: Reconstructed datasets, Part 4.

Quality of the models which have been successfully reconstructed differs. The main reason for the lower quality of the reconstruction is the lower density of features, which is especially apparent at the Transformer dataset depicted in Figure 5.2(a). An interesting result is the reconstruction of a planar object "Catalog" which is depicted in Figure 5.2(b). The object "Lego" in Figure 5.2(b) could not be reconstructed with the default settings of COLMAP, it was however reconstructed with a lowered threshold for minimal inlier ratio in the PnP registration, which allowed registration of the images towards the object.

5.3 Comparison to the single body SfM

Apart from the datasets which depict the object on some background, we have reconstructed some of the objects without any background in the classical single body SfM COLMAP [3]. The motivation was to determine whether and eventually in which cases can the presence of the background actually help with the reconstruction of the object. The numbers of the images in the datasets are given in Table 5.3. The models reconstructed with the background have been adjusted using the alternating minimization, which is proven to be more suitable in Section 5.4.

For each object, we compare two models, first of which has been reconstructed with the background using our method, while the second one has been reconstructed without the background using a standard SfM pipeline. The parameters which we compare are the number of reconstructed points, the median reprojection error, and the quality of the reconstruction (e.g.

Dataset	MBSfM	Single body
Colosseum	294	326
Vatican	270	304
Ganesha	160	187
Salt lamp	330	302
Pillow	278	275
Transformer	329	290
Ship	277	294
Catalog	150	151
Lego	264	307

Table 5.3: Number of images from which the object is reconstructed

Figure 5.7: Example of an image from the single body dataset without the background.

whether all parts of the object have been reconstructed). Table 5.4 shows the numbers of points and Table 5.5 shows the reprojection errors.

Dataset	MBSfM	MBSfM filtered	Single body
Colosseum	23535	22932	49418
Vatican	17175	17094	38833
Ganesha	19199	19192	34539
Salt lamp	2322	2320	10049
Pillow	92166	86351	198728
Transformer	2070	2037	14372
Ship	4665	4636	20571
Catalog	5047	4466	10143
Lego	354	90	-

Table 5.4: Number of points from the object

It is apparent from Table 5.4 that the object reconstructed with the background using our method has usually fewer points than the same object reconstructed without the background. The possible reason for this is that only the points towards which a camera has been registered in the sequential PnP can be recognized to belong either to the background or to the object. Therefore some of the points which have been reconstructed with the background are not present in the final reconstruction. Another possible reason may be that the features on the object are ignored because of stronger features on the background. For some of the objects, the images in the dataset without the background outnumber the images in the dataset with the background (Table 5.3), but the difference is too low to make such a difference in the numbers of the points.

The most significant difference in the numbers of points is at the objects which have a low density of the features such as the Salt lamp, the Transformer (Figure 5.8(a), 5.8(b)) and the Ship. The difference at the objects which have a higher density of features is less significant and visually unrecognizable. (Figure 5.8(c) 5.8(d))

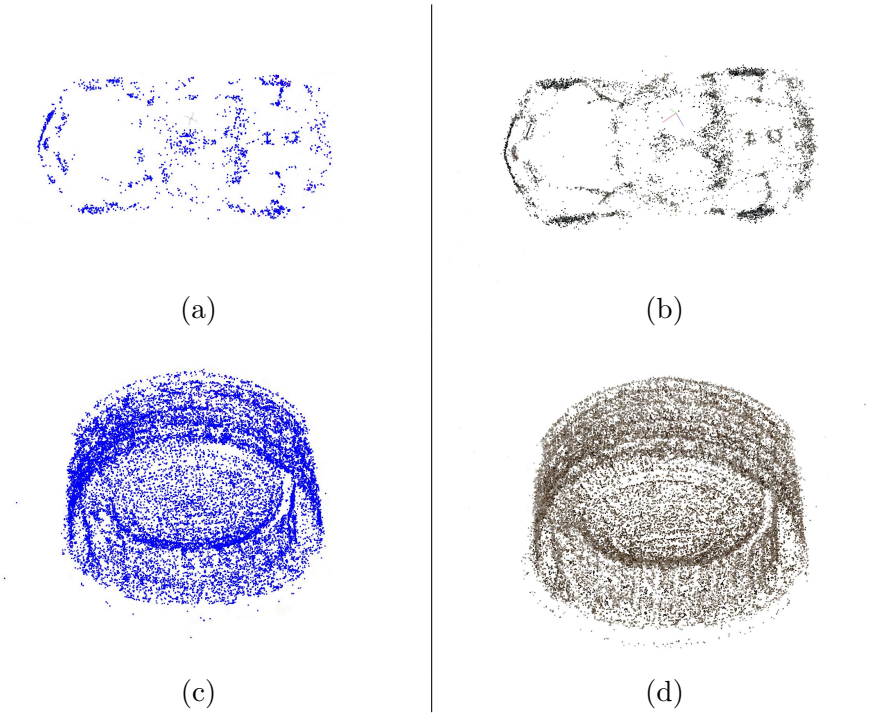


Figure 5.8: Comparison of the models reconstructed with and without the background. The models on the left side have been reconstructed with the background.

Dataset	MBSfM	MBSfM filtered	Single body
Colosseum	3.2662	3.2085	1.4825
Vatican	3.4622	3.4427	1.7924
Ganesha	0.8209	0.8208	1.8522
Salt lamp	4.1629	4.1610	1.8889
Pillow	1.0564	0.9710	2.3420
Transformer	6.3239	6.1637	2.6443
Ship	6.0914	6.0112	1.7392
Catalog	2.0319	2.0035	2.8318
Lego	20.4614	7.2846	-

Table 5.5: Median reprojection error E_O

For the majority of the objects, the reprojection error of the reconstruction with the background is higher than the error of the reconstruction without the background. The exceptions are the Ganesha, the Catalog and the Pillow.

5.3.1 Planar objects

Our method allows a reconstruction of planar objects by placing them onto a non-planar background. The object with the background are then reconstructed as a general object and we can afterward separate the object from the background. This has been demonstrated on the object "Catalog", whose reconstruction can be seen in Figure 5.9(a). However, the single-body pipeline COLMAP managed to reconstruct the dataset, as well (Figure 5.9(b)).

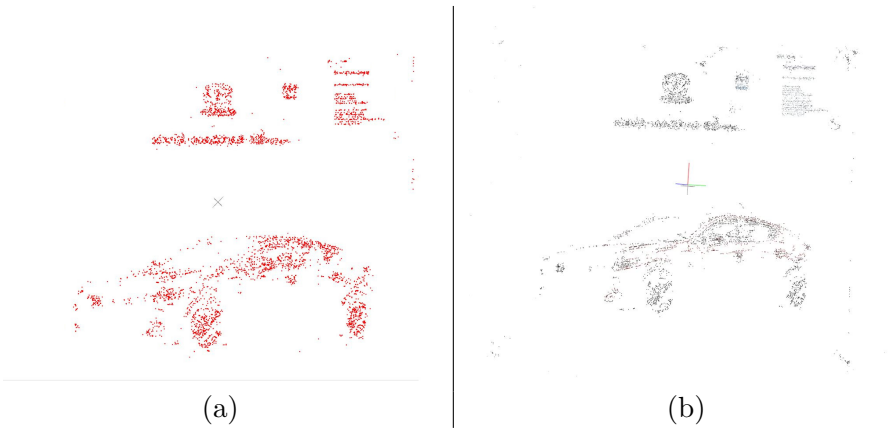


Figure 5.9: Comparison of the models of the "Catalog" object reconstructed with and without the background. The model (a) has been reconstructed with the background.

5.3.2 Repetitive objects

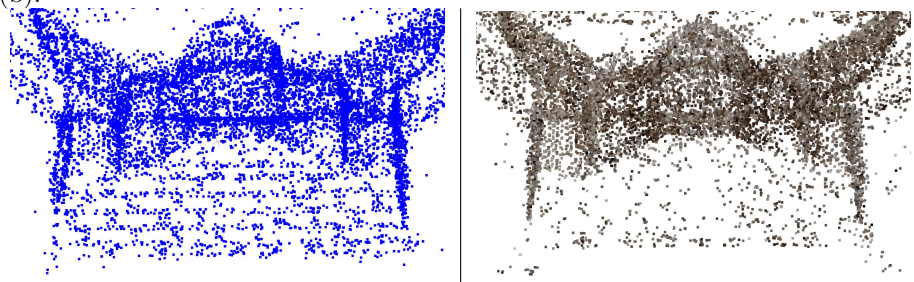
In most cases, the reconstruction without the background has higher a number of reconstructed points and lower reprojection errors at the same time. In some cases, however, our method can perform better. One of these cases are objects with repetitive patterns.

According to Table 5.5, a repetitive object "Pillow" is one of the objects which have lower reprojection error after the reconstruction with the background, than without the background.

Objects "Daliborka" and "Vatican" contain repetitive patterns (Figure 5.10). These patterns could be reconstructed using our method. Figure 5.11 shows that the repetitive back side of the Vatican object has been reconstructed better with the background than without it.

(a) (b)

Figure 5.10: Examples of repetitive patterns on "Daliborka" (a) and "Vatican" (b).



(a) (b)

Figure 5.11: Comparison of the reconstruction of the back side of the "Vatican" object. The model (a) has been reconstructed with the background.

The object "Lego" is repetitive and almost planar, and therefore it is difficult to reconstruct. The standard single body pipeline failed to find the initial

pair of cameras and therefore could not reconstruct the object. Our method managed to reconstruct the object, as the relative pose could be found using the points from the background. This model is shown in Figure 5.12

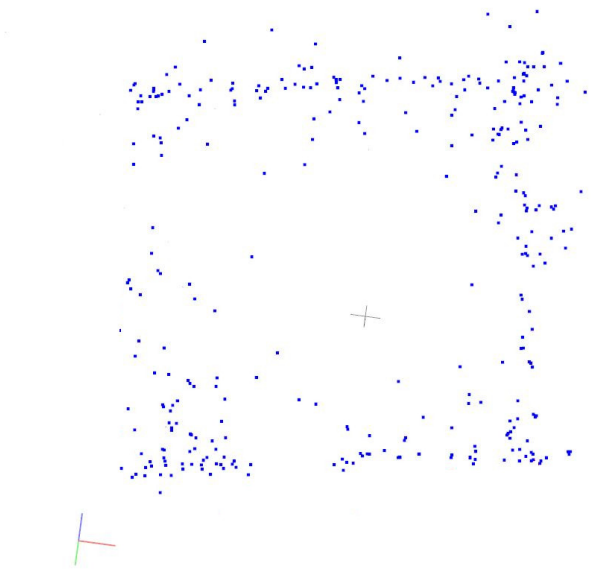


Figure 5.12: Model of the "Lego" object.

5.4 Review of approaches to the Bundle Adjustment

In Section 4.8 we have proposed two approaches to the bundle adjustment, namely the gradient descent and the alternating minimization. We are interested in the median reprojection error E, as well as in the partial reprojection errors E_O, E_B. E_O is calculated only from the observations of points which belong to the object and E_B is calculated only from the observations of the points which belong to the background. If the labels of the background and the object have been swapped, E_O is calculated from the true object, which has been labeled as the background.

For each dataset, we compare the values E, E_O, E_B of the model before the adjustment, after the adjustment using the gradient descent, and after the adjustment using the alternating minimization in order to determine which of the approaches is more appropriate. Table 5.6 shows values for E, Table 5.7 shows E_O and Table 5.8 shows E_B. The best value is bold; if the error after the adjustment is worse, the value is red.

Dataset	Original	Gradient	Alternating
Daliborka	3.4294	1.8704	**1.5888**
Lycan	2.3397	2.0448	**1.3930**
Colosseum	5.9503	3.4171	**2.6892**
Vatican	6.6690	3.2737	**2.8050**
Ganesha	2.8834	1.4558	**1.3600**
Salt lamp	8.9735	**2.3805**	2.4062
Pillow	7.8455	**1.4150**	1.4340
Transformer	15.9971	2.2689	**2.2509**
Ship	11.3332	2.8786	**2.8458**
Catalog	9.8959	4.8998	**4.5814**
Lego	9.6033	2.0808	**2.0773**

Table 5.6: Overall median reprojection error E

Dataset	Original	Gradient	Alternating
Daliborka	2.4018	3.3470	**2.1918**
Lycan	1.6981	3.5872	**1.6719**
Colosseum	3.3818	5.7818	**3.2662**
Vatican	3.5849	5.7980	**3.4622**
Ganesha	1.9734	0.8508	**0.8209**
Salt lamp	**3.4133**	7.4074	4.1629
Pillow	6.9275	**1.0436**	1.0564
Transformer	6.9450	9.2377	**6.3239**
Ship	6.5760	9.2265	**6.0914**
Catalog	5.5661	2.3659	**2.0319**
Lego	50.6459	20.7596	**20.4614**

Table 5.7: Median reprojection error E_O of the object

According to Table 5.6 the alternating minimization outperforms the gradient descent in the most cases. In addition to that, Table 5.7 shows, that in 7 of 11 cases the partial error E_O is made worse by the gradient descent. The same thing happens only at one dataset in the case of the alternating minimization. Therefore the alternating minimization is more suitable for our purpose.

Dataset	Original	Gradient	Alternating
Daliborka	4.7270	1.2718	**1.2373**
Lycan	3.3340	1.1679	**1.1408**
Colosseum	9.4377	2.3787	**2.3417**
Vatican	10.4341	2.5497	**2.5125**
Ganesha	7.5913	5.8102	**4.4917**
Salt lamp	9.5084	**2.2928**	2.3459
Pillow	13.8701	36.6333	**6.7692**
Transformer	16.8831	2.0978	**2.0942**
Ship	13.6002	**2.3304**	2.3812
Catalog	14.5464	**8.1427**	9.4299
Lego	9.4550	2.0490	**2.0440**

Table 5.8: Median reprojection error E_B of the background

Yet another advantage of the alternating minimization is that the reconstruction is naturally stabilized during the bundle adjustment, as in each step either the cameras or the points are fixed.

Chapter 6

Future work

We have solved a relaxed version of the MBSfM problem, where the number of the objects is limited to 2 and the images depict several static configurations of a scene and it is known which image belongs to which scene. This algorithm can be improved in many ways.

In order to dispose of the dependence on the thresholds in the clustering phase, the uncertainties of the cameras can be utilized. In that case, a series of experiments would have to be performed to show which of the options, or the combination of both, would perform better.

The solution can be generalized in such a way, that the number of moving objects would be arbitrary. In that case, it would be challenging to handle the situations where some of the objects remain static between some of the takes, while the other ones move. The algorithm can be improved such that it would be able to assign the images to their takes, so the labels of the images would not be necessary.

In order to improve the quality of the reconstruction, new points could be triangulated from camera pairs which do not belong to the same object. Many points have been reconstructed but they have not been assigned to the object nor to the background. An additional labeling of these points can be performed.

The sequential PnP can be replaced by another motion segmentation method. This algorithm can as well be used in a general version of the MBSfM.

Chapter 7

Conclusion

In this thesis we have proposed a solution to the problem introduced in Section 1.3, where the task is to reconstruct a scene consisting of two objects. This scene is captured by multiple takes, which depict different static configurations of the scene. We have implemented the method as an extension of a COLMAP pipeline [3] and we have demonstrated its functionality on real data.

We have reconstructed several objects using our method, where the background served as the second object. In most cases the reconstruction of an object with a background using our method produces a model, which is worse than the model reconstructed in a single-body pipeline without the background. We have however shown, that for some objects, especially the planar ones or the objects which contain repetitive structures, our method produces better models than the single-body SfM pipeline.

Bibliography

[1] R. Hartley and A. Zisserman. Multiple view geometry in computer vision. 2003.

[2] S. M. Seitz N. Snavely and R. Szeliski. Photo tourism: Exploring photo collections in 3d. *SIGGRAPH, pages 835–846*, 2006.

[3] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[4] Pierre Moulon, Pascal Monasse, Renaud Marlet, and Others. Openmvg. `https://github.com/openMVG/openMVG`.

[5] Tomáš Pajdla. Elements of geometry for robotics. 2019.

[6] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision, 2004*, 2004.

[7] A. Fussielo E. Maset, F. Arrigoni. Practical and efficient multi-view matching. *ICCV*, 2017.

[8] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with apphcatlons to image analysis and automated cartography. *ACM 0001-0782/81/0600-0381 00.75*, 1981.

[9] R. I. Hartley B. Triggs, P. F. McLauchlan and A. W. Fitzgibbon. Bundle adjustment - a modern synthesis. *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, 2000.

[10] T. E. Boult and L. G. Brown. Factorization-based segmentation of motions. *Visual Motion*, 1991.

[11] J. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *Int. Journal of Computer Vision, vol. 29, no. 3*, 1998.

[12] S. Soatto R. Vidal, Y. Ma and S. Sastry. Two-view multibody structure from motion. *IJCV, 68(1):725,*, 2006.

[13] R. Vidal S. Rao, R. Tron and Y. Ma. Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. *PAMI, 32(10):18321845,*, 2010.

[14] S. S. Sastry S. R. Rao, A. Y. Yang and Y. Ma. Robust algebraic segmentation of mixed rigid-body and planar motions from two views. *IJCV, 88(3):425-446*, 2010.

[15] M. Irani and P. Anandan. A unied approach to moving ob ject detection in 2d and 3d scenes. *IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 20, no. 6, pp. 577-589*, 1998.

[16] C. S. Kenney M. Zuliani and B. S. Manjunath. The multiransac algorithm and its application to detect planar homographies. *ICIP, pages 153-156*, 2005.

[17] K. Schindler and D. Ozden. Two-view multibody structure-and-motion with outliers through model selection. *IEEE Transactions on Pat-tern Analysis and Machine Intelligence, 28(6):983–995*, 2006.

[18] W. Zhang and J. Košecká. Nonparametric estimation of multiple structures with outliers. *European Conference on Computer Vision, volume 4358, pages 60–74*, 2006.

[19] R. Toldo and A. Fusiello. Robust multiple structures estimation with j-linkage. *European Conference on Computer Vision, volume 5302, pages 537–547*, 2008.

[20] L. Magri and A. Fusiello. T-linkage: a continuous relaxation of j-linkage for multi-model fitting. *Computer Vision and Pattern Recognition, pages 3954–3961*, 2014.

[21] Jin Yu Trung Thanh Pham, Tat-Jun Chin and David Suter. The random cluster model for robust geometric fitting. *Pattern Analysis and Machine Intelligence, 36(8):1658– 1671*, 2014.

[22] L. Magri and A. Fusiello. Robust multiple model fitting with preference analysis and low-rank approximation. *BMVC*, 2015.

[23] Z. Li X. Xu, L. F. Cheong. Motion segmentation by exploiting complementary geometric models. 2018.

[24] C. Rabe A. Wedel J. Klappstein, T. Vaudrey and R. Klette. Moving object segmentation using optical flow and depth information. *PSIVT, pages 611-623*, 2008.

[25] J. Ju H. Jung and J. Kim. Rigid motion segmentation using randomized voting. *CVPR, pages 1210-1217*, 2014.

[26] V. Murino C. Rubino, M. Crocco and A. D. Bue. Semantic multi-body motion segmentation. *WACV, pages 1145-1152*, 2015.

[27] Filip Šrajer. Image matching for dynamic scenes. *http://cmp.felk.cvut.cz/srajefil/theses/filip-srajer-diploma-thesis.pdf*, 2016.

[28] H Wang K Schindler, D Suter. A model-selection framework for multi-body structure-and-motion of image sequences. *International Journal of Computer Vision, 79(2):159–177*, 2008.

[29] P. Hruby and T. Pajdla. Multi-body structure from motion experiments. 2019.

[30] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing, 17 (4)*, 2007.

[31] D. S. Blostein K. S. Arun, T. S. Huang. Least-squares fitting of two 3-d point sets. *IEEE transactions on pattern analysis and machine intelligence, vol. PAMI-9, No. 5. September 1987*, 1987.