CENTER FOR
MACHINE PERCEPTION

CZECH TECHNICAL
UNIVERSITY IN PRAGUE

MASTER'S THESIS

Multi-Body Structure from Motion

Bc. Jan Krček

krcekjan@fel.cvut.cz

May 25, 2017

Thesis Advisor: Ing. Tomáš Pajdla, Ph.D.

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
fax +420 2 2435 7385, phone +420 2 2435 7637, www: http://cmp.felk.cvut.cz

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

DIPLOMA THESIS ASSIGNMENT

Student:	Bc. Jan K r č e k
Study programme:	Open Informatics
Specialisation:	Computer Vision and Image Processing
Title of Diploma Thesis:	Multi-Body Structure from Motion

Guidelines:

1. Review the state of the art in multibody structure from motion [1-11] and in particular analyze the algorithm used in [1].
2. Suggest and develop a method improving of the multi-body structure from motion over [1].
3. Implement the method, demonstrate its performance on real data.

Bibliography/Sources:

[1] F. Srajer. Image Matching for Dynamic Scenes. MSc Thesis, CTU 2016. https://dspace.cvut.cz/bitstream/handle/10467/64771/F3-DP-2016-Srajer-Filip-filip-srajer-diploma-thesis.pdf
[2] Tianwei Shen, Siyu Zhu, Tian Fang, Runze Zhang, and Long Quan.Graph-Based Consistent Matching for Structure-from-Motion. ECCV 2016.
[3] R. Vidal and R. Hartley. Three-view multibody structure from motion. PAMI, 30(2):214-227, Feb 2008.
[4] A. W. Fitzgibbon, A. Zisserman. Multibody Structure and Motion: 3-D Reconstruction of Independently Moving Objects. ECCV 2000 (http://www.robots.ox.ac.uk/~vgg/publications/2000/Fitzgibbon00/fitzgibbon00.pdf)
[5] P. Ji, H. Li, M. Salzmann, Y. Zhong. Robust Multi-body Feature Tracker: A Segmentation-free Approach. CoRR abs/1603.00110 (2016)
[6] Shubhangi L. Vaikole, S. D. Sawarkar. Moving Object Segmentation with camera in motion Using GMEC and Change Detection Method. JMPT 6(2): 53-60 (2015)
[7] C. Rubino, M. Crocco, V. Murino, A. Del Bue. Semantic Multi-body Motion Segmentation. WACV 2015: 1145-115.
[8] R. Sabzevari, D. Scaramuzza. Monocular simultaneous multi-body motion segmentation and reconstruction from perspective views. ICRA 2014: 23-30
[9] G. Pan, K-Y. K. Wong. Multi-body Segmentation and Motion Number Estimation via Over-Segmentation Detection. ACCV Workshops (2) 2010: 194-203, 2005.
[10] N. Thakoor, J. Gao, V. Devarajan. Multibody Structure-and-Motion Segmentation by Branch-and-Bound Model Selection. IEEE Trans. Image Processing 19(6): 1393-1402 (2010)
[11] A. Delong, A. Osokin, H. Isack, Y. Boykov. Fast Approximate Energy Minimization with Label Costs. IJCV 2012. http://www.csd.uwo.ca/~yuri/Abstracts/ijcv10_lc-abs.shtml

Diploma Thesis Supervisor: Ing. Tomáš Pajdla, Ph.D.

Valid until: the end of the summer semester of academic year 2017/2018

L.S.

prof. Dr. Ing. Jan Kybic	prof. Ing. Pavel Ripka, CSc.
Head of Department	**Dean**

Prague, January 6, 2017

Acknowledgment

I wish to express my sincere thanks to Ing. Tomáš Pajdla Ph.D., my thesis advisor. I am extremely thankful and indebted to him for sharing expertise, and sincere and valuable guidance and encouragement extended to me. My thanks to Ing. Filip Srajer for his help and patience with all of my questions. And finally I want to wholeheartedly thank my family for all of their support.

Abstract

This thesis first focuses on analyzing a C++ library YASFM [1], especially with regard to the situations, when the resulting models are not satisfactory. The work tries to document what causes those unsatisfactory results via experimenting with different input scenes and setups and then chooses a specific issue to focus on. Then it proposes an improvement of YASFM [1], which would address that issue, mitigating its impact on the result and further improving the output model's quality. The principle of this improvement lies in recycling potential 3D points that were removed during the reconstruction phase of the original model. The thesis then further describes the steps of this process, comparing the results with the original YASFM [1] and showing its benefit through experimental results.

Keywords computer vision, 3d reconstruction, structure from motion, yasfm, c++

Abstrakt

Diplomová práce se nejdříve zabývá analýzou C++ knihovny YASFM [1], se zaměřením na případy, kde výsledné modely generované knihovnou nejsou příliš dobré a zkoumáním důvodů proč k tomu dochází. Výzkum je prováděn formou dokumentovaných experimentů. Z objevených problémů je poté vybrána menší oblast, kterou se práce následně zabývá více do hloubky. Zjištění jsou využita k návrhu rozšíření knihovny, které tyto nedostatky řeší. Princip leží především ve zhušťování výsledných modelů o body, které byly původním YASFM [1] vyřazeny v průběhu rekonstrukce. Práce postupně popisuje jednotlivé úpravy oproti původní funkčnosti knihovny a na výsledcích experimentů ukazuje jejich přínos.

Klíčová slova počítačové vidění, 3d rekonstrukce, scéna v pohybu, yasfm, c++

Author statement for undergraduate thesis:

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.

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl vešekeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Dne ....

Contents

1 Intro

With the ever growing quality of image capturing methods and hardware, there is also growing demand for better 3D reconstruction techniques. One such technique is Structure from Motion (SFM), nowadays an important research topic within the field of computer vision. It estimates 3D structures from a sequence of 2D images, usually coupled with local motion signals. It has wide array of possible uses within applications working with complex scenes in motion. One such example can be video post-production software.

Multiple different approaches to this problem are being researched. A common approach used for estimating the 3D model usually works in several steps. After the initial stage of adjusting the images, the next step is detecting the features and finding correspondences between images using SIFT descriptors. This first part of the process is usually very similar among most of the approaches to estimating SFM. The biggest difference lies in the next part, where the segmentation and reconstruction the specific structures is done. One of the commonly used methods for this task is employing some homography or epipolar geometry constraints. This work focuses on a slightly modified approach using a fast greedy algorithm implemented in YASFM [1].

This approach proposed in Filip Šrajer's master's thesis [1] detects multiple homographies and groups them together into motion groups.

When used for 3D reconstruction of a scene in motion, it yields good results on many scenes and data-sets, however there is also a number of other cases where the resulting model is flawed in some way. There are situations, where multiple objects end up being merged together, when they should be reconstructed separately. The opposite case is the reconstructed model being incomplete, either missing some part or just being too sparse. Those two issues are the most general and noticeable ones among all of the problems that can occur. So if we can identify those issues and find their cause, then we can try to mitigate or even fully remove them with the right modifications to the current method itself and its implementation.

1.1 Contributions

This work builds upon the functionality of YASFM [1]. There are two main contributions.

First being the in-depth analysis and describing the algorithm used within the original library and exploration of different flaws occurring within the results.

Secondly, we propose a modification of the original process, capable of dealing with the issue of sparse models, missing unreconstructed parts. This modification makes use of calculating complementary model from unreconstructed tracks for each original one and then using epipolar constraints to merge the two models.

1.2 Thesis structure

The content of this work is structured into the four main parts:

1. (Section 2). The discussion of the State of the Art approaches to 3D reconstruction and SFM.

2. (Section 3). Analysis of the YASFM [1] library run, with the focus on input cases resulting in flawed models. Then in-depth study of the issue, chosen to be addressed in the next chapter.

3. (Section 4). The theoretical background and description of the new functionality added to the YASFM [1] library.

4. (Section 5). The documentation of the classes and methods added to the YASFM [1] library [1].

5. (Section 6). Description of the specific setup for each experiment performed and the discussion of the results.

6. (Section 7). Discussion of further work and possible improvements.

7. Section 8. Final summary of results and the proposed method's benefits.

Programming languages used for the scripting and experiments were MATLAB and C++. The Daliborka data-set was provided by CMP and the YASFM [1] library by its author Filip Srajer.

1.3 Frequently used terms

Feature (point) is a detected signature spot within the image. In case of YASFM [1] library, feature points are detected and assigned a SIFT descriptor. [2]

Correspondence is a pair of feature points matched through their descriptors.

Track is a set of feature points detected among the images and connected via chain of correspondences among them.

Model is a resulting point-cloud of the 3D reconstruction, ideally representing one moving object within the scene.

Homography is a special relationship describing mainly the transformation between two points within different image planes. [3]

Segment is a homography group assigned to every correspondence between two images. This term is introduced mostly to better differentiate between global track groups that form nViewMatches and the homography groups, local to each image pair.

Group is how we refer to the global track groups.

2 Related work

Structure from motion and the problem of multi-body segmentation can be broken down into several sub-problems, which are then each dealt with in a suitable way. This chapter will cover some of those commonly used methods.

2.1 Feature point detection and matching

The first step of any Structure-From-Motion pipeline is the detection of objects within the captured images. What most classic approaches have in common is that they begin with searching for points of importance, also called features, that define the objects within the input image sequence. Features can be detected and described via multiple methods, with the currently most widely used being SIFT [2], also used within YASFM [1].

Having detected the feature points, the next step is matching them based on their descriptors. This is usually done by nearest neighbour search and further filtering the outliers via some criterion such as Lowe ratio [2].

2.2 Object segmentation

More specifically making use of motion information to model and segment the scene. A review of some popular methods for segmentation of the scene with focus on the Change direction method is done by Shubhangi L. Vaikole and S. D. Sawarkar in their work [4]. The simplest traditional way of modelling the scene starts with two-view relations, choosing the best pair and gradually adding more views one at a time and thus building the model **incrementally**. This approach was also used by Filip Šrajer in the original YASFM [1]. To further robustify this process, we could increase the number of views, similarly like in the method proposed by R. Vidal and R. Hartley in their work [5] using a three-view approach instead. Past that they stick to the classical approach of using epipolar geometry to model the scene. Another such approach is described by A. W. Fitzgibbon and A. Zisserman in their work [6] also using epipolar geometry to model the scene via calibrating cameras modelling each object's motion separately. Most of the other similar approaches usually search for such model that has the highest support and discard the rest. YASFM [1] works differently in the sense that it groups together "similar" homographies using a greedy algorithm, with each of the resulting final groups modelling one object within the scene, thus keeping as much information as possible.

This homography grouping step is essentially a type of image matching followed by search for consistent groups. Thus applying some advanced graph-based matching approaches, like the work of Tianwei Shen et al. [7] could also improve the results.

2.3 3D Reconstruction

The actual process of reconstructing a 3D scene begins after we have found at least the two-view matches among the images (cameras). What follows is usually a process of starting from the best pair and further gluing more cameras one-by-one as the pool of tentative matches changes with each addition. During this process, we often want to eliminate the outliers within the reconstructed model and enforce consistency of the inliers via bundle adjustment [8]. In YASFM [1] this step is performed by a system inspired by Bundler [9] even saving the output in its default format, so that it can be loaded and used by any tool compatible with it.

YASFM's [1] main idea focuses on using the inliers of every detected model, not only the most dominant one. Thus allowing for fast and still reasonably accurate reconstructions.

The modification we propose in Section 4 further builds on this idea, by making use of as many tracks as possible. To that goal we aim to reduce the amount of unreconstructed points during the final model reconstruction.

2.4 RANSAC

The random sample consensus i.e. RANSAC [10] is among the most useful tools used to further optimize the results while keeping fast processing time of the whole pipeline. When modelling a scene using either homographies or epipolar geometry, we need some points that will define the model. Searching for a near-optimal subset of points is thus almost a necessary step, if we want the results to be accurate.

An interesting alternative is proposed by Cosimo Rubino et al. [11], where they use a general purpose detector to construct a sampling function based on its semantic information. This function then allows the grouping of similar features together. Where this approach is viable it can replace the standard approach to fitting a model to a scene via multiple calls of RANSAC [10].

2.5 Different approaches to SFM

Besides all of the more standard methods mentioned previously, multiple different approaches that might not seem that related to 3D reconstructions from motion can be applicable. Such as the segmentation-free approach to multi-body feature tracking described by P. Ji et al. [12]. Within their work, they introduce a method of tracking rigid scene motion by modelling all motions at once with the use of epipolar constraints.

Reza Sabzevari and Davide Scaramuzza chose to tackle the problem of single camera mounted on a car, which by itself is a rather complex scene. By factorization of projective trajectory matrix, while omitting estimation of depth, they focus on generating multiple hypotheses using epipolar geometry to estimate both 3D structures and the motions.

Other interesting approaches are the the work of N. Thakoor et al. [13] using Branch-and-Bound method, while G.Pan et al. [14] uses Over-Segmentation Detection to estimate the number of motions within the scene.

3 YASFM Analysis

In this chapter we take a closer look at the inner workings within the incremental pipeline of YASFM [1]. However we take a different approach to how Filip Šrajer described it in his own work. Our goal is to analyze and evaluate it with future improvements in mind. So that we can choose which area to focus on further on in this work.

3.1 YASFM pipeline

Steps:

1. Input image sequence processing - preprocessing and then scanning for feature points and generating SIFT descriptors.
2. Matching - searching for correspondences among every pair of images.
3. Geometric verification - creating and assigning homography groups modelling motion to every correspondence.
4. Model building - generating tracks and grouping them together via algorithm described in Section 3.2.
5. 3D reconstruction and bundle adjustment - triangulating points from the best two images for the specific group, then gluing the remaining tracks via correspondence chains over multiple images.

For full and detailed description of the first steps of the pipeline, please refer to the work by Filip Šrajer [1]. This analysis is solely focused on the last two steps - the merging of tracks into the n-view-matches and then running bundle adjustment [8] on them.

Next we are going to take a closer look at the former of the two. The merging step uses a rather greedy approach which could be represented by an undirected graph G.

G is such an undirected graph whose vertices are the basic two-view matches outputted from the previous step and assigned their local homography group ID. An edge connects such vertice to all others sharing the same homography group. A single vertice can be part of multiple homography groups, each for a different image pair.

Idea Local homography groups for one pair of images very often overlap with local homography groups for other pairs consisting of either one of the two images and a different image. This means that in G, such pairs are always connected with edges. Thus if we find strongly connected components [15] within G, we create the groups of tracks among multiple images, also sometimes called n-view-matches and each n-view-match will ideally represent one object within the scene.

The last step of the original pipeline takes each resulting group from the previous step and then performs the camera gluing. Starting with the best found pair, then running first bundle adjustment [8] optimization using Ceres solver [16]. There are two extra filters for outliers based on reprojection error and small ray angle thresholds done after the bundle adjustment [8]. Upon removing outliers, the next best camera to add is chosen and the gluing step is repeated, reconstructing matches, running bundle adjustment again and filtering outliers. As this process goes on, the pool of tentative correspondences connecting the current reconstructed result with the tentative cameras is changing accordingly.

3.2 Algorithm

```
input  : F ... set of features(keys) found for every camera
         P ... set of CameraPair structures containing all matched
tracks and which segment they belong to, for every pair of cameras
output: G ...set of track groups, contains the label of group to which
         each track belongs
/* Create a group of size one for every track.          */
G = InitGlobalGroups(F,P);
/* Map of camera-pair-segment triples per group, so that
   groupMap[jg] = {group((2,3),1), group((5,8),0), ... }
   means camera pair 2,3 was segmented to segment with
   local id 1 and belongs to n-view match global group Id
   jg, ...                                              */
groupMap = InitGroupMap(F,P,G);
/* Merging the identical groups.  Identical means they
   match within all their camera-pair-segment triples.  */
G = MergeIdenticGroups(G);
/* Here we prepare a structure to keep information about
   group relations, structure has to support removal of
   whole rows, so Map<int Key,Map<int Key,int Value>>
   structure is used in current implementation.         */
commonSegmentsInPairsCount[groupsCount][groupsCount];
for $ig$ = $0...groupsCount$ do
    for $jg = ig + 1...groupsCount$ do
        /* Search for identical camera-pair-segment triples
           between global groups ig and jg.             */
        commonSegmentsInPairsCount[ig][jg] :=
         countCommonPairs(groupMap,ig,jg);
    end
end
```

Algorithm 1: Initialization phase and merging of identic groups.

```
/* Main merging loop                                          */
Set candidates; int maxCount = -1, maxJ, maxI;
while maxCount != 0 do
    findMaxCount(commonSegmentsInPairsCount,maxCount,maxJ,maxI);

    /* Merge groups by adding all non-common entries from
       groupMap[maxJ] to groupMap[maxI], each G[i]=maxJ
       becomes G[i]=maxI                                      */
    mergeGroups(groupMap, G, maxI, maxJ);
    /* Replace all occurences of the removed group with the
       now merged one.                                        */
    fixGroupAssignment(G,maxI,maxJ);
    /* Push groups that had common image-pair-segments with
       group maxI and maxJ                                    */
    candidates.push(nonZeroGroups(commonSegmentsInPairsCount[maxI]));

    candidates.push(nonZeroGroups(commonSegmentsInPairsCount[maxJ]));

    /* Remove rows maxI and maxJ from
       commonSegmentsInPairsCount                             */
    removeRow(commonSegmentsInPairsCount,maxI);
    removeRow(commonSegmentsInPairsCount,maxJ);
    /* Go through all groups left in
       commonSegmentsInPairsCount                             */
    candidates.push(commonSegmentsInPairsCount.findAllRowsContaining(maxI));

    commonSegmentsInPairsCount.removeElement(maxI);
    candidates.push(commonSegmentsInPairsCount.findAllRowsContaining(maxJ));

    commonSegmentsInPairsCount.removeElement(maxJ);
    /* Recalculate common camera-pair-track counts for all
       candidates                                             */
    maxCount = 0;
    ig = maxI;
    for each jg in candidates do
        commonSegmentsInPairsCount[ig][jg] :=
         countCommonPairs();
        if commonSegmentsInPairsCount[ig][jg] > maxCount then
            maxCount := commonSegmentsInPairsCount[ig][jg];
            maxJ := jg;
            maxI := ig;
        end
    end
end
```
Algorithm 2: Group merging loop. End of YASFM pipeline Step 4.

```
/* Single camera gluing step                                  */
input  : t ... single n-view-match (group of tracks) outputted from
         the track merging step. Contains coordinates and image
         indices for matches defining each track.
output: m ... the resulting model.
bestPair = findBestPairAmongCameras(t);
m = reconstructPoints(bestPair);
m = bundleAdjust(m);
m = filterOutliers(m);
camToAdd = findNextBestCam(m,t);
while camToAdd!=null do
    m = reconstructAndAddPoints(camToAdd,t); m =
      bundleAdjust(m);
    m = filterOutliers(m);
    camToAdd = findNextBestCam(m,t);
end
```

Algorithm 3: YASFM pipeline Step 5 - Camera gluing step.

This algorithm describes the reconstruction of single model performed for each group outputted from the previous step separately.

3.3 Common issues

As the task of reconstructing a 3D model from a scene in motion is rather difficult problem, so within complex scenes it is quite likely that the reconstruction won't result in a perfect model. The most commonly occuring flaws are the following:

Unseparated objects is the situation, when multiple separately moving objects end up being reconstructed as a single model. This issue mostly occurs when there are two or more input images where the motion of such objects is insignificant, close to standing still. Thus a single homography group can cover correspondences found on all such objects. Then within the algorithm above it can be observed that such tracks are grouped together into a single nViewMatch, thus resulting into a merged model. This usually results into multiple objects merging together with the background.

Fragmented objects is the opposite situation, where big objects end up being fragmented into multiple smaller ones. This issue is considered a little less severe, as it is much easier to merge such models together afterwards compared to splitting incorrectly merged pointclouds.

Sparse model is the result when a part of the scene does not end up being reconstructed or the point density of some part of the model is very low. In extreme cases the whole model can be discarded. There are multiple factors affecting this, starting with the input image quality, the amount of detected and matched points and many more. But most of those problems are not going to be covered by this work. For our purposes the most important cause of this flaw is within the YASFM [1] pipeline, specifically the bundle adjustment step.

A significant amount of tracks is often fully discarded during the bundle adjustment process. So the unreconstructed points relevant to those are all lost. This is mostly caused by the bundle adjustment's approach of discarding the outliers. In some cases the outliers can even form the majority of the tracks being reconstructed.

With this problem in mind, we designed the modification proposed in the next section, aiming to reduce the loss of reconstructed points within this step.

Other factors like low quality of the input images, including blur, occlusion and many other effects, have rather high impact on the initial steps of extracting feature points and further matching them into corresponding pairs.

4 The proposed modification

We propose a two step modification of the original YASFM pipeline, aiming to minimize the loss of potentially reconstructable points. These two steps are performed for each model separately. The additional two steps are added to the end of the original YASFM [1] pipeline, beginning just as the bundle adjustment step is finished. At that point we have the reconstructed model and also most importantly its unreconstructed outliers which are key for the first step to work.

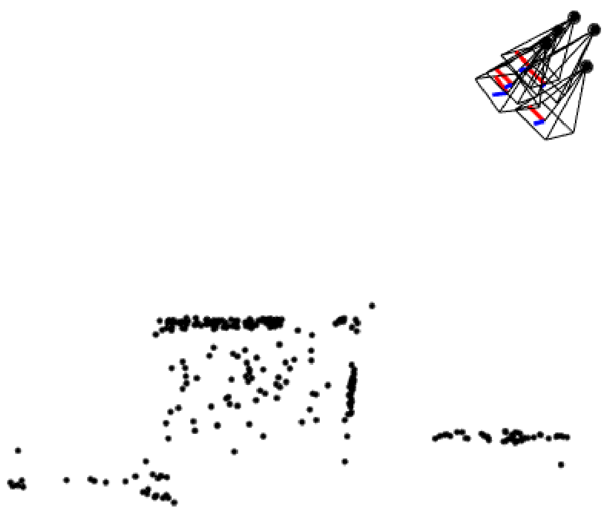


Figure 1: Example of a model reconstructed by the original YASFM pipeline.

First step - Complementary model Within the first step, we take the unreconstructed outlier tracks and run a second bundle adjustment only on those tracks. The resulting model is complementary to the original. We can repeat this step recursively until the amount of outliers is acceptably low enough. However this approach can further complicate the following step. We designed Experiment 4 described in Section 6.7 to test how many points form an average complementary model and thus can be recovered. This experiment also proved the necessity of the following step, as the complement gets reconstructed separately, thus with different scale, translation and rotation compared to the original model.

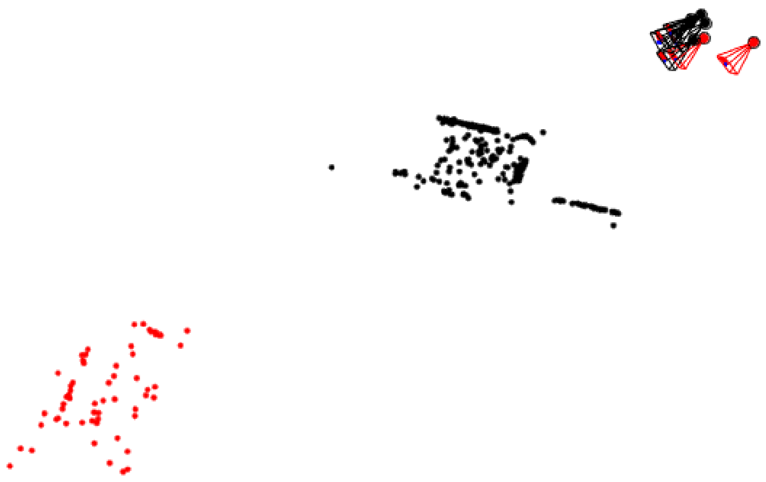


Figure 2: Example of the original model (black) and its complement (red).

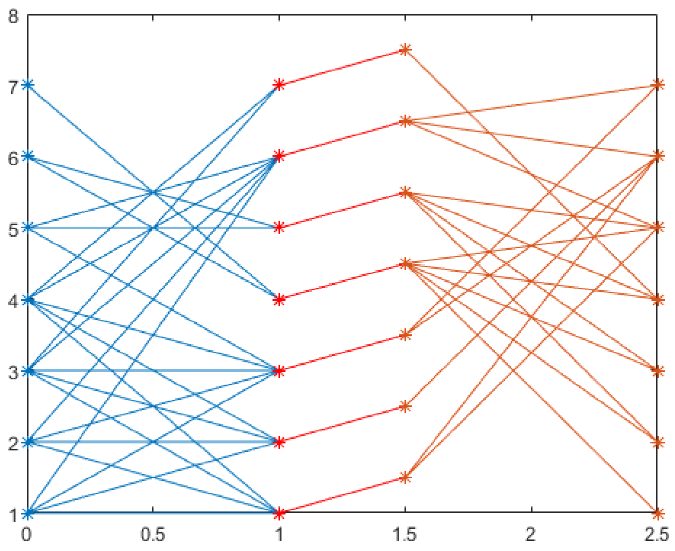


Figure 3: Example graph of which cameras (on the left and mirrored on the right) observe the original(blue lines) models and their complements (orange lines).

Second step - Merge After reconstructing both the original and its complementary model, we need to find a transformation that would allow us to fit one within the coordinate system of the other and thus merge them accurately. For this purpose, we can make use of the following idea.

Idea Some tracks end up being split due to some of the correspondences being visible only on several images. So both the original model and the complement can contain a reconstructed point originating from the same track. With knowledge of three or more such points, it is possible to calculate a reliable rotation, translation and scale transformations. For this we use SVD decomposition to calculate the transformation. In cases where we have more than three common points, we can use RANSAC [] to find the best triple resulting in smallest transformation error.

If the only three such points lie on a line or very close to such degenerate setup, the rotation transformation might require some additional manual tuning. An example of such behaviour is shown in Section 6.9.

We can see that in the worst case the proposed modification will result in the same amount of reconstructed points as the original model. However in general case it will yield denser models, with even the possibility of recovering a fully discarded model.

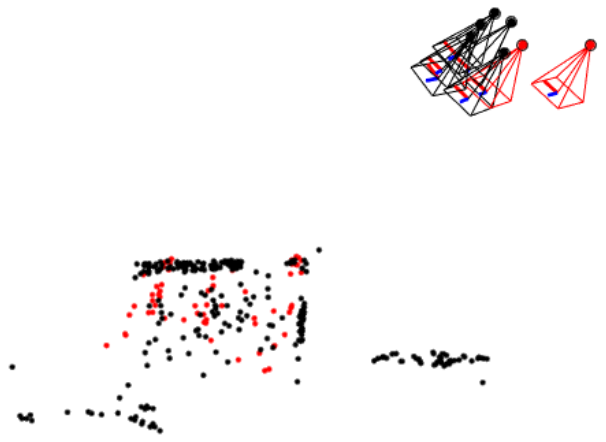


Figure 4: Example of the merged original model (black) with its complement (red).

4.1 Two step modification algorithm

```
input  : T ... n-view-matches (groups of tracks) outputted from the
         track merging step.
output: M ...pair of models, complement is transformed into the
         original's coordinates.
for each t in T do
    /* Step 1 - Create models                                  */
    original = reconstructByCameraGluing(t);
    M.insert(original);
    t' = getUnusedTracks(t);
    while t' is big enough do
        complement = reconstructByCameraGluing(t');
        /* Step 2 - Find transformation                        */
        commonPoints = findCommonPoints(original,complement);
        if size(commonPoints) >= 3 then
            /* Transform.rotation ...  3x3 rotation matrix */
            /* Transform.scale ...  scalar number          */
            /* Transform.rotation ...  3x1 translation vector
               */
            transform = findTransformBySVD(commonPoints);
            for each point in complement.points do
                point = transform.scale*transform.rotation*point +
                  transform.translation;
            end
            M.insert(complement);
            t' = getUnusedTracks(t');
        end
    end
end
```

Algorithm 4: Two step modification

This algorithm modification calls the Camera gluing process once for the original model and then recursively until we deplete all the unreconstructed tracks.

Implementation note: Only the first step of the recursive complement creation is performed, as within the experiments even after the first step the amount of unreconstructed tracks was nearly always close to zero.

4.2 Finding the transformation

This section explains what happens inside of $findTransformBySVD$ function called in the Algorithm 4 describing the proposed modification of YASFM [1]. To find the relevant rotation, scale and translation, we use Singular Value Decomposition also known as SVD [17], based approach inspired by the one described by S. Umeyama [18].

The task - We have two sets A and B, each consisting of n points (3x1 column vectors) such that the first point in A corresponds to the first point in B. This holds for all pairs. Our goal is to find such rotation(R), scale(s) and translation(t) which would allow us to fit set A onto B.

$$B = RsA + t \tag{1}$$

Rotation - We begin by finding centroids C_A and C_B of the two sets.

$$\begin{aligned} C_A &= \frac{1}{n}\sum_{i=1}^{n} Point_A^i \\ C_B &= \frac{1}{n}\sum_{i=1}^{n} Point_B^i \end{aligned} \tag{2}$$

After finding the centroids, we can re-centre both sets by subtracting them and assemble a direction correlation matrix K. K is 3x3 square matrix accumulating all of the pair coordinate vector products.

$$K = \sum_{i=1}^{n} Point_A^i * (Point_B^i)^T \tag{3}$$

Now we can perform the SVD and finally estimate the rotation from the decomposed square matrices.

$$\begin{aligned} [U, -, V] &= SVD(K) \\ S &= diagonal([1\ 1\ det(U) * det(V)]) \\ R &= USV^T \end{aligned} \tag{4}$$

Scale - s can be computed as the ratio between euclidean norms of both sets.

$$\begin{aligned} s_A &= \|Point_A\| \\ s_B &= \|Point_B\| \\ s &= \frac{s_A}{s_B} \end{aligned} \tag{5}$$

Translation - Finding t at this point is simple, all we need to do is subtract the transformed centroid of the set we want to shift from the other.

$$t = C_B - RsC_A \tag{6}$$

5 Implementation

Within this section we make an overview of the changes done to the original YASFM [1] library and also describe

As the original YASFM [1] library is implemented within C++ programming language, we made modifications to the code described in previous Section 3 and then used MATLAB scripts to complete the last step of the new pipeline. The implementation was meant mostly to prove the concept and thus we omitted the recursive approach to the first step (Complementary model creation) of the modification.

5.1 Implemented features

The proposed modification - The final two steps of the pipeline.

First step - Implemented within the C++ code of YASFM [1], makes use of the unused/split tracks left after running bundle adjustment and reconstructing the original model. Then does the same with the leftover tracks.

Saving common points - A new utility function was added for saving the ID's of known common points within each original model and its complement. Saves the data into text (.txt) files, two ID's per line, first the original and second the complement.

The second (merging step) - Realized partly via additional MATLAB scripts also included within the whole project.

Tools for the analysis

Camera time-stamp attribute - We added functions to extract and keep track of the time of capture for each image.
Homography group visualization - A C++ utility function for saving the data and also a MATLAB tool for plotting out the inliers of each homography group after the merging step.

Result visualization - Whole set of MATLAB scripts written for this task is included within the final build. These allow plotting of resulting 3D models, their back projection into the original images and plotting the graph of model visibility per camera.

Additional library For the transformation estimation via SVD and some of the visualizations, we use MATLAB library functions provided by the thesis adviser, T. Pajdla.

5.2 Code modifications

Within Filip Šrajer's thesis proposing YASFM [1], he describes the general use and functionality of the main classes used within the pipeline. So in a similar fashion, we would like to point out the necessary additions and changes in order for the modified pipeline to work. However the main classes remain mostly unchanged.

Camera class was extended to hold information of the image capture time-stamp. This wasn't necessary for the two-step modification of the pipeline, but proved helpful with the analysis. Also it can be used by any future modifications and additions to the YASFM [1].

main.cpp is the file, where the algorithm of the proposed modification is implemented. It contains all of the separate steps of the incremental pipeline.

utils_io.cpp contains most of the tools to save and load data. Within this file was added most of the utility functions necessary to pass the data to MATLAb for visualization and the analysis as well the common point indices necessary for the final step of the modified pipeline.

1. Homography group data - saved to files named homographies_a_b.txt, where a and b are the two image indices in the input sequence.
   Format - The first row holds the amount of groups detected. For each group the file contains a block, with a number of its inliers first

and then twice that amount of lines. Odd lines contain the first image coordinates whilst the even ones the ones in second image.

2. Common point indices - saved to files named commonIdx_model$n$.txt, where n is the number of the generated model.
   Format - First row holds the amount of common points, whilst two following columns are the indices within first the original and second the complementary models.

6 Experiments

The main goal of running all the following experiments was to evaluate and compare changes resulting from modifying and improving the current YASFM [1] library. Each of the experiments focuses on the specific newly added modification, showing its effect upon the resulting point-cloud and object segmentation within the images.

6.1 Camera technical specifications

The images were captured with a regular iPhone 4S camera with the following specifications:

Lens focal length:	4.28 mm
Sensor size:	1/3.2" diagonal (4.54 mm x 3.42 mm)
Sensor model:	Sony IMX105 Exmor R
Sensor technology:	Back-side illuminated (BSI) CMOS (Complementary Metal Oxide)
Sensor resolution:	8 megapixel (3,264 x 2,448 pixels)
Macro working distance:	6.5 cm

6.2 YASFM settings

Common YASFM settings for all of the experiments:

Points re-projection error threshold:	8 [px]
Ray angle re-projection error threshold:	2 [px]
Similarity transform threshold:	20 [px]
Affine transform threshold:	10 [px]
Homography threshold:	5 [px]

6.3 Experiment setups and results

6.4 Experiment 1 - Shuffling books

Within this experiment we try to run YASFM [1] upon a scene modelling the situation where both the camera and the objects move around. It is one of the more complex cases, so our goal here was to observe the results and confirm our hypothesis of expected results based on the previous theoretical analysis.

Scene setup:
6 Images
3 planar objects (book piles)
Textured background
Moving camera position

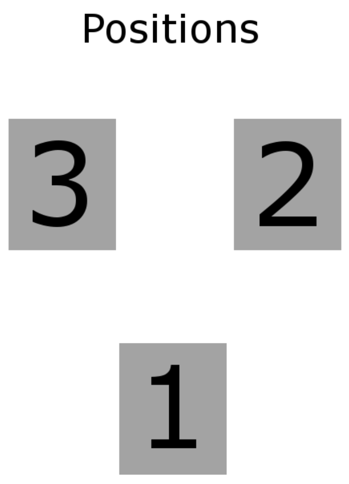


Figure 5: Book pile positions being shuffled during the experiment.

We placed the books upon the textured background and shuffled their positions in the following sequence, making sure to capture images from different angles in a circle around the scene. Colours in the table relate to the top book cover for each pile.

Image	Position 1	Position 2	Position 3
1	Green	Yellow	Blue
2	Blue	Green	Yellow
3	Yellow	Blue	Green
4	Green	Blue	Yellow
5	Yellow	Green	Blue
6	Blue	Yellow	Green

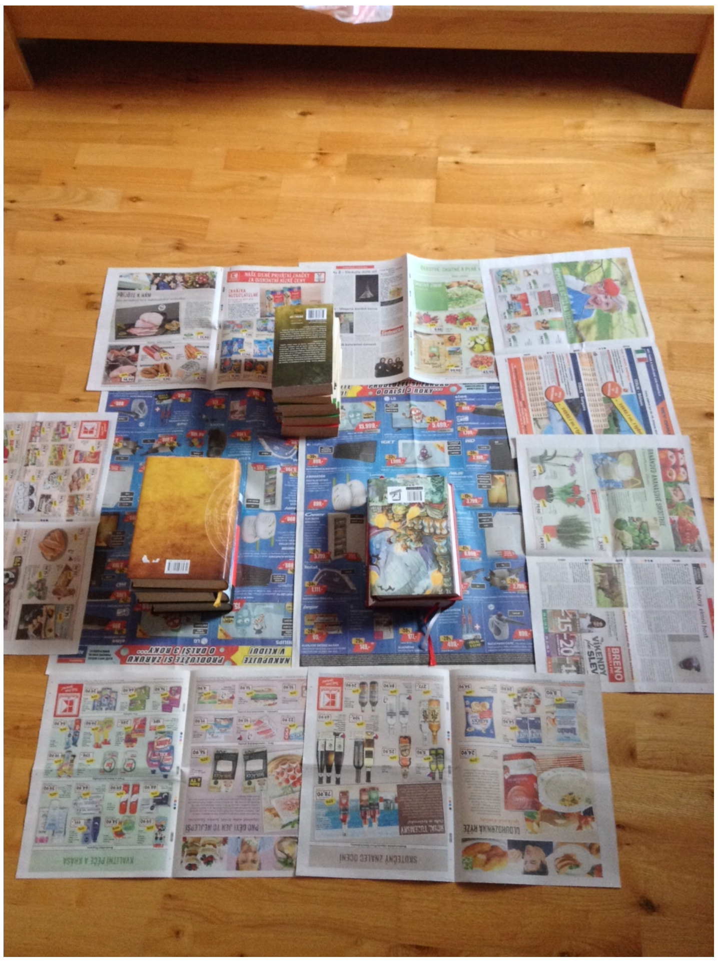


Figure 6: Example - first image in the sequence.

Results and observations

The expectations weren't too far from the actual results. The textured background ended up reconstructed as the biggest model and the correspondences upon book piles appearing in the same spots within multiple images got merged into it. The resulting model doesn't contain very good reconstruction of the book piles themselves, due to the yellow and green books having very few feature points detected and matched at the beginning of the pipeline. For future experiments, more suitable objects were used.

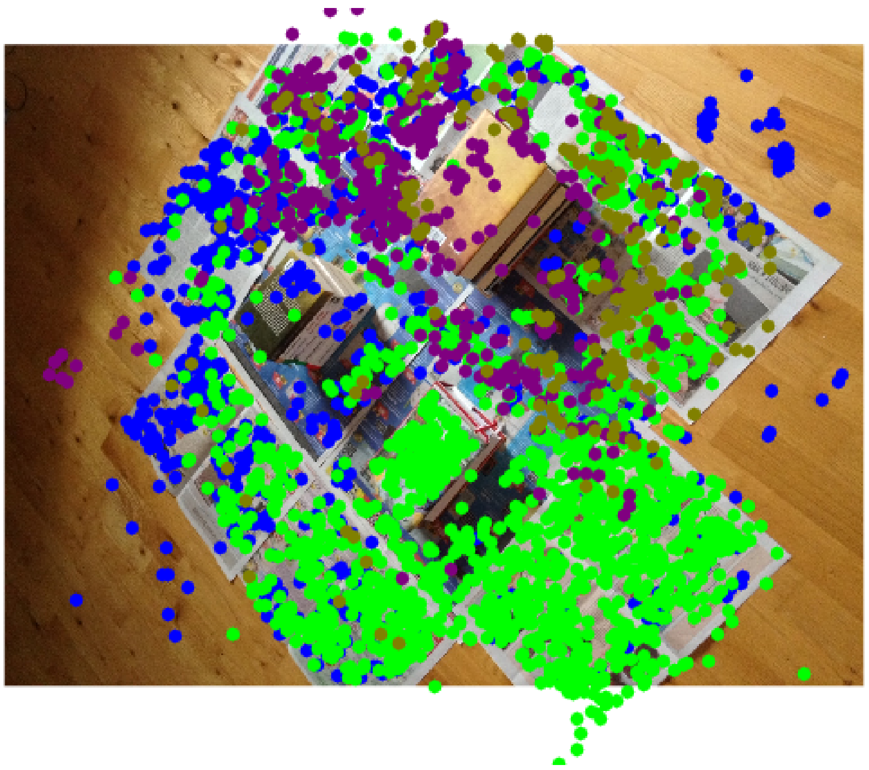


Figure 7: All models projected back into the image

6.5 Experiment 2 - Daliborka

Here we test the YASFM [1] library upon images of Daliborka model. This time there is just one object, which stays still and only the camera is moving around it in a half-circle. This is the only data-set captured using a higher-spec camera and its main purpose is to show that the modification retains the original functionality for input sets, where it is not needed.

Scene setup:
10 images
1 3D object (Daliborka)
No background
Moving camera position

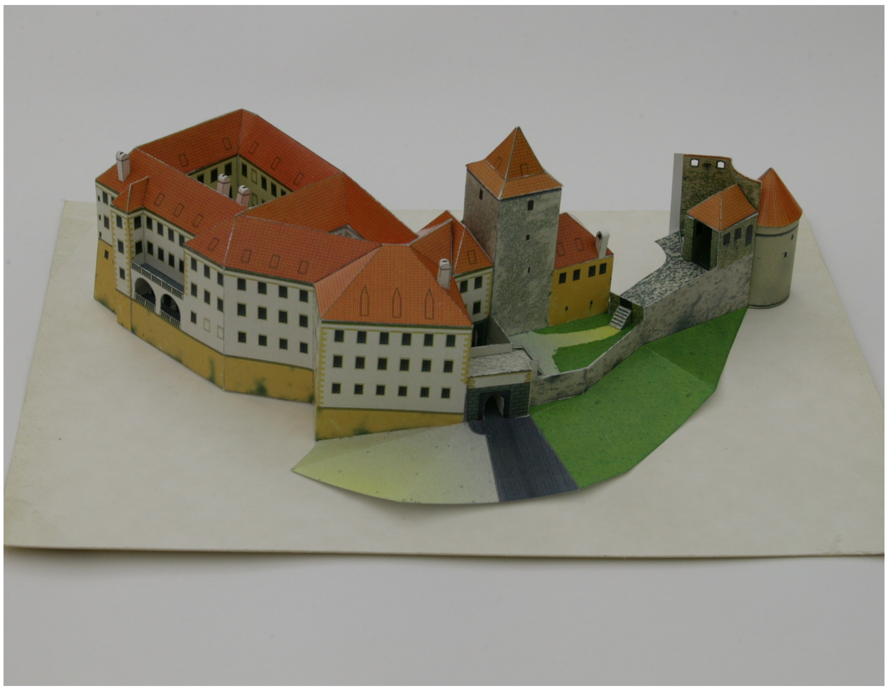
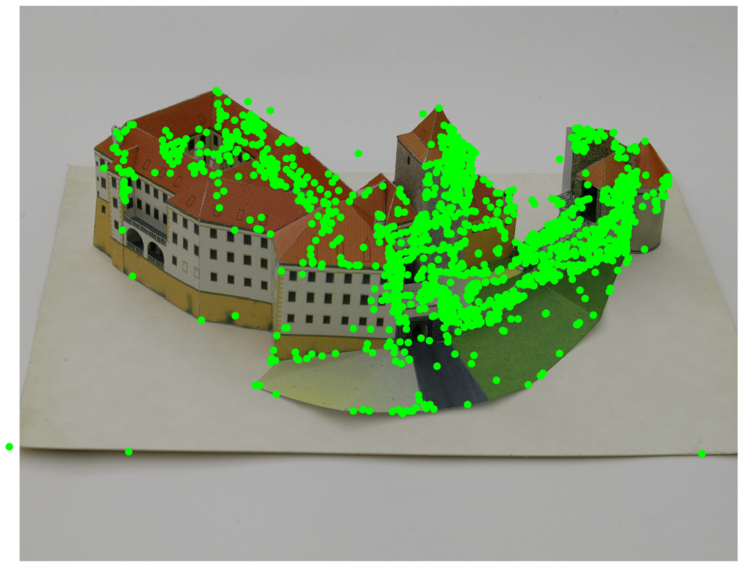
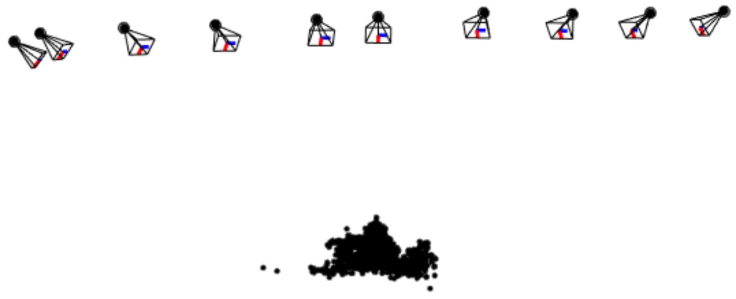


Figure 8: Sample image

Results and observations

For simple scenes like this one, where just one object is present with no background, the results match the ones produced by the unmodified original pipeline in Filip's work. For this specific data-set, we get almost no outliers after the bundle adjustment, so there is no need to use the modified pipeline here. However this occurs only under very specific conditions. The scene must be very simple, ideally without background and the input image quality needs to be very high.





6.6 Experiment 3 - Moving books

Within this scene, we have one moving object while the camera and the rest of the scene remain still. Here we test the hypothesis, that only the moving object should get segmented and the static objects merge with the background.

Scene setup:
1 moving object (single book)
1 static object (pile of books)
Wooden floor background
Static camera position

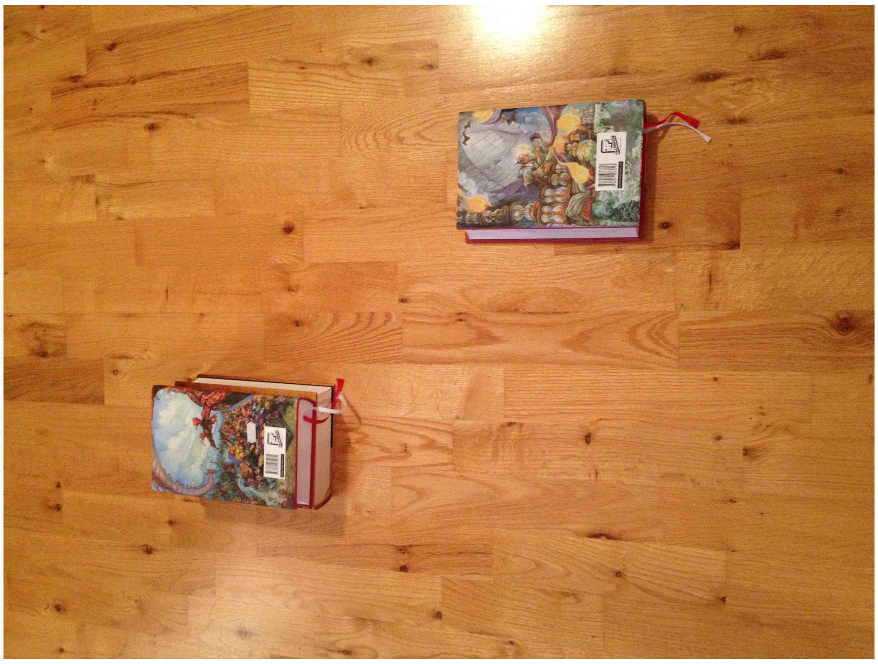


Figure 9: Sample image

Results and observations
The hypothesis seems to be proven right, according to the following figures displaying the results. We can see that the static pile of books is merged with the background. This also helped with the analysis of the track group merging algorithm described in 3.2.

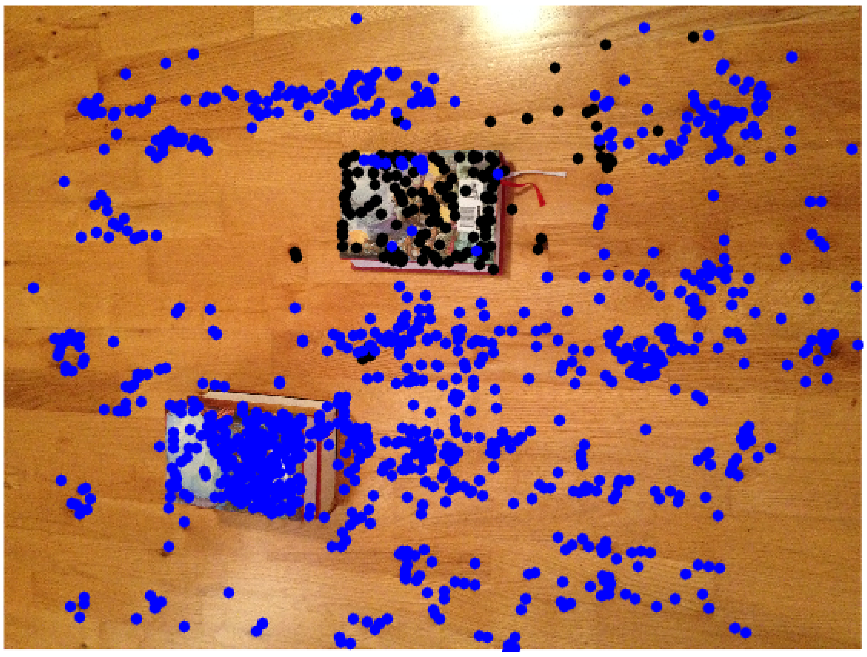


Figure 10: Experiment 3 resulting models. Blue is the background merged with the static bookpile, while the black model is the segmented moving object.

6.7 Experiment 4 - Moving books, no background

Almost the same setup as in the previous experiment, only this time without a textured background and using only two books. Within this experiment we try to test the newly added functionality of creating a complementary model for each of the original ones.

Scene setup:
7 images
1 moving object (single book)
1 static object (single book)
No background
Static camera position

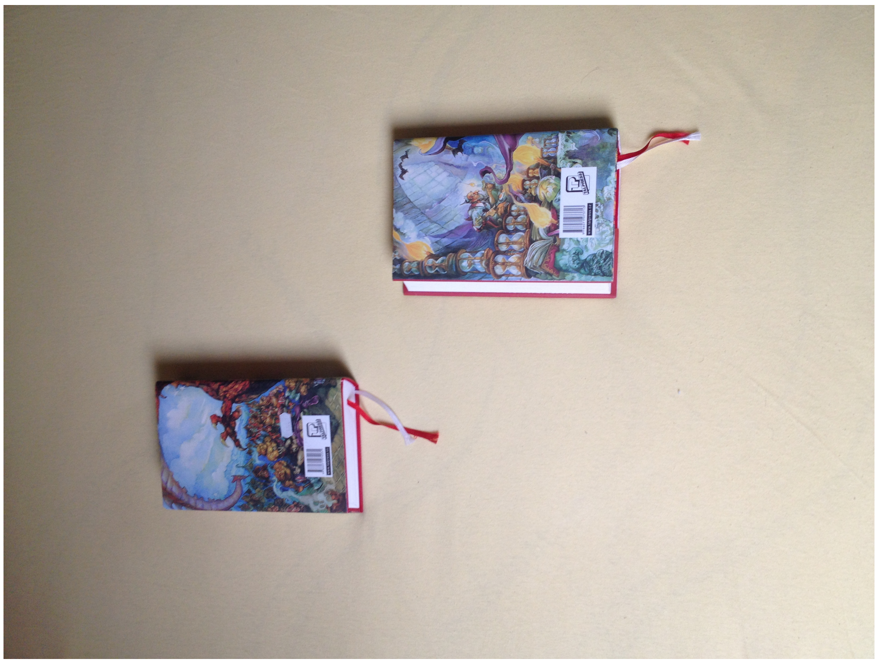


Figure 11: Sample image

Results and observations

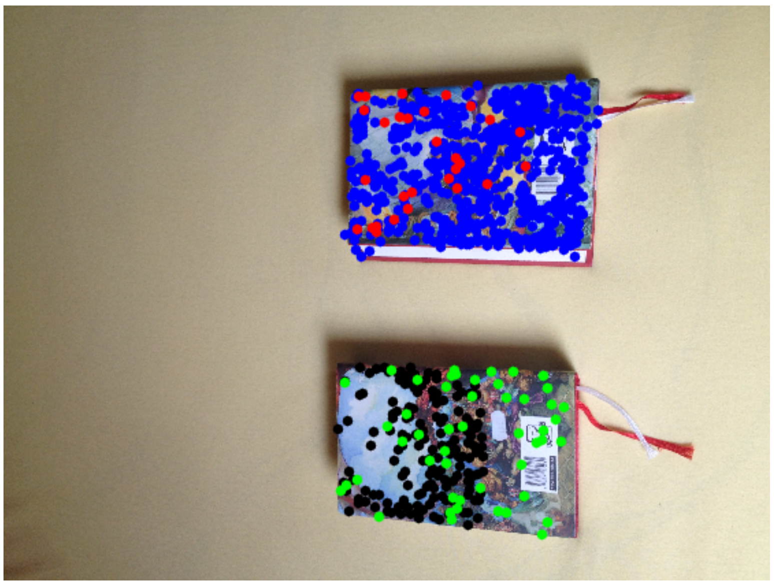


Figure 12: Resulting original models are black and blue, while green and red are the complements.

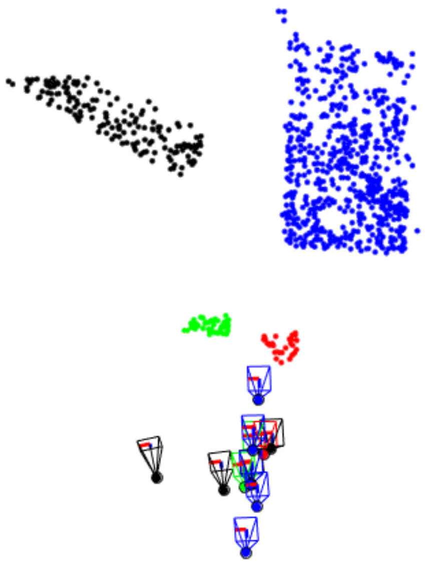


Figure 13: 3D reconstruction of the scene. Black circled green dots represent reconstructed cameras and their directions.

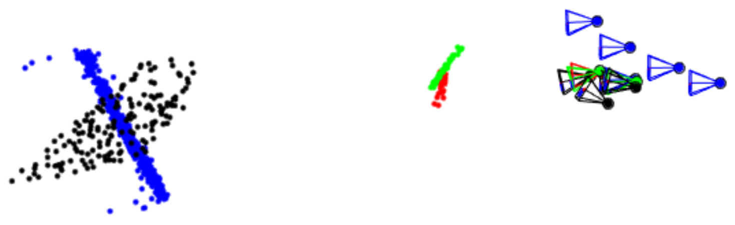


Figure 14: 3D reconstruction of the scene, side view. Black circled green dots represent reconstructed cameras and their directions.

From the previous figures we can see, that by performing the first step of the proposed modification will result in a complementary model displaced and differently scaled and rotated from the original. This shows the necessity of the second step.

The following table compares the amount of reconstructed points within the original and the complementary models.

Model	Original	Complement
1. Black	167	51
2. Blue	657	32

6.8 Experiment 5 - Box and books, no background

In the fifth experiment, we decided to add a little more complexity to the scene, by adding a bigger and more three dimensional object. With moving the camera around enough, we can get some split tracks by their matched correspondences not being in the field of vision anymore. Those split tracks are then used for the merging of original and complementary models, as described in Section 4.

Scene setup:
3 moving objects (two books and a box)
No background
Moving camera position

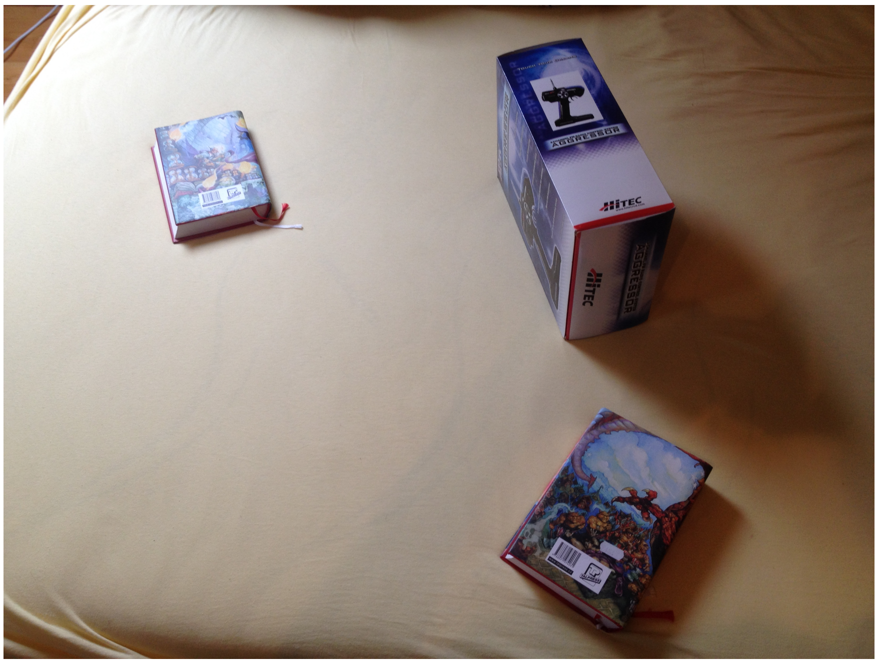


Figure 15: Sample image

Results and observations

Finally adding the last step of the modified pipeline, we can now see the results of the whole process upon the following examples of the merged models. A thing worth noticing is that usually the complement is created from different cameras than the original model.

Model 1

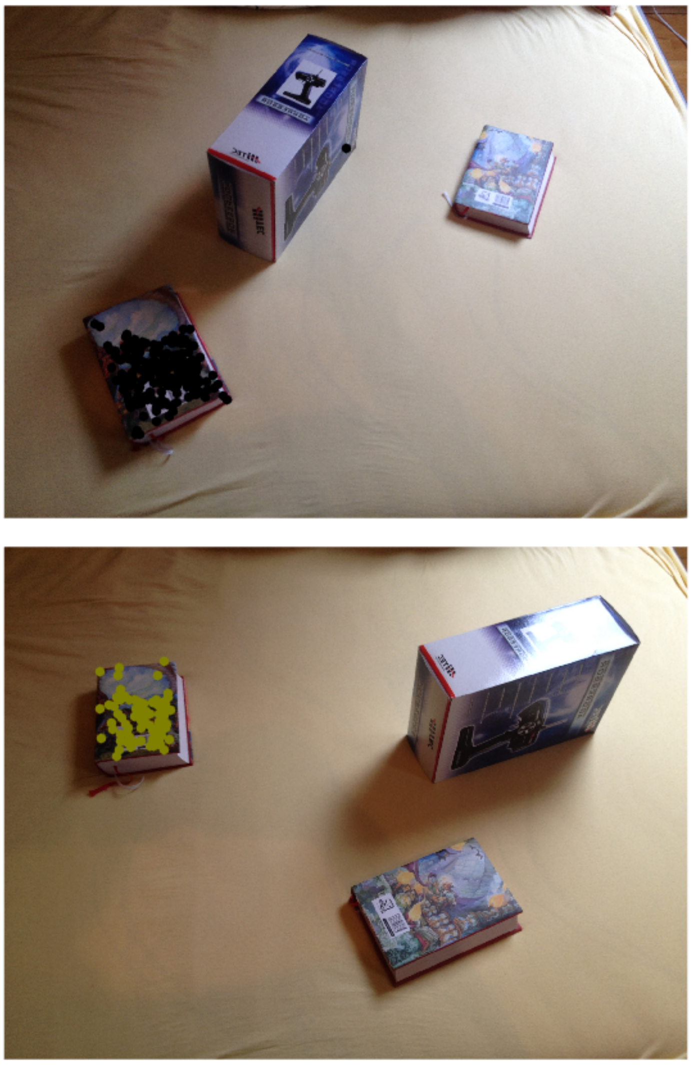


Figure 16: Cameras capturing scene that changed too much often result in creating the complement, as its matches are labeled inconsistent during bundle adjustment and removed from the original model.

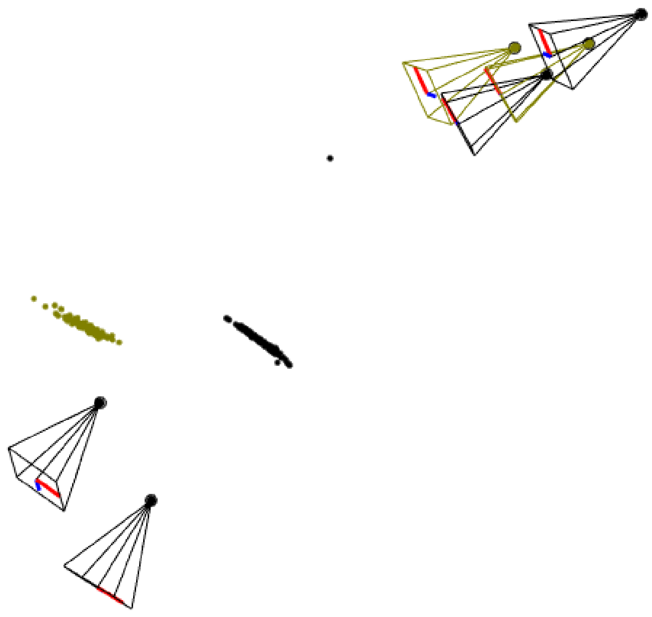


Figure 17: Original model (black) and the complement (gold) with their respective cameras before merging within the second step of the modification.

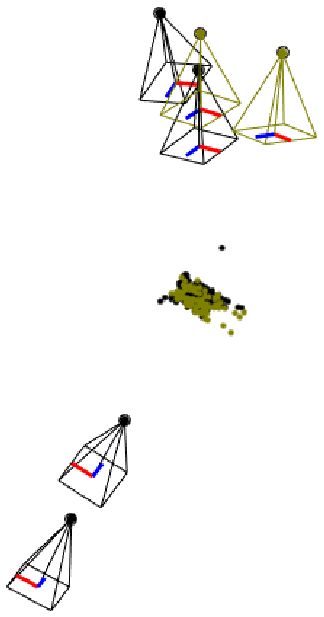


Figure 18: Result of the merging.

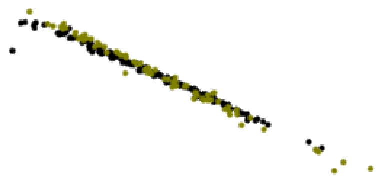


Figure 19: Result of the merging in a side view.

Model 4

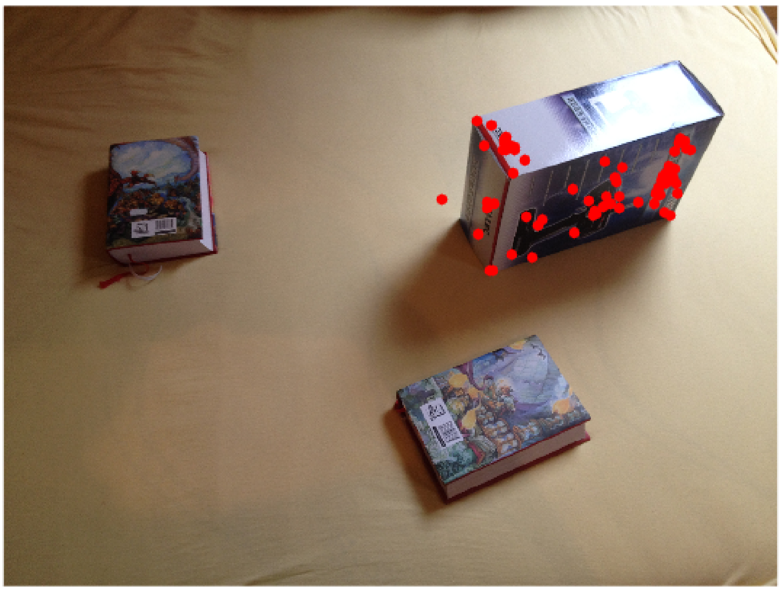


Figure 20: Original model (red) observed by cameras 6 and 7. Example image is from camera 6.

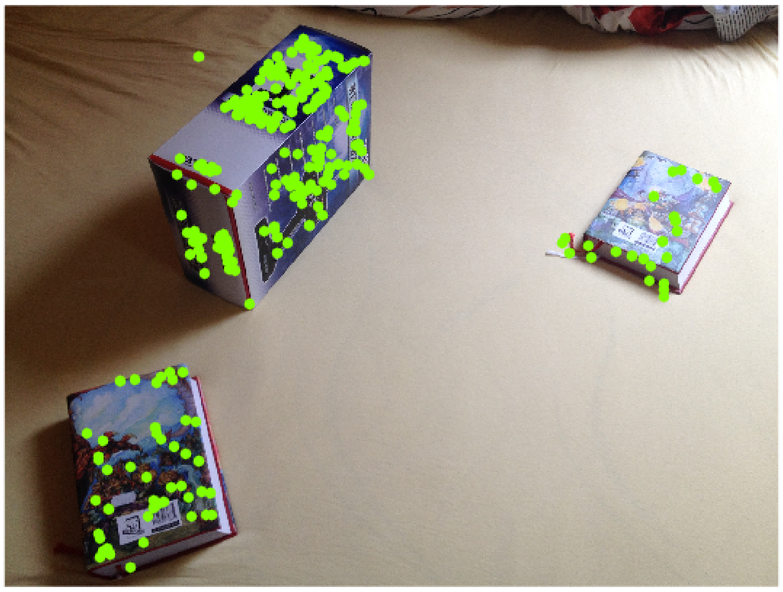


Figure 21: The complementary model (green), visible by cameras 1,2,3,4 and 5. Example image is from camera 1.

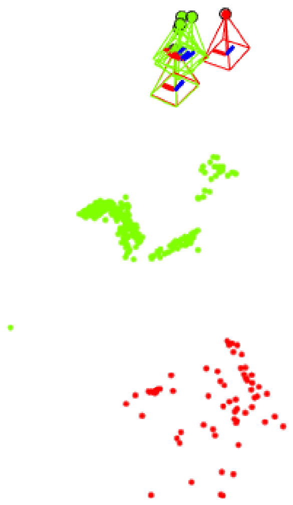


Figure 22: Original model (red) and the complement (green) with their respective cameras before merging within the second step of the modification.

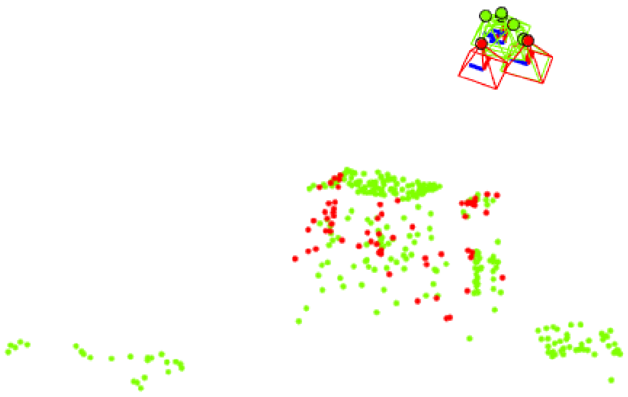


Figure 23: Result of the merging, side view.

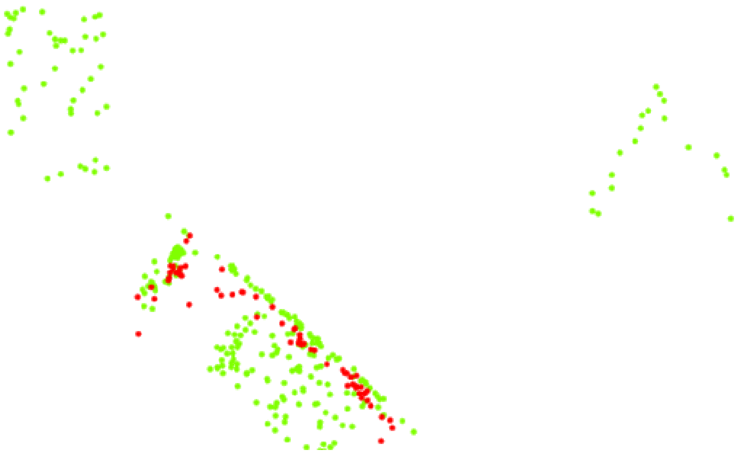


Figure 24: Result of the merging, top view.

In the following table we compare all of the models with regard to their amounts of reconstructed points. We can see that models 3. and 4. are nearly identical, only reversed in order of the tracks within the original and complement. This is possible due to the fact, that the YASFM's [1] track grouping algorithm allows tracks to belong within multiple groups. Thus being part of more than a single resulting model.

Model	Original	Complement	Common pts.
1. Black	195	81	71
2. Blue	97	16	0
3. Green	306	67	5
4. Red	67	306	5
5. Purple	12	61	0
6. Turquoise	20	28	0
7. Pink	30	29	0

6.9 Experiment 6 - Multiple objects, no background

Within the final experiment, we tried capturing higher amount of images with more complex scene, which is also changing more than in the previous ones.

Scene setup:
20 images
Multiple moving objects
No background
Moving camera position

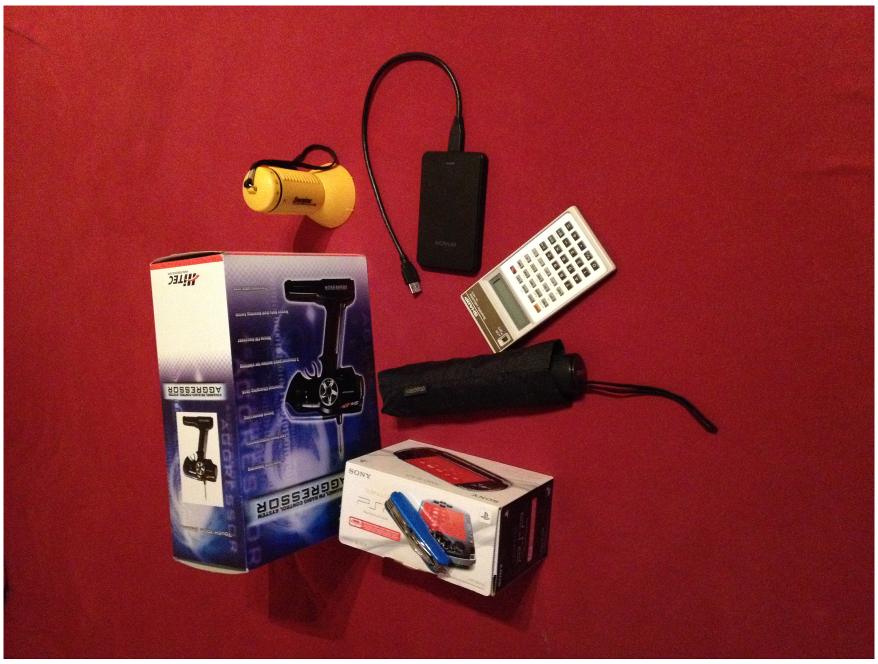


Figure 25: Sample image

Results and observations

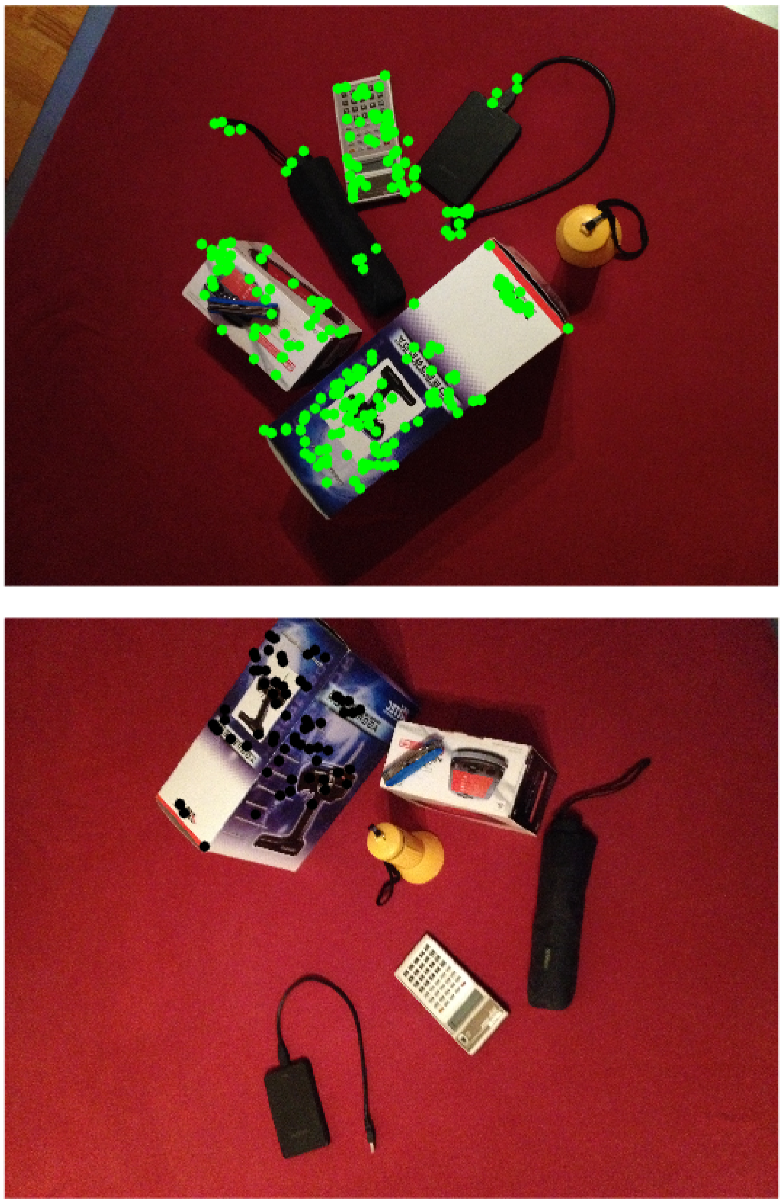


Figure 26: Example model (green) and its complement (black).

Figure 27: Result of the successful merging.

Within this experiment we also noted the occurrence of the degenerate case resulting into wrong rotation transformation.

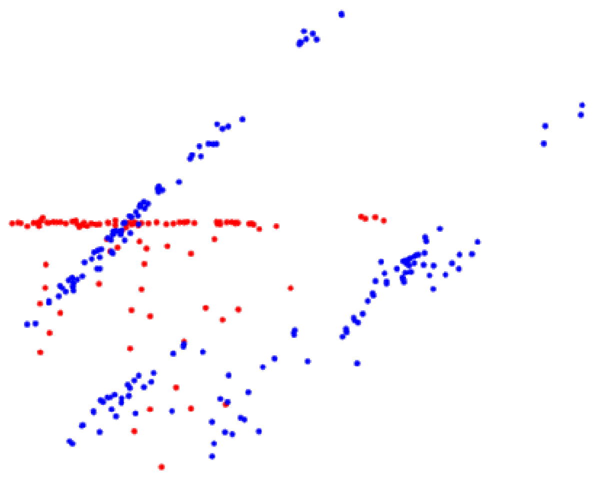


Figure 28: Degenerated result

Again, we compare all of the models with regard to their amounts of reconstructed points.

Model	Original	Complement	Common pts.
1. Black	88	184	0
2. Blue	185	116	3
3. Green	257	70	11
4. Red	106	190	7
5. Purple	18	255	0
6. Turquoise	63	327	1
7. Pink	106	105	2
8. Gold	114	106	6
9. Orange	58	106	0
10. Yellow	34	0	0

7 Future work

Among the issues covered within the Section 3, the proposed modification deals only with the problem of sparse models. Addressing any of the other issues and especially the biggest one being the merging or splitting of objects incorrectly, would dramatically improve the results upon complex scenes and scenes containing multiple similarly shaped objects.

One such approach could be keeping track of multiple possible assignments to future models for each track and not making a strict decision until the final reconstruction.

The weakest point of the current approach is the condition of having knowledge of three common points in general position. (Not in line)

Possible improvement to this method would be additional modification, that could find the transformation between the original and its complement without the prior knowledge of their common points or to guarantee the existence of at least three such points for any general model and its complement. Searching for specific additional matches could be one of the ways to deal with this.

Lastly implementing any of the advanced approaches described in Section 2 should also lead to better results, as most of the current YASFM [1] incremental pipeline's steps are implement by rather basic methods that could be further improved. Especially the graph-based greedy homography grouping algorithm analyzed in 3.2. It could be modified to prevent the merging of objects with background in scenes where on several images the object does not move as can be seen on figures 7 and 10.

8 Conclusion

The main purpose of this work was to perform an in-depth study of the YASFM [1] library. With the use of specifically set up experiments identifying its weaknesses and then working towards the goal of removing them.

Narrowing the focus, we chose to address the issue of big amounts of unreconstructed tracks. With this problem in mind, we proposed a modification to the original pipeline. This two-step modification firstly attempts to reconstruct the leftover tracks as complementary model to the original, then finding a transformation allowing us to merge these two models back together with as low error as possible. For this merging step, we make use of knowledge of common points within both models, reconstructed from split tracks.

Upon implementing these changes, we ran several experiments to prove the concept and evaluate its strengths and limitations. We also cover the special case, resulting in non-standard behaviour. The finalized pipeline is described in the following table.

Step	Code	State
1. Input processing	C++	Unmodified
2. Matching	C++	Unmodified
3. Geometric verification	C++	Unmodified
4. Homography grouping	C++	Unmodified
5. Reconstruct original model	C++	Unmodified
6. Complementary model	C++	Added
7. Merging	C++, MATLAB	Added

Results of the performed experiments were satisfactory and often showed promising potential of this approach to greatly increase the density of models reconstructed by the modified YASFM [1]. The only important requirement upon the input sequence is that the chosen object for reconstruction should be captured from multiple different angles, which greatly increases the amount of split tracks. Otherwise some minor manual adjustment of the complementary model's point-cloud might be necessary.

References

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

[2] David G. Lowe. Distinctive image features from scale-invariant keypoints. In *Proceedings of the International Conference on Computer Vision*, pages 1150–1157, 2004.

[3] Tomas Pajdla. Elements of geometry for computer vision. `http://cmp.felk.cvut.cz/~pajdla/gvg/GVG-2016-Lecture.pdf`, 2016.

[4] Shubhangi L. Vaikole and S. D. Sawarkar. Moving object segmentation with camera in motion using gmec and change detection method. *JMPT*, 6(2):53–60, 2015.

[5] René Vidal and Richard Hartley. Three-view multibody structure from motion. *PAMI*, 30(2):214–227, 2008.

[6] A. W. Fitzgibbon and A. Zisserman. Multibody structure and motion: 3-D reconstruction of independently moving objects. In *European Conference on Computer Vision*, pages 891–906. Springer-Verlag, 2000.

[7] Tianwei Shen, Siyu Zhu, Tian Fang, Runze Zhang, and Long Quan. Graph-based consistent matching for structure-from-motion. *ECCV*, 2016.

[8] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *In Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, pages 298–372, 2000.

[9] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. *SIGGRAPH*, pages 835–846, 2006.

[10] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[11] C. Rubino, M. Crocco, V. Murino, and A. Del Bue. Semantic multibody motion segmentation. *WACV*, 6(2):53–60, 2015.

[12] P. Ji, H. Li, M. Salzmann, and Y. Zhong. Robust multi-body feature tracker: A segmentation-free approach. 2016.

[13] N. Thakoor, J. Gao, and V. Devarajan. Multi-body structure-and-motion segmentation by branch-and-bound model selection. *IEEE Trans. Image Processing*, 19(6):1393–1402, 2010.

[14] G. Pan and K-Y. K. Wong. Multi-body segmentation and motion number estimation via over-segmentation detection. *ACCV Workshops*, (2):194–203, 2005.

[15] Micha Sharir. A strong connectivity algorithm and its applications to data flow analysis. *Computers and Mathematics with Applications*, 7(1):62–72, 1981.

[16] S. Agarwal, K. Mierle, and et al. Ceres solver. `http://ceres-solver.org`, 2010.

[17] J. C. Nash. The singular-value decomposition and its use to solve least-squares problems. In *Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation*, chapter 3, pages 30–48. Adam Hilger, England, 2 edition, 1990.

[18] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *Pattern Analysis and Machine Intelligence*, 1991.