Body reconstruction of moving target with static camera

Master program in Computer Science Engineering
Politecnico of Milano
Email: matteo1.frosi@mail.polimi.it

Abstract

This report describes various topics and methodologies related to the problem of 3D reconstruction of a moving object, given images taken from a single static camera. The mostly used technique is the Visual Hull, which combines information about the silhouette of the target and its pose in the world, to form an approximate reconstruction. The main problem can so be decomposed in sub-tasks, each solvable in different ways. Four demos will be presented, increasing in complexity, and each will follow the same order of steps, that includes silhouette extraction, estimation of the pose of the target and reconstruction from Visual Hull.
Section 1 will be an overview of the problem and the state of the art. Sections 2, 3 and 4 will describe in more detail the techniques used in our project to face respectively the problems of silhouettes extraction, pose estimation and volumetric reconstruction. Section 5 will present the results obtained for the 4 demos. Section 6 will give a brief description of the code architecture, highlighting the relations between modules and the flow of operations used for each demo. Lastly, section 6 will present the conclusion of the author and some ideas for future projects.

Contents

1 Introduction

The main purpose of this project consists in a volumetric reconstruction (application of texture is not required) of a moving target, not necessarily human. A static camera records the target movement and provides a sequence of images describing the scene.
Since we are interested only in the reconstruction of the target and not of the whole scene, that is another whole topic, we would like to isolate the target, namely distinguish foreground and background for every available image of the scene. We can take advantage of the fact that the camera is static, and so is the background in every scene. However, we need to specify the meaning of "***static background***": on the contrary of what the name suggest, static doesn't mean that the background remains immovable. As it may seem intuitive, we would not expect to take two images, even at a short time distance, of the same scene and have them equal to each other. Various elements influence the ambience such as illumination, wind, moving elements and so on. So we will use the term static just as a reference to the fixed camera pose, that takes images where the boundaries of the scene don't change.
Once we have separated foreground from background, we also would like to know the position of the target with respect to the camera. As it is stated, the problem involves the knowledge of the shape of the target and the location of the points belonging to it. We can indeed reformulate the problem comparing it to the movement of a camera around a still target.
We'll give an intuitive explanation here, followed in the relative section by the demonstration, of such duality. Suppose we are given a set of images representing a human or an object from different directions and distances, taken by a single camera (this is the only knowledge that we have). The images are comparable to the previous described extracted foreground. A question arises: did the camera move or the target? Without any further information we are not able to tell.
Once we have reformulated the target pose estimation as camera pose estimation, we can proceed with known algorithms and methods, suitable for 3D reconstruction and points tracking. Among them we mention Structure from Motion approaches [1] and Visual SLAM [2], even though the latter can be considered a particular case of the first. Briefly, while the majority of SfM approaches works on an unordered set of images by exploiting feature matching, vSLAM is supposed to work real-time on an ordered sequence of images, acquired from a fixed camera set-up (both monocular or stereo vision cases). In our project, we will adopt classic SfM techniques, trying to improve them and adapting them to the problem.
Lastly, once we have both silhouettes and camera poses, we can proceed with the Visual Hull reconstruction, well described in [3, 4]. Visual Hull can be computed robustly and efficiently, however it may be very slow, especially when dealing with large scale objects. Moreover, the resulting shape is significantly larger then the true object shape, which makes the approach feasible only when used in approximated variants. Modern approaches use surface-based representation instead of volumetric representation of the scene. In our project we will use a version of Visual Hull that exploits volume quantization, to improve performance: Voxel Carving [5].
The underlying idea is extremely simple, yet efficient. First, the 3D space is split up into a grid of voxels, small cube shaped portions of space that make it discrete. Each voxel is then intersected with the volume generated by each silhouette, determined by the position of the camera associated with the corresponding image. Only voxels that lie inside the silhouette volumes remain part of the final shape. This space carving proceeds iteratively, using all the extracted silhouettes.

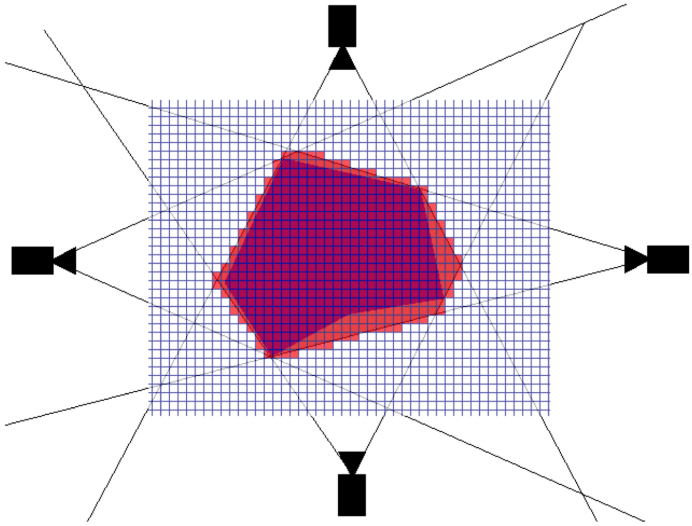

Figure 1: View form above of the voxel carving technique. The purple area describes the real volume occupied by the object, while the red area is the reconstructed volume exploiting silhouette re-projection and carving. As it can be seen in figure, complex shapes, such as convex ones, may present some difficulties in the reconstruction. This is due to the fact that silhouettes do not have any information about object cavities.

Aside from Voxel Carving, different algorithms are used for volumetric and non-volumetric reconstruction, including: Marching intersections, Image-based visual hulls and Exact polyhedral methods.

2 Silhouette extraction

Given $E(u, v, z)$ as the eye vector, a point on a surface $\sigma(u, v, z)$ with surface normal $N(u, v, z)$ is a silhouette point if $E(u, v, z) \bullet N(u, v, z) = 0$, that is, the angle between $E(u, v, z)$ and $N(u, v, z)$ is 90 degrees. This definition includes internal silhouettes as well as the object's outline, or halo. It is important to note that the silhouette set of an object is view dependent, that is the edges of a model that are silhouettes change based on the point from which the object is viewed.

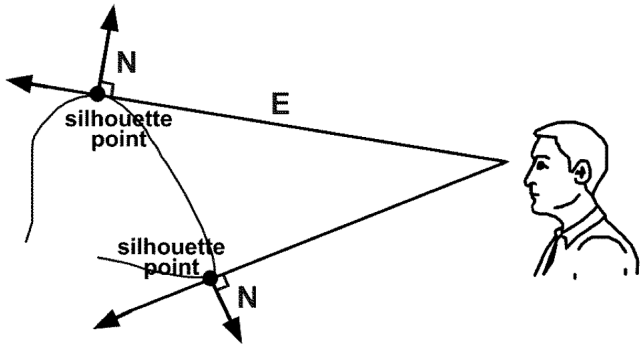

Figure 2: Silhouette definition from a two dimensional point of view

One of the steps of our project is to extract the silhouettes of the target in each image. Since we deal with a static camera, we can assume available, along with the images containing the target, a set of frames of the background, taken from a short video of it (1∼2 minutes). The problem of silhouette extraction becomes then background subtraction, and in the last 40 years plenty of methods and algorithms have been developed and discussed. Following, we present four techniques, along with their advantage and disadvantages, comparing them.

2.1 RGB thresholding with parameters estimated by genetic algorithm

A standard and basic technique to segment an image is grayscale thresholding. An image is first converted, if needed, from RGB to Grayscale color space. Each pixel in the image is then replaced by a black pixel if the image intensity $I_{i,j}$ is less than some fixed constant T, or a white pixel if the image intensity is greater than that constant. The technique is extremely simple, but it fails to capture information about the color channels. When converting to Grayscale color space, we simply average all the channels, so we lose knowledge about how colors are distributed.

An improvement, that is what we use in the project, is to apply thresholding separately on each color channel. This leads to a problem: while in the monochromatic case, the threshold value T could be manually selected with a small amount of effort, using three separate channels leads to an explosion in the number of combinations of T_R, T_G and T_B. Suppose that the user has a trained eye and is able to say that the threshold lies in a range of 30 values. In the first case he has to try at worst 30 times the value, before finding the best. In the other he must try 30·30·30 times, surely not feasible.

We then propose a genetic algorithm [6] to detect the best threshold combination, given a single silhouette sample. We consider a population of n chromosomes (user selected) with 3 genes each, representing the combination of color thresholds to use later. The algorithm evolves in a standard way, doing fitness evaluation, best chromosome selection, crossover and mutations. The fitness function for a chromosome is the number of correctly classified pixels, obtained using its genes values as threshold for image segmentation, compared to

the silhouette sample. The algorithm ends after a fixed number of iterations (also user selected) or when the fitness function provides a good minimal result.
Once the parameters are selected, a simple model for each color channel of the background is estimated, doing the mean of all the available images. Then each frame of the sequence containing the target is compared, again considering each color channel separately, with the computed background, using the previously found thresholds.

2.2 HSV thresholding

It has been noticed that image segmentation in different color spaces than RGB provides better results [7]. In our project we also try one of these technique, to compare against the others, using the HSV color space, that is more robust towards external lightning changes. As before, the background is modelled as a simple mean of all its available images, then it is converted in HSV color space. Each frame of the sequence containing the target is first converted, then confronted with the background model, using a bit-wise XOR comparison, giving positive output if two pixels differ. The output is then positive thresholded and smoothed applying a median filter. Lastly, we do denoising, clustering similar regions in the binary image. The last two steps, median filtering and denoising, are done in order to "correct" the errors of the thresholding that has been done without any model of the variance of the background.

2.3 Statistic modeling

In the previous subsections we modeled the background as a simple average of the available images. This is surely a fast approach, but lacks in robustness and does not capture any information about ambience changes, such as variation of illumination or weather factors such as rain or wind. In the 1999, Horprasert proposed a statistic approach to do background subtraction, described in [8]. First a model of the background is built, considering both brightness and color distortions, then it proceeds to classify each pixel of a target image in four classes: original background, shaded background or shadow, highlighted background or moving foreground object. The main advantage of such approach is that it provides a robust model of the background, that can be reused for different applications. However, it is very computational extensive, since it computes the distortions for each pixel of the desired background image. Another major fault that makes the method almost not feasible for large datasets is that pixels with small variation in chromaticity distortion lead to very high values of normalized distortions, likely to exceed the foreground threshold. Hence a default minimum value b_i has to be found with a trial and error approach, further extending the computational time.

2.4 Efficient statistic modeling for shadow removal

The approach described in [9] uses robust statistical descriptors to model the background image, obtaining a noise estimate. Foreground pixels are extracted, and another statistical approach combined with geometrical constraints are adopted to detect and remove shadows. The core of the proposed algorithm consists of a simple background subtraction scheme, in which the background model is extracted in a training stage, and foreground pixels are obtained in the test stage. After the initial foreground extraction, morphological operators are applied to remove isolated responses, and the remaining foreground pixels are processed for shadow removal. Lastly, morphological operators are applied again, for noise cleaning.

2.5 Adaptive Background Mixture Model

The authors of [10, 11] describe a method to model each background pixel by mixture of K Gaussian distributions (usually from 3 to 5). Different Gaussians are assumed to represent different colors. The weight

parameters of the mixture represent the time proportions that those colors stay in the scene. The background components are determined by assuming that the background contains B highest probable colors. These are the ones which stay longer and more static. Static single-color objects trend to form tight clusters in the color space, while moving ones form widen clusters due to different reflecting surfaces during the movement. To allow the model to adapt to changes in illumination a scheme based upon selective updating is used. Every new pixel value is checked against existing model components in order. The first matched model component will be updated and if no match is found, a new Gaussian component is added with the mean at that point, along with a large covariance matrix and a small value of the weighting parameter.

3 Target pose estimation

The target pose estimation is a complex problem that involves many phases. In our project we deal with it first reformulating the problem as if it was camera pose estimation, then adopting the state of the art strategies, namely structure from motion algorithms. There is although an important consideration to do when dealing with finding correspondences between images. Usually, when dealing with camera pose estimation, we expect to find a huge number of correspondences, since the whole scene is captured from different angles. In other words, feature matching between images consists in both features from the foreground and background. For target pose estimation (or the dual problem, discussed later) this is not so: since the background is static with respect to the camera taking images, the background provides no information about the movement of elements in the scene, and only features extracted from the foreground can be used in SfM algorithms. It is correct and intuitive to assume that the number of features extracted from only the foreground (moving target) is much less than the number of features of the whole scene. This leads to incorrect results when dealing with simple matching algorithms. In our project, we propose an estimation algorithm based on the KLT tracking algorithm [14, 15], that for every frame boosts the points tracked by the algorithm with new features extracted from the image. Lastly, we will use SURF features [16], since they provide good results and performances [17].

3.1 Duality of the target pose and camera pose estimation problems

As stated in the introduction, the target pose estimation problem has a dual version, the camera pose estimation. Before proceeding, some concepts of geometry about rotation matrices are required.

Theorem 1. *A rotation matrix can be expressed as a series of three successive rotations about coordinate axes.*
Proof We will prove the theorem for a generic three dimensional rotation, considering the Cartesian coordinate system described by the three axes versors: $[1,0,0]^T$, $[0,1,0]^T$ and $[0,0,1]^T$. It is trivial to demonstrate it in different dimension or coordinate system.
Consider a rotation around each of the three axes, respectively of angles ϕ, θ and ψ. Consider then the resulting rotation matrix, obtained as composition of a rotation around the z, y axis and x axis in the reverse order.

$$R = \begin{bmatrix} cos\psi & -sin\psi & 0 \\ sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & -sin\phi \\ 0 & sin\phi & cos\phi \end{bmatrix}$$

$$= \begin{bmatrix} cos\psi cos\theta & cos\psi sin\theta sin\phi - sin\psi cos\phi & cos\psi sin\theta cos\phi + sin\psi sin\phi \\ sin\psi cos\theta & sin\psi sin\theta sin\phi + cos\psi cos\phi & sin\psi sin\theta cos\phi - cos\psi sin\phi \\ -sin\theta & cos\theta sin\phi & cos\theta cos\phi \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \quad (1)$$

It is trivial to associate each element of R with a particular expression of angles.

Theorem 2. *The rotation matrix around a simple axis, describing an angle ψ in anticlockwise sense and the rotation matrix around the same axis, describing the same angle in clockwise sense are one the inverse of the other.*
Lemma 2. *The two matrices are also one the transpose of the other.*
Proof Let's consider the rotation around the z axis (for the two other axes, the demonstration is left to the reader) and calculate the inverse.

$$R_z(\psi)^{-1} = \begin{bmatrix} cos\psi & -sin\psi & 0 \\ sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} cos\psi & sin\psi & 0 \\ -sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} cos(-\psi) & -sin(-\psi) & 0 \\ sin(-\psi) & cos(-\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} = R_z(-\psi) \quad (2)$$

Since the rotation matrix is orthonormal, it is true that it's inverse is also the transpose, as stated in Lemma 2.

Theorem 3. *A rotation matrix and its transpose are one the inverse of the other.*
Proof We apply the result of the previous two theorems to prove the theorem for a generic rotation R.

$$\begin{aligned} R(\psi,\theta,\phi) * R(\psi,\theta,\phi)^T &= [R_z(\psi)R_y(\theta)R_x(\phi)] * [R_z(\psi)R_y(\theta)R_x(\phi)]^T \\ &= [R_z(\psi)R_y(\theta)R_x(\phi)] * [R_x^T(\phi)R_y^T(\theta)R_z^T(\psi)] \\ &= [R_z(\psi)R_y(\theta)R_x(\phi)] * [R_x(-\phi)R_y(-\theta)R_z(-\psi)] \\ &= R_z(\psi)R_y(\theta)R_x(\phi)R_x(-\phi)R_y(-\theta)R_z(-\psi) \\ &= R_z(\psi)R_y(\theta)I_3R_y(-\theta)R_z(-\psi) = R_z(\psi)I_3R(-\psi) = I_3 \end{aligned} \quad (3)$$

Now we have all the basic knowledge to reformulate the pose estimation problem, and the duality is straightforward. Let C be a column vector describing the location of the camera center in world coordinates, and let R_C be the rotation matrix describing the camera's orientation with respect to the world coordinate axes. The transformation matrix that describes the camera's pose is then $[R_C \mid C]$. We make the matrix square by adding an extra row of $[0, 0, 0, 1]$. Then the extrinsic matrix is obtained by inverting the camera's pose matrix:

$$\begin{bmatrix} R & t \\ \underline{0} & 1 \end{bmatrix} = \begin{bmatrix} R_C & C \\ \underline{0} & 1 \end{bmatrix}^{-1} = \begin{bmatrix} R_C^T & -R_C^T C \\ \underline{0} & 1 \end{bmatrix} \quad (4)$$

A 3D world point $\widetilde{X}$, in inhomogeneous coordinates, is then projected by the camera to the 2D point u by: $u = K * \begin{bmatrix} R_C^T & -R_C^T C \\ \underline{0} & 1 \end{bmatrix} * \begin{bmatrix} \widetilde{X} \\ 1 \end{bmatrix} = K * R(\widetilde{X} - C)$.

Theorem 4.
Consider an image point u, corresponding to a world point X', obtained rotating around the origin a point X. Let Q be the rotation matrix describing the movement and R_C and C be respectively the rotation matrix and the location in world coordinates of the camera taking the image of point X'. Consider then an image point u', corresponding to world point X, but taken with a camera which rotation matrix and location are transformed by Q^T. Then u and u' are the same image points.
Proof The proof is straightforward and takes into account that $X' = QX$ and that to transform a rotation matrix we left multiply it by the transformation matrix.

$$u = KR(\widetilde{X}' - C) = KR(Q\widetilde{X} - C) = KR(Q\widetilde{X} - QQ^T C) = K * (RQ) * (\widetilde{X} - Q^T C) = u' \quad (5)$$

$$(RQ) = R_C^{*^T} \Rightarrow R_C^* = Q^T R^T = Q^T R_C \quad (6)$$

From C to QQ^TC we can go using the result of Theorem 3, multiplying C by the identity matrix and then decomposing it.

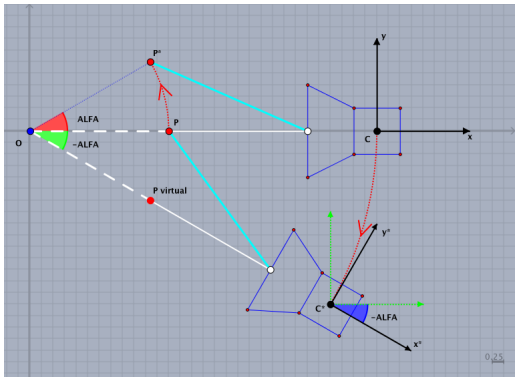

Figure 3: Geometric interpretation of Theorem 4

The figure on the side gives a simple description of the geometric meaning of the theorem. Point P is rotated anticlockwise of an angle $ALFA$ and the camera with center in C takes its image. Also, the camera rotation is described by R and is represented by the black axes y and x. The theorem tells us that the image of point P^* taken by such camera, is not distinguishable by the image of point P taken by the same camera but which rotation and location of the center are rotated clockwise of the same angle $ALFA$ (e.g. rotate anticlockwise by $-ALFA$). The intuition comes from observing two parts of the figure. Consider the camera mentioned first, points P and P^* and the rotation angle. Consider then the last mentioned camera, points $P\,virtual$ and P and the rotation angle. Separately, without any information about the world frame, they represent the same situation. Moreover, it is trivial to see that a simple rotation of the world frame changes from the first scenario to the second one.

3.2 Camera pose estimation

Each frame corresponds to a specific camera virtual pose (virtual because it is static, but for Theorem 4 we can image a fake moving camera). The estimation of such poses can be solved exploiting well known epipolar geometry [12]. We can fix the initial camera pose, given by the first (properly chosen) view in the world frame, meaning that the first camera pose is represented by the matrix $P_1 = [I_3 \mid \underline{0}]$. The goal is to find all the remaining poses, of the form $P_i = [R_{C_i} \mid C_i]$, $i \in \{2, ..., \#Frames\}$. The first step is to find P_2, looking for correspondent points in the first two frames. Depending on the availability of the camera intrinsic parameters K, that obviously remain the same for all the virtual cameras, we can estimate the Fundamental (without K) or the Essential (with K) matrices. In our project we have the availability to calibrate the camera, hence we will focus on the Essential matrix computation. At least 6 correspondences are needed in theory to compute E_{12}, but for a robust approach more correspondences can be used [13]. Let E_{ij} be the Essential matrix related to cameras i and j, then it holds that $E_{ij} = [t]_{\otimes}R$, where t is the translation between the cameras and R their relative rotation and $[]_{\otimes}$ is an operator such that $[t]_{\otimes} = \begin{bmatrix} 0 & t_y & -t_z \\ -t_y & 0 & t_x \\ t_z & -t_x & 0 \end{bmatrix}$. In our case we have fixed first camera pose, hence $E_{12} = [t_2]_{\otimes}R_2$, where $t_2$and R_2 represent respectively the translation and rotation of the camera w.r.t. the first one.

However, the information is correct up to a scale factor, that can be obtained exploiting knowledge of the scene. Suppose to have two world points, X_1 and X_2, whose distance d is known, along with their image coordinates in two frames. Once we have extracted the parameters from the Essential matrix, we can triangulate the points in both frames and then compute the scale factor.

The estimation process goes on iteratively, finding correspondences between frames i and j, estimating the Essential matrix E_{ij}, extracting the pose of camera j w.r.t. camera i and rescaling it.

3.3 Structure from motion by simple matching

To provide better results, the sequence of frames are first ordered w.r.t. the target rotation and cleaned of duplicates. Moreover, since the SfM algorithm accumulates errors when the number of different poses grows, a good idea is to break the sequence into two rotation, clockwise and anticlockwise, starting from a particular

frame F_k selected by the user, and estimate the poses separately. Once this preprocessing is done, the algorithm works as follow:

1. Extract SURF features from the first and second frames
2. Match corresponding features, using an algorithm described in [18, 19]
3. Estimate the Essential matrix E_{12} and compute the scale factor between images
4. Extract SURF features from the next frame
5. Match corresponding features with the previous frame
6. Estimate the camera pose exploiting the known poses and triangulated 3D world points
7. If the current frame is a keyframe, do bundle adjustment [20] to optimize poses and triangulated points coordinates
8. If the current frame is the last of the sequence STOP, otherwise GOTO 4

We consider keyframe an image that is located in the sequence after a fixed value. In our project we consider keyframes the images that are at an even position in the sequence, starting from the fourth one. Such a tight gap is due to the fact that we aim to obtain the most accurate result possible, even increasing the computational cost.
The algorithm performs well when dealing with a large number of features, providing accurate results. However, for small resolution sequences or sequences where the target contains a low number of features (e.g. monochromatic dresses) it performs worse, giving not acceptable outcomes or even failing some steps (BA fails often, because it requires a quite large amount of triangulated world points having small reprojection error and high confidence).

3.4 Structure from motion by Interrupted KLT tracking

The way KLT tracking algorithm works is pretty simple. In its basic form, it tries to find a shift that an interest point (detected feature) might have taken. Its framework is based on local optimization, usually a squared distance criterion over a local region that has to be optimized w.r.t. the transformation parameters (e.g. displacement in x and y). To solve this problem, it approximates the feature displacement with a linear term. This method can also be extended to take into account registration based on more complex transformations such as rotations, scaling or shearing. Since the basic KLT tracking algorithm can deal with small pixel displacements and we have no information about the rotation of the target between images, we use a variant of the algorithm, which exploits a pyramidal representation of the images [21].
Our proposed algorithm works as follow, sharing the same structural similarity with the one described in the previous subsection:

1. Extract SURF features from the first frame
2. Track the previously found features in the second frame
3. Estimate the Essential matrix E_{12} and compute the scale factor between images
4. Extract SURF features from the current frame and add them to the tracked ones
5. Track the previously found features (tracked + extracted) in the next frame, interrupting the old tracking flow

6. Estimate the camera pose exploiting the known poses and triangulated 3D world points
7. If the current frame is a keyframe, do bundle adjustment [20] to optimize poses and triangulated points coordinates
8. The next frame becomes the current frame
9. If the current frame is the last of the sequence STOP, otherwise GOTO 4

Simply applying the tracking algorithm is useless: starting from the initial frame, considering that the sequence is ordered w.r.t. the target rotation, an extracted feature will be tracked for a small amount of subsequent frames, leading to fewer and fewer points as the frames are analyzed. To avoid such problem and still exploit the good performances of the KLT algorithm, we "boost" the tracked points in image i with the extracted features of the same image, that will include additional points, not visible in the previous frames. Originally we intended to adopt a $k-$sliding window technique, so to "boost" an image only after $k-1$ iterations. Turns out that the best results are given for $k=2$, meaning that for each image, except the first one, we both consider the previously tracked and the currently extracted features.
As before, the keyframe is an image that is at an even position in the sequence, starting from the fourth one.
With respect to the previous algorithm, described in 3.3, it allows to get the poses with higher confidence and without the need to break the sequence (while the ordering and cleaning are still extremely useful).

3.5 Preprocessing of the sequence

As mentioned in the previous two subsection, it is good practice to work on the sequence of images before estimating the relative poses. Following we present, in order, all the procedures that can be applied to the sequence to improve performances and results.

3.5.1 Sequence ordering

Although the sequence of frames comes from a video, and so it is supposed to be ordered, it may include some redundant images. Suppose, for example, that the target spins two times. The corresponding sequence will be half useless, since if we assume a constant angular velocity of the target and a constant rate of frame extraction, the second half of images will be the same of the first one. Keeping all the frames will result in heavily increasing the computational time, already extended by the pose estimation algorithms. The idea is then to completely order the sequence, depicting a complete rotation from the first to the last image of the new sequence, and no more than one. Exploiting the technique described in [22] we are able to juxtapose equal or similar images.

3.5.2 Sequence cleaning

As stated before, the ordered sequence will contains redundant images, almost copies of each other. The idea is then to remove images that are too much similar. Since the images are ordered, we expect that two consequent frames in the sequence, call them F_i and F_j, have corresponding features distant from each other of a certain length, corresponding to the rotation velocity, and their color histograms are not too similar. The similarity check consists in the following procedure:

1. Start from the first image in the ordered sequence, named F_i and the second image, F_j
2. Extract and match features from F_i and F_j
3. If the mean distance between the matched features less than a selected threshold, the images are the same and the second one is flagged as "copy". F_j is the new F_i and the next frame is considered as F_j. If there is no next frame, GOTO 6, otherwise GOTO 2

4. If the cross-bin distance for each color histogram of the images is less than a selected threshold, the images are the same and the second one is flagged as "copy". F_j is the new F_i and the next frame is considered as F_j. If there is no next frame, GOTO 6, otherwise GOTO 2

5. Slide the sequence window: F_j is the new F_i and the next frame is considered as F_j; if there is no next frame, GOTO 6, otherwise GOTO 2.

6. Once all the images are flagged, cut the "copy" ones from the sequence.

The cross-bin distance is computed with the Earth Mover Distance algorithm (EMD), described in [23]. We could also have used EMD for the ordering the sequence, but the proposed method relies also on features matching, making the procedure more robust. *Remember, in fact, that histogram based methods have a major fault: they cannot distinguish a tomato from a small red ball*.

3.5.3 Frames selection

Not all the images will be useful in the camera pose estimation phase: since we are dealing with a small foreground, it may happen that two subsequent images do not share enough matching points to provide an accurate estimation, especially when dealing with the algorithm described in 3.3. We further filter the sequence selecting the largest subset of images, computed as if the sequence was a circular list of images, which share at least a fixed number of matching features.

3.5.4 Sequence breaking

To avoid errors accumulating when estimating the camera poses, noticeable when using a lot of images, we break the sequence into two sub-sequences, showing respectively an half-anticlockwise and half-clockwise rotation of the target. The break point, or reference image, is user selected and detected in the sequence exploiting once again the EMD.
This step serve two different goals, depending on the pose estimation algorithm considered. If we use the SfM from simple matching algorithm (3.3), the main purpose of sequence breaking improves the result accuracy, but if we take the SfM from Interrupted KLT algorithm (3.4), aside from slightly improve the results, it is needed to bound the computational time. At each step, the algorithm grows in the number of active features, greatly increasing as the number of images considered. Breaking the sequence halves the number of iterations, reducing the almost exponential cost.

4 Voxel Carving

As stated in the introduction, voxel carving is a procedure that consists in the progressive elimination of voxels, starting from a dense 3D discretization of the space surrounding the target, based on the extracted silhouettes and the corresponding camera poses.
Since we allow some inaccuracy in the silhouette extraction, we model a voxel as a four dimensional object. The first three dimension correspond to the voxel center, while the fourth represents a not normalized fuzzy discrete value for the voxel. The uncertainty of the silhouette extraction is translated into the fact that a voxel does not just belong or not to the target, but it belongs to it with a certain degree of truth. For each available sequence of images, the algorithm works in the following way:

1. Create a grid of voxels which contains the target. It requires the knowledge of the target position in the world with respect to the camera frame. The volume occupied by the voxels influences both the computational time of the algorithm and the precision, in an inverse relation. All the voxels are initialized with a fuzzy value corresponding to the number of images in the whole starting sequence
2. Consider the first silhouette of the sequence as the current image
3. Project the voxels on the current image using the corresponding camera parameters
4. If the voxel overlaps the silhouette, it is kept, otherwise its fuzzy value is decremented by 1. If the value is less than a selected threshold, the voxel is discarded
5. If there is no remaining frame in the sequence STOP, otherwise the current image is the next frame

We adopt a discrete fuzzy modeling of the voxels to have more freedom in the reconstructed model. Visualizing the model bounding the showed voxels to their fuzzy value also provides some information on the accuracy of the silhouette extraction method.
In the first point of the algorithm we stated that we need the target position. To do this we perform a pre-carving, on a wider space. This volume is created considering the camera positions as the boundaries of this parallelepiped-shaped space (the height is more than two times the base length) and discretised in 10 million voxels. From the obtained carving, we take the maximum and minimum values for each axis, among all the points of the reconstruction, incrementing them by 10% to avoid errors: they will be the limit values for the dense carving (20 million voxels) described before.

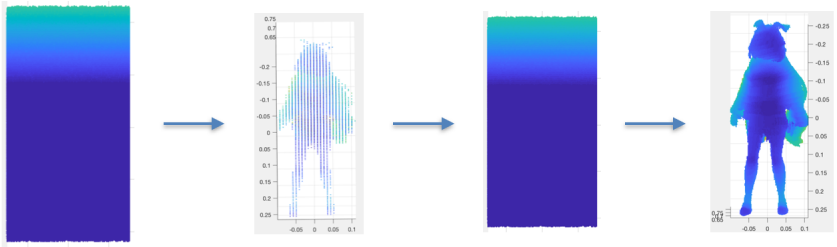

Figure 4: Carving high level phases

5 Results

In this section we describe all the results obtained for each demo, in every step described in the previous sections. The demos presented in the project are 4, and each is characterized by its own set of advantages and disadvantages. Moreover, we imposed an increasing difficulty in the reconstruction, described in the following list.

Demo 1

- Almost dichromatic background, with small brightness variations, allowing an easy background subtraction (ADV).
- The moving target is rigid: all its body parts move together, allowing a better voxel carving (ADV).
- The good camera focus provides images with well defined edges (ADV).
- There are very few features on the sides and back of the target, making hard to estimate all the poses (DISADV).

Demo 2

- The moving target is rigid: all its body parts move together, allowing a better voxel carving (ADV).
- The good camera focus provides images with well defined edges (ADV).
- There are enough features on the sides of the target and on the back, making easier to estimate all the poses (ADV).
- The background represents a scene subjected to weather conditions, such as variable wind and small illumination changes (DISADV).

Demo 3

- The background is almost stationary, with small chromatic and brightness variations (ADV).
- The moving target is not rigid: some body parts move w.r.t. each other, producing inaccuracies when carving (DISADV).
- There are very few features on the sides and back of the target, making hard to estimate all the poses (DISADV).
- The focus of the camera is not properly tuned, producing images slightly blurred in some regions (DISADV).

Demo 4

- There are a lot of features in the scenes, allowing an optimal pose estimation (ADV).
- The moving target is almost rigid, since only the head and torso are considered and it is unlikely that they move differently w.r.t. each other (ADV).
- The focus of the camera is not properly tuned, producing images slightly blurred in some regions (DISADV).
- The scene is poorly illuminated and the source of light is located right behind the camera (DISADV).

5.1 Demo 1

SUBJECT: character model Unity-chan, rendered with Unity3D
CAMERA FEATURES: Mac OSX screen capture + Unity3D virtual camera
DISTANCE AND CAMERA POSITION: the target is about 70 cm distant from the camera, located parallel w.r.t. the ground at 25 cm height.

5.1.1 Inputs

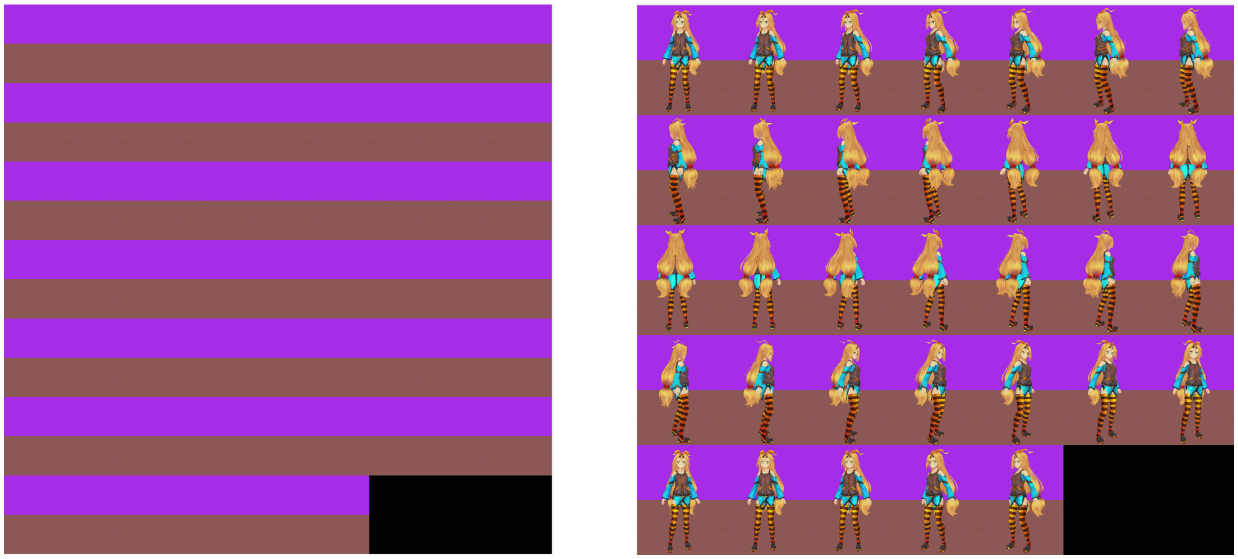

Figure 5: Background images and target images

5.1.2 Silhouettes extraction

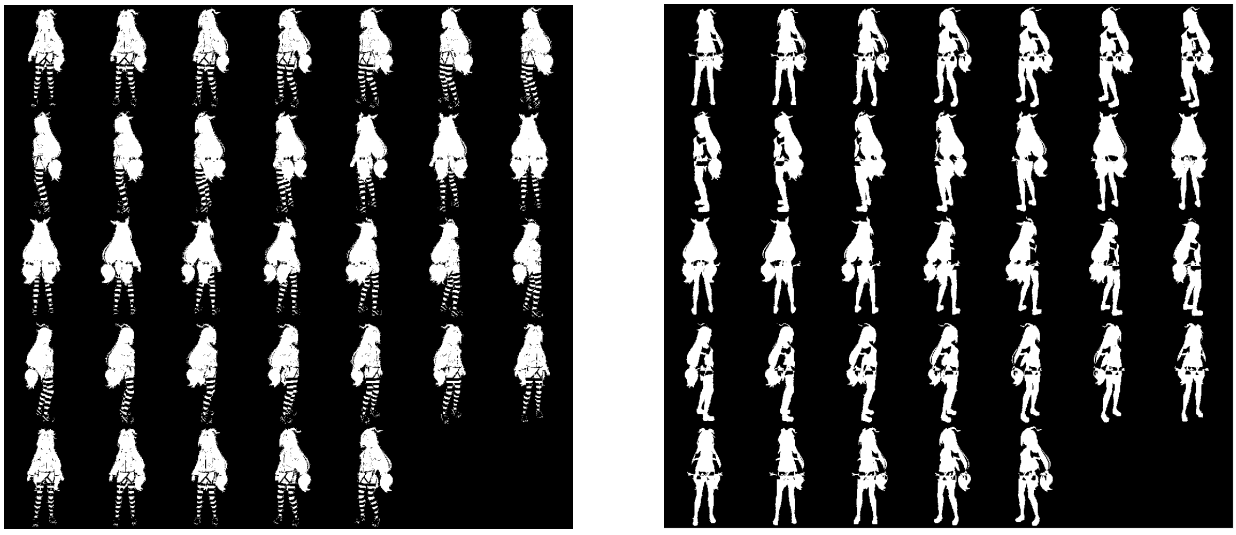

Figure 6: Silhouettes extracted using RGB (left) and HSV (right) thresholding

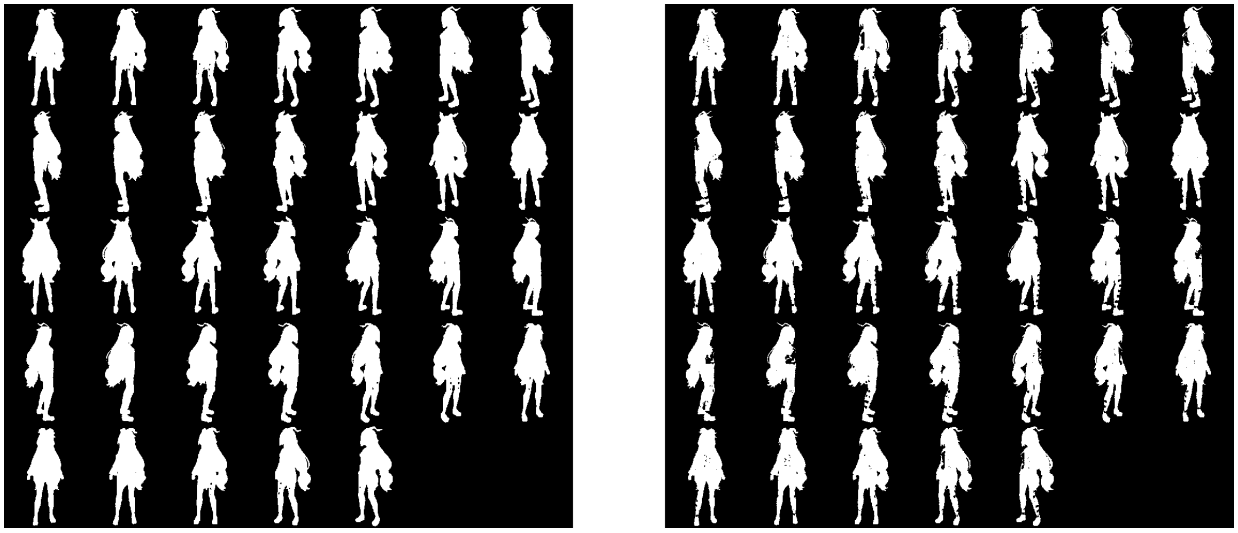

Figure 7: Silhouettes extracted using statistical modeling (left) and efficient statistical modeling (right)

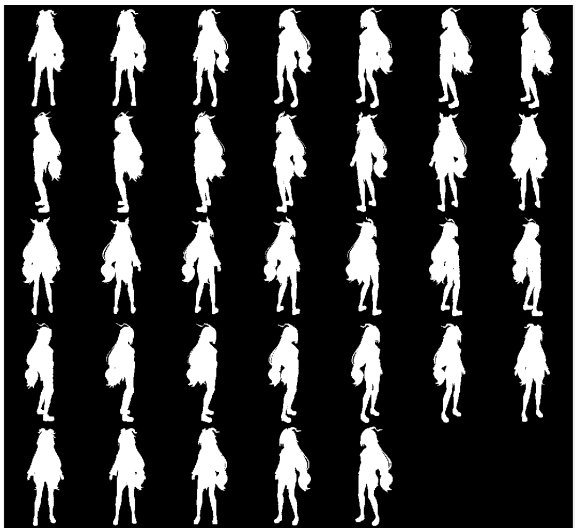

Figure 8: Silhouettes extracted using adaptive background mixture modeling

	RGB+GA	**HSV**	**Statistical modeling**	**Efficient statistical modeling**	**Adaptive background mixture model**
Performance	95,73%	97,21%	99,23%	98,98%	100%

The performance is evaluated considering the Adaptive Background Mixture Model silhouettes as ground truth images, since the method provides the best results. For each set of silhouettes we do an average over the correctly classified pixels (not only just belonging to the silhouette, but also black pixels related to the background): this explains the high values in the table.

5.1.3 Pose estimates

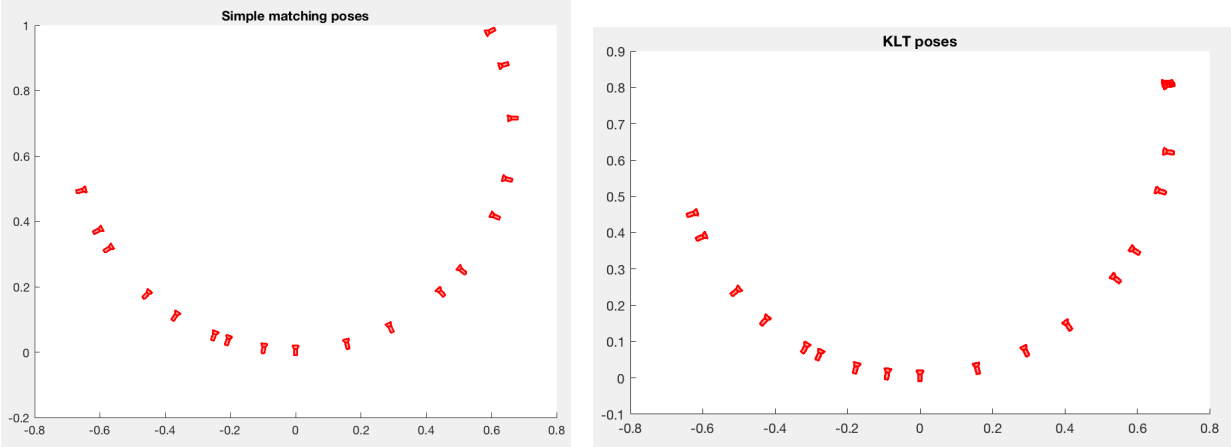

Figure 9: Camera poses estimates from above

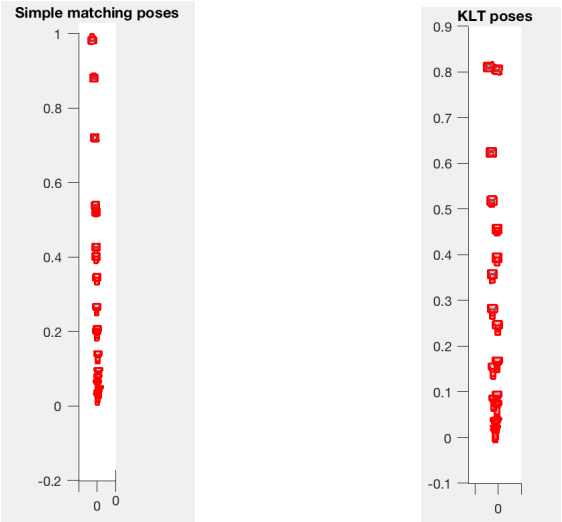

Figure 10: Camera poses estimates from the side

Only a small amount of frames can be used in the pose estimation due to the lack of features. In this demo, both estimation methods provide an accurate results, even if with small difference. When doing voxel carving, we will use the first, since it has a wider range of motion and better represents the parallelism of the camera and the ground.

5.1.4 Volumetric reconstruction

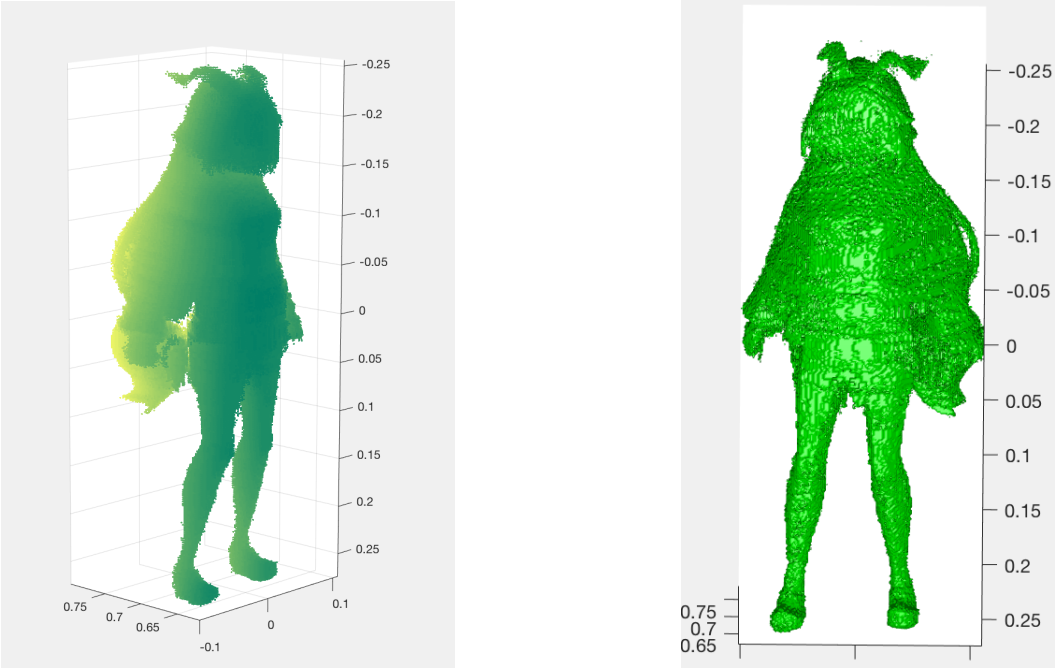

Figure 11: Reconstruction after voxel carving from poses estimated with simple matching

5.2 Demo 2

SUBJECT: character model Sapphire-chan, rendered with Unity3D
CAMERA FEATURES: Mac OSX screen capture + Unity3D virtual camera
DISTANCE AND CAMERA POSITION: the target is about 1.40 m distant from the camera, located inclined of 10° w.r.t. the ground, at 35 cm height.

5.2.1 Inputs

Figure 12: Background images and target images

5.2.2 Silhouettes extraction

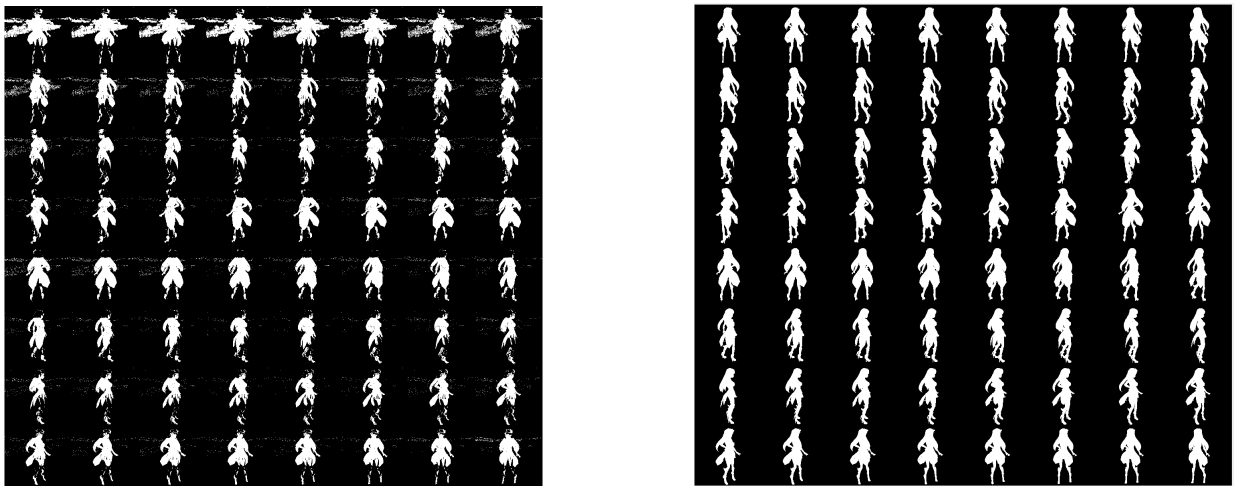

Figure 13: Silhouettes extracted using RGB (left) and HSV (right) thresholding

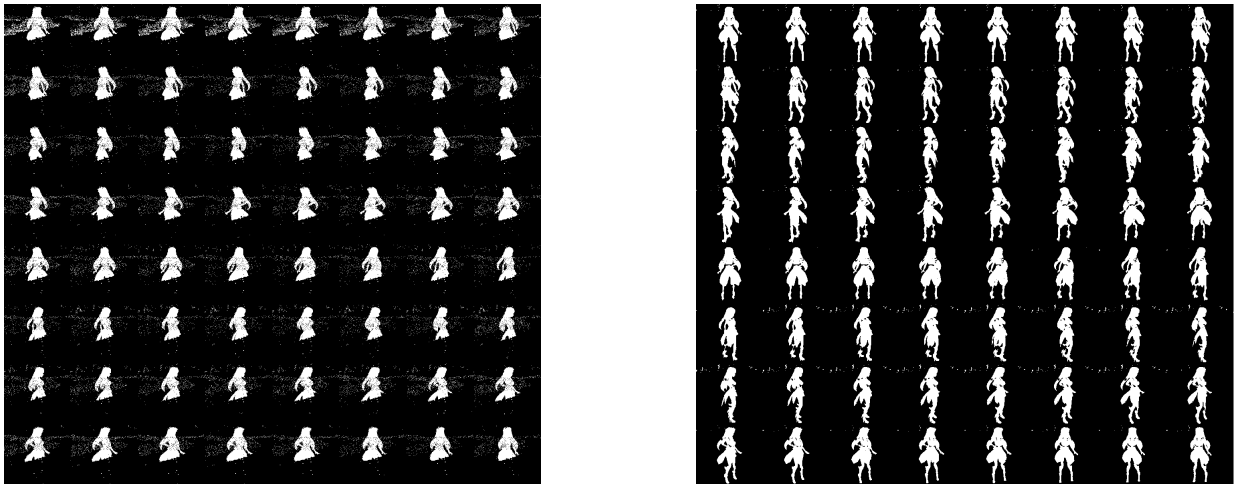

Figure 14: Silhouettes extracted using statistical modeling (left) and efficient statistical modeling (right)

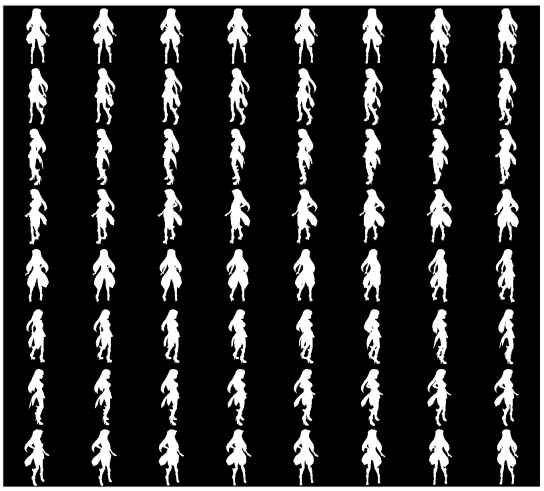

Figure 15: Silhouettes extracted using adaptive background mixture modeling

	RGB+GA	HSV	Statistical modeling	Efficient statistical modeling	Adaptive background mixture model
Performance	95,14%	99,54%	93,48%	99,14%	100%

The performance is evaluated considering the Adaptive Background Mixture Model silhouettes as ground truth images, since the method provides the best results. For each set of silhouettes we do an average over the correctly classified pixels (not only just belonging to the silhouette, but also black pixels related to the background): this explains the high values in the table.

5.2.3 Pose estimates

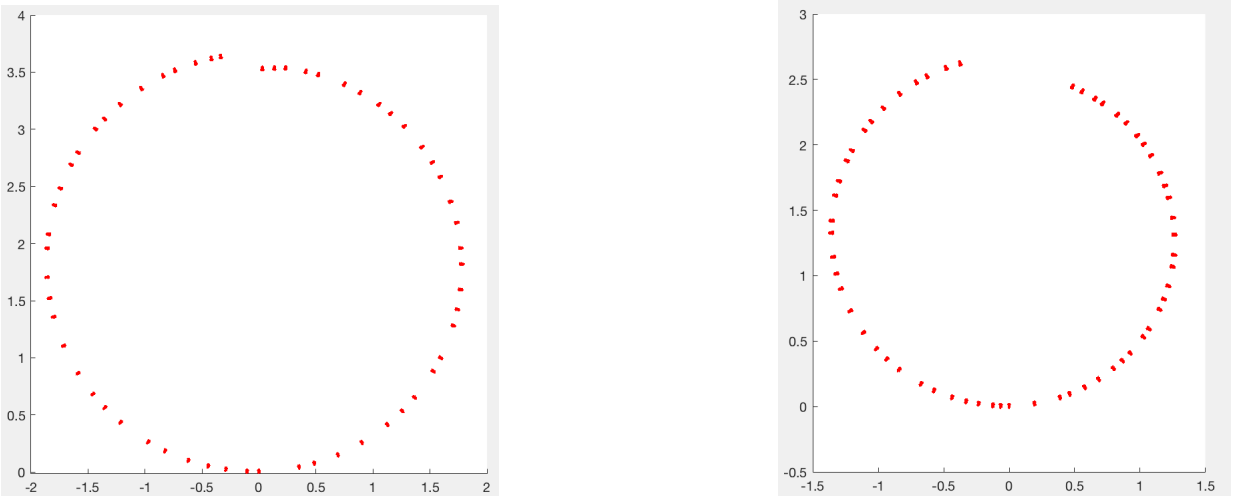

Figure 16: Camera poses estimates from above, left from simple matching and right from interrupted KLT

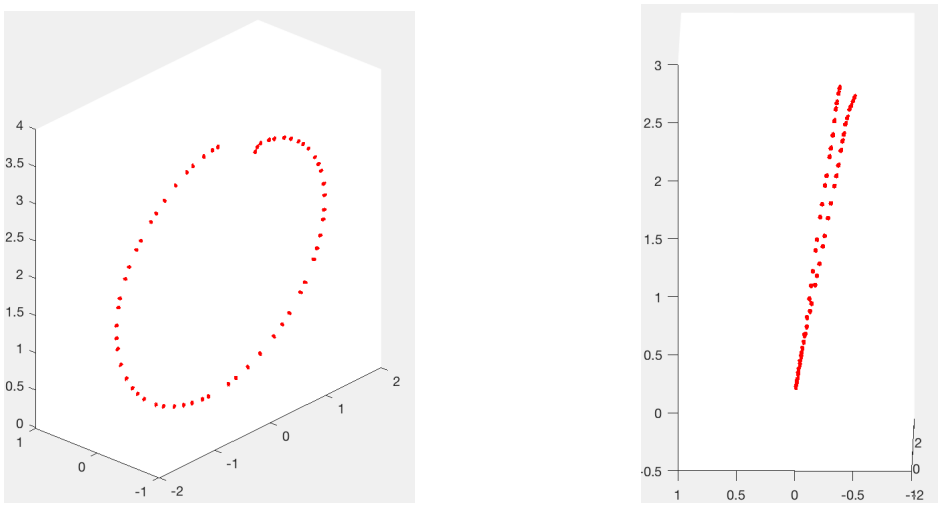

Figure 17: Camera poses estimates from the sides, left from simple matching and right from interrupted KLT

Both methods correctly estimated the angle of the camera w.r.t. the ground (seen by the oblique positions in the space, when the initial camera is considered reference with no rotation and no displacement). It is noticeable however that the second method results in less displacement and more accurate results, while the first gives an overestimation of the distance and closes, inducing an error, the loop. In fact, the preprocessing leaves an open space within the sequence of images, so that the target does not perform a full rotation. Interrupted KLT correctly estimates this gap.

5.2.4 Volumetric reconstruction

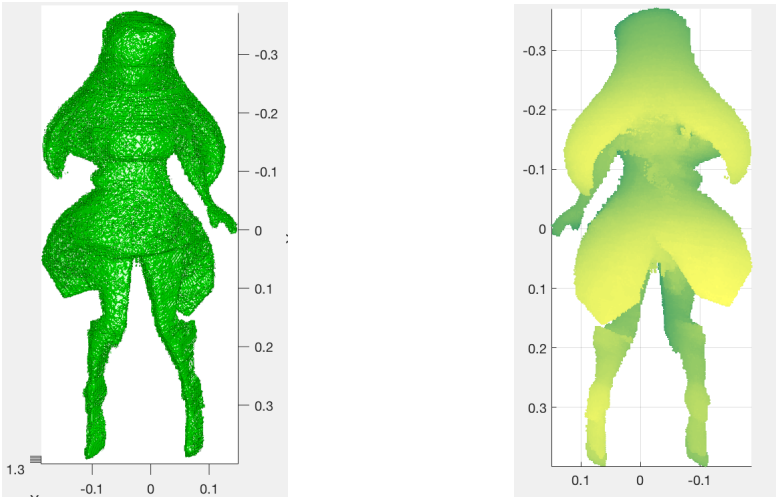

Figure 18: Reconstruction after voxel carving from poses estimated with interrupted KLT

5.3 Demo 3

SUBJECT: project author, Matteo Frosi
CAMERA FEATURES: Huawei P8 Lite frontal camera, 720x1280 px resolution
DISTANCE AND CAMERA POSITION: the target is about 1 m distant from the camera, located inclined to the left w.r.t. the ground.

5.3.1 Inputs

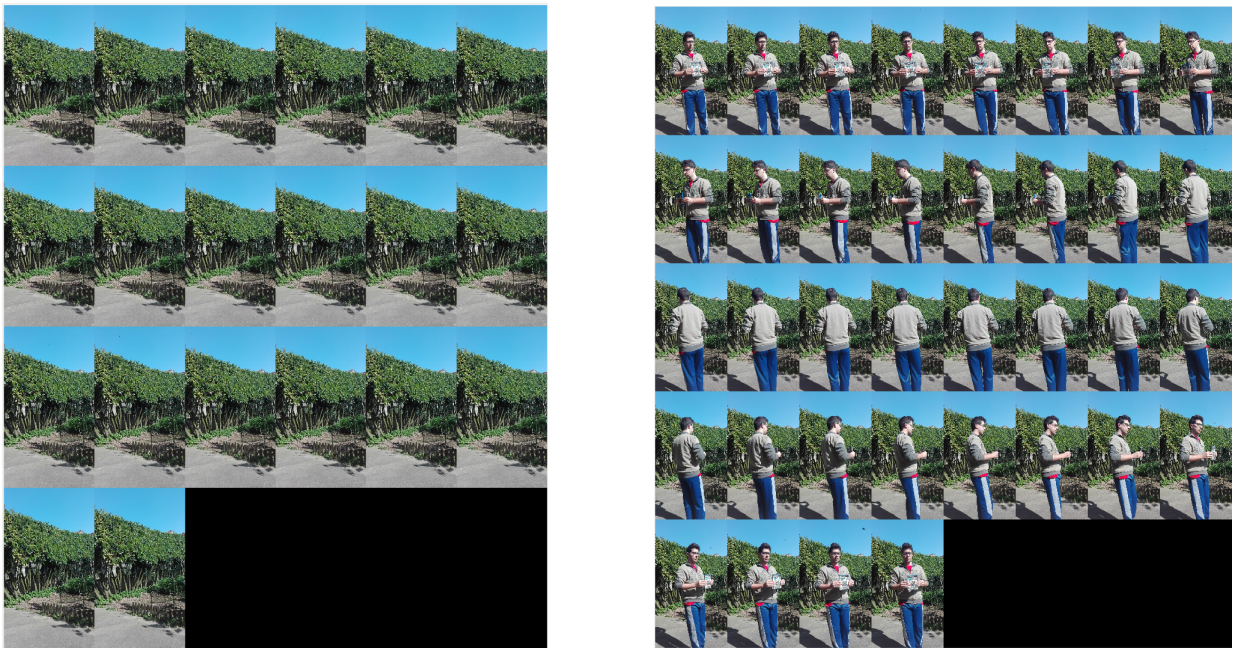

Figure 19: Background images and target images

5.3.2 Silhouettes extraction

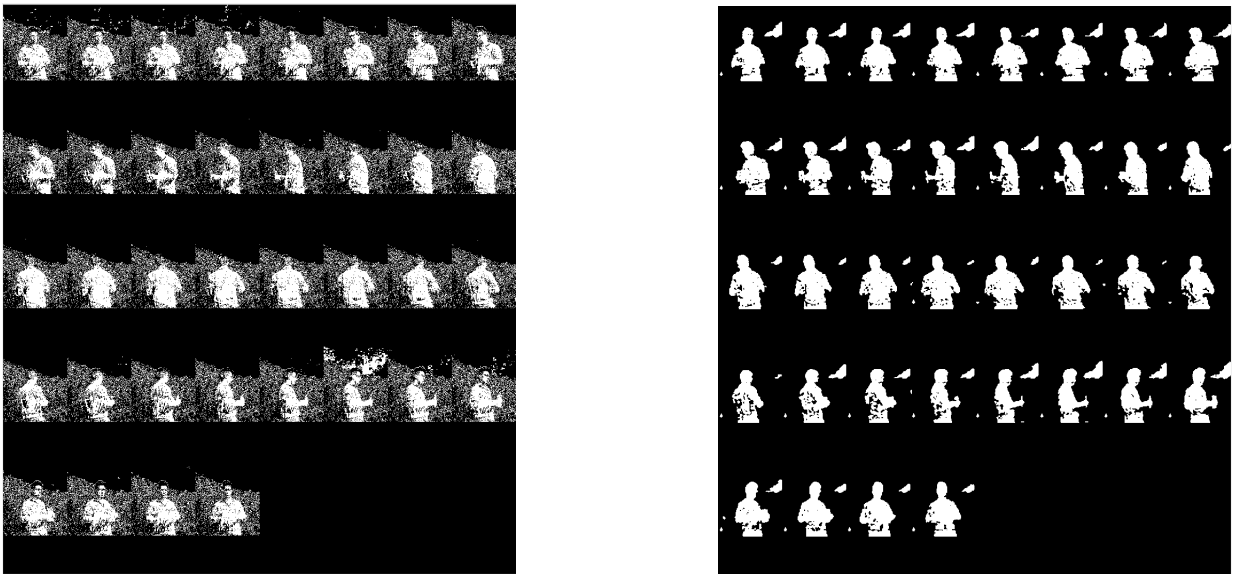

Figure 20: Silhouettes extracted using RGB (left) and HSV (right) thresholding

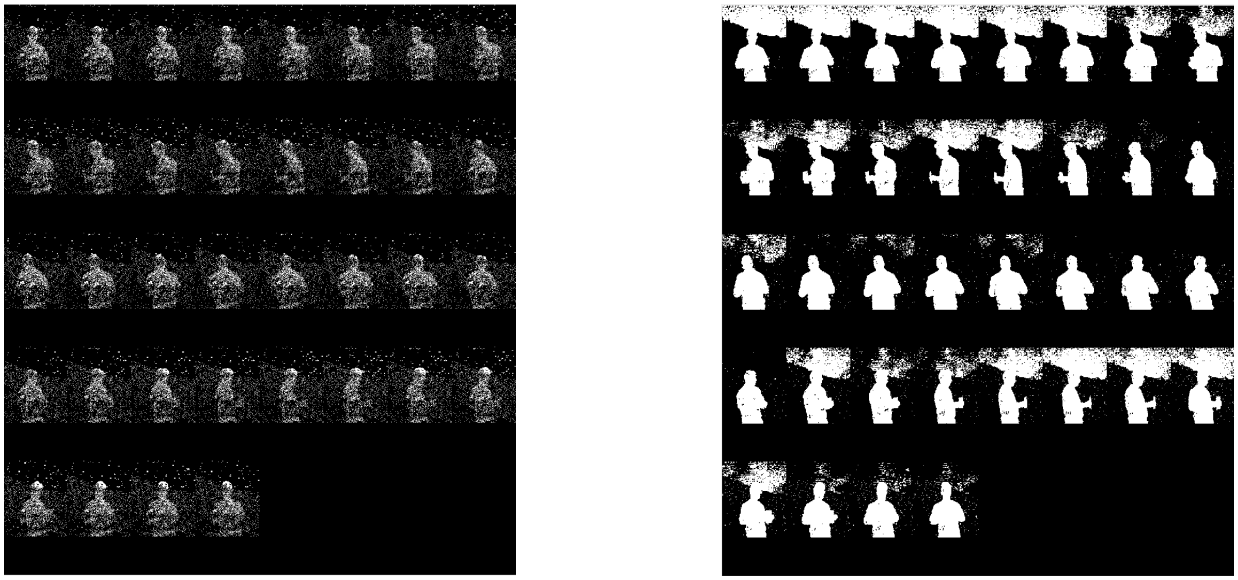

Figure 21: Silhouettes extracted using statistical modeling (left) and efficient statistical modeling (right)

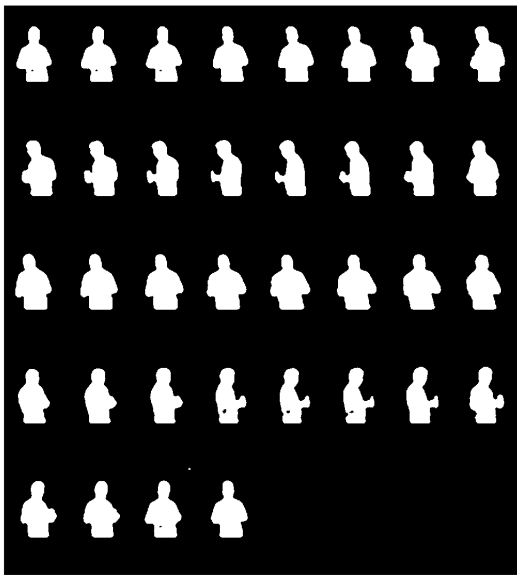

Figure 22: Silhouettes extracted using adaptive background mixture modeling

	RGB+GA	**HSV**	**Statistical modeling**	**Efficient statistical modeling**	**Adaptive background mixture model**
Performance	86,87%	95,91%	85,77%	89,38%	100%

The performance is evaluated considering the Adaptive Background Mixture Model silhouettes as ground truth images, since the method provides the best results. However, it is clear that it overestimates the silhouettes (resulting also in an overestimation of the volumetric reconstruction): this is due to the face that different morphological operators are applied to the Gaussian mixture model outputs, to remove noise and fill holes.

5.3.3 Pose estimates

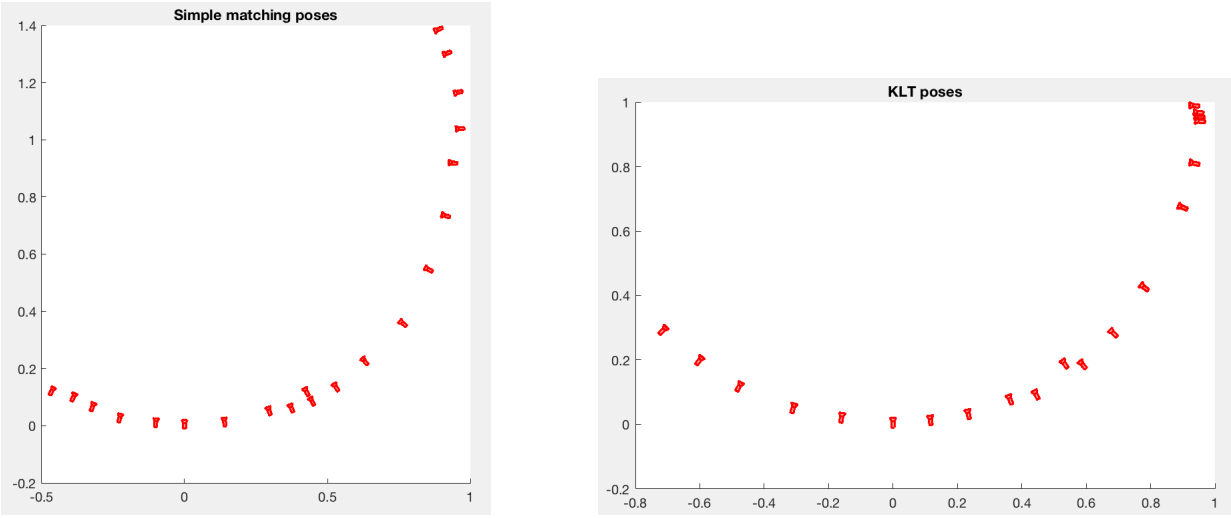

Figure 23: Camera poses estimates from above

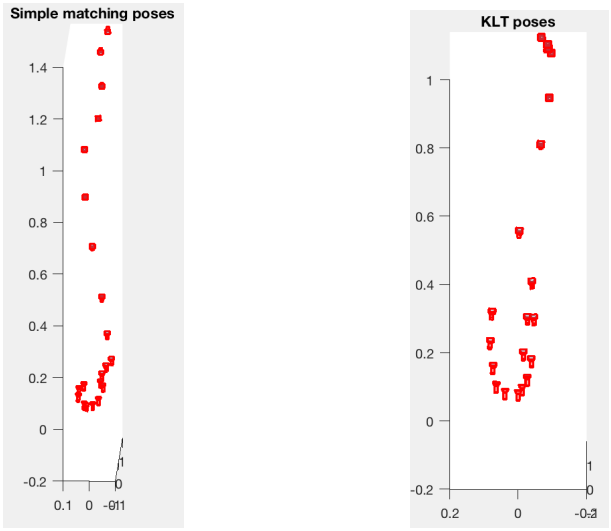

Figure 24: Camera poses estimates from the sides

Interrupted KLT provides more accurate results in a short range of motion, depicting also the movement of the target while rotating (up and down when moving the feet and hips around). Simple matching covers a wider range of motion, but it does so incorrectly, making a wrong estimation of the poses. In both cases they provide an advantage and a disadvantage in the volumetric reconstruction, as it will be seen in the next subsection.

5.3.4 Volumetric reconstruction

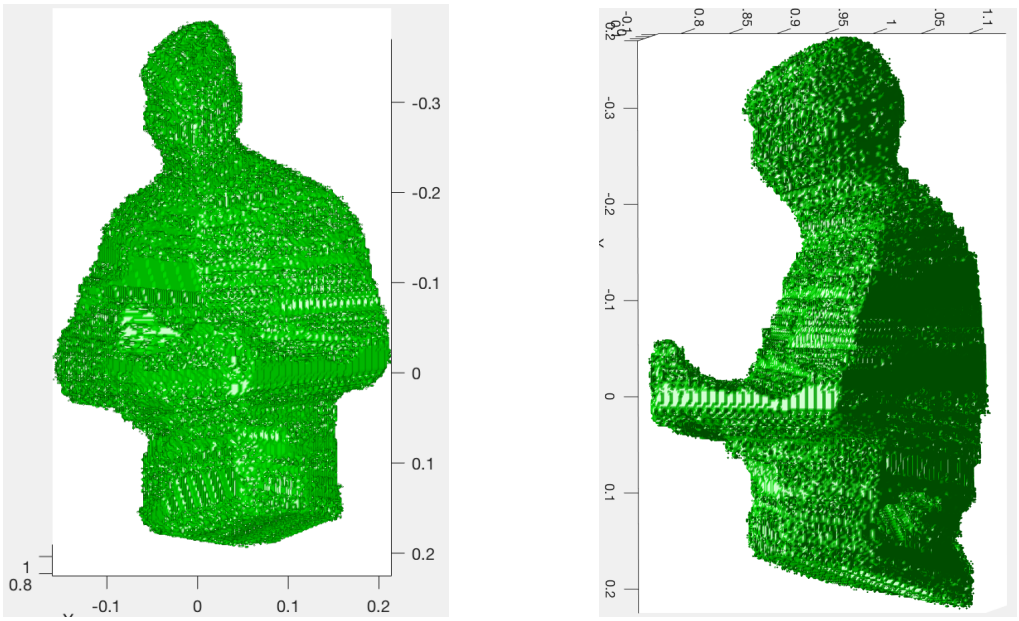

Figure 25: Reconstruction after voxel carving from poses estimated with interrupted KLT

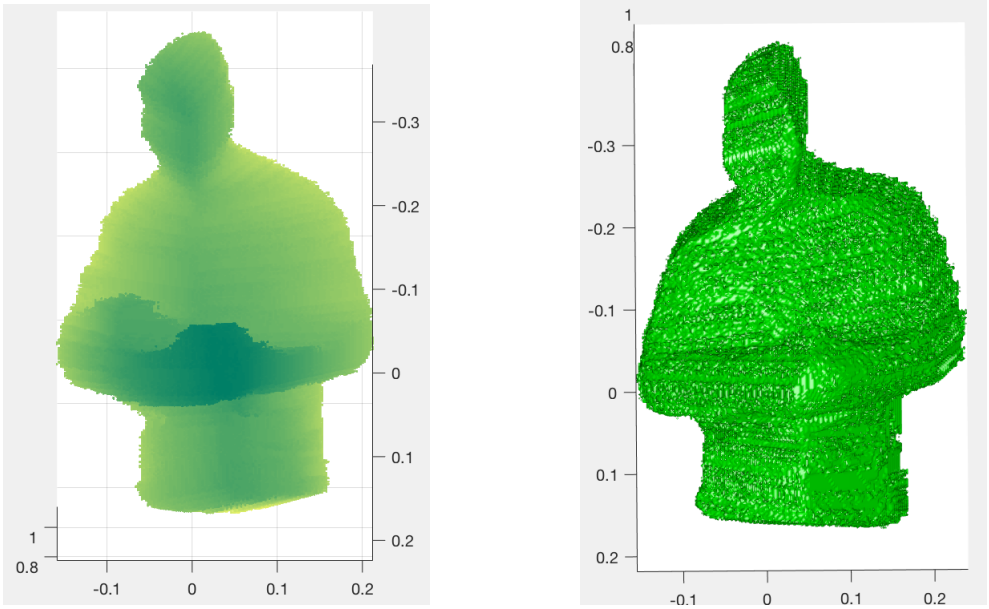

Figure 26: Reconstruction after voxel carving from poses estimated with simple matching

The reconstruction by tracking is smooth and accurate, especially on semi-sharp edges, but it is imperfect since it derives from a small range of poses, not fully carving the space (noticeable around arms or covered parts). The reconstruction by simple matching gives a complete rough shape of the model, but it brings to light the inaccuracy of the pose estimates.
Once again, this demo proves us that tracking may leads to less carving but with more accuracy, while simple matching has wider range of motion but with poor precision.

5.4 Demo 4

SUBJECT: project author, Matteo Frosi
CAMERA FEATURES: Huawei P8 Lite frontal camera, 1088x1980 px resolution
DISTANCE AND CAMERA POSITION: the target is about 80 cm distant from the camera, located inclined w.r.t. the ground.

5.4.1 Inputs

Figure 27: Background images and target images

5.4.2 Silhouettes extraction

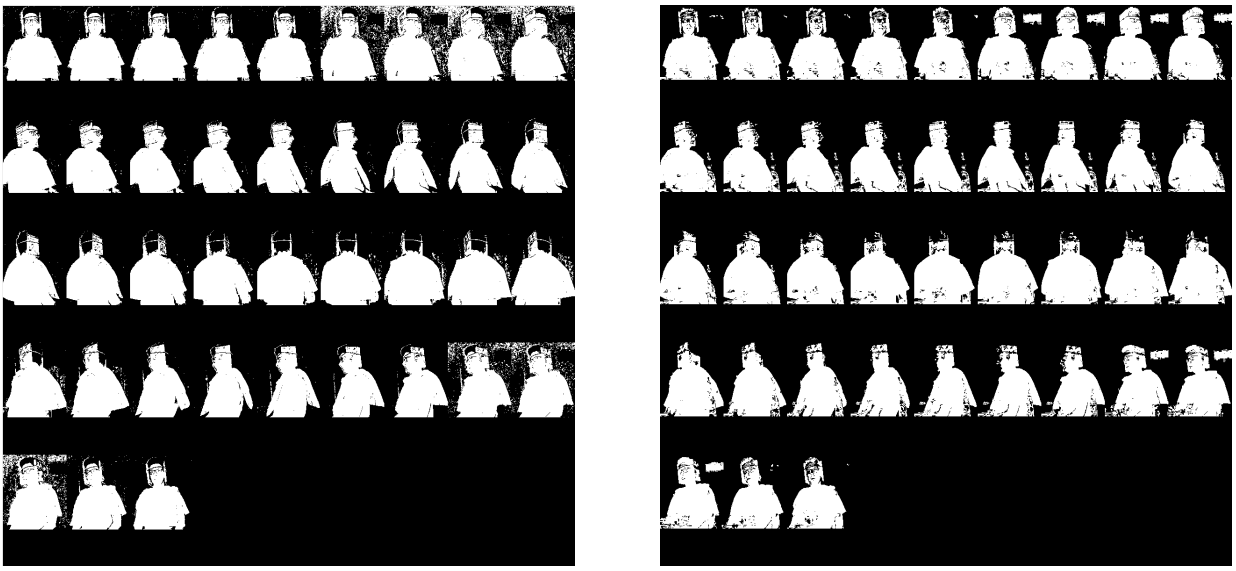

Figure 28: Silhouettes extracted using RGB (left) and HSV (right) thresholding

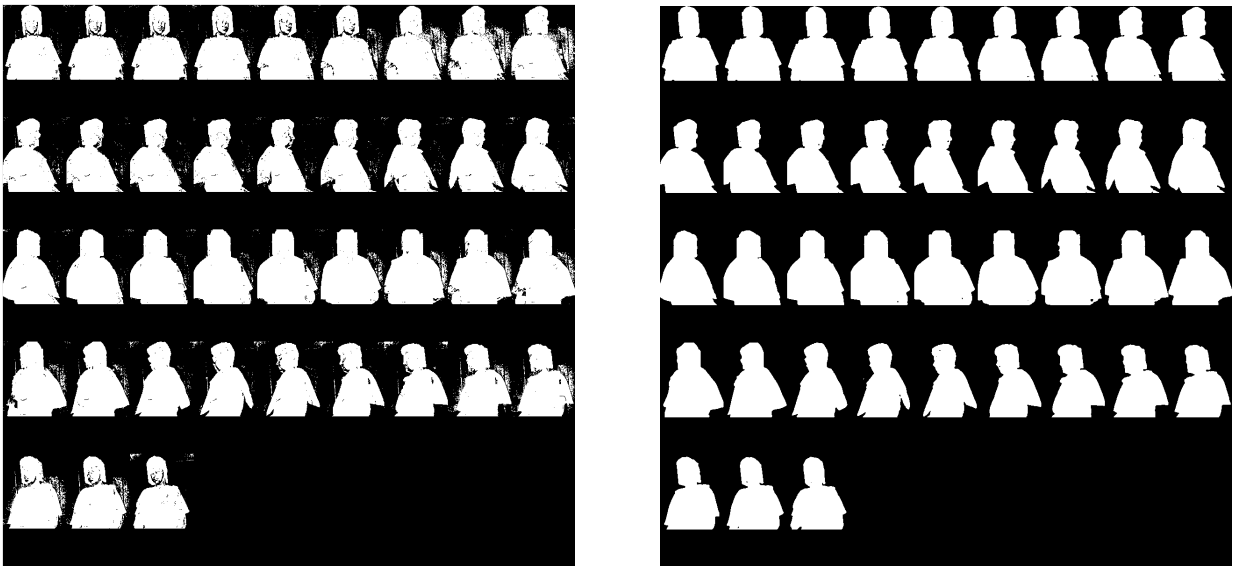

Figure 29: Silhouettes extracted using statistical modeling (left) and adaptive background mixture modeling (right)

	RGB+GA	HSV	Statistical modeling	Efficient statistical modeling	Adaptive background mixture model
Performance	93,97%	94,51%	0*%	97,38%	100%

The performance is evaluated considering the Adaptive Background Mixture Model silhouettes as ground truth images, since the method provides the best results. In this demo, statistical modeling is not considered because this is one of the cases where we need to tune the algorithm to deal with small chromatic variations, and it takes too much time (since it can only be performed by random guesses over the minimum threshold).

5.4.3 Pose estimates

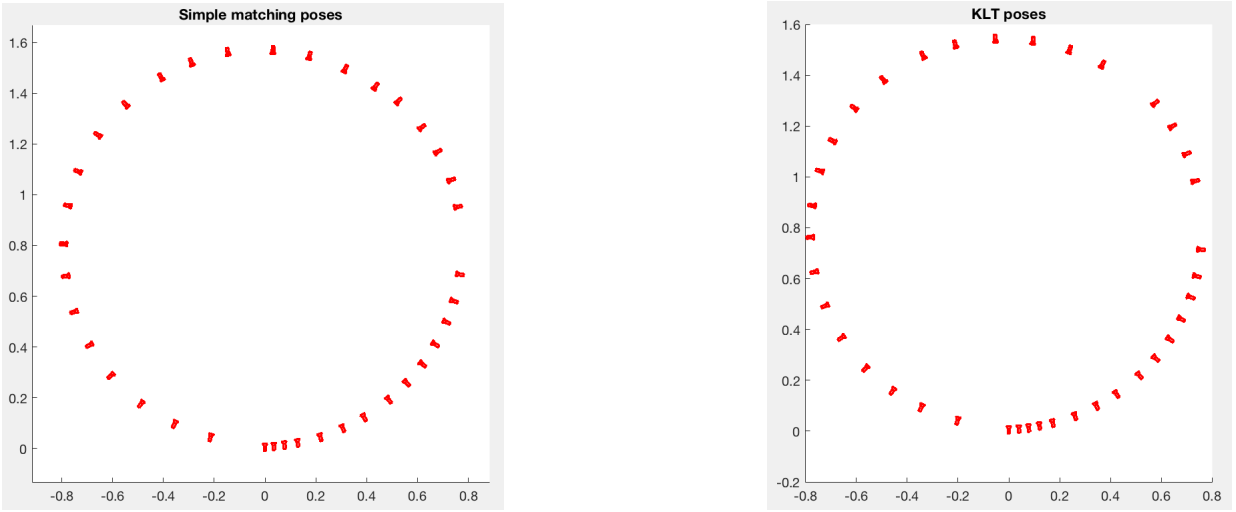

Figure 30: Camera poses estimates from above

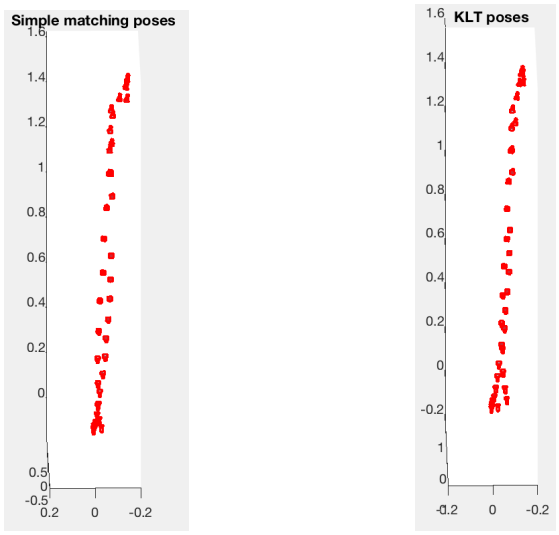

Figure 31: Camera poses estimates from the sides

Since in this demo a great number of features is available, both estimation algorithms provide the same results. The up and down behavior of the poses, related to the small movements of the body when rotating, includes also the small orientation of the camera. It is noticeable however, that Interrupted KLT captures better this behavior, making the motion between scenes appear smoother.

5.4.4 Volumetric reconstruction

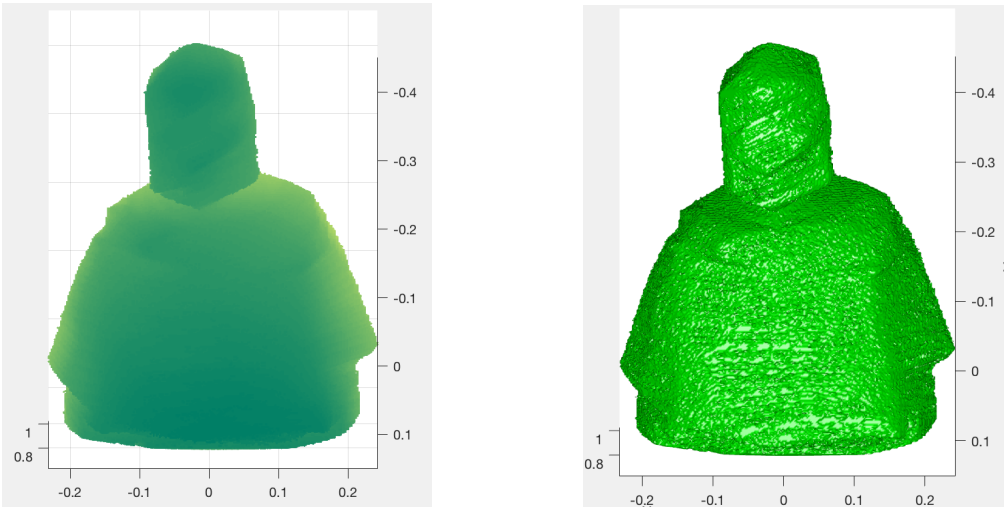

Figure 32: Reconstruction after voxel carving from poses estimated with interrupted KLT (front view)

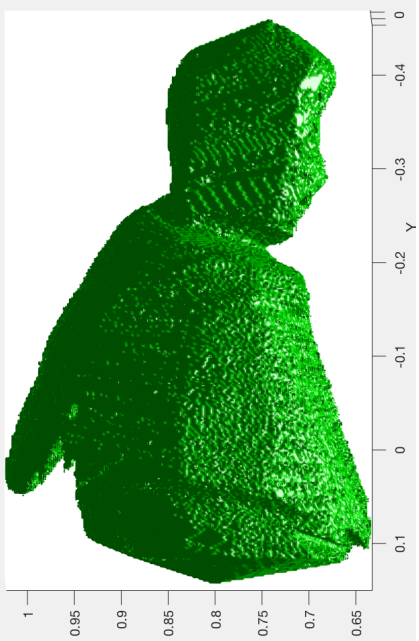

Figure 33: Reconstruction after voxel carving from poses estimated with interrupted KLT (side view)

The funny setup (pillows and videogame casings) allows a detailed reconstruction, with a full range of camera poses. However it fails to represent hidden entities, such as the shape of the eyes or nose (covered by the casings). This problem is intrinsic to voxel carving, as described in the introduction. Notice also that pillows, shirt and flat surfaces are well visible in the reconstruction.

6 Code architecture

In this section we briefly describe the architecture of the project code, relying on the figure below.

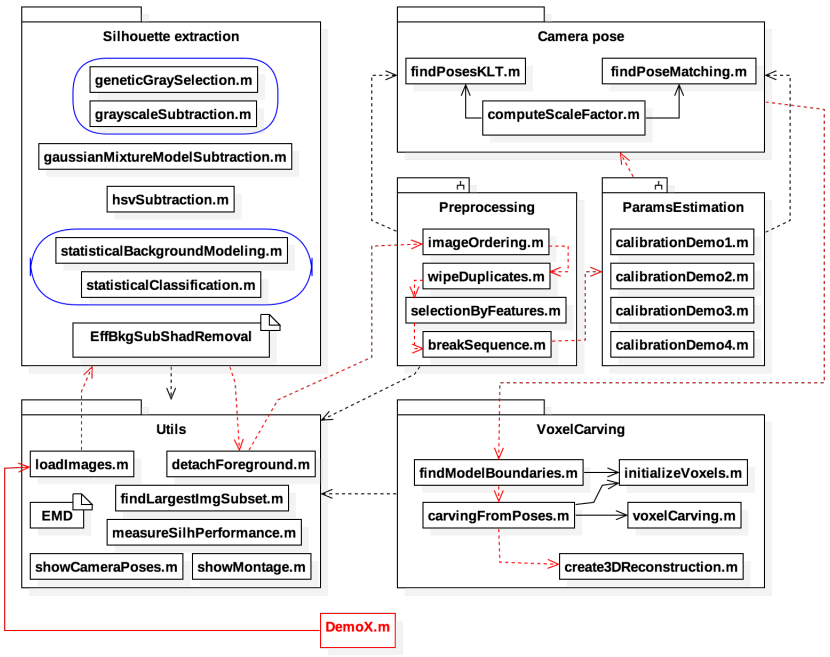

Figure 34: High view code architecture

The project is composed by small modules, each related to a phase previously described.

- The SilhouetteExtraction module contains all the sub-modules (files or directories) that perform, as implied by the name, extraction of the foregrounds for each image. When a very small number of sub-modules work together they are considered a group (surrounded by the blue ellipses in figure), but the sub-modules are usually stand alone files (with the exception of the efficient background subtraction and shadow removal sub-module).
- The CameraPose module contains two major sub-module directories and 3 files. The directories perform camera calibration, depending on the current demo, and sequence preprocessing, described in section 3. The files are instead related to the estimation algorithms: Interrupted KLT and simple matching.
- The VoxelCarving module contains all the files used for both the carving and visualization of the reconstructed model.
- The Utils module contains all the files that help the other modules to do their job. They perform simple tasks such as loading a set of images, showing a sequence, showing the camera poses and so on.

In red we pointed out the flow of operations to be done for each demo. First, the background and target images are loaded. Then we perform silhouette extraction and select the best results, detaching the foreground from the input images. Follows the preprocessing previously described and the calibration of the camera that took the images. We do voxel carving, starting from finding the boundaries of the reconstruction and then performing a dense discretisation. Lastly, we visualize the 3D obtained model.

7 Conclusions and notes

7.1 Conclusions

For each phase of the project we can make some considerations. The best performances in foreground extraction are obtained both by HSV subtraction and Gaussian Mixture modeling, with the first slightly worse than the second, although faster and easier to implement. This doesn't mean that modeling the background as a mixture of Gaussian distributions is the silver bullet: there are some cases when it fails even to distinguish differences in the backgrounds. It may happen that the target occludes the light source and changes the background brightness and chromaticity. Since the method relies on training with the background images, the sudden change when the training ends (and the target comes in the scene), makes it fail when classifying the foreground.
Regarding pose estimation, it is not feasible to pretend that a moving target maintains its body rigid. This can be done only by expert people or by a target placed on a rotating disk. Since in the project we dealt with general cases, we do not take for granted any ability or positioning of the target. This leads to incorrect volumetric reconstructions. Moreover, the amount and quality of detected features also strongly influences the reconstruction, because they lead to poor results in estimating the poses, inducing less accuracy and covered range of motion. Once again, we consider general situations, with normal dresses (we didn't want to cheat and wear particular shirts or pants full of different features).
Interrupted KLT exploits many features, but its computational cost increases with the number of fed images. It also performs as good as simple matching when lots of features can be extracted from the image sequence, making the last preferable due to its speed. A solution to the cost problem could be introducing a threshold over the number of current features used at each iteration of the algorithm, to avoid "costly explosions".
Lastly we consider voxel carving: it is very easy to implement and work with, and the algorithm shows high flexibility with respect to the problem needs. It has, however, faults: it relies on the quality of camera poses and silhouettes and to represent hidden parts it requires scenes taken from different angles all around the space surrounding the target (above, sides, below, ...), not just from the front, back and sides as done in this project. Voxel carving is a double edged sword: from one side the idea behind it is simple and efficient, but it absolutely requires a static target (making not usable in real-time videos, where there is plenty of movements).

7.2 Ideas and proposals

Instead of performing a whole volumetric reconstruction with voxel carving, we can first break each image of the sequence into small fragments [24], namely the standard parts of the human body. Then we perform voxel carving for each part, paying attention to ambiguous situations (e.g. body side image with upper arm overlapping the torso). This procedure is good in theory, but brings great disadvantages:

1. It requires a CNN (https://github.com/shiba24/deep-learning-for-human-part-discovery-in-images) to classify each image of the sequence, greatly increasing the computational time and resources needed.

2. Since it has sense only for the problem described in this project (static camera - moving target), breaking the foregrounds in small fragments makes even smaller the number of features available for each subset of the sequence, making extremely difficult, if not impossible, to estimate correctly the poses of each chunk of body.

Because the second problem appears particularly hard to overcome, we also propose reconstruction methods beyond voxel carving. Instead of forcing volumetric reconstruction over a dense discretisation of the 3D space, we can rely on inferential methods [25] and model based reconstruction techniques. The main advantage w.r.t. voxel carving is that we already start with a prior knowledge of the target shape, that is refined continuously

through the sequence of input images. Moreover we have no constraints over the target movement, that can assume every possible position [26, 27, 28].

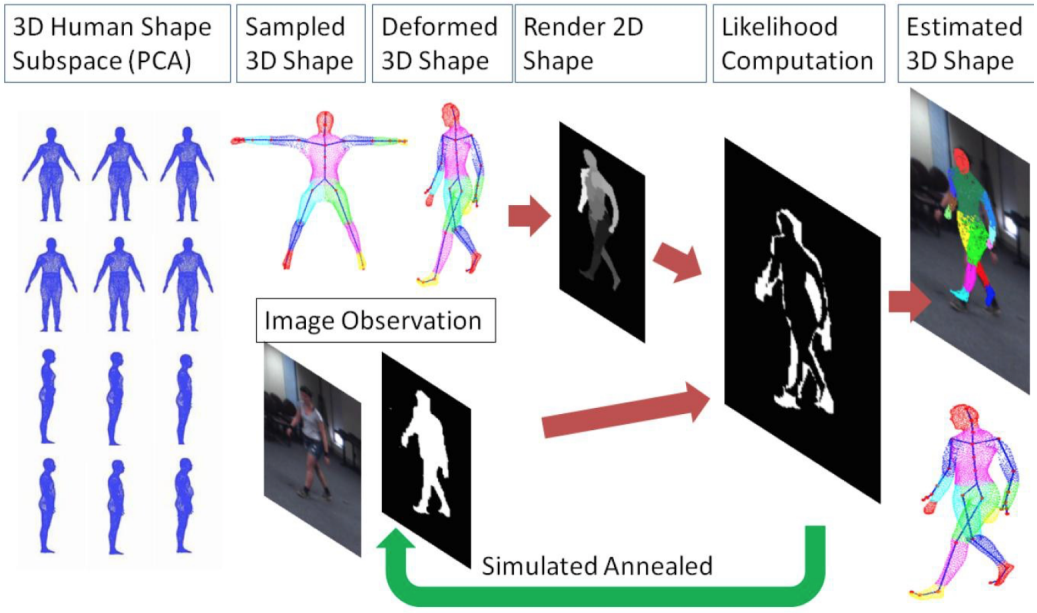

Figure 35: Work-flow of model based Visual Hull reconstruction

8 Bibliography

References

[1] Onur Ozyesil, Vladislav Voroninski, Ronen Basri, Amit Singer, "*A Survey of Structure from Motion*". 1

[2] Takafumi Taketomi, Hideaki Uchiyama, Sei Ikeda, "*Visual SLAM algorithms: a survey from 2010 to 2016*". 1

[3] Aldo Laurentini, "*The Visual Hull Concept for Silhouette-Based Image Understanding*", IEEE transactions on pattern analysis and machine intelligence, vol. 16, no. 2, February 1994, pp. 150-162 1

[4] David Schneider, "*Visual Hull*". 1

[5] K. Kutulakos, S. M. Seitz, "*A theory of shape by space carving*", Int. Journal of Computer Vision, 38(3), July 2000, pp. 198-218. 1

[6] K. F. Man, K. S. Tang, and S. Kwong, "*Genetic Algorithms: Concepts and Applications*", IEEE Transactions on Industrial Electronics, vol. 43, no. 5, October 1996, pp. 519-534. 2.1

[7] Khamar Basha Shaik, Ganesan P.,V. Kalist, B.S.Sathish, J.Merlin Mary Jenith, "*Comparative Study of Skin Color Detection and Segmentation in HSV and YCbCr Color Space*", Procedia Computer Science, vol. 57, 2015, pp. 41-48. 2.2

[8] Thanarat Horprasert, David Harwood, Larry S. Davis, "*A Statistical Approach for Real time Robust Background Subtraction and Shadow Detection*". 2.3

[9] Cláudio Rosito Jung, "*Efficient Background Subtraction and Shadow Removal for Monochromatic Video Sequences*", IEEE Transactions on Multimedia, vol. 11, no. 3, April 2009, pp. 571-577. 2.4

[10] Kaewtrakulpong, P. and R. Bowden, "*An Improved Adaptive Background Mixture Model for Realtime Tracking with Shadow Detection*", In Proc. 2nd European Workshop on Advanced Video Based Surveillance Systems, AVBS01, VIDEO BASED SURVEILLANCE SYSTEMS: Computer Vision and Distributed Processing, September 2001. 2.5

[11] Stauffer, C. and W.E.L. Grimson, "Adaptive Background Mixture Models for Real-Time Tracking, Computer Vision and Pattern Recognition", IEEE Computer Society Conference, vol. 2, August 1999, pp. 246-252. 2.5

[12] Richard Hartley, Andrew Zisserman, "*Multiple View Geometry in computer vision 2nd ed.*", pp. 239-259. 3.2

[13] Zuzana Kukelova, Martin Bujnak, Tomas Pajdla, "*Polynomial Eigenvalue Solutions to the 5-pt and 6-pt Relative Pose Problems*", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, Issue 6, June 2004. 3.2

[14] Lucas, Bruce D., Takeo Kanade, "*An Iterative Image Registration Technique with an Application to Stereo Vision*", Proceedings of the 7th International Joint Conference on Artificial Intelligence, April 1981, pp. 674–679. 3

[15] Tomasi, Carlo, Takeo Kanade, "*Detection and Tracking of Point Features, Computer Science Department*", Carnegie Mellon University, April 1991. 3

[16] Andreas Ess, Tinne Tuytelaars, Luc Van Gool, "*Speeded-Up Robust Features (SURF) Herbert Bay*". 3

[17] Farman Ali, Sajid Ullah Khan, Muhammad Zarrar Mahmudi, Rahmat Ullah, "*A Comparison of FAST, SURF, Eigen, Harris, and MSER Features*", International Journal of Computer Engineering and Information Technology, vol. 8, no. 6, June 2016. 3

[18] Muja, M., and D. G. Lowe, "*Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration*", International Conference on Computer Vision Theory and Applications,VISAPP, 2009. 2

[19] Lowe, David G., "*Distinctive Image Features from Scale-Invariant Keypoints*", International Journal of Computer Vision, vol. 60, no. 2, pp. 91–110. 2

[20] B. Triggs, P. McLauchlan, R. Hartley, A. Fitzgibbon, "*Bundle Adjustment: A Modern Synthesis*", Proceedings of the International Workshop on Vision Algorithms, Springer-Verlag, 1999, pp. 298-372. 7, 7

[21] Jean-Yves Bouguet, "Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm". 3.4

[22] Philbin, J., O. Chum, M. Isard, J. Sivic, A. Zisserman, "*Object retrieval with large vocabularies and fast spatial matching*", CVPR 2007. 3.5.1

[23] Yossi Rubner, Carlo Tomasi, Leonidas J. Guibas, "*The Earth Mover's Distance as a Metric for Image Retrieval*". 3.5.2

[24] Gabriel L. Oliveira, Abhinav Valada, Claas Bollen, Wolfram Burgard, Thomas Brox, "*Deep Learning for Human Part Discovery in Images*". 7.2

[25] Kristen Grauman, Gregory Shakhnarovich, Trevor Darrell, "*Virtual Visual Hulls: Example-Based 3D Shape Inference from Silhouettes*", In Proceedings of the 2nd Workshop on Statistical Methods in Video Processing, in conjunction with ECCV, Prague, Czech Republic, May 2004. 7.2

[26] Atul Kanaujia, Nicholas Kittens, Narayanan Ramanathan, "*Part Segmentation of Visual Hull for 3D Human Pose Estimation*", CVPR Workshop 2013. 7.2

[27] Atul Kanaujia, Niels Haering, "*3D Human Pose and Shape Estimation from Multi-view Imagery*", CVPR Workshop 2011. 7.2

[28] Stefano Corazza, Lars Mundermann, Emiliano Gambaretto, Giancarlo Ferrigno, Thomas P. Andriacchi, "*Markerless Motion Capture through Visual Hull, Articulated ICP and Subject Specific Model Generation*", Int J Comput Vis (2010) 87, pp. 156–169.

7.2