NAIST-IS-MT1251124

Master's Thesis

Free-viewpoint Image Generation-based Human Motion Reenactment from a Single RGB-D Video Stream

Fabian Lorenzo Dayrit

March 13, 2014

Department of Information Science
Graduate School of Information Science
Nara Institute of Science and Technology

A Master's Thesis
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
MASTER of ENGINEERING

Fabian Lorenzo Dayrit

Thesis Committee:

Professor Naokazu Yokoya	(Supervisor)
Professor Hirokazu Kato	(Co-supervisor)
Associate Professor Tomokazu Sato	(Co-supervisor)
Assistant Professor Yuta Nakashima	(Co-supervisor)

Free-viewpoint Image Generation-based Human Motion Reenactment from a Single RGB-D Video Stream*

Fabian Lorenzo Dayrit

Abstract

Videos are invaluable for trainers demonstrating complex actions. They enable learners to study the motion of these actions at any time. However, conventional video is only 2-dimensional. Generally, viewers are able to compensate for the loss of the third dimension by making good inferences about the motion of the action. However, if some ambiguity is present in the action, the motion may become harder to comprehend.

Free-viewpoint image generation techniques attempt to solve this problem by allowing viewers to watch the action from any viewpoint they choose. However, most existing free-viewpoint image generation systems also require the action to be simultaneously captured from multiple viewpoints. Such systems require large camera arrays and are difficult to set up.

This thesis proposes a system to generate free-viewpoint images of human motion using a single RGB-D sensor by synthesizing viewpoints other than the one that was originally captured. Our system is composed of two stages: capturing the motion of a human subject's action, and displaying a reenactment of the action, which is a free-viewpoint rendering of the action using augmented reality. To generate an image from a novel viewpoint, we find the most appropriate texture among our captured frames based on the subject's pose and viewpoint similarity and apply this texture to a rough three-dimensional model.

* Master's Thesis, Department of Information Science, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT1251124, March 13, 2014.

We have implemented a prototype which runs on a mobile computer. We have conducted a user study to determine if i) the system increases comprehension of 3-dimensional motion, ii) the system allows users to compare the reenactment's motion to a real human's motion, and iii) the system has other real-world applications.

Keywords:

Augmented reality, free-viewpoint image generation, human motion capture/reenactment

単一 RGB-D ストリームに基づく 自由視点画像生成による人物動作の再現[*]

Fabian Lorenzo Dayrit

内容梗概

舞踊などのように特定の動作を学習する場合, 指導者の動作見本の映像記録は, 学習者がいつでもその動きを学習することが可能となることから, 極めて有用である. しかし, 通常の映像は三次元の動きを二次元に投影して記録するため, 学習者が映像のみから複雑な三次元の動きを推測することが困難な場合がある.

この問題に対して, 自由視点画像生成技術を利用することで, 学習者が任意の視点からその動きを観測することを可能にするシステムが提案されている. これらのシステムは, 通常の映像では失われていた奥行方向の動きについても, 視点を変更することにより観測可能となる. しかし, このようなシステムは多くの場合, 対象となる人物のすべての方向からの見えを記録するために複数の視点から撮影された映像や対応する奥行マップが必要となることから, 一般の指導者や学習者が手軽に利用することはできない.

そこで本研究では, 単一の RGB-D センサーのみを利用して動きを撮影することにより, 任意の視点からその動きを観測可能にするシステムを提案する. 単一視点から撮影された映像には, 例えば対象の人物の背面からの視点の画像を生成する場合に, テクスチャが存在しないという問題が生じる. 提案システムでは, 撮影されたすべてのフレームから最も適切なフレームをテクスチャとして利用することでこの問題を解決する.

実験では, 事前に撮影された人物動作に基づいて, モバイルデバイス上でリアルタイムに任意の視点の画像を生成するプロトタイプを作成し, ユーザスタディより提案システムの有用性を検証する.

[*] 奈良先端科学技術大学院大学 情報科学研究科 情報科学専攻 修士論文, NAIST-IS-MT1251124, 2014 年 3 月 13 日.

キーワード

拡張現実, 自由視点画像生成, 人物動作の取得・再現

Contents

1. Introduction **1**

2. Related Literature **4**

2.1 Learning motion with AR 4

2.2 Free-viewpoint image generation 6

2.2.1 Using RGB frames 6

2.2.2 Using RGB-D sensors 7

2.2.3 Using human pose estimation 8

2.3 Summary and relation to our work 9

3. The AR Reenactment System **12**

3.1 Overview of AR reenactment system 12

3.2 Capturing stage 15

3.2.1 RGB-D sensor pose estimation 15

3.2.2 Motion capture 15

3.2.3 Rough 3D model building 16

3.3 Reenactment stage 18

3.3.1 Viewer camera pose estimation 18

3.3.2 Appropriate texture selection 18

3.3.3 Applying the 3D model 20

3.3.4 Applying texture 20

4. Implementation **25**

5. Experimental Results **28**

5.1 Reenactment results 28

5.2 User study 31

5.2.1 Setup 31

5.2.2 Results 36

6. Conclusion **38**

Acknowledgements **40**

References **41**

List of Figures

1 Free-viewpoint image generation for human action [1]. A person is rendered from arbitrary viewpoints for the purpose of generating character animation. 1

2 Examples of capturing setups for free-viewpoint image generation systems for moving scenes. Most capture from multiple cameras, such as [2] (top row) and [3] (bottom row). 2

3 Examples of an RGB frame captured by a user (left) and a reenactment by the proposed system (right). 3

4 The post-stroke rehabilitation system proposed by Hondori et al [4]. Users are directed by the system to perform specific hand motions. In the first column, users reach for boxes to play sounds; in the second column, users pour virtual water on the blue spot; in the third column, users grasp randomly positioned circles. 4

5 Dancing with a robot [5]. The robot carries a screen that simulates the appearance of a missing dancer. 5

6 The AR instruction system proposed by Henderson and Feiner [6]. The AR arrows instruct the user on how to move in 3D space. . . 6

7 AR "magic mirror" [7]. The ideal motions are overlayed on top of a mirror image of the user. 7

8 Synthesizing a new viewpoint of a dance [2]. Left and right frames are interpolated to produce the image in the center. 8

9 Detailed model and motion of a subject captured by De Aguiar et al. [8] 9

10 Human pose estimation and free-viewpoint image generation system by Carranza et al. [9] First column: general 3D model. Second column: two frames of input. Third and fourth column: 3D model fit to the input. 10

11 Capturing two users with three handheld Kinects simultaneously [10]. 11

12 Overview of the proposed system. 13

13 Coordinate systems and the transformations relating them. 14

14 (a) The skeleton representation of the subject. Circles are joints, and segments are body parts. (b) Corresponding depth frame with definitions of some angles. (c) Rectangles fitted to each body part. 16
15 Vector transformations. 19
16 First column: cylinder model. Second column: Texturing using only the head body part's projection. Third column: Texturing with individual body part projection. 21
17 Left: cylinder model in the viewer's camera. Right: RGB frame for coloring with overlaid cylinders. $\mathbf{\Omega}$ is the transformation that relates them. 22
18 Rotating a cylinder so that it faces front. 23
19 Our representation of the skeleton. Circles: joints, rounded rectangles: body parts. 26
20 Left: frame from an RGB video. Right: equivalent reenactment with double-limb error. 29
21 Comparison of the same frame of a reenactment in our system from different angles. 29
22 Left column: sample frames from an RGB video. Right column: equivalent reenactments. 30
23 Left: misregistered pose applied to the cylinder model. Right: texture applied to the misregistered pose. 31
24 Left column: sample frames from the RGB video that we used in the user study. Right column: equivalent reenactments. 32
25 Left column: sample frames from the RGB video that we used in the user study. Right column: equivalent reenactments. 33
26 Left column: sample frames from the RGB video that we used in the user study. Right column: equivalent reenactments. 34
27 Evaluation results for questions our user study. 1 means strong disagreement and 5 means strong agreement. 37

List of Tables

1 Questions asked in our user study (originally in Japanese). 35

1. Introduction

In recent years, watching and sharing videos over the Internet has become more popular, thanks to video sharing services such as YouTube. Anyone can use these services to share their own videos as well as explore what others have shared. Among the types of videos shared are instructional videos, which are videos which aim to educate the viewer. A subset of these are videos of people demonstrating some physical action: perhaps a martial arts kick, or a difficult dance move.

These videos are convenient because learners and instructors do not have to be in the same room together. Anyone can watch them at any time. However, they do have a drawback. A video holds a two-dimensional (2D) projection of the motion that was originally three-dimensional (3D). Normally, this is fine, as we humans can usually infer motion, even from a 2D video. If there is any ambiguity in the video, however, it may become difficult to comprehend the motion.

Perhaps augmented reality (AR) can offer a solution. AR is the integration of virtual elements with the real world [11]. AR applications augment a real environment in some way in order to provide users with a more entertaining or educational experience (e.g., [6, 7]).

Free-viewpoint image generation techniques also tackle this problem by rendering the video, or a part of the video such as its subject, from an arbitrary viewpoint (e.g., Fig. 1). However, in order to render moving scenes, most of these systems require simultaneous capture from multiple viewpoints (Fig. 2). This means that users must set up multiple cameras, which prevents the average user from being able to capture scenes with the system.

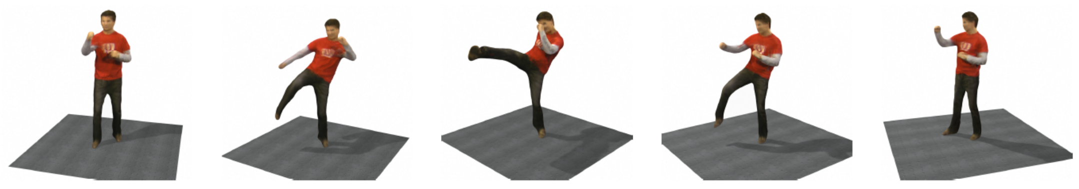


Figure 1: Free-viewpoint image generation for human action [1]. A person is rendered from arbitrary viewpoints for the purpose of generating character animation.

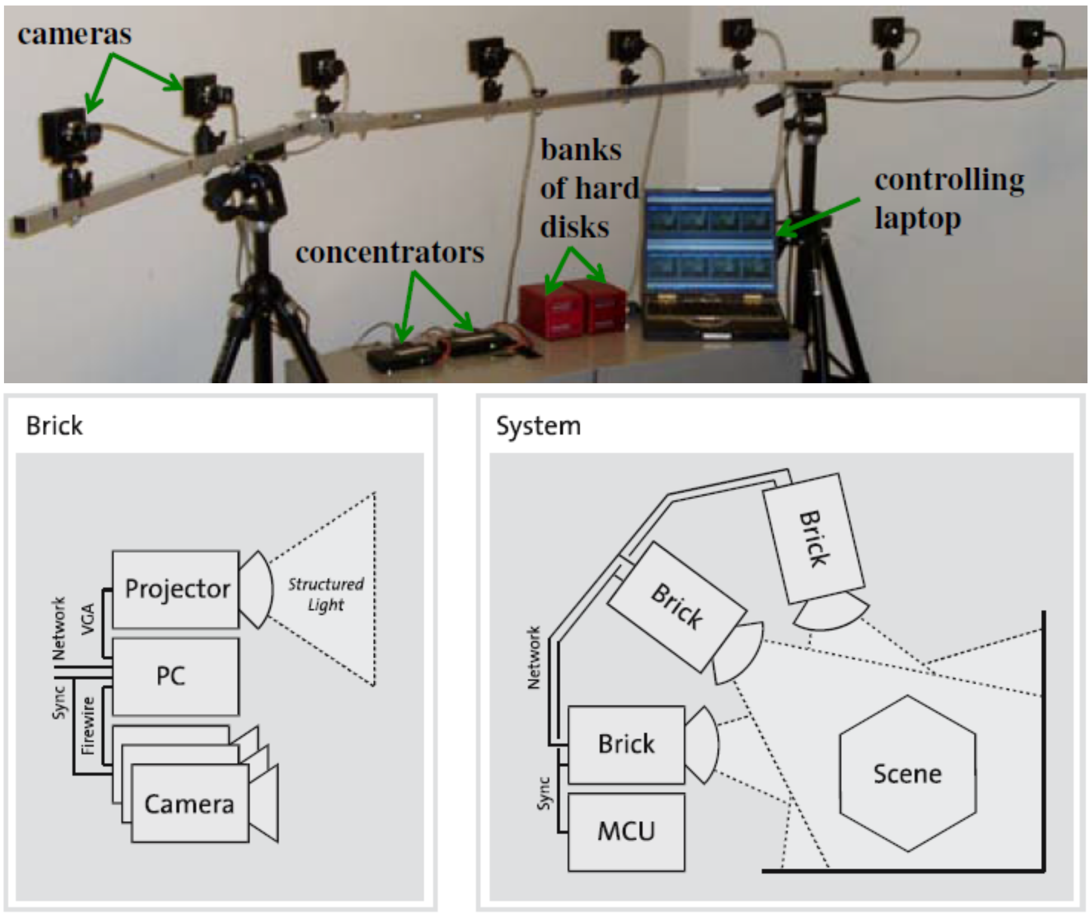


Figure 2: Examples of capturing setups for free-viewpoint image generation systems for moving scenes. Most capture from multiple cameras, such as [2] (top row) and [3] (bottom row).

This thesis will discuss a special case of free-viewpoint image generation, focusing solely on the human subject of the video. From now on, we will refer to such a synthesis of a moving human subject as a reenactment. As a special case of free-viewpoint image generation, a reenactment may also be viewed from an arbitrary viewpoint. If this viewpoint is not one of the original capturing viewpoints, it must synthesize the most likely appearance of the subject from the viewpoint. Finally, a reenactment must follow the recorded motions of the subject in the video sequence.

We propose a system that uses a single RGB-D sensor to capture and record

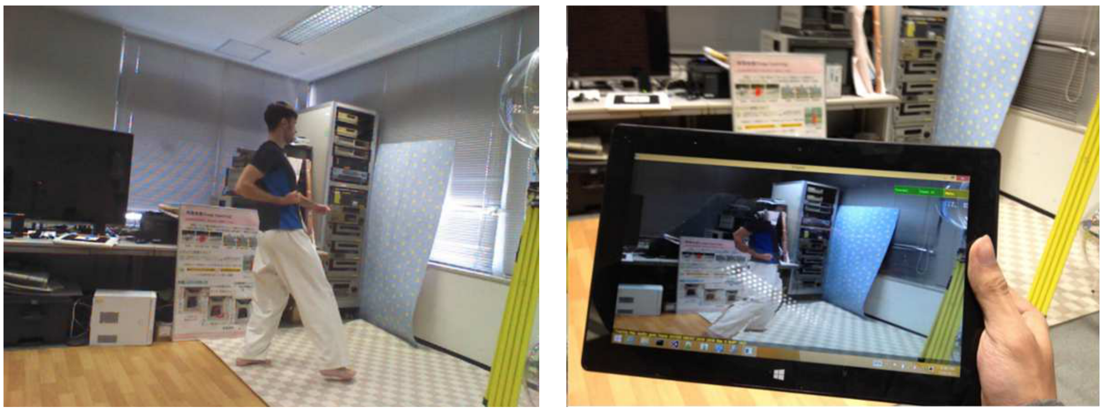


Figure 3: Examples of an RGB frame captured by a user (left) and a reenactment by the proposed system (right).

a moving subject, then generates an AR reenactment to be overlaid on top of an RGB stream (Fig. 3). In our system, a user captures and records the subject with an RGB-D sensor, and the data is stored in a database. Upon a viewer's request, the motion is reenacted based on the stored data. Finally, the reenactment is overlaid on top of the viewer's camera.

Requiring only a single RGB-D sensor makes our proposed system easy to use. The challenge is synthesizing a good reenactment from our single-sensor stream, because multiple viewpoints of the subject are necessary. Our observations in this regard are that (i) the subject's rough 3D model can be generated based on the depth frame and the pose of the subject in that frame, and that (ii) the subject may, over the course of the entire recording, expose different angles of himself or herself to the camera, which we may then use to color the rough 3D model.

This thesis is organized as follows. In section 2 we will discuss systems that are related to our proposed system: AR learning systems, and free-viewpoint image generation systems, with a focus on those that make use of RGB-D sensors and human pose estimation. In section 3 we will outline and describe each section of our proposed system in detail. We will cover capturing the motion of a subject and rendering the reenactment. In section 4 we will go into the specifics of how we implemented the system, including the hardware and software we used. In section 5 we will describe our user study and discuss its results. We conclude our thesis in section 6.

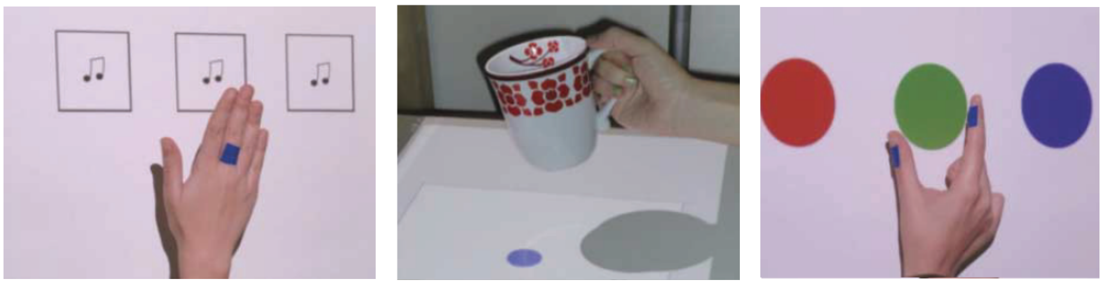


Figure 4: The post-stroke rehabilitation system proposed by Hondori et al [4]. Users are directed by the system to perform specific hand motions. In the first column, users reach for boxes to play sounds; in the second column, users pour virtual water on the blue spot; in the third column, users grasp randomly positioned circles.

2. Related Literature

This section discusses related research work. We discuss AR systems with a similar focus on human motion. We will then discuss the difference between our proposed system and existing free-viewpoint image generation systems. We also take a special look at free-viewpoint image generation systems that utilize RGB cameras, RGB-D sensors, and human pose estimation.

2.1 Learning motion with AR

AR is the combination of a scene from the real world with a virtual object. Our system aims to display a virtual subject with AR. There exist similar AR applications that were developed to provide viewers with an augmented view of motion in order to aid users' learning.

Hondori et al. [4] propose a system for users who have suffered a stroke. The rehabilitation process usually requires repetitive motion on the part of the user. The system makes the user perform motions that are commonly used in daily life, such as reaching, grasping, and pointing. The AR portion of the system generates virtual targets for the user to interact with (Fig. 4).

Tsuchida, Terada, and Tsukamoto [5] propose a learning support system specifically for dancers in a formation. In the place of a missing dancer, they

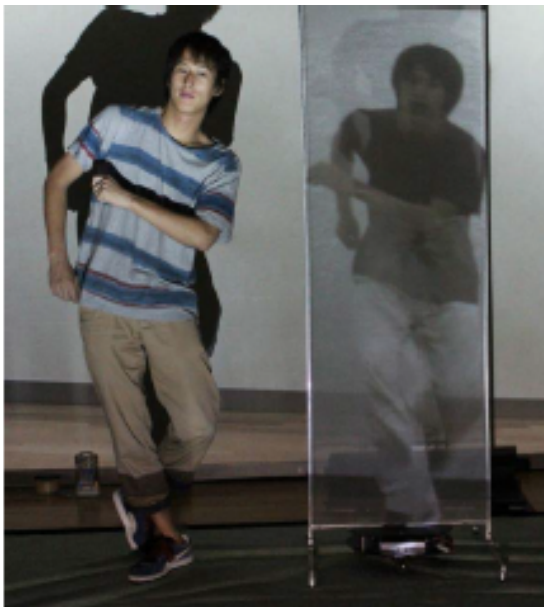


Figure 5: Dancing with a robot [5]. The robot carries a screen that simulates the appearance of a missing dancer.

used a self-propelled robot with a screen displaying the appearance of the dancer (Fig. 5). The robot moves in space according to how the dancer would have moved. Users who tried the system danced more accurately, i.e. closer to the actual trajectory, with the robot than without.

The system proposed by Henderson and Feiner [6] shows the user instructions on how to do a specific procedure, by way of arrows and labels in 3D space attached to key objects (Fig. 6). Users wore an optical see-through head-mounted display, and the instructions were overlayed on top of their view. Users who were surveyed preferred to use the proposed system over an instructional video with similar content displayed on an LCD monitor.

The YouMove system [7] first records and tracks the motion of the subject using an RGB-D sensor. Afterwards, viewers stand in front of a "magic mirror" (Fig. 7) and the system overlays the subject's motions on their reflection. In this way, viewers can more easily copy difficult motions. The system also provides a comparison between the subject and a viewer using 3D stick figures, which the viewer can rotate in order to view the motions from different directions. However,

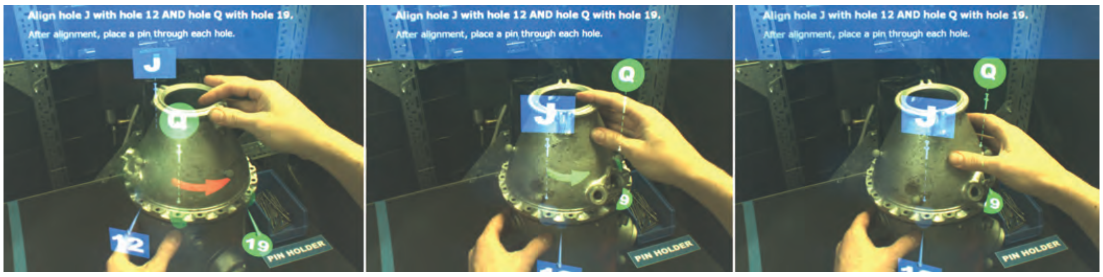

Align hole J with hole 12 AND hole Q with hole 19.
After alignment, place a pin through each hole.
J
Q
12
19
9
PIN HOLDER

Figure 6: The AR instruction system proposed by Henderson and Feiner [6]. The AR arrows instruct the user on how to move in 3D space.

as useful as they are, stick figures lack realism and naturalness.

2.2 Free-viewpoint image generation

Considering this, free-viewpoint image generation techniques can be used to provide complete renderings of video or parts of video. Free-viewpoint image generation systems use various methods in order to generate images of a scene from arbitrary viewpoints, including using RGB cameras, using RGB-D sensors, and using human pose estimation. This section introduces some of the research work that has been done on these techniques.

2.2.1 Using RGB frames

Several free-viewpoint image genration methods generate a 3D scene from multiple RGB frame captures of the same scene. Some of the first developed of these systems used the image-based visual hull (IBVH) [12]. IBVH-based techniques build a 3D model of an object by capturing it from multiple viewpoints and integrating the sillhouettes in each image. The original IBVH system was used for static objects. However, Würmlin et al. [13] use this technique in order to generate free-viewpoint image sequences of a moving subject. Since the subject was in motion, his sillhouette changed every frame. To resolve this problem, they made use of multiple cameras, all capturing the subject simultaneously.

Similar to these are the systems based on voxels and marching cubes [14, 15]: Matsuyama and Takai [16] and Starck, Miller, and Hilton [1]. These systems first generate, using multiple cameras, a voxel representation of the subject, and

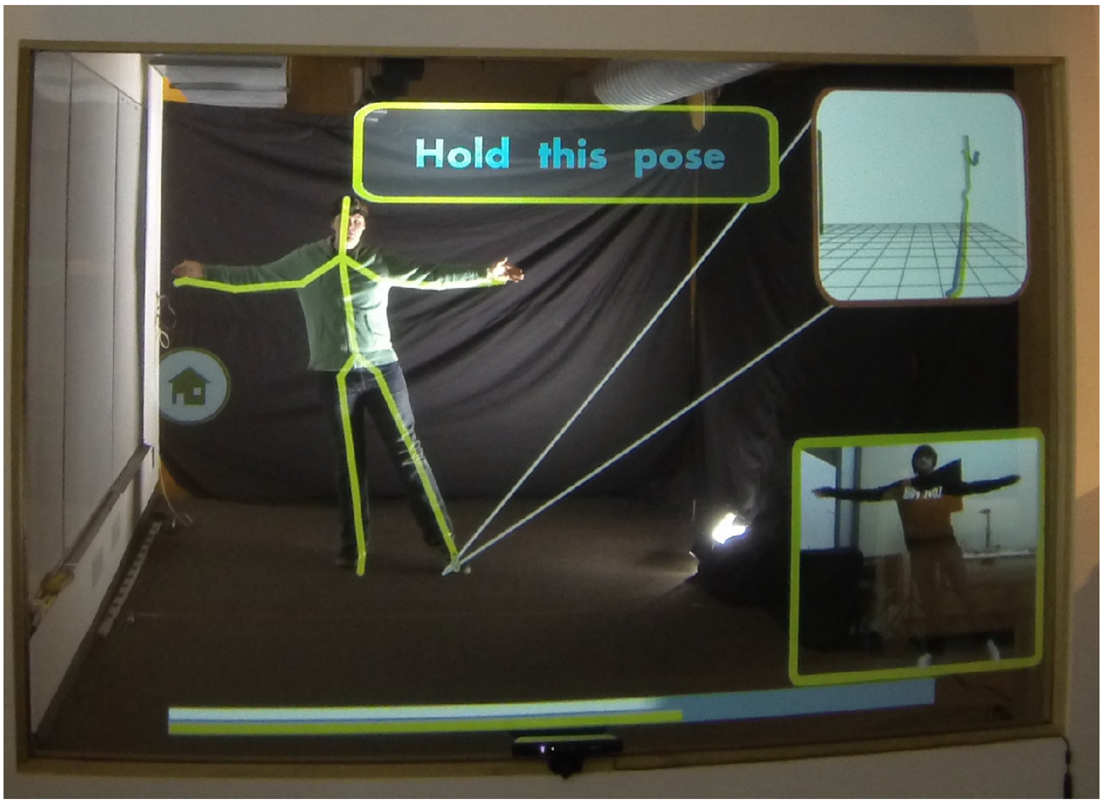


Figure 7: AR "magic mirror" [7]. The ideal motions are overlayed on top of a mirror image of the user.

then convert it into a 3D mesh using the marching cubes algorithm. The mesh is colored using view dependent textures in [16], while [1] blends the RGB frames from each camera into one integrated texture.

Another class of these systems interpolates captured views in order to generate novel ones. Zitnick et al. [2] and Karsten et al. [17] segment frames into layers and then blend the layers captured from two cameras in order to render an image from a virtual viewpoint that is somewhere in between the two cameras.

2.2.2 Using RGB-D sensors

Other systems use a combination of depth and RGB data in order to generate free-viewpoint images. Dai and Yang [18] capture and render a subject in real time, from an arbitrary viewpoint, using multiple RGB-D sensors. Each sensor's

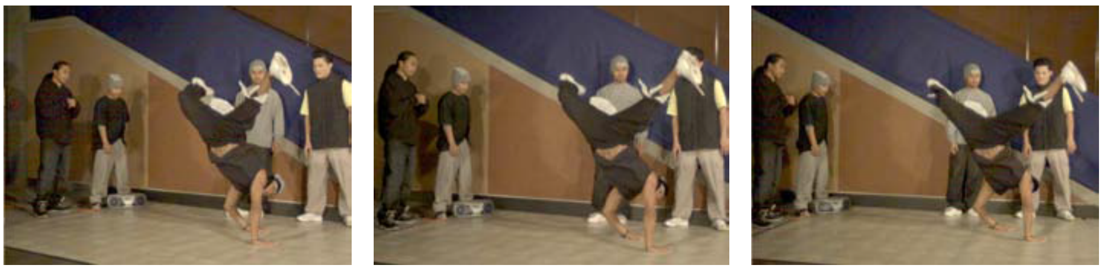


Figure 8: Synthesizing a new viewpoint of a dance [2]. Left and right frames are interpolated to produce the image in the center.

foregound layer is merged to produce the final result. Alexiadis, Zarpalas, and Daras [19] also capture a dynamic scene using multiple RGB-D sensors. Each sensor's output is converted into 3D meshes and merged, taking care to remove redundant polygons.

Depth data also can be converted into 3D point clouds. Point clouds from multiple viewpoints can be integrated to form a representation of the entire scene. Waschbüsch et al. [3] capture an RGB-D stream from projector-camera combinations around the scene (Fig. 2 (bottom)), and then convert the color and depth data into 3D points, which they then integrate. Kainz et al. [20] also make use of multiple RGB-D streams. They use a combination of point cloud integration and IBVH. The point cloud method usually produces noisy edges and IBVH does not detect concavities, but an intersection of the two produces a model with clean edges and proper concavities.

Other interesting systems include De Aguiar et al. [8], who in addition to using RGB frames, also use a laser scanner to construct a 3D mesh model of the subject in advance, which is similar to using a depth sensor. They then capture the subject's motion and use keypoints in each frame to transform the model. In order to locate the 3D positions of the keypoints, they capture from multiple cameras simultaneously. Using this method, they are able to capture a detailed mesh with motion (Fig. 9).

2.2.3 Using human pose estimation

Some free-viewpoint image generation techniques dedicated to synthesizing novel images of human actions also use human pose estimation at some point.

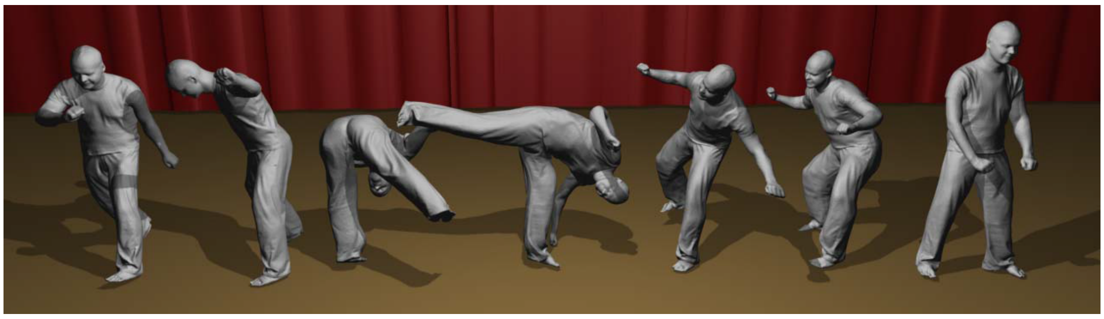


Figure 9: Detailed model and motion of a subject captured by De Aguiar et al. [8]

Carranza et al. [9] first initialize a general 3D model to the body shape of a subject (Fig. 10). Then, by capturing that subject using multiple cameras, they are able to obtain sillhouettes from multiple viewpoints over multiple frames. For each frame, they then find the pose of the 3D model that fits to each sillhouette.

Shotton et al. [21] developed an algorithm that estimates human motion in real time for the Microsoft Kinect. This was used for the systems proposed by Ye et al. [10] and Malleson et al. [22]. In [10], three Kinects are used in order to generate free-viewpoint images of human motion (Fig. 11). Each Kinect captures a point cloud of the scene, similar to the systems above. In order to correctly integrate the point cloud, they use a number of constraints, such as the extrinsic parameters of each Kinect, and the pose of the subjects. Using this method, they are able to generate free-viewpoint image sequences of up to two subjects. In [22], on the other hand, only a single Kinect is used. They build a 3D model of a subject using voxels and apply the subject's motion to the model in order to generate a free-viewpoint image sequence. To accomplish this, they capture the subject's pose in each frame, and then assign voxels to defined body parts.

2.3 Summary and relation to our work

In this section we discussed ways to possibly increase users' comprehension of motion, for the purpose of learning. The two methods that we focused on were AR and free-viewpoint image generation. AR systems are useful for comparisons between virtual and real world objects. Free-viewpoint image generation systems are useful for viewing different perspectives of the same object.

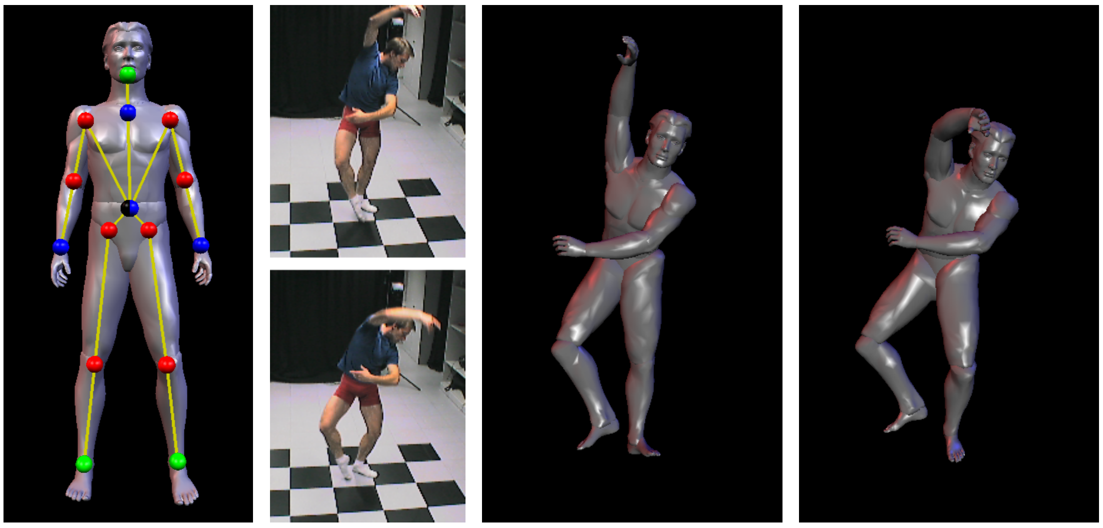


Figure 10: Human pose estimation and free-viewpoint image generation system by Carranza et al. [9] First column: general 3D model. Second column: two frames of input. Third and fourth column: 3D model fit to the input.

Our proposed system renders reenactments, which are free-viewpoint images of a human subject, e.g. the output of [1, 8, 9]. Additionally, so that we are not forced to use multiple cameras, we also propose a method to capture reenactments with a single RGB-D sensor, using human pose estimation. The advantage of our system compared to existing AR-based learning support systems is the fact that we synthesize the complete appearance of the subject that the viewer is attempting to copy. The existing systems are more abstract; this means that their output is not so realistic or natural. In the next part we will go into the details of our AR reenactment system.

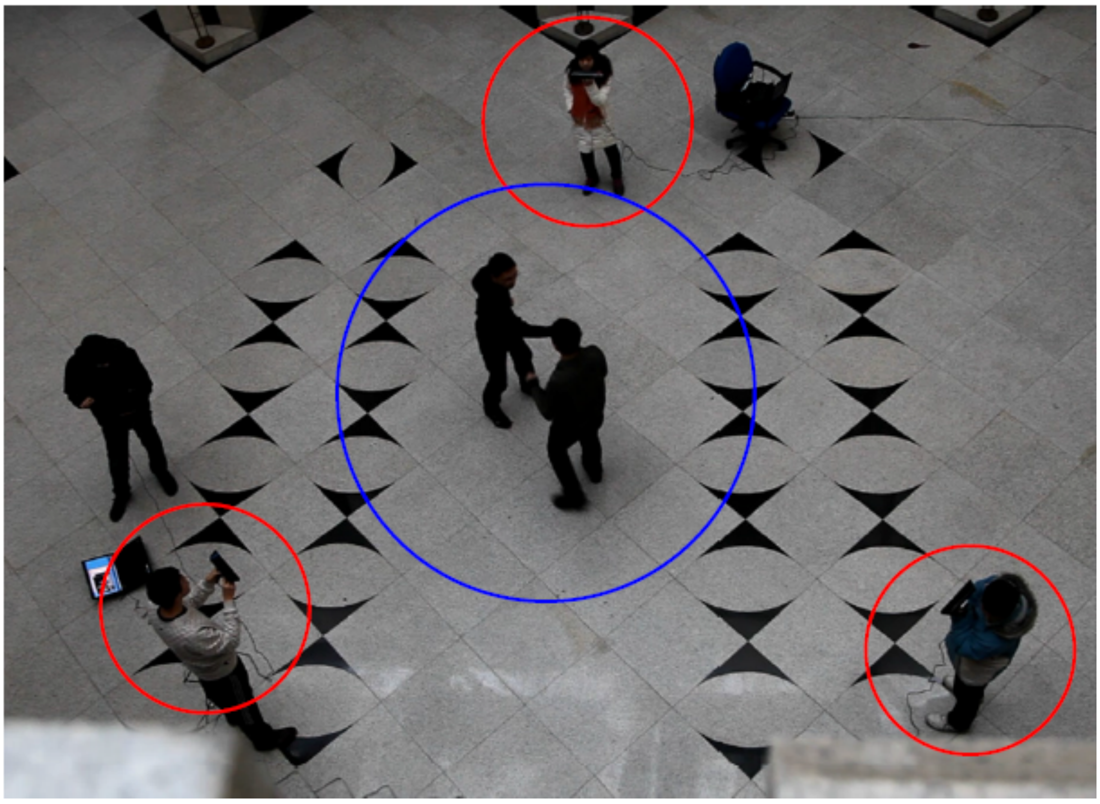


Figure 11: Capturing two users with three handheld Kinects simultaneously [10].

3. The AR Reenactment System

In this section, we will discuss the details of our proposed AR reenactment system. We will go into how a user can capture the motion of a subject, and how a viewer can view the synthesized reenactment.

3.1 Overview of AR reenactment system

As shown in Fig. 12, the proposed AR reenactment system consists of two stages: capturing stage and reenactment stage.

During the capturing stage, a user of the proposed system captures the motion of a subject using a single RGB-D sensor, resulting in a stored video stream consisting of N_{V} video frames. The RGB-D sensor's pose is estimated using a visual SLAM technique [23] in an arbitrarily set world coordinate system (Fig. 13). Each video frame consists of an RGB frame, a depth frame, and the subject's estimated pose, which is represented by the positions of a predefined set of joints, called a skeleton, in the world coordinate system. We then generate a rough 3D model of the subject based on the skeletons and the depth frames. The 3D model consists of one cylinder per body part and only defines their shapes; we will assign their position and rotation during the reenactment stage. The video stream and the 3D model are stored in a database for later use in the reenactment stage. Additionally, the SLAM technique generates a set of map points in order to track the camera's pose. We store those so we can use the same world coordinate system in the reenactment stage.

During the reenactment stage, we synthesize an image of the subject from the stored video stream from the viewpoint of a viewer's camera. The pose of the viewer's camera is estimated using the visual SLAM technique and the 3D map stored in the database. We next synthesize the k-th frame of the subject's reenactment by applying the rough 3D model which we generated during the capturing stage to the k-th skeleton in the stored video stream, and afterward texturing it with the RGB frame from an appropriate stored video frame. Finally, the proposed system presents the reenactment superimposed on the real-time RGB frame captured by the viewer's camera.

Since the proposed system involves three coordinate systems, i.e., the world,

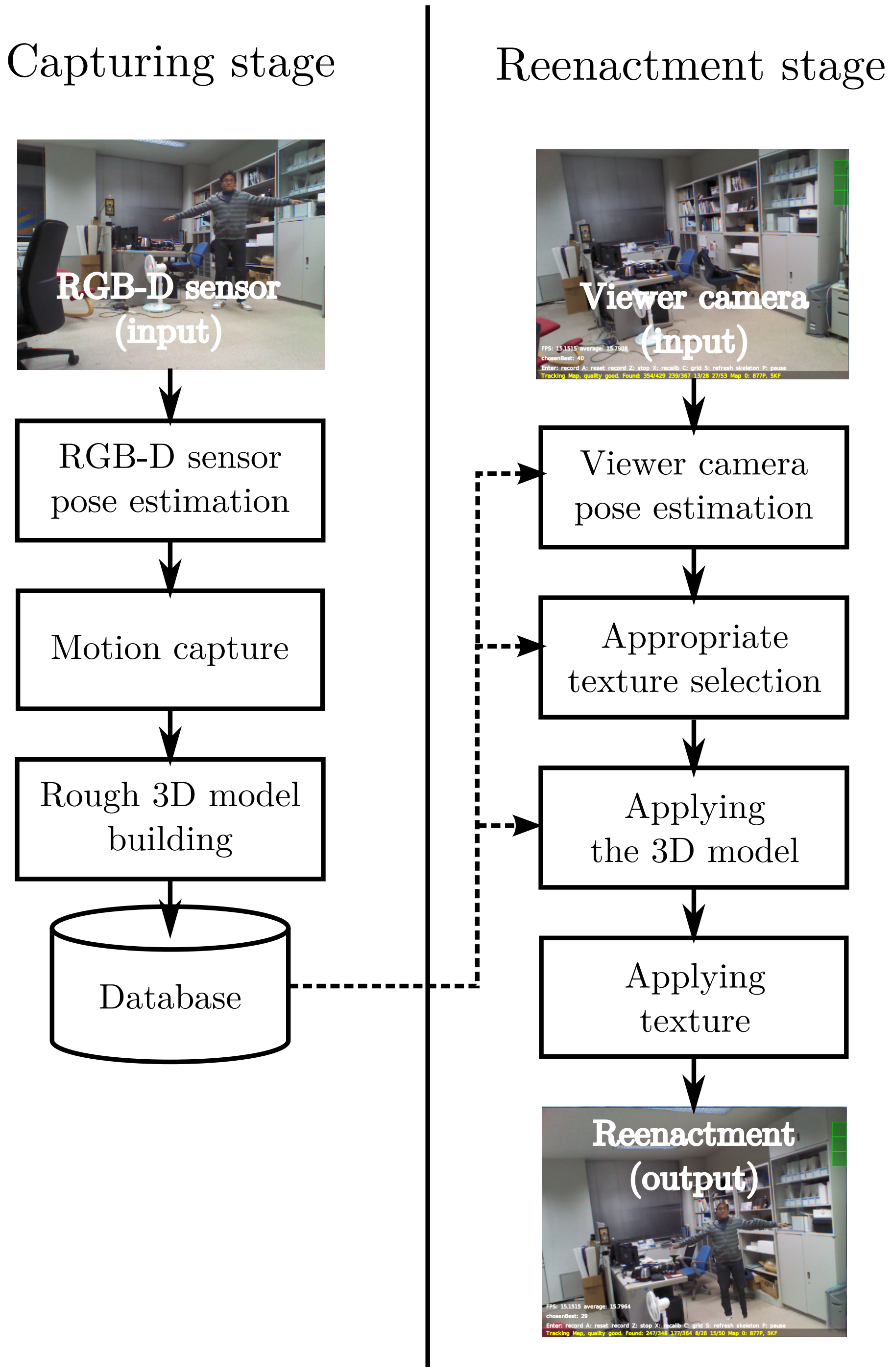


Figure 12: Overview of the proposed system.

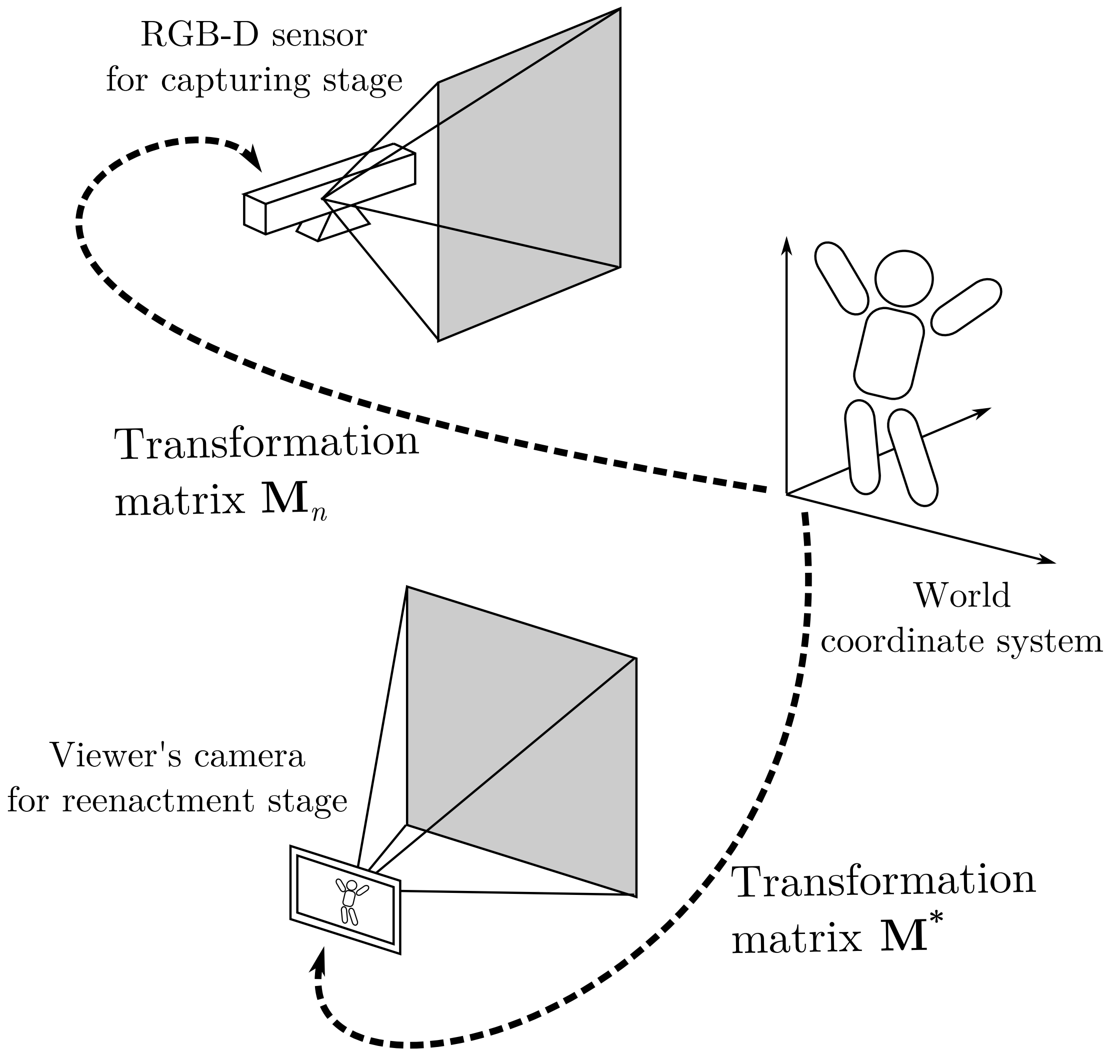


Figure 13: Coordinate systems and the transformations relating them.

the RGB-D sensor's, and the viewer's, we need to transform between them (Fig. 13). The subject's skeleton is initially in the RGB-D sensor's coordinate system and is transformed into the world coordinate system before it is stored in the database. This transformation is done using transformation matrix $\mathbf{M}_n$, which is obtained through camera pose estimation. For synthesizing the subject's reenactment, the stored skeleton is converted from the world coordinate system to the viewer camera's coordinate system. The transformation matrix used for this conversion is $\mathbf{M}^*$, which is again obtained using camera pose estimation.

3.2 Capturing stage

We first estimate the RGB-D sensor's pose in the world coordinate system. Next, we track and capture the subject's skeleton based on the depth frame obtained from the RGB-D sensor. Lastly, we generate a rough 3D model of the subject's body using the stored video stream.

3.2.1 RGB-D sensor pose estimation

As we mentioned, our RGB-D sensor may move while capturing. Therefore, we set an arbitrary world coordinate system and track the sensor's pose within it. For this purpose, we employ a visual SLAM system (e.g. PTAMM [23]) to constantly track our RGB-D sensor's translation and rotation. For the n-th video frame of the video stream, we estimate the RGB-D sensor's pose as extrinsic camera parameters $\mathbf{M}_n$.

3.2.2 Motion capture

Figure 14 (a) shows the N_J joints and N_BP body parts that compose a skeleton of the subject, where a body part is the segment formed by a specific pair of joints. The skeleton of the subject's body in the n-th frame can be extracted and tracked using a human pose estimation technique, e.g., [21]. Assuming a single subject in the scene, we denote the skeleton in the n-th frame by

$$\mathbf{S}_n = \{\mathbf{s}_{n,i} | i = 1, \ldots, N_\mathrm{J}\}, \tag{1}$$

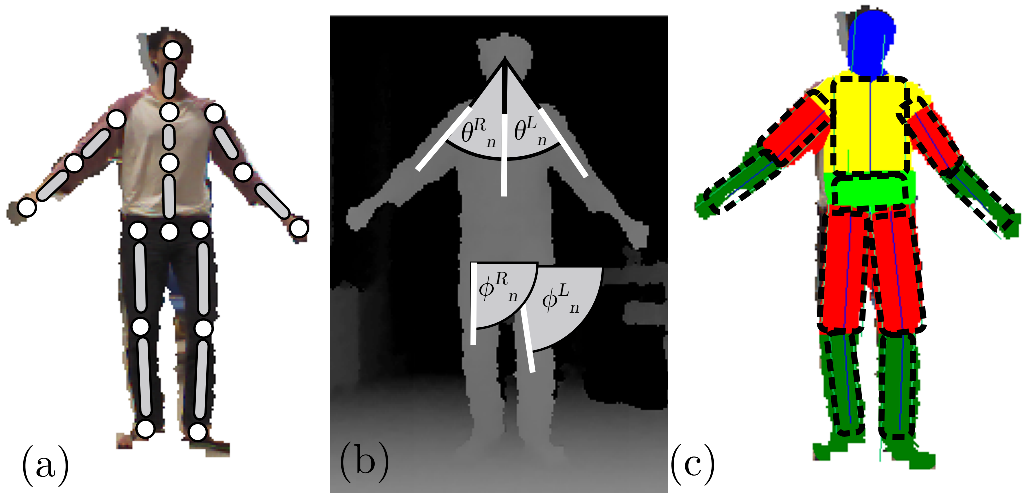


Figure 14: (a) The skeleton representation of the subject. Circles are joints, and segments are body parts. (b) Corresponding depth frame with definitions of some angles. (c) Rectangles fitted to each body part.

where $\mathbf{s}_{n,i}$ is the 3D position of the i-th joint of the skeleton. Since human pose estimation gives the joints positions in the RGB-D sensor's coordinate system, shown in Fig. 13, using the inverse of $\mathbf{M}_n$, which maps a 3D coordinate from the world coordinate system to the RGB-D sensor's coordinate system, we transform the 3D joint positions in $\mathbf{S}_n$ by $\mathbf{s}'_{n,i} = \mathbf{M}_n^{-1}\mathbf{s}_{n,i}$ for all i in $\mathbf{S}_n$ and define the skeleton in the world coordinate system as $\mathbf{S}'_n = \{\mathbf{s}'_{n,i} | i = 1, \dots, N_\mathrm{J}\}$. We store the n-th video frame, i.e., skeleton $\mathbf{S}'_n$, the RGB frame I_n, and depth frame D_n in the database.

3.2.3 Rough 3D model building

To render the reenactment of the subject, we use a 3D model of the subject. In this step, we build the shape of the 3D model. In this work, we chose to represent each body part with cylinders. Since the heights of the cylinders are trivially determined from the length of the body part in the skeleton of a stored video frame, we only need to determine the radii of the cylinders. For this, we first find the index of a single representative frame $\hat{n}$ from the stored video stream and then fit rectangles to the subject's region in the depth frame of the representative frame $D_{\hat{n}}$, which can viewed as a projection of the cylinders onto the image plane of the RGB-D sensor.

To easily obtain radii and heights of the cylinders based on the rectangles that can be considered as the cylinders' projection, the directions of their heights should be perpendicular to the optical axis of the RGB-D sensor and body parts must not overlap. This means that the representative frame should contain the subject's appearance that meet the following requirements: (i) both arms should be away from the body, (ii) the line segments formed by the joints corresponding to both hands should be parallel to the image plane as possible, and (iii) the legs should be uncrossed. The camera should, as much as possible, not be rotated for capturing the representative frame.

These requirements ensure that the representative frame has body parts that are separate from each other as shown in Fig. 14 (a), which makes it easier to build an accurate 3D model of the subject's body. Such a pose may be requested by the user who captures the action, but it may also be captured during the normal course of recording. Using notations as in Fig. 14 (b), these requirements can be empirically encoded as

$$F(n) = \theta_n^{\mathrm{L}} l_n^{\mathrm{L}} + \theta_n^{\mathrm{R}} l_n^{\mathrm{R}} + \lambda g(\phi_n^{\mathrm{R}}, \phi_n^{\mathrm{L}}). \tag{2}$$

The first and second terms, $\theta_n^{\mathrm{L}} l_n^{\mathrm{L}}$ and $\theta_n^{\mathrm{R}} l_n^{\mathrm{R}}$, reward poses where the subject holds his/her left and right arms out to the side and parallel to the image plane, as θ_n^{L} and θ_n^{R} are the angles between the torso and the left and right arms in $\mathbf{S}'_n$, and l_n^{L} and l_n^{R} are the x-components of the normalized left and right arm vectors. The third term $g(\phi_n^{\mathrm{R}}, \phi_n^{\mathrm{L}})$, which rewards poses with legs uncrossed, is defined as

$$g(\phi_n^{\mathrm{R}}, \phi_n^{\mathrm{L}}) = \begin{cases} 1: & \text{if } \phi_n^{\mathrm{R}} > \phi_n^{\mathrm{L}} \\ 0: & \text{otherwise} \end{cases}. \tag{3}$$

λ is an empirically-defined constant.

We obtain the index of the most appropriate frame in the sense of the above criterion by maximizing F, i.e.,

$$\hat{n} = \arg\max_n F(n). \tag{4}$$

We then find the rectangles that fit each body part in $D_{\hat{n}}$, as in Fig. 14 (c). Radius $r_{a,b}$ of the cylinder for the body part with joints a and b is then given as the length of the line segments perpendicular to the body part segment. For compensating the differences in the body part segment length from frame to frame, we store in the database the radius ratio given by $w_{a,b}/\|\mathbf{s}_{\hat{n},a} - \mathbf{s}_{\hat{n},b}\|$ for each body part, where $w_{a,b}$ is the width of the fitting rectangle.

3.3 Reenactment stage

In the reenactment stage, a viewer with a mobile device captures an RGB-only video stream from a camera installed in the device. The proposed system synthesizes the reenactment sequentially for this real-time stream. For each frame, we estimate extrinsic camera parameters $\mathbf{M}^*$ of the viewer's camera again by the SLAM technique based on the 3D map stored in the database. The reenactment stage then transforms the skeleton to the viewer's camera coordinate system using $\mathbf{M}^*$, applies the rough 3D model stored in the database, and colors it based on the appropriate RGB frame in the database. Finally, the reenactment is superimposed on the frame of real-time RGB video stream to be displayed on the viewer's mobile device.

3.3.1 Viewer camera pose estimation

In the reenactment stage, we track the viewer camera's pose in order to convert the subject's skeleton from the world coordinate system to the viewer camera coordinate system. In order to use the same world coordinates as in our capturing stage, we use the 3D map generated during the capturing stage.

3.3.2 Appropriate texture selection

Since our 3D model of the subject is very rough and no color is assigned to it, we apply textures to our 3D model. For a static scene, view-dependent texture mapping proposed by Debevec et al. [24] works well for this purpose by assigning as textures those images which were captured from the viewpoint close to that of the novel image to be synthesized. But we cannot adopt it naively because the proposed system captures a moving subject and uses only a single RGB-D sensor and thus there are no RGB frames that show the same scene at the same time from different viewpoints. Our idea for solving this problem is based on our

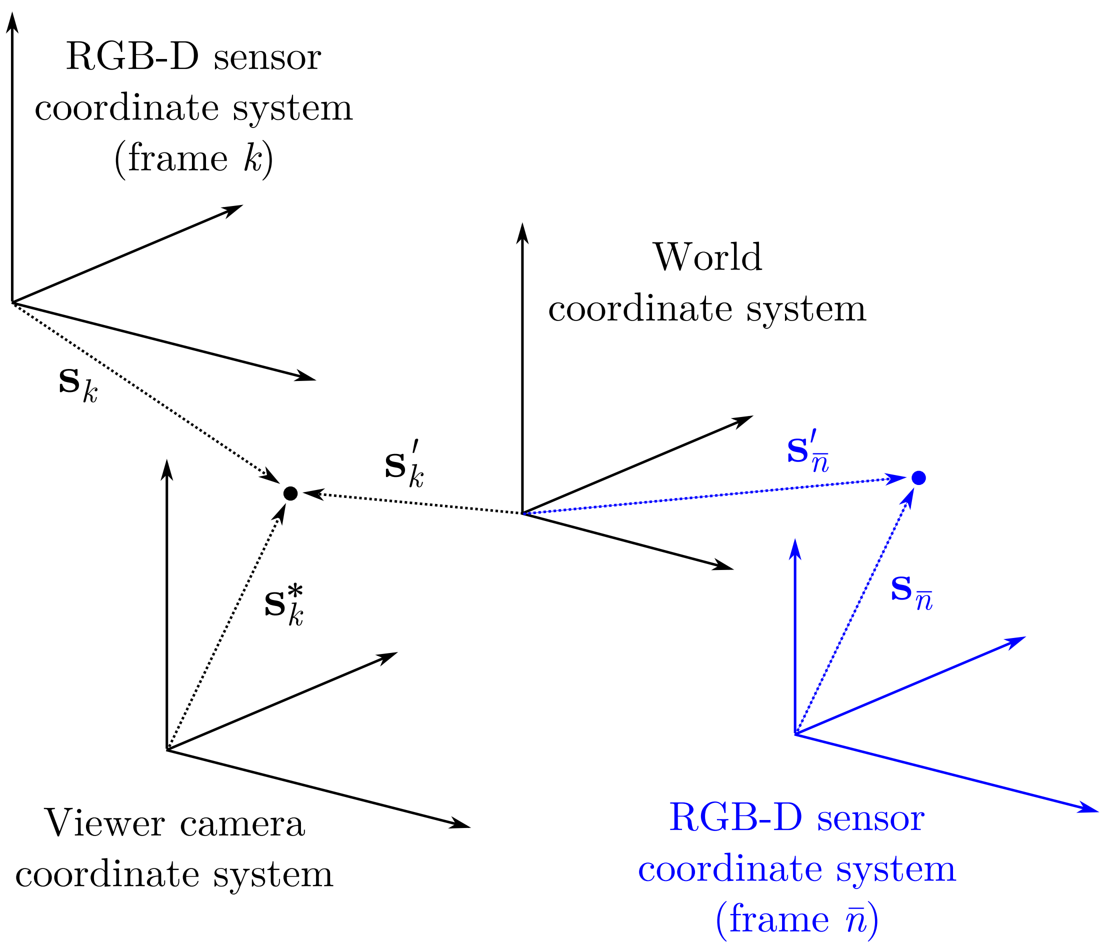


Figure 15: Vector transformations.

observation that there may be RGB frames that show the subject in a similar pose, which means that we can select a frame such that the joint positions in the selected frame are close to those in the novel image to be synthesized.

When reenacting the subject's appearance from the k-th skeleton, $\mathbf{S}_k'$, we first transform joint $\mathbf{s}_{k,i}'$ in the world coordinate system into the viewer camera's coordinate system using $\mathbf{M}^*$, giving us $\mathbf{S}_k^*$. We also transform $\mathbf{S}_n'$ for all n into its original RGB-D sensor's coordinate system using $\mathbf{M}_n$, giving us $\mathbf{S}_n$. Figure 15 shows the three coordinate systems and the Since the position of the subject in the world coordinate system differs frame by frame, to make the selection translation invariant, the position of a specific joint is subtracted from each joint's position in order to make the specific joint coincide with the origin. In this work, we choose the neck joint shown in Fig. 14 (a) as the origin. We select the appropriate stored video frame, where the associated skeleton $\mathbf{S}_n$ (in the original RGB-D sensor's coordinate system) is closest to the $\mathbf{S}^*_k$ (in the viewer camera's coordinate system). To summarize, we find appropriate frame index $\bar{n}$ by

$$\bar{n} = \arg\min_n \sum_{i=1}^{N_\mathrm{J}} \|(\mathbf{s}^*_{k,i} - \mathbf{s}^*_{k,\mathrm{neck}}) - (\mathbf{s}_{n,i} - \mathbf{s}_{n,\mathrm{neck}})\|, \tag{5}$$

where $\mathbf{s}^*_{k,\mathrm{neck}}$ and $\mathbf{s}_{n,\mathrm{neck}}$ are the neck joint position of $\mathbf{S}'_k$ and $\mathbf{S}'_n$, respectively.

This method of texture selection does not preserve the subject's facial expression. For our purpose, however, it is sufficient to make the subject's motion comprehensible.

3.3.3 Applying the 3D model

We now render the cylinder model based on the radius ratios stored in the database, the frame in the sequence that we are reenacting, and the joints in the viewer camera coordinate system. For the body part associated with joint indices a and b, the cylinder axis endpoints will be at $\mathbf{s}^*_{k,a}$ and $\mathbf{s}^*_{k,b}$. The radius will be the length of the cylinder axis multiplied by the radius ratio computed earlier: $r_{k,a,b} = \|\mathbf{s}^*_{k,a} - \mathbf{s}^*_{k,b}\| r_{a,b}$, where $r_{a,b}$ is the radius ratio associated with the body part.

3.3.4 Applying texture

Now that we have selected an appropriate RGB frame, we can apply it to the cyinder model. However we cannot just copy the texture naively. Since the poses represented by $\mathbf{S}^*_k$ and $\mathbf{S}_{\bar{n}}$ are not exactly the same, naively copying the cylinder to the selected RGB frame can lead to inconsistency between the cylinders and the frame (Fig. 16). We thus find transformation $\mathbf{\Omega}_{k,\bar{n}}$ individually for each cylinder that compensates for the difference between the subject's poses in $\mathbf{S}^*_k$ and $\mathbf{S}_{\bar{n}}$. We apply this transformation to each point on the cylinder to find its corresponding pixel location on the RGB frame, and determine its color from that pixel.

We want to transform the cylinder with axis endpoints $\mathbf{s}^*_{k,a}$ and $\mathbf{s}^*_{k,b}$ and radius $r_{k,a,b}$, which the cylinder in viewer camera coordinates, into the cylinder with axis

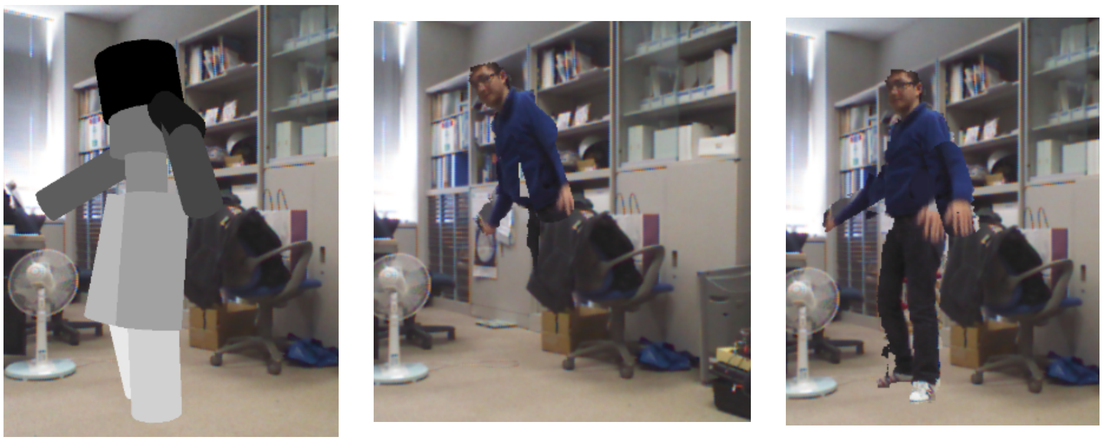


Figure 16: First column: cylinder model. Second column: Texturing using only the head body part's projection. Third column: Texturing with individual body part projection.

endpoints $\mathbf{s}_{\bar{n},a}$ and $\mathbf{s}_{\bar{n},b}$ and radius $r_{\bar{n},a,b}$, which is the corresponding cylinder in the RGB frame we are extracting from (Fig. 17).

All equations from here will refer to the body part formed by joints a and b. We omit the subscript unless we need to refer to one of the two joints in particular.

First we find the axis vectors $\mathbf{v}^*_k$ for the viewer camera cylinder and $\mathbf{v}_{\bar{n}}$ for the RGB frame cylinder:

$$\mathbf{v}^*_k = \mathbf{s}^*_{k,a} - \mathbf{s}^*_{k,b} \tag{6}$$

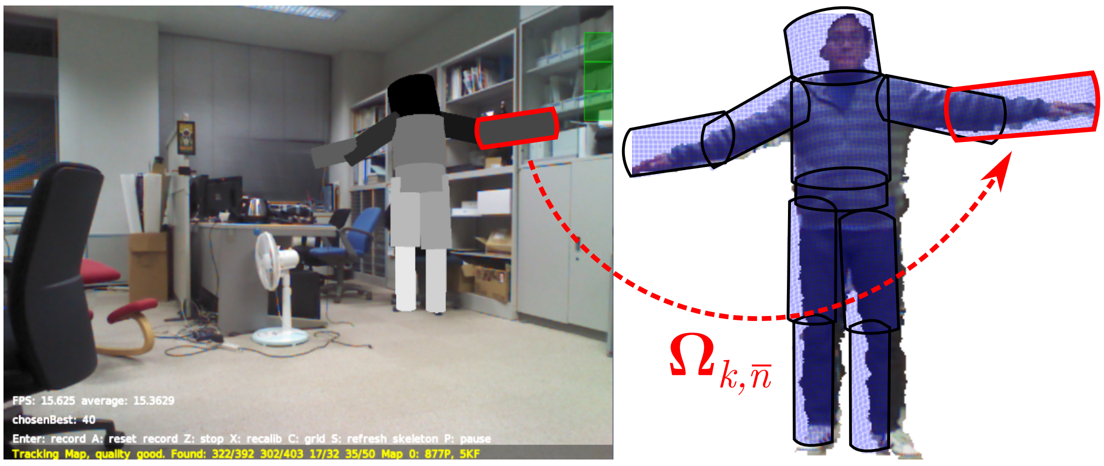
$$\mathbf{v}_{\bar{n}} = \mathbf{s}_{\bar{n},a} - \mathbf{s}_{\bar{n},b}. \tag{7}$$

The rotation is then calculated by:

$$\mathbf{R}_{k,\bar{n}} = \mathbf{\Theta}\left(\mathbf{v}^*_k \times \mathbf{v}_{\bar{n}}, \cos^{-1}\left(\frac{\mathbf{v}^{*T}_k \mathbf{v}_{\bar{n}}}{\|\mathbf{v}^*_k\|\|\mathbf{v}_{\bar{n}}\|}\right)\right). \tag{8}$$

$\mathbf{\Theta}(\mathbf{w}, \theta)$ is the Rodrigues rotation algorithm, i.e. the algorithm that finds the rotation matrix given an axis and an angle. The axis is the cross product between the axis vectors, and the angle is the angle between them. The scale is the ratio between the lengths of the two vectors:

$$s_{k,\bar{n}} = \frac{\|\mathbf{v}^*_k\|}{\|\mathbf{v}_{\bar{n}}\|}, \tag{9}$$



```
FPS: 15.625 average: 15.3629
chosenBest: 40
Enter: record A: reset record Z: stop X: recalib C: grid S: refresh skeleton P: pause
Tracking Map, quality good. Found: 322/392 302/403 17/32 35/50 Map 0: 877P, 5KF
```

Figure 17: Left: cylinder model in the viewer's camera. Right: RGB frame for coloring with overlaid cylinders. $\mathbf{\Omega}$ is the transformation that relates them.

the translation is the difference between the joints with index a:

$$\mathbf{t}_{k,\bar{n}} = \mathbf{s}_{\bar{n},a} - \mathbf{s}^*_{k,a}, \tag{10}$$

and the resulting transformation is (Fig. 17):

$$\mathbf{\Omega}_{k,\bar{n}} = \left[\begin{array}{c|c} s_{k,\bar{n}}\mathbf{R}_{k,\bar{n}} & \mathbf{t}_{k,\bar{n}} \end{array}\right] \tag{11}$$

Finally, we need to compensate for facing, which means ensuring that the resulting cylinder faces front, i.e. the side of the cylinder that the viewer can see. This means that we need to find the vector that faces front, which is the unit vector orthogonal to the cylinder axis that is closest to the ray from the axis endpoint to the viewing point (Fig. 18). We calculate the front vector of the first cylinder and apply $\mathbf{\Omega}_{k,\bar{n}}$ to it to see where it ends up. Then, we calculate the ideal facing vector of the second cylinder and rotate it along its axis.

We take the viewing ray from an axis endpoint (e.g., $-\mathbf{s}^*_{k,a}$). The front vector of a cylinder is calculated by taking the vector rejection of that viewing ray from the cylinder axis:

$$\mathbf{f}^*_k = \hat{\mathbf{v}}^*_k \times (-\mathbf{s}^*_{k,a} \times \hat{\mathbf{v}}^*_k) \tag{12}$$

Applying $\mathbf{\Omega}_{k,\bar{n}}$ to it, we obtain the intermediate facing:

$$\mathbf{f}'_{k,\bar{n}} = \mathbf{\Omega}_{k,\bar{n}}(\mathbf{s}^*_{k,a} + \mathbf{f}^*_k) \tag{13}$$

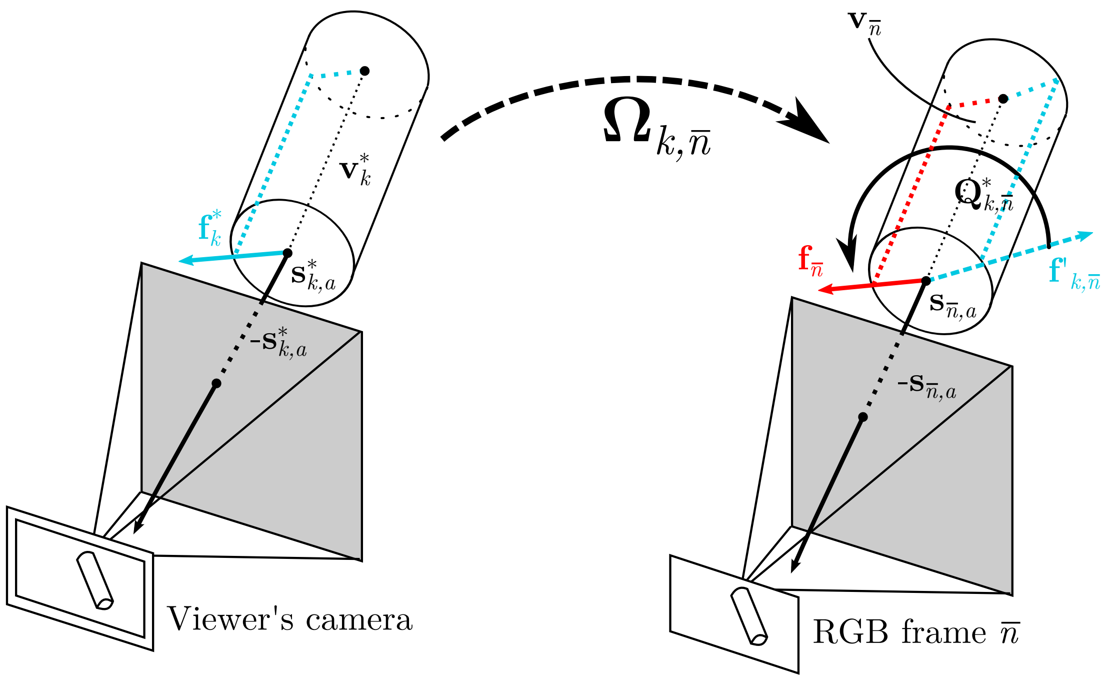


Figure 18: Rotating a cylinder so that it faces front.

We now calculate the ideal facing for the cylinder on frame $\bar{n}$, again using the vector rejection:

$$\mathbf{f}^*_{\bar{n}} = \hat{\mathbf{v}}^*_{\bar{n}} \times (-\mathbf{s}^*_{\bar{n},a} \times \hat{\mathbf{v}}^*_{\bar{n}}) \tag{14}$$

Finally, we rotate the intermediate facing into the ideal facing by using the cylinder axis on frame $\bar{n}$ and the angle between the two facings:

$$\mathbf{Q}^*_{k,\bar{n}} = \boldsymbol{\Theta}\left(\mathbf{v}_{\bar{n}}, \cos^{-1}\left(\frac{\mathbf{f}'_{k,\bar{n}}{}^T\mathbf{f}_{\bar{n}}}{||\mathbf{f}'_{k,\bar{n}}||||\mathbf{f}_{\bar{n}}||}\right)\right) \tag{15}$$

The final transformation is thus, for viewer camera coordinate point $\mathbf{p}^*_k$ and RGB frame coordinate point $\mathbf{p}_{\bar{n}}$:

$$\mathbf{p}_{\bar{n}} = \mathbf{Q}^*_{k,\bar{n}}\boldsymbol{\Omega}_{k,\bar{n}}\mathbf{p}^*_k \tag{16}$$

We then apply the intrinsic camera matrix to $\mathbf{p}_{\bar{n}}$ in order to get the pixel value at (x, y) on the RGB frame. We use our human pose estimation technique in order to determine if it belongs to the subject, or to the background. In the latter case, we ignore the pixel.

Finally, we superimpose the reenactment on the real-time RGB frame captured from the viewer's camera.

4. Implementation

We implemented the capturing stage of the proposed system on an Intel i7 with a 3.20GHz processor and 32GB of RAM. We used a Microsoft Kinect as our RGB-D sensor. We implemented the reenactment stage on a Microsoft Surface Pro 2 with a 1.60GHz processor and 4GB of RAM. We used its embedded camera as the viewer's camera. The intrinsic parameters for Kinect and Surface's cameras were preliminarily calibrated. We used OpenGL [25] to render the reenactment. We set constant λ for 3D model building to be 2000.

For camera pose estimation in both capturing and reenactment stages, we employed Parallel Tracking and Multiple Mapping (PTAMM) [26], which is capable of storing the 3D map for later uses, in order that we can use the shared world coordinate system in these stages. We set the world coordinate system according to PTAMM.

In order to continuously track the subject's joints, we made use of OpenNI [27] and NiTE [28] skeleton tracking. Our representation of the skeleton contains 16 joints and 11 body parts (Fig. 19). Unfortunately, the NiTE skeleton tracker returns joints in the depth sensor's coordinate system, a different coordinate system from PTAMM. Thus we converted the skeleton joints captured by the skeleton tracker into the coordinate system used by PTAMM. To do this, we calculate a transformation matrix (i.e., rotation, translation, and scale) between the two coordinate systems.

Happily, PTAMM tracks a number of map points, i.e. feature points with which it estimates the camera pose, as well as their 3D coordinate in the camera coordinate system. We now query the 3D camera coordinate representation of each map point. Next, we take a depth frame captured by our RGB-D sensor, and for each map point, find the corresponding depth pixel and convert it into a 3D point in the RGB-D sensor's coordinate system, according to our sensor's driver. We now need to find the transformation between the RGB-D sensor points and the PTAMM map points. Given that $\mathbf{p}_i$ is the i-th map point and $\mathbf{q}_i$ is the corresponding point projected to the RGB-D sensor's coordinate system, we find the transformation, i.e. rotation $\mathbf{R}$, translation $\mathbf{t}$, and scale s, in the sense of

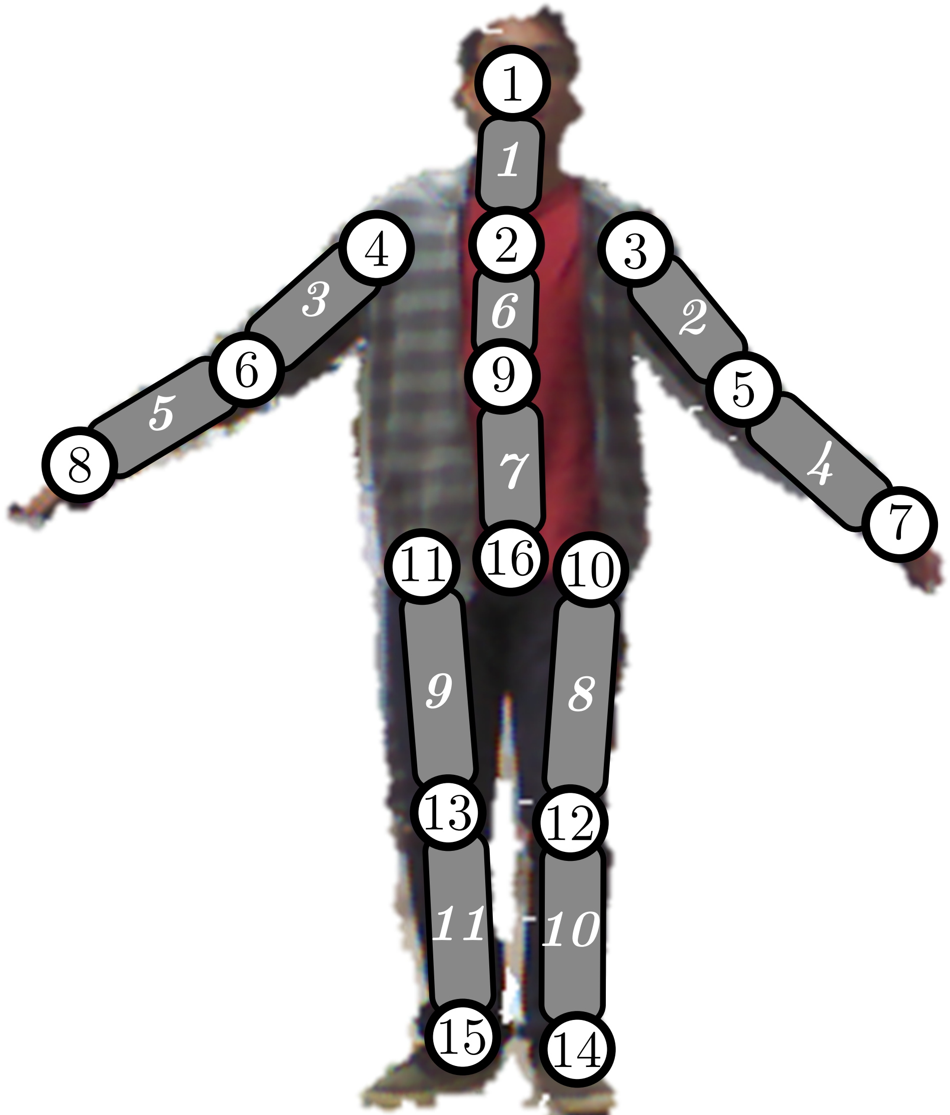


Figure 19: Our representation of the skeleton. Circles: joints, rounded rectangles: body parts.

least squares, as:

$$(\bar{\mathbf{R}}, \bar{\mathbf{t}}, \bar{s}) = \underset{(\mathbf{R},\mathbf{t},s)}{\arg\min} \sum_{i=1}^{N} \|\mathbf{p}_i - (s\mathbf{R}\mathbf{q}_i + \mathbf{t})\|^2. \tag{17}$$

According to [29], we can obtain $\bar{\mathbf{R}}$ by

$$\bar{\mathbf{R}} = \mathbf{U} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(\mathbf{U}\mathbf{V}^T) \end{bmatrix} \mathbf{V}^T, \tag{18}$$

where $\mathbf{U}$ and $\mathbf{V}^T$ are obtained from the singular value decomposition of the covariance matrix of the two sets of points:

$$\mathbf{U\Sigma V^T} = \sum_{i=1}^{N} \mathbf{p}_i' \mathbf{q}_i'^T, \tag{19}$$

where $\mathbf{p}_i'$ and $\mathbf{q}_i'$ are the points adjusted by their centroid:

$$\mathbf{p}_i' = \mathbf{p}_i - \frac{1}{N} \sum_{j=1}^{N} \mathbf{p}_j, \tag{20}$$

$$\mathbf{q}_i' = \mathbf{q}_i - \frac{1}{N} \sum_{j=1}^{N} \mathbf{q}_j. \tag{21}$$

We can use the rotation and the centroid-adjusted points to form another least-squares equation to find the scale:

$$\bar{s} = \underset{s}{\arg\min} \sum_{i=1}^{N} \|\mathbf{p}_i' - s\bar{\mathbf{R}}\mathbf{q}_i'\|^2, \tag{22}$$

which we solve by finding the ordinary least squares estimator:

$$\bar{s} = \frac{\sum_{i=1}^{N} \mathbf{p}_i'^T \bar{\mathbf{R}} \mathbf{q}_i'}{\sum_{i=1}^{N} \|\bar{\mathbf{R}}\mathbf{q}_i'\|^2}. \tag{23}$$

Lastly, the translation is the difference between the properly rotated and scaled centroids.

$$\bar{\mathbf{t}} = -\bar{s}\bar{\mathbf{R}} \sum_{\mathbf{j=1}}^{\mathbf{N}} \mathbf{q_j} + \sum_{\mathbf{j=1}}^{\mathbf{N}} \mathbf{p_j} \tag{24}$$

For any point $\mathbf{q}^*$ that we obtain from the RGB-D sensor, we can thus convert it to the correct coordinate system by $\mathbf{p}^* = \bar{s}\bar{\mathbf{R}}\mathbf{q}^* + \bar{\mathbf{t}}$.

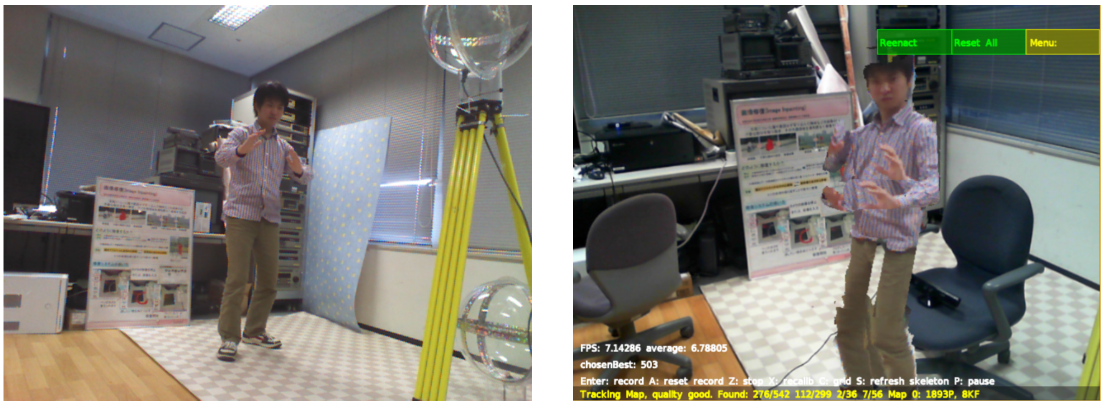
5. Experimental Results

We evaluated the system using our prototype, and we also conducted a user study to find out how users compare the reenactments to conventional video. In this section we describe the reenactment results, as well as the results of the user study.

5.1 Reenactment results

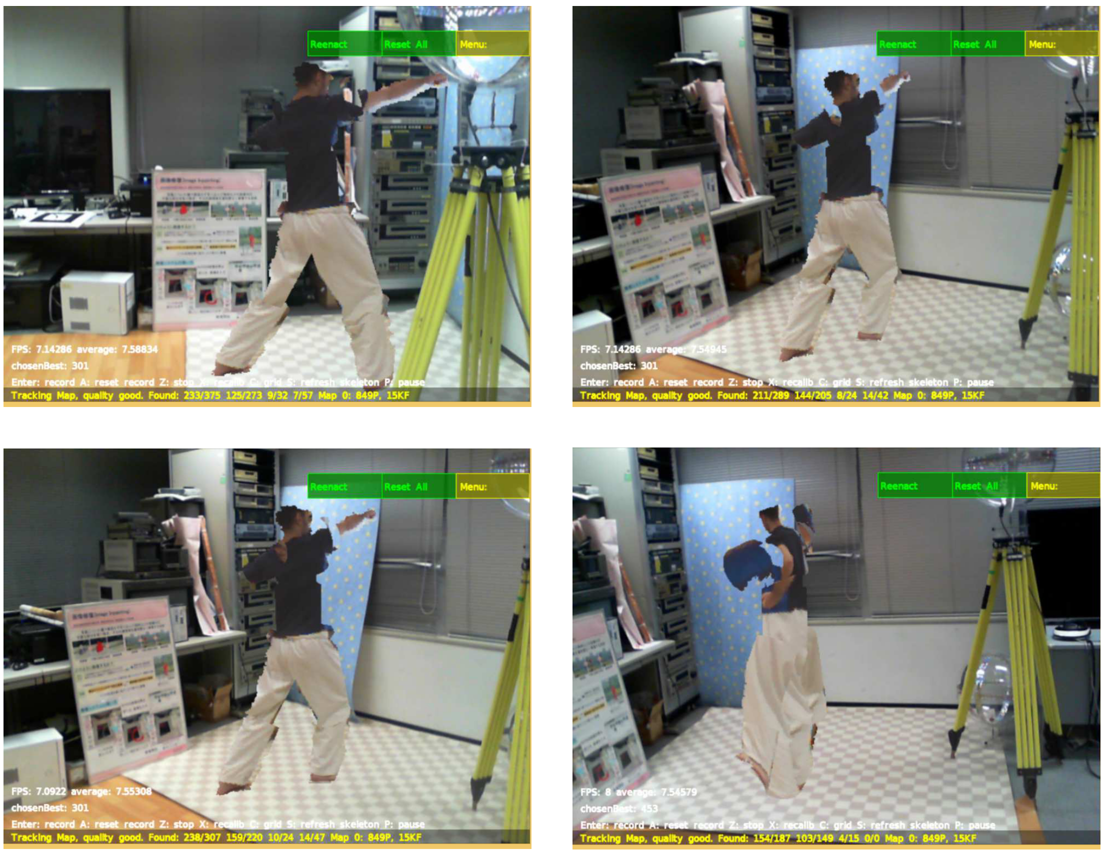
Figures 20–26 show the reenactments generated by the proposed system. We can see that the reenactments generated by the system are coherent, i.e. they are consistent with regards to the recorded motion, the subject's appearance, and the change in viewpoint.

However, there are some major errors that would possibly impede comprehension. The most noticeable error is the doubling of body parts, which can be seen in Fig. 20. This is the result of self-occlusion by the subject. The 3D model coloring process divides the RGB frame into body parts for texturing each cylinder in the rough 3D model. But the texture extraction does not account for occlusion, which sometimes results in unwanted objects being included in the texture. At large changes in viewing angle, we observe a different type of error. In the image on the first column and first row of Fig. 21, the subject's legs are incomprehensible. Also, large artifacts have appeared in the arm regions. This is the result of choosing inappropriate RGB frame to color the cylinders, or simply that more appropriate RGB frames do not exist. This type of error makes it harder to comprehend the output, as the body parts appear to face the wrong way and adjacent body parts do not combine well. Another type of error can be seen in Fig. 22. If the environment has changed since the time of capture, the reenactment may be rendered in the wrong place. Lastly, the skeleton tracker sometimes fails to register the correct joint positions (Fig. 23). This appears to happen often when the subject moves quickly or faces away from the RGB-D sensor. The effects are two-fold. For one, it introduces a skeleton into the sequence that is inconsistent with the other skeletons. If this skeleton is textured, it produces bad results, as in the figure. For another, the skeleton is misregistered with the associated RGB frame. If we use that frame as a texture, we risk extracting textures from the



```
FPS: 7.14286 average: 6.78805
chosenBest: 503
Enter: record A: reset record Z: stop X: recalib C: grid S: refresh skeleton P: pause
Tracking Map, quality good. Found: 276/542 112/299 2/36 7/56 Map 0: 1893P, 8KF
```

Figure 20: Left: frame from an RGB video. Right: equivalent reenactment with double-limb error.



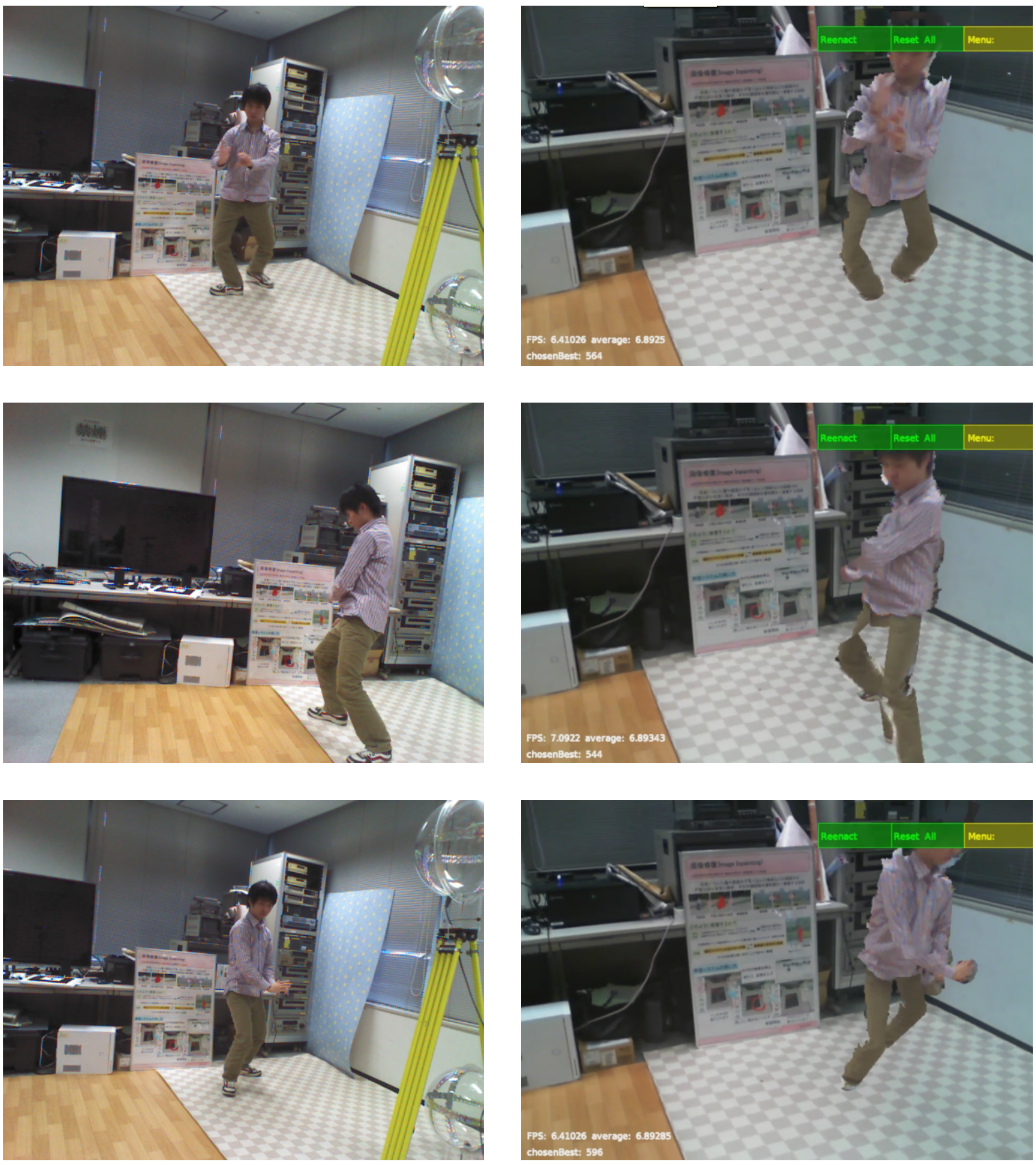
```
FPS: 7.14286 average: 7.58834
chosenBest: 301
Enter: record A: reset record Z: stop X: recalib C: grid S: refresh skeleton P: pause
Tracking Map, quality good. Found: 233/375 125/273 9/32 7/57 Map 0: 849P, 15KF
```

```
FPS: 7.14286 average: 7.54945
chosenBest: 301
Enter: record A: reset record Z: stop X: recalib C: grid S: refresh skeleton P: pause
Tracking Map, quality good. Found: 211/289 144/205 8/24 14/42 Map 0: 849P, 15KF
```

```
FPS: 7.0922 average: 7.55308
chosenBest: 301
Enter: record A: reset record Z: stop X: recalib C: grid S: refresh skeleton P: pause
Tracking Map, quality good. Found: 238/307 159/220 10/24 14/47 Map 0: 849P, 15KF
```

```
FPS: 8 average: 7.54579
chosenBest: 453
Enter: record A: reset record Z: stop X: recalib C: grid S: refresh skeleton P: pause
Tracking Map, quality good. Found: 154/187 103/149 4/15 0/0 Map 0: 849P, 15KF
```

Figure 21: Comparison of the same frame of a reenactment in our system from different angles.

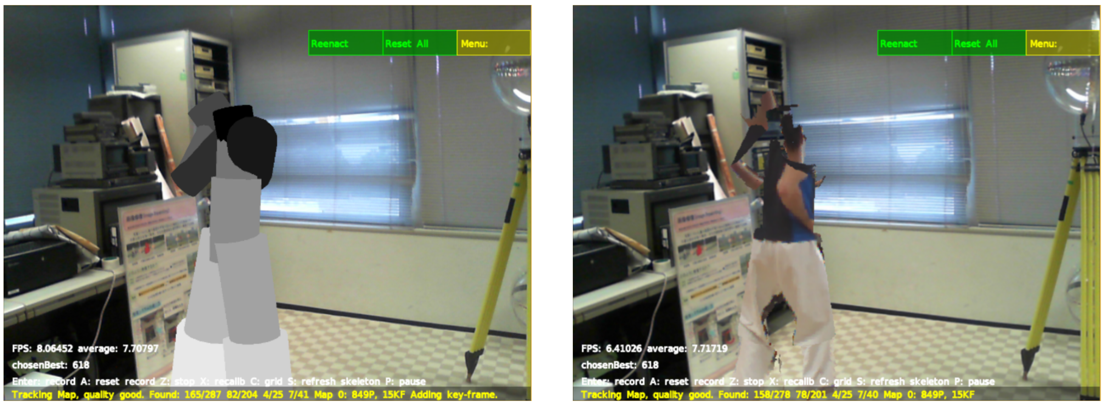


```
FPS: 6.41026 average: 6.8925
chosenBest: 564
```

```
FPS: 7.0922 average: 6.89343
chosenBest: 544
```

```
FPS: 6.41026 average: 6.89285
chosenBest: 596
```

Figure 22: Left column: sample frames from an RGB video. Right column: equivalent reenactments.



```
FPS: 8.06452 average: 7.70797
chosenBest: 618
Enter: record A: reset record Z: stop X: recalib C: grid S: refresh skeleton P: pause
Tracking Map, quality good. Found: 165/287 82/204 4/25 7/41 Map 0: 849P, 15KF Adding key-frame.
```

```
FPS: 6.41026 average: 7.71719
chosenBest: 618
Enter: record A: reset record Z: stop X: recalib C: grid S: refresh skeleton P: pause
Tracking Map, quality good. Found: 158/278 78/201 4/25 7/40 Map 0: 849P, 15KF
```

Figure 23: Left: misregistered pose applied to the cylinder model. Right: texture applied to the misregistered pose.

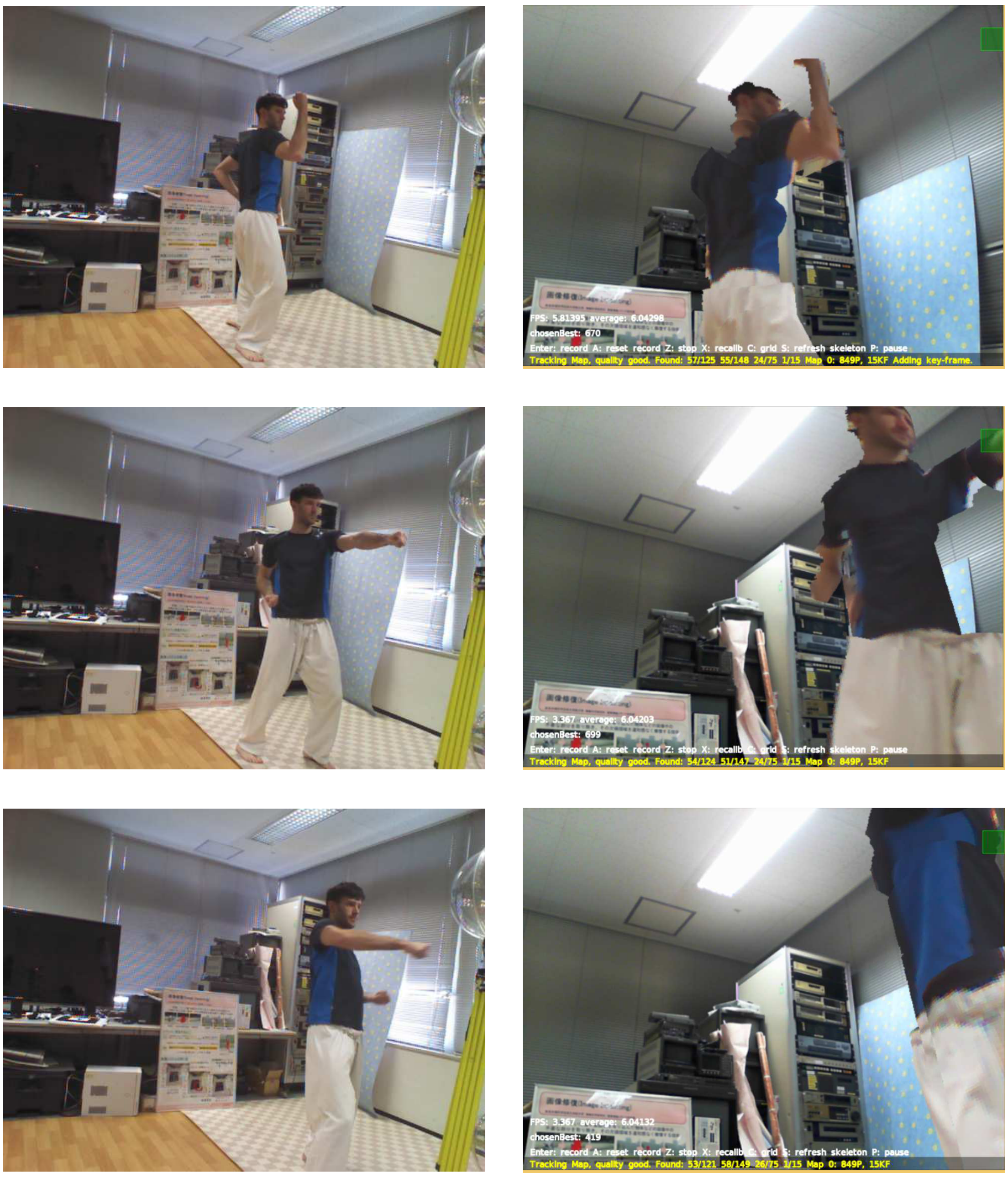
wrong parts of the RGB frame.

On average, the system runs at 9fps on the Surface during reenactment.

5.2 User study

5.2.1 Setup

We asked a martial artist to serve as our subject and captured him performing a Taekwondo form. We captured the motion with a fixed-position Kinect, considering that users who capture an RGB-D video stream generally do not move while capturing. Before capturing, we made a 3D map with the PTAMM system. The resulting RGB-D stream was 23 seconds long and consisted of 580 frames. For comparison between the proposed system and conventional video, we also compiled the RGB component of our captured stream into a separate video. Figs. 24, 25, and 26 show some example frames from the RGB images, compared with the AR reenactment generated by our proposed system.

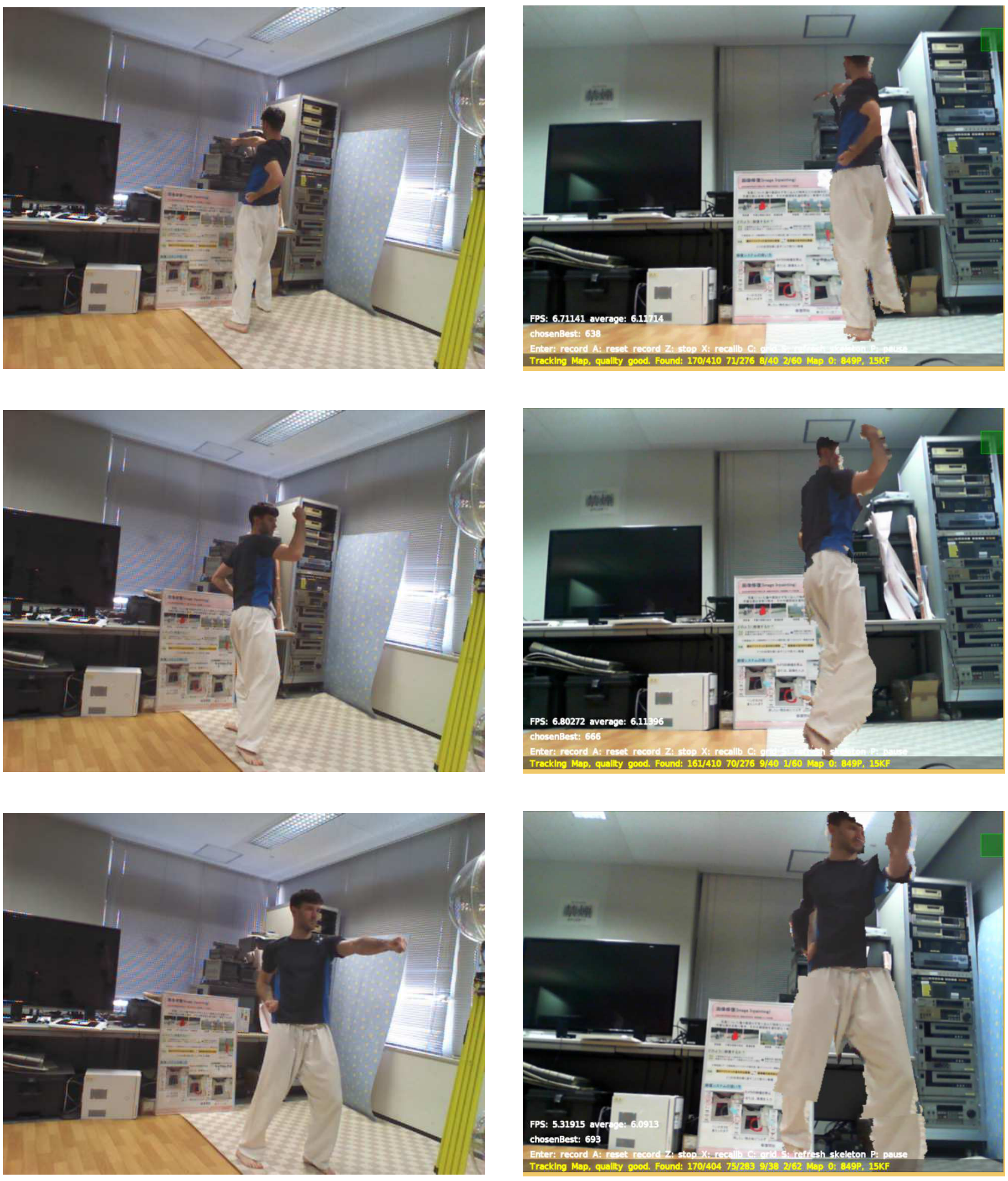


```
FPS: 5.81395 average: 6.04298
chosenBest: 670
Enter: record A: reset record Z: stop X: recalib C: grid S: refresh skeleton P: pause
Tracking Map, quality good. Found: 57/125 55/148 24/75 1/15 Map 0: 849P, 15KF Adding key-frame.
```

```
FPS: 3.367 average: 6.04203
chosenBest: 699
Enter: record A: reset record Z: stop X: recalib C: grid S: refresh skeleton P: pause
Tracking Map, quality good. Found: 54/124 51/147 24/75 1/15 Map 0: 849P, 15KF
```

```
FPS: 3.367 average: 6.04132
chosenBest: 419
Enter: record A: reset record Z: stop X: recalib C: grid S: refresh skeleton P: pause
Tracking Map, quality good. Found: 53/121 58/149 26/75 1/15 Map 0: 849P, 15KF
```

Figure 24: Left column: sample frames from the RGB video that we used in the user study. Right column: equivalent reenactments.

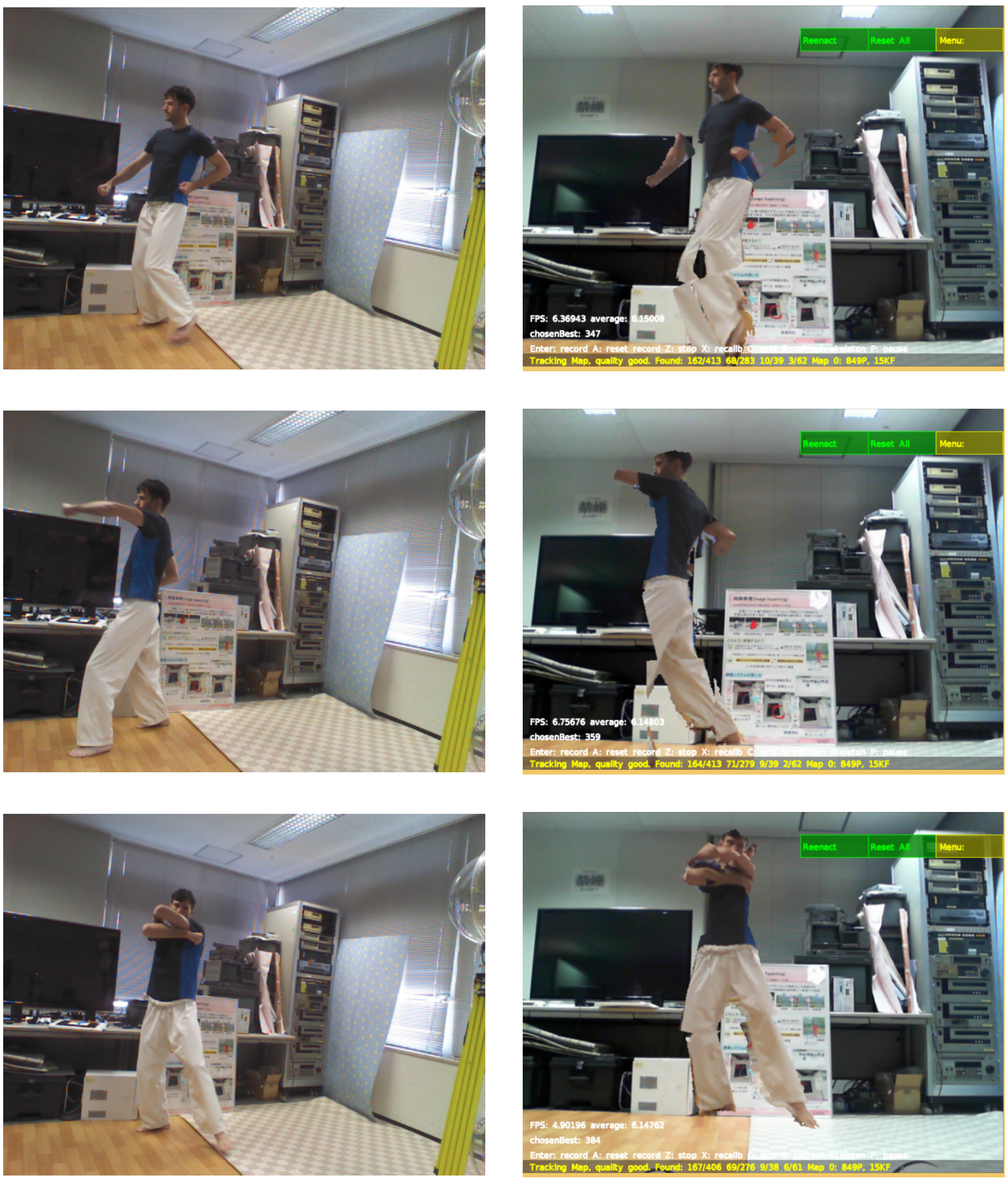


```
FPS: 6.71141 average: 6.11714
chosenBest: 638
Enter: record A: reset record Z: stop X: recalib C: grid S: refresh skeleton P: pause
Tracking Map, quality good. Found: 170/410 71/276 8/40 2/60 Map 0: 849P, 15KF
```

```
FPS: 6.80272 average: 6.11396
chosenBest: 666
Enter: record A: reset record Z: stop X: recalib C: grid S: refresh skeleton P: pause
Tracking Map, quality good. Found: 161/410 70/276 9/40 1/60 Map 0: 849P, 15KF
```

```
FPS: 5.31915 average: 6.0913
chosenBest: 693
Enter: record A: reset record Z: stop X: recalib C: grid S: refresh skeleton P: pause
Tracking Map, quality good. Found: 170/404 75/283 9/38 2/62 Map 0: 849P, 15KF
```

Figure 25: Left column: sample frames from the RGB video that we used in the user study. Right column: equivalent reenactments.



```
FPS: 6.36943 average: 6.15008
chosenBest: 347
Enter: record A: reset record Z: stop X: recalib C: grid S: refresh skeleton P: pause
Tracking Map, quality good. Found: 162/413 68/283 10/39 3/62 Map 0: 849P, 15KF
```

```
FPS: 6.75676 average: 6.14803
chosenBest: 359
Enter: record A: reset record Z: stop X: recalib C: grid S: refresh skeleton P: pause
Tracking Map, quality good. Found: 164/413 71/279 9/39 2/62 Map 0: 849P, 15KF
```

```
FPS: 4.90196 average: 6.14762
chosenBest: 384
Enter: record A: reset record Z: stop X: recalib C: grid S: refresh skeleton P: pause
Tracking Map, quality good. Found: 167/406 69/276 9/38 6/61 Map 0: 849P, 15KF
```

Figure 26: Left column: sample frames from the RGB video that we used in the user study. Right column: equivalent reenactments.

Table 1: Questions asked in our user study (originally in Japanese).

#	Questions	Translation
Q1	普通のカメラで記録した人物が環境中のどこにいるか把握できていると思いますか？	Were you able to comprehend the recorded motion on the conventional video?
Q2	提案システムで記録した人物が環境中のどこにいるか把握できていると思いますか？	Were you able to comprehend the recorded motion on the proposed system?
Q3	提案システムでは、視聴位置によって人物の位置の把握しやすさが変化しますか？	Did it become easier to comprehend the recorded motion on the proposed system as you moved the perspective?
Q4	普通のカメラの映像と比べて、提案システムで視聴した場合、人物の動作は把握しやすいと思いますか？	Was it easier to comprehend the recorded motion on the proposed system, than on the conventional video?
Q5	普通のカメラの映像と比べて、提案システムで生成した画像の質は満足できますか？	Were you satisfied by the quality of the image generated by the proposed system, compared to the conventional video?
Q6	実際の人物の動作と普通のビデオで記録された人物の動作は比較しやすいと思いましたか？	Were you easily able to compare the motions of the real person with the recorded motion on the conventional video?
Q7	実際の人物の動作と提案システムで記録された人物の動作は比較しやすいと思いましたか？	Were you easily able to compare the motions of the real person with the recorded motion on the proposed system?
Q8	普通のカメラで記録された映像に比べて、提案システムでのパフォーマンスの視聴は楽しいと思いますか？	Is the proposed system more fun to use than conventional video?
Q9	提案システムはどのようなアプリケーションに有用だと思いますか、特定の動作の習得？	Do you think that this system would be useful for learning specific motions?
Q10	提案システムはどのようなアプリケーションに有用だと思いますか、パフォーマンスの視聴？	Do you think that this system would be useful for watching performances?

The user study was in three parts. During Part 1, participants were asked to watch the 2D conventional video first, and then the reenactment generated by the proposed system. Both were presented on the Microsoft Surface's display. The participant was allowed to move around while watching the proposed system's reenactment. This part investigates whether the proposed systems aids user comprehension of 3D motion. During Part 2, a real person attempted to copy the motion of the recorded subject. The participants first simultaneously watched the conventional video and the motions of the real person; afterward, they simultaneously watched the reenactment and the motions of the real person. Participants were again allowed to move freely while watching the reenactment. Part 2 is dedicated to show the effectiveness of the proposed system over the conventional video for learning and training of a specific motion, considering that if the participants were easily able to see the differences between an expert and a learner, this would imply an easier learning process. The questions in Part 3 were designed to show the applicability of the proposed system. After watching the video, they were asked to answer the questions in Table 1. The answers were multiple choice and free entry. The multiple choice answers were on a scale of 1, I strongly think not, to 5, I strongly think so.

5.2.2 Results

Figure 27 shows the participants' answers. For part 1, the answers show that the proposed system did not improve much from the conventional video. In terms of pure comprehension, the results of Q2 over Q1 shows that the system improved slightly. Almost all of the participants agreed that viewing the reenactment from a different viewpoint improved comprehension (Q3). Q4 was split almost in half, meaning that half of the participants rated the proposed system as more comprehensible than conventional video. And for Q5, the participants unanimously answered in the negative, meaning that nobody was satisfied with the quality of the reenactment. Some comments from the participants were that the output was jittery, that the sillhouette wasn't smooth, and that if they moved drastically, the image would become corrupted.

In comparisons with a real person for part 2, the proposed system improved slightly over conventional video (Q6 and Q7). A common comment was that it

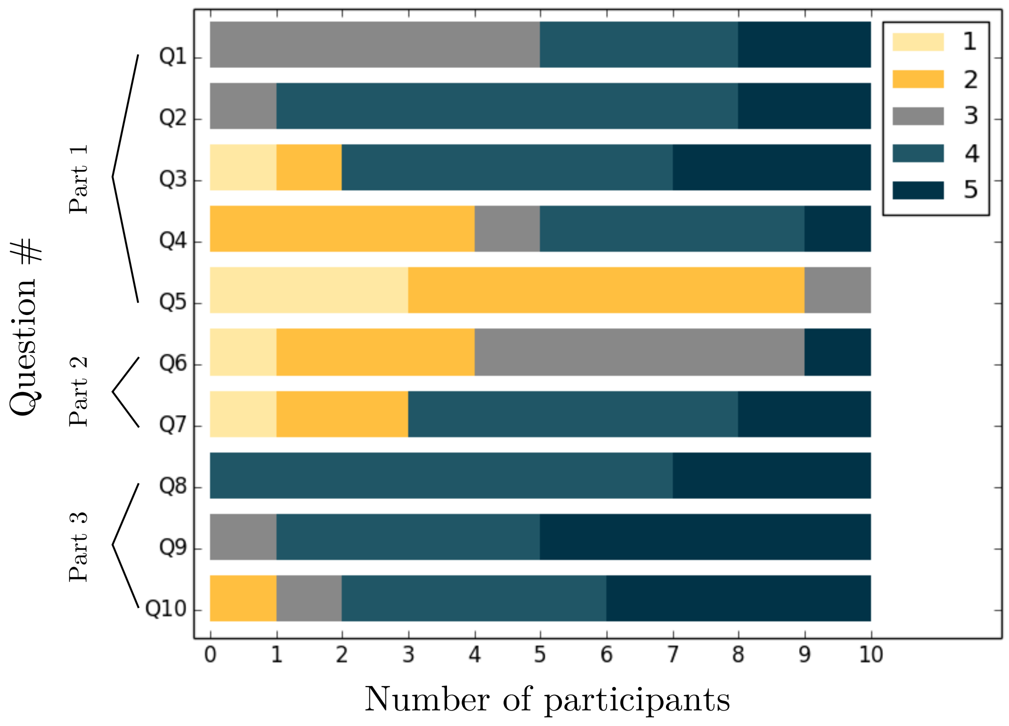


Figure 27: Evaluation results for questions our user study. 1 means strong disagreement and 5 means strong agreement.

was easier to compare the motions of the real human with the reenactment.

Finally, in part 3, most of the participants thought that the system had good applicability, from Q8–Q10 as well as the comments.

6. Conclusion

In this work, we introduce the concept of reenactments for the purpose of learning. To realize the reenactments, we developed a system that can synthesize a free-viewpoint image of a moving subject using a single RGB-D sensor.

We conducted a user study to gauge the effectiveness of the system for the purpose of learning a motion. Our results showed that the system was comprehensible when viewed from viewpoints close to the capturing viewpoint. In these cases, the reenactments were able to synthesize the subject's appearance and motion close to what was expected from viewing the RGB video. Another strength of the system was making comparisons in real time. The reenactment and the real world appear on the same screen, making it trivial for viewers to see the difference. This is a direct benefit over conventional video, where viewers must switch their attention between the screen and the real world.

On the other hand, the system has some limitations. The reenacted motion is limited to what the system can see, meaning that any motions that the subject does that the sensor does not capture can not be reenacted. Adding to this, the appearance of the subject is drawn from the actual frames captured. This means that if, for example, the captured frames do not include a particular viewpoint, the output may suffer when that viewpoint is requested.

Also, the reenactments also contained some rendering errors. The further away from the original viewpoint, the more errors appeared. Since the motivation of the system was to increase comprehensibility at viewpoints other than the original, this is an undesirable result. We identified some possible causes. First, the texture selection may be choosing inappropriate textures. We must redefine the criteria for an appropriate texture. In our current implementation, the two skeletons being compared must simply be similar distance-wise. However, there may exist a definition that gives better results. Next, the original RGB frames are badly affected by self-occlusion. Currently, we use each RGB frame as-is, meaning we do not perform any processing before using them as textures. This means that the cylinder projection for texture extraction simply extracts the whole cylinder shape. We need to isolate each body part from its neighbors and fill in the gaps in order to solve this problem. Lastly, the skeleton tracker sometimes fails to track the correct position of the joints. This happens when the motion is too fast, and sometimes when the subject's back is to the camera. However, filtering or correcting for its output may improve results.

The system also has the potential to grow in other ways. One way is to replace the RGB-D sensor used for capturing with an RGB camera, and to do skeleton tracking on the RGB frames. This would increase the accessibility of the system by allowing mobile users to capture reenactments. Another way is to introduce a method to relocate reenactments. Currently, the reenactment must be rendered over the originally captured environment. Naively relocating it introduces rotation and scaling errors, so the new environment must be carefully considered.

During the user study, most of those surveyed answered that the system could also be applicable to many things, e.g. performance, and this is certainly another possible avenue for future development. However, our ultimate goal is to implement the system as a social reenactment-sharing network. Users will be able to capture reenactments wherever they want, and potential viewers will be able to see if a previous user captured a reenactment close to their location.

Acknowledgements

First of all, I would like to sincerely thank Professor Naokazu Yokoya for taking me in and welcoming me to the Media and Vision Computing laboratory. His generosity gave me the chance to complete my Master's course in Japan. I am grateful for all the help and support I received from his laboratory.

I would also like to thank Professor Hirokazu Kato, Professor Tomokazu Sato, and Professor Yuta Nakashima. Thanks to them, I was able to produce better work. Although I can safely say that I deserved most of the headaches that came my way, a combination of their good ideas, direction, and some tough love, went into my research.

Professor Hiroyuki Seki was my introduction to NAIST, and he was very accomodating towards me. He made my transition here as painless as possible. He was also head of the international program that I studied under.

The members of the Media and Vision Computing laboratory welcomed me like a family. Thank you for the fruitful conversations and the good advice, and the companionship during late nights. Additional thanks go to Professor Norihiko Kawai and Takahito Aoto for meeting me at at the station when I came to NAIST and helping me take care of my affairs.

I'd also like to throw "shout-outs" to all my NAIST friends. I met many colorful characters during my stay here. You're all super cool. I wish everyone the best of luck in research (and life). Many thanks also to Max Krichenbauer and Satoshi Nakao for performing for my data set! You guys got some smooth moves.

For my Master's course I was supported by the Docomo scholarship. Thank you for this wonderful opportunity! I hope to pay it forward to the next generation of dreamers.

Maraming salamat to my family, who seem to have gone above and beyond the call of duty to support me. It feels like you were right there with me every step of the way. Thanks always for your love!

References

[1] J. Starck, G. Miller, and A. Hilton, "Video-based character animation," in *Proc. ACM SIGGRAPH Eurographics Symposium on Computer Animation*, pp. 49–58, 2005.

[2] C. Zitnick, S. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 600–608, 2004.

[3] M. Waschbüsch, S. Würmlin, D. Cotting, F. Sadlo, and M. Gross, "Scalable 3D video of dynamic scenes," *The Visual Computer*, vol. 21, no. 8-10, pp. 629–638, 2005.

[4] H. Hondori, M. Khademi, L. Dodakian, S. Cramer, and C. V. Lopes, "A spatial augmented reality rehab system for post-stroke hand rehabilitation," in *Proc. Conference on Medicine Meets Virtual Reality*, 2013.

[5] S. Tsuchida, T. Terada, and M. Tsukamoto, "A system for practicing formations in dance performance supported by self-propelled screen," in *Proc. Augmented Human International Conference*, pp. 178–185, 2013.

[6] S. Henderson and S. Feiner, "Augmented reality in the psychomotor phase of a procedural task," in *Proc. IEEE International Symposium on Mixed and Augmented Reality*, pp. 191–200, 2011.

[7] F. Anderson, T. Grossman, J. Matejka, and G. Fitzmaurice, "YouMove: Enhancing movement training with an augmented reality mirror," in *Proc. ACM Symposium on User Interface Software and Technology*, pp. 311–320, 2013.

[8] E. De Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun, "Performance capture from sparse multi-view video," in *ACM Transactions on Graphics*, vol. 27, 10 pages, 2008.

[9] J. Carranza, C. Theobalt, M. Magnor, and H. Seidel, "Free-viewpoint video of human actors," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 569–577, 2003.

[10] G. Ye, Y. Liu, Y. Deng, N. Hasler, X. Ji, Q. Dai, and C. Theobalt, "Free-viewpoint video of human actors using multiple handheld kinects," *IEEE Transactions on Cybernetics*, vol. 43, no. 5, pp. 1370–1382, 2013.

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

[12] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan, "Image-based visual hulls," in *Proc. Conference on Computer Graphics and Interactive Techniques*, pp. 369–374, 2000.

[13] S. Würmlin, E. Lamboray, O. Staadt, and M. Gross, "3D video recorder," in *Proc. Pacific Conference on Computer Graphics and Applications*, pp. 325–334, 2002.

[14] W. B. Culbertson, T. Malzbender, and G. Slabaugh, "Generalized voxel coloring," in *Proc. International Workshop on Vision Algorithms: Theory and Practice*, pp. 100–115, 2000.

[15] W. Lorensen and H. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *ACM SIGGRAPH Computer Graphics*, vol. 21, pp. 163–169, 1987.

[16] T. Matsuyama and T. Takai, "Generation, visualization, and editing of 3D video," in *Proc. International Symposium on 3D Data Processing Visualization and Transmission*, pp. 234–245, 2002.

[17] K. Müller, A. Smolic, K. Dix, P. Merkle, P. Kauff, and T. Wiegand, "View synthesis for advanced 3D video systems," *EURASIP Journal on Image and Video Processing*, 11 pages, 2009.

[18] B. Dai and X. Yang, "A low-latency 3D teleconferencing system with image based approach," in *Proc. ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, pp. 243–248, 2013.

[19] D. Alexiadis, D. Zarpalas, and P. Daras, "Real-time, full 3-D reconstruction of moving foreground objects from multiple consumer depth cameras," *IEEE Transactions on Multimedia*, vol. 15, no. 2, pp. 339–358, 2013.

[20] B. Kainz, S. Hauswiesner, G. Reitmayr, M. Steinberger, R. Grasset, L. Gruber, E. Veas, D. Kalkofen, H. Seichter, and D. Schmalstieg, "OmniKinect: Real-time dense volumetric data acquisition and applications," in *Proc. ACM Symposium on Virtual Reality Software and Technology*, pp. 25–32, 2012.

[21] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013.

[22] C. Malleson, M. Klaudiny, A. Hilton, and J. Guillemaut, "Single-view RGBD-based reconstruction of dynamic human geometry," in *Proc. IEEE Workshop on Dynamic Shape Capture and Analysis*, pp. 307–314, 2013.

[23] R. Castle, G. Klein, and D. Murray, "Video-rate localization in multiple maps for wearable augmented reality," in *Proc. IEEE International Symposium on Wearable Computers*, pp. 15–22, 2008.

[24] P. Debevec, C. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach," in *Proc. Conference on Computer Graphics and Interactive Techniques*, pp. 11–20, 1996.

[25] "OpenGL." `http://opengl.org/`, 2014. [Online; accessed January 28, 2014].

[26] R. Castle, G. Klein, and D. Murray, "Parallel Tracking and Multiple Mapping source code." `http://www.robots.ox.ac.uk/~bob/research/research_ptamm.html`, 2014. [Online; accessed January 28, 2014].

[27] PrimeSense, "OpenNI." `http://openni.org/`, 2014. [Online; accessed January 28, 2014].

[28] PrimeSense, “NiTE.” http://www.openni.org/files/nite/, 2014. [Online; accessed January 28, 2014].

[29] W. Kabsch, “A discussion of the solution for the best rotation to relate two sets of vectors,” *Acta Crystallographica Section A*, vol. 34, no. 5, pp. 827–828, 1978.