Capturing Detailed Deformations of Moving Human Bodies

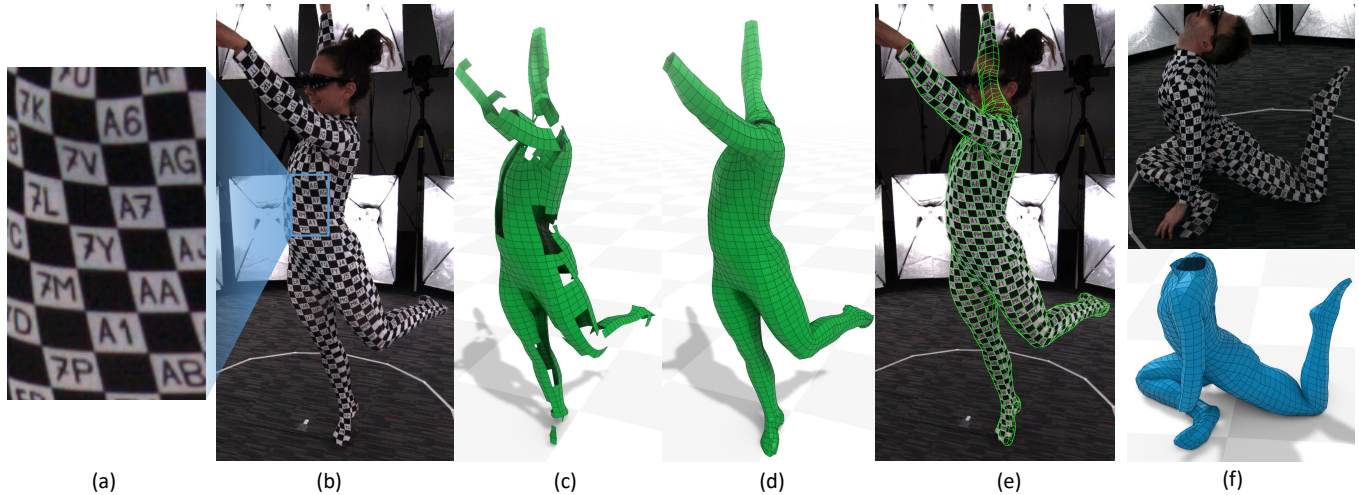HE CHEN, HYOJOON PARK, KUTAY MACIT, and LADISLAV KAVAN, University of Utah, USA

Fig. 1. (a) Our novel motion capture suit with a special pattern; (b) An example input image from our multi-camera capture system; (c) Raw 3D reconstruction of **labeled** corners from the suit (raw data, no body model was used); (d) The result after interpolating missing observations (here, a body model is used); (e) Our reconstructed mesh aligns very closely with the original input images; (f) Our method works even in uncommon poses with many self-occlusions, such as this yoga pose.

We present a new method to capture detailed human motion, sampling more than 1000 unique points on the body. Our method outputs highly accurate 4D (spatio-temporal) point coordinates and, crucially, automatically assigns a unique *label* to each of the points. The locations and unique labels of the points are inferred from individual 2D input images only, without relying on temporal tracking or any human body shape or skeletal kinematics models. Therefore, our captured point trajectories contain all of the details from the input images, including motion due to breathing, muscle contractions and flesh deformation, and are well suited to be used as training data to fit advanced models of the human body and its motion. The key idea behind our system is a new type of motion capture suit which contains a special pattern with checkerboard-like corners and two-letter codes. The images from our multi-camera system are processed by a sequence of neural networks which are trained to localize the corners and recognize the codes, while being robust to suit stretching and self-occlusions of the body. Our system relies only on standard RGB or monochrome sensors and fully passive lighting and the passive suit, making our method easy to replicate, deploy and use.

Authors' address: He Chen, ankachan92@gmail.com; Hyojoon Park, hjoonpark.us@gmail.com; Kutay Macit, ucanbizon@gmail.com; Ladislav Kavan, ladislav.kavan@gmail.com, University of Utah, Electrical and Computer Engineering 50 S. Central Campus Drive MEB Lab 3335, Salt Lake City, UT, 84112, USA.

Our experiments demonstrate highly accurate captures of a wide variety of human poses, including challenging motions such as yoga, gymnastics, or rolling on the ground.

CCS Concepts: • **Computing methodologies** → **Motion capture**.

Additional Key Words and Phrases: human animation, motion capture, skin deformation

1 INTRODUCTION

In most real-world images, the human body is occluded by clothing, making precise body measurements difficult or impossible. A significant amount of previous work focuses on approximate but robust pose estimation in the wild [Cao et al. 2018; Güler et al. 2018]. However, a small muscle twitch or the speed of breathing may contain signals that are critical in certain contexts, e.g., in the context of social interactions, minute body shape motion may reveal important information about the person's emotional state or intent [Joo et al. 2018]. Detailed human body measurements are highly relevant also in orthopedics and rehabilitation [Zhou and Hu 2008], virtual cloth try on [Giovanni et al. 2012; Ma et al. 2020], or building realistic avatars for telepresence and AR/VR [Barmpoutis 2013; Lombardi et al. 2018].

When precise measurements are needed, prior work utilized either 1) reflective markers attached to a motion capture suit or glued to the skin [Park and Hodgins 2006], or 2) painting colored patterns on the skin [Bogo et al. 2017]. The traditional reflective ("mocap") markers present certain limitations. Because all of the markers look alike (Fig. 2a), marker labeling relies strongly on temporal tracking and high frame-rate cameras. However, robust marker labeling is a hard problem [Song and Godøy 2016] which often requires manual corrections, especially for markers that have been occluded for too long. The difficulty of this problem grows with the number of markers [Park and Hodgins 2006], thus sparse marker sets are most common in the industry. Sparse marker sets are sufficient for fitting a low-dimensional skeletal body model, but not for capturing the details of flesh deformation or motion due to breathing.

To capture moving bodies with high detail, the DFAUST approach [Bogo et al. 2017] starts by geometrically registering a template body model to 3D scans [Hirshberg et al. 2012; Pons-Moll et al. 2015] and then uses colored patterns on the skin to obtain high-accuracy temporal correspondences via optical flow. These colored patterns serve a similar purpose as the checkerboard-like corners on our suit, i.e., they enable precise localization of points on the surface of the body. The key difference of our approach is that our suit contains also unique two-letter codes adjacent to each corner, allowing us to label the corners directly by recognizing the codes. This is not possible with the DFAUST's patterns, because they are self-similar, created by applying colored stamps to the skin. Instead, the DFAUST approach relies on the initial geometric registration and temporal tracking, which can suffer from error accumulation and may lead to incorrect local minima in more challenging poses or fast motions. The DFAUST dataset contains a variety of high-detailed human body animations, but is restricted to upright standing-type motions. In contrast, we demonstrate captures of a wider variety of motions, including gymnastics exercises, yoga poses or rolling on the ground, see Fig. 19 and our accompanying video and data.

Our new motion capture method was enabled by recent advances in deep learning and high-resolution camera sensors. The key idea is to use a new type of motion capture suit with special fiducial markers, consisting of checkerboard-like corners for precise localization and two-letter codes for unique labeling. Our localization and labeling process is very robust, because it does not rely on temporal tracking or any type of body model; in fact, our approach succeeds even if only a small part of the body is visible in the image, e.g., the zoom-ins in the right part of Fig. 14. A similar advantage exists also in the temporal domain. Because our localization and labeling approach can process each image independently, there are no issues due to occlusions and dis-occlusions which complicate traditional temporal tracking in both marker-based as well as marker-less methods.

Even though the automatic localization and labeling are very robust, achieving this functionality is non-trivial, because our methods need to be robust against significant stretching of the tight-fitting suit as well as projective distortion. Fortunately, checkerboard-like corners remain to be checkerboard-like even despite significant stretching of the suit. We apply three convolutional neural networks combined with geometric algorithms. The first of our convolutional neural networks (CNNs) is a corner detector which localizes all of the corners in an input image (4000×2160). To rectify the distortion of the two-letter codes inside the white squares, we connect candidate four-tuples of corners into quadrilaterals (quads) and apply homography transformation which rectifies both suit stretching as well as projective distortion. Another CNN called *RejectorNet* then performs quality control and legible and upright-oriented codes. The remaining codes are passed to *RecogNet* which reads the characters in the two-letter code. Because the orientation of our codes is unique (we avoided symmetric symbols such as "O" or "I" as well as ambiguous pairs like "6" and "9"), recognition of the code allows us to uniquely label each of the adjacent corners, see Fig. 2c.

The labels of our corners establish correspondences both in time and in space, i.e., between individual cameras in our multi-view system, which means that we can easily triangulate the 2D corner locations into 3D points. However, the 3D reconstructed (triangulated) points will inevitably miss observations due to self-occlusions and the limited number of our cameras, see Fig. 1c. To fill (interpolate) these missing observations, we start by fitting the STAR model [Osman et al. 2020] and then refine it for each of our actors using a point trajectory from a calisthenics-type motion sequence captured using our method. This refinement ensures that we obtain the best possible low-dimensional model for each of our actors, since high quality is our main objective. We use this refined body model to interpolate the missing corners in the rest pose, resulting in the final mesh without any holes, see Fig. 1d.

Our goal was to make each two-letter code as small as possible, so we can recover the highest possible number of points on the body. We created our special capture suits in two sizes, one "medium" (with 1487 corners) and one "small" (with 1119 corners) and we captured three actors: one male and two females (two of the actors used the "medium" suit).

We evaluated both the geometric accuracy through reprojection error of 3D reconstruction, and the quality of the temporal correspondences by computing the optical flow between the synthetic image and the real image. Results show 99% of our reconstructed points has a reprojection error of less than 1.01 pixels, and 95% of the pixels on the optical flow have a optical flow norm of less than 1.2 pixels. In our camera setup, 1 pixel approximately converts to 1mm on a person 2 meters away from the camera.

Contributions: (1) We propose a new method to measure 3D marker locations at each frame and automatically infer corresponding **marker labels**. This is achieved without any priors on human body shapes or kinematics, which means that our data are "raw measurements", immune to any type of modeling or inductive bias.
(2) We introduce a novel type of fiducial markers and capture suit, which enables marker localization and unique labeling using only local image patches. Our approach does not utilize temporal tracking, which makes it robust to marker dis-occlusions and also invites parallel processing, because each frame can be processed independently.
(3) All results in this paper were obtained using an experimental multi-camera system utilizing 16 commodity RGB cameras and passive lighting. High-end multi-camera systems based on machine vision cameras such as those built by Facebook [Lombardi et al.

2018] or Google [Guo et al. 2019] require significant hardware investments and engineering expertise. In contrast, our system is easy to build from inexpensive off-the-shelf parts. We provide our data as supplemental material and we invite individual researchers, independent studios or makers to replicate our setup and capture new actors and motions.

2 RELATED WORK

Optical systems based on reflective markers [Menache 2000] are the most widely used approaches to capture the human body. While typically only sparse marker sets are used, [Park and Hodgins 2006, 2008] pushed the resolution of reflective markers based system up to 350 markers to capture the detailed skin deformation. However, difficulties in marker labeling [Song and Godøy 2016] complicate further increases of resolution by adding even more markers. Recent work utilizes self-similarity analysis [Aristidou et al. 2018] and deep learning [Han et al. 2018; Holden 2018] to reduce the expensive manual clean-up in marker labeling procedure. An alternative to the classical reflective markers is the use of colored cloth, enabling the capture of certain types of garments [Scholz et al. 2005; White et al. 2007] or hand tracking using colored gloves [Wang and Popović 2009].

Early work in markerless motion capture [Gavrila and Davis 1996] and [Bregler et al. 2004] inferred human poses directly from 2D images or videos. [Kehl and Van Gool 2006] integrates multiple image cues such as edges, color information and volumetric reconstruction to achieve higher accuracy. [Brox et al. 2009] tracks a 3D human body on 2D image by combining image segments, optical flows and SIFT features [Lowe 1999]. [De Aguiar et al. 2008] deforms laser-scan of the tracked subject under the constraints of multi-view videos to capture spatio-temporally coherent body deformation and textural surface appearance of the actors. Silhouettes [Liu et al. 2013; Sand et al. 2003; Starck and Hilton 2007; Vlasic et al. 2008] or visual hulls [Corazza et al. 2010] can be used to obtain more detailed human body deformations. [Gall et al. 2010] introduce a multi-layer framework that combines stochastic optimization, filtering, and local optimization to tack 3D human poses. [Stoll et al. 2011] model the human body via a sums of Gaussians, representing both shape and appearance of the captured actors.

Deep learning enabled estimation of 2D or 3D human poses from multi-view [Pavlakos et al. 2017] or monocular multi-person images [Newell et al. 2016; Pishchulin et al. 2016; Raaj et al. 2019; Wei et al. 2016], more recently also with hands and faces [Cao et al. 2018, 2017; Hidalgo et al. 2019]. 3D pose or even dense 3D surface of the human body can also be predicted from a single image [Choutas et al. 2020; Güler et al. 2018; Mehta et al. 2017; Xiang et al. 2019; Xu et al. 2019]. Morphable human models can be learned from multi-person datasets [Pons-Moll et al. 2015; Robinette et al. 2002]. Models such as SCAPE [Anguelov et al. 2005], SMPL [Loper et al. 2015] and STAR [Osman et al. 2020] focus on the body, while models such as Adam [Joo et al. 2018] and SMPL-X [Pavlakos et al. 2019] also include the face and the hands. Focusing on high-quality rendering rather than geometry, [Guo et al. 2019; Meka et al. 2020] proposed methods for photo-realistic relighting of moving humans, including clothing and accessories such as backpacks.

The idea of a motion capture suit with special texture is related to fiducial markers used e.g., in robotics or augmented reality, such as ARTag [Fiala 2005], AprilTag [Olson 2011; Wang and Olson 2016], ArUco [Garrido-Jurado et al. 2014] and many others, but these fiducial markers are typically assumed to be non-deforming. They are also not easy to read for humans, which would complicate their annotations. The localization of our fiducial marker is related to corner detection. Many corner detection methods have been developed to meet different use-case scenarios. For localizing the corners, there are methods that are designed to detect general corners features that naturally exists in nature, like [DeTone et al. 2018; Rosten and Drummond 2006], Another class of corner detectors focuses on rigid calibration checkerboards [Bennett and Lasenby 2014; Chen et al. 2018; Donné et al. 2016; Hu et al. 2019], particularly useful in camera calibration. Because these methods assume the checkerboard pattern to be rigid, they will not work on our checkerboard-like suit which can have significant deformations (Fig. 2b). The code recognition component of our method is related to text recognition. As discussed in [Long et al. 2020], the text recognizer generally performs poorly on text with large spatial transformations. One possible solution is based on generating region proposals [Jaderberg et al. 2014; Ma et al. 2018] to rectify the spatial transformation.

High-resolution temporal correspondences can be obtained by registering a template mesh to RGB-D images or 3D scans. The registration can be based on solely geometric information [Allain et al. 2015; Li et al. 2009], or combined with RGB images to align [Bogo et al. 2015] to reduce the tangential sliding. Model-less approaches are also possible [Collet et al. 2015; Dou et al. 2015; Newcombe et al. 2015]. Those methods focus on registering sequential motions frame to frame, with the assumption of small displacements between subsequent frames. Therefore, they can suffer from error accumulation, resulting in drift over time [Casas et al. 2012]. Aligning non-sequential motions is also possible [Huang et al. 2011; Tung and Matsuyama 2010], but it is challenging to establish correspondence between very different poses [Boukhayma et al. 2016; Prada et al. 2016]. Deformation models can be trained from 3D scans [Allen et al. 2003; Anguelov et al. 2005], with non-rigid scan registration being the technical challenge [Hirshberg et al. 2012].

Similarly to our new motion capture suit, the FAUST [Bogo et al. 2014] and DFAUST [Bogo et al. 2017] methods paint high-frequency colored patterns directly to the skin. We chose to work with a suit because putting it on and off is easy and fast compared to applying colored stamps and washing them off after the capture session. Even though we only experimented with basic tight-fitting suits in this paper, future improvements such as adhesive suits or non-permanent tattoos are possible, see Section 7. Our capture system is significantly simpler and less expensive: we use only on 16 standard (RGB) cameras with passive uniform lights, while [Bogo et al. 2014, 2017] used 22 pairs of stereo cameras, 22 RGB cameras and 34 speckle projectors (active light). Perhaps more important are the technical differences between our approach and DFAUST, in particular the fact that our codes are unique as opposed to the self-repetitive patterns used in FAUST and DFAUST. Rather than creating a dataset, our goal was to create a universal and practical method to enable future research on advanced human body modeling and its applications in areas ranging from graphics to sports medicine.
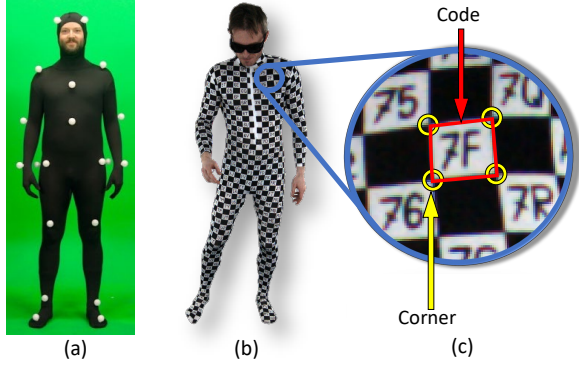
Fig. 2. (a) A classical motion capture suit with reflective markers. (b) The design of our motion capture suit with fiducial markers. (c) Each or our markers consists of checkerboard-like corners and codes.
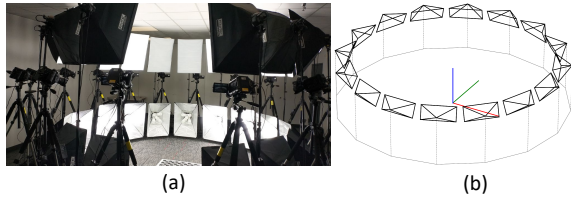

Fig. 3. (a) A photo of our 16 camera setup. (b) The cameras form a circle surrounding the capture volume.

3 3D RECONSTRUCTION OF LABELED POINTS

Suit. To create our special motion capture suit, we started by purchasing a tight-fitting unitard, originally intended for dance or performing arts. Fortuitously, one of the manufacturer-provided patterns was precisely the black-and-white checkerboard texture reminiscent of computer vision calibration boards (in fact this provided some of the original inspiration for this project). We purchased two suits, one "medium" and one "small" and augmented them by writing codes into the white squares using a marker pen. The medium suit contains 1487 corners and 625 two-letter codes; the small suit has 1119 corners and 456 codes. For our two-letter codes, we only used symbols whose upright orientation is unique and non-ambiguous, specifically: "1234567ABCDEFGJKLMPQRTUVY".

An alternative to hand-drawn suits is to use a cloth printing service, capable of printing an arbitrary high-resolution image on textile. However, this approach is more complex because it requires us to design sewing patterns, ship them to the printing service, then cut and sew the printed textile into a suit. We have explored this approach, using a computer font that is quite different from our handwriting, see Fig. 5c,d. This experimental suit contains 1473 corners and 618 two-letter codes, but it contains large black areas due to suboptimal sewing patterns.

Camera system. Our multicamera setup contains 16 standard (RGB) cameras arranged into a circle surrounding the capture volume (Fig. 3b). Each camera captures 4000×2160 images at 30 FPS in RAW format. The camera shutters are synchronized via genlock with negligible synchronization error, which means that human

motion is captured as if "frozen" in time. Surrounding the capture volume, we positioned 32 softboxes that generate uniform diffuse light. The bright light allows us to use a small aperture and very fast 0.5*ms* shutter speeds, guaranteeing sharp images even with the fastest human motions. The cameras are calibrated by waving a traditional calibration checkerboard in from of them. The intrinsic and extrinsic camera parameters are calibrated using the well-established method [Zhang 2000] for which we use OpenCV's checkerboard corner detector for rigid calibration boards [Bradski 2000]. Next, the camera parameters and the 3D checkerboard corner positions in the world coordinates are further refined using bundle adjustment [Triggs et al. 1999]. We use the Levenberg-Marquardt algorithm and the Ceres library [Agarwal and Mierle 2012].

Image processing pipeline. The calibrated cameras generate sequences of images, which are processed by our pipeline outlined in Fig. 4. We start by detecting checkerboard-like corners in the input image with sub-pixel accuracy (Fig. 4a, Section 3.1). Next, we need to uniquely label the detected corners by recognizing the adjacent two-letter codes. Because the codes are written in the white squares surrounded by four corners, we generate candidate quadrilaterals (quads) by connecting four-tuples of corners. Only a few four-tuples of corners correspond to the white squares, but it is okay to generate a quad that does not correspond to a white square, because it will be discarded later; hence we use the term "candidate quads", see Fig. 4b. Since the quads are generated by connecting four corners, we naturally have correspondences between the corners and the quads. The candidate quads are rectified by mapping them into a regular square using homography (Fig. 4c, Section 3.2) to remove suit stretching and perspective distortion, and then passed as input to *RejectorNet* which performs quality control and checks whether the quad actually corresponds to a white square with a code (Fig. 4d). The *RejectorNet* also ensures the correct upright orientation of the code. The images accepted by *RejectorNet* are then passed to *RecogNet*, which reads the two-letter code that finally enables us to uniquely label each corner (Fig. 4e).

We would like to point out that our method is local by design, i.e., each stage of the pipeline works with small patches of the input image. This gives us a several advantages: *a)* Our method is capable of extracting reliable geometric information of the human body and - crucially - correspondences even from a small patch of the suit. This makes our method very robust to occlusions or partial views of the human body e.g. due to zoomed-in cameras. *b)* By decomposing the suit into small quads and undistorting them using homography, we can counteract much of the projective distortions and suit stretching (see Fig.4c), simplifying the learning task. *c)* The CNN quad classifier includes a quality control mechanism, rejecting white squares with dubious quality and further improving the robustness of our method.

3.1 Corner Detector

The corner detector's task is to detect and localize all checkerboard-like corners in the input image. This task is non-trivial because there are corner-like features in the background, the suit stretches along with the skin and there are significant lighting variations.

Our corners have two key properties: *a)* The corners are sparsely and approximately uniformly distributed on the suit; *b)* The corners
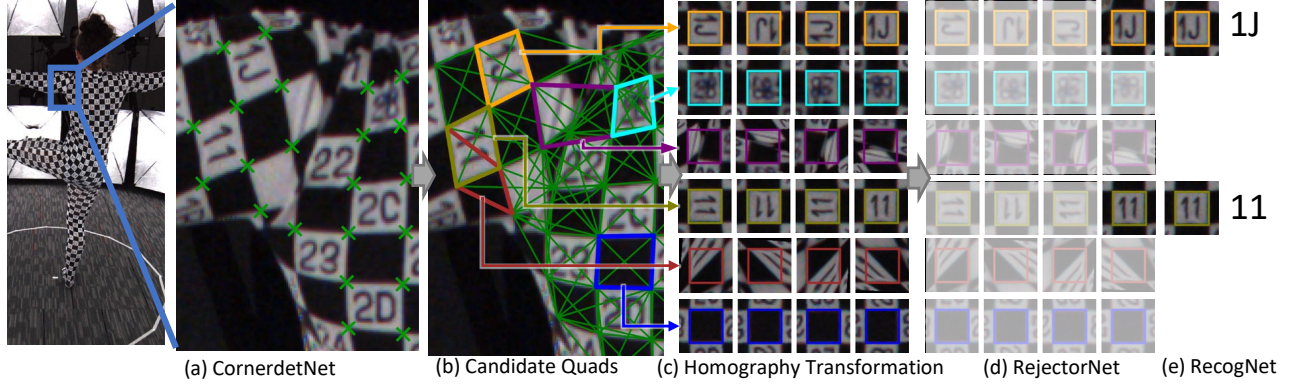
Fig. 4. Corner localization and labeling pipeline: (a) *CornerdetNet* CNN detects and localizes our suit corners; (b) Four-tuples of corners are connected into candidate quads; (c) homography transformations using all four possible orientations (we selected a few quads and outlined them with different colors for illustration purposes); (d) *RejectorNet* CNN: a binary classifier accepting only valid white squares with upright oriented two-letter codes; (e) *RecogNet* CNN recognizes the two characters in the valid codes.
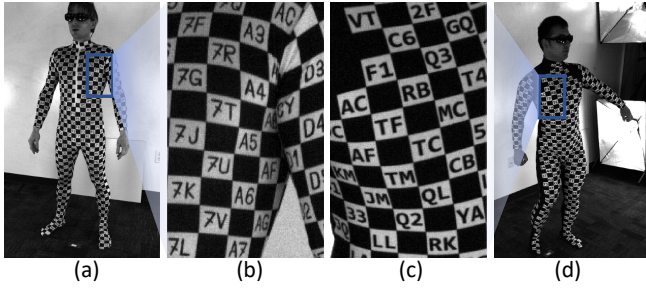

Fig. 5. Comparison of (a, b) hand-drawn and (c, d) printed suits.

are defined locally, i.e., a small image patch is enough to identify and localize a corner. We divide our input image into a grid of 8×8 cells with the assumption that there could be at most one checkerboard-like corner in each 8×8 cell, and apply *CornerdetNet* CNN to detect and localize a checkerboard-like corner from each cell separately. The design of *CornerdetNet* is inspired by single shot detectors [Liu et al. 2016; Redmon and Farhadi 2017], which perform prediction and localization simultaneously. Please see Supplementary Material A.1 for more details about *CornerdetNet*.

3.2 Corner Labeling and 3D Reconstruction

At this point we have detected typically several hundreds of corners in each input image. The next step is to read the codes and link them to the corners, which will give us a unique label for each corner. As shown in Fig. 1, the deformations of the suit and its codes can be significant, including not only projective transformations, but also stretching and shearing because the tight-fitting suit is highly elastic. First we have to detect the white squares with two letter codes. We know that each such white square is surrounded by four corners. Therefore, we generate quadrilaterals (quads) by connecting four-tuples of corners. In theory we could connect any four-tuples of corners into a quad, but in practice we can immediately discard concave quads (which do not correspond to correct sequences of corners) or quads that would cover too few or too many pixels (which would make it impossible to contain a legible code). We call
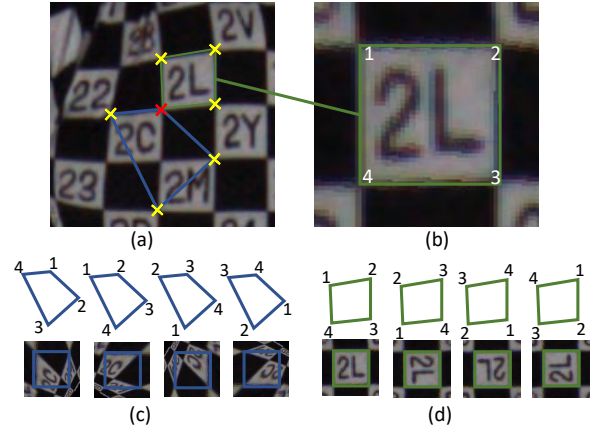

Fig. 6. Quad generation: (a) For a given corner (red), we find three other corners (yellow) within a bounding box and compute their convex hull to generate a candidate quad. Here we visualize two example candidate quads (green and blue); (b) Our corner numbering convention with respect to upright code orientation; (c) and (d): The four possible orientations of the candidate quads, and their corresponding homography transformations. (c) is an invalid candidate, but the first orientation in (d) is valid.

the resulting quads "candidate quads", because they may - but are not guaranteed to - contain a correct two-letter code. We transform the four corners of each candidate quad to a standardized square using a homography transformation to simplify subsequent processing. The standardized square is a 64×64 pixel image with 20 pixel margin on each side, i.e., 104×104 total, see Fig. 6. The 20 pixel margin allows the *RejectorNet* to detect errors stemming from incorrect corner detections. Since we do not know the correct upright orientation of our two-letter code yet, we generate all four possible orientations, see Fig. 6c,d. Fig. 6c shows an example of an invalid quad and Fig. 6d demonstrates a valid one.

Quad classifiers. We trained two quad classifiers, *RejectorNet* and *RecogNet*. *RejectorNet* is a binary classifier predicting whether a candidate quad is valid, i.e., whether the four corners are at the
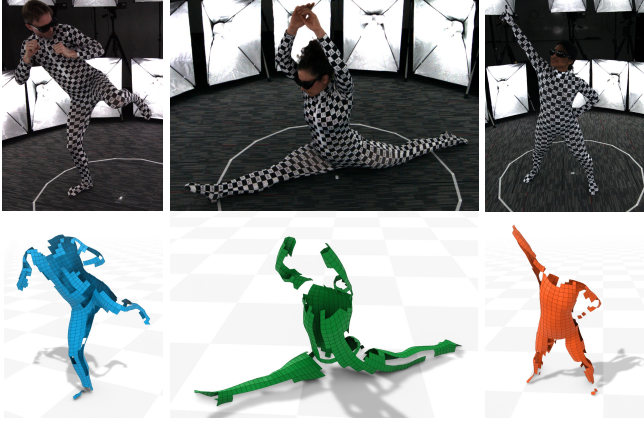
Fig. 7. Example input images (top) from our multi-camera system and the corresponding raw 3D reconstructions (bottom). The reconstructed 3D points are meshed according to the patterns in our suits.

correct locations and their order is correct relative to the upright code orientation, see Fig. 6b. Also, the white square surrounded by a valid quad needs to contain a clearly legible code. Invalid quads are discarded, and the valid ones are passed to *RecogNet* which reads the codes, such as "2L" in Fig. 6b. *RecogNet* is a multi-class classifier with two heads, one for each character of the two letter code. The details of candidate quad generation and quad classifiers can be found in Supplementary Material A.2. We use standard cross entropy losses to train those classifiers. The training of our CNNs is discussed in Section 6.

Corner labeling and 3D Reconstruction. At this point, the two-letter codes of the valid quads have been recognized, including their upright orientation. The next step is to uniquely label each corner. We define a labeling function $l(code, i_q)$ which maps a two-letter *code* and corner index $i_q \in \{1, 2, 3, 4\}$ (see Fig. 6b) to an integer which represents a unique corner ID. The unique corner IDs are defined for each suit. Many corners have *two* two-letter codes adjacent to them. If both of the two-letter codes are visible, we can leverage this fact as a redundancy check, detecting potential errors of *RecogNet*. Given unique corner IDs, we can convert corresponding 2D corners in more than two views into labeled 3D points. Let C_i be the set of cameras that see corner i, $k \in C_i$ is a camera that sees corner i, $\mathbf{c}_i^k \in \mathbb{R}^2$ be the corner i's location in image coordinate system of camera k, and $f^k : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ is the projection function of camera k. We computed 3D reconstructed corner $\mathbf{p}_i$ by minimizing the reprojection error:

$$\mathbf{p}_i = \arg\min_{\mathbf{p}_i \in \mathbb{R}^3} \sum_{k \in C_i} ||f^k(\mathbf{p}_i) - \mathbf{c}_i^k||_2^2 \qquad (1)$$

This is a non-linear least square optimization problem; we compute an initial guess of $\mathbf{p}_i$ using the Linear-LS method [Hartley and Sturm 1997] and optimize it using non-linear least squares solver [Agarwal and Mierle 2012].

Error Filtering. The label consistency check discussed above works only if two adjacent two-letter codes are present in the suit and

visible in the images. If this is not the case, a corner can be assigned the wrong label if *RejectorNet* or *RecogNet* make a mistake. This kind of labeling errors will typically result in non-sensical correspondences with large reprojection errors, which we detect and correct by a RANSAC-type method discussed below. Specifically, for a corner with label i, let C_i be the set of the cameras that claim to see this corner. We assume that outliers in C_i, i.e., the cameras that mislabeled corner i, should only be a minority. We iterate over all pairs of cameras (j, k) in C_i, and 3D reconstruct the corner i from each pair. Among all of the pairs, we pick the 3D reconstruction that has the lowest reprojection error averaged over all cameras in C_i and assume this is the correct 3D location $\mathbf{p}_i$. Next, we analyze the reprojection errors of $\mathbf{p}_i$ into all of the cameras C_i. The reprojection error should be low in cameras with correct labeling, but high if there was a labeling error. We use the $1.5 \times IQR$ (interquartile range) rule [Upton and Cook 1996] to detect the outliers in terms of reprojections errors. We re-compute the triangulation of $\mathbf{p}_i$ after removing the outliers from the cameras C_i. This RANSAC-type outlier filter does not work when there are only two cameras that see one corner. Therefore, we additionally discard reconstructed corners with an average reprojection error larger than 1.5 pixels. These tests are designed to be conservative, because mistakenly discarded points are not a major problem, just missing observations which can be inpainted as discussed in Section 5.

4 DATA ACQUISITION AND NEURAL NETWORK TRAINING

A key feature of our approach is that all of our networks are trained only on small image patches, e.g., see Fig. 6c,d. This allows our trained model to generalize to different suits, capture environments, camera configurations and body poses that are not in the training set, because our local fiducial markers exhibit significantly less variability than images of full human poses. This is quite different from deep-learning based methods that perform global pose-prediction, looking for the body as a whole. The training of our networks does not require large training sets. We have prepared our training data ourselves, without the use of any external annotation services or existing data sets. Our dataset contains 28 manually annotated images, 24 of them are randomly selected from captures of our three actors wearing the hand-drawn suits. We also captured a different (fourth) actor wearing the printed suit and annotated four images of it in order to evaluate our approach on printed characters as opposed to hand-drawn. The data from the printed suit will only be used for testing the performance of CNNs. For each image, we apply two types of annotations: corner annotation (Fig. 8a,b) and quad annotation (Fig. 8c,d). In the corner annotation, we manually annotate all of our checkerboard-like corners on the suit with sub-pixel accuracy. In the quad annotation, we manually connect the corners annotated in the previous step into quads. Specifically, we create quads that correspond to valid white squares with two-letter codes in the suit and the annotators also write down the code of each annotated quad. We ensure the quad vertices are in a clockwise order and start from the top-left corner, defined by the upright orientation of the code (see Fig. 6b). It takes about two person-hours to annotate one image. These annotations are then automatically converted into training
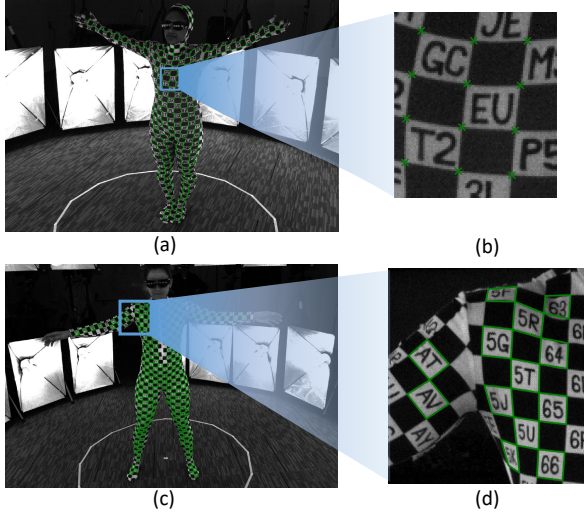
Fig. 8. Two types of annotations we applied: (a, b): corner annoatation; (c, d): quad annotation.

data for our networks using techniques described in Supplementary Material B.1.

We apply data augmentation to our training data through geometric and color space operations. We also generate synthetic training data by rendering textured human body models. The details can be found in Supplementary Material B.2.

5 FILLING MISSING OBSERVATIONS WITH REFINED BODY MODEL

Our method for 3D reconstruction of labeled points will inevitably result in missing observations because the human body often self-occludes itself and is observed only by a limited number of cameras, see Fig. 7. In this section, we propose a method to interpolate (inpaint) the missing corners. Even though we could use any existing multi-person human body model [Loper et al. 2015; Osman et al. 2020] for this purpose, we can achieve higher quality, because our pipeline gives us highly accurate measurements of the actor's body and its deformations. Therefore, instead of relying on previous statistical body shape models, we capture example motions of a given actor using our method and use this data to create a more precise *refined body model*, i.e., a model with parameters refined for a specific person.

Our body model has two types of parameters: shape parameters that are invariant in time, and pose parameters that change from frame to frame as the body moves. The shape parameters are only optimized during the model refinement process. After the body model refinement process is done, we fix the shape parameters and only allow the pose parameters to change. However, even after the refinement, the low-dimensional body model will not fit the 3D reconstructed corners exactly (Fig. 10c). We call the remaining residuals "non-articulated displacements", because they correspond to motion that is not well explained by the articulated body model. The non-articulated displacements arise due to breathing, muscle activations, flesh deformation, etc. Therefore, in addition to our refined

body model we also interpolate the non-articulated displacements mapped to the rest pose via inverse skinning. The combination of the refined body model with the non-articualted displacement interpolation enables us to achieve high quality inpainting.

5.1 Actor-Specific Model Optimization

Our body model is based on linear blend skinning (LBS) [Magnenat-Thalmann et al. 1988]. Let $\mathbf{v}_i \in \mathbb{R}^4$ for $i = 1, 2, \ldots, N$ be the deformed vertices in homogeneous coordinates, where N is the number of all the vertices on our body model. We denote the skinning model as: $\mathbf{v}_i = D_i(\tilde{\mathbf{v}}_i, \mathbf{J}, \mathbf{W}, \theta)$, where $\tilde{\mathbf{v}}_i \in \tilde{\mathbf{V}} = (\tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_N)$, $\tilde{\mathbf{V}}$ are rest pose vertex positions, $\mathbf{J} = (\mathbf{j}_1, \ldots, \mathbf{j}_M)$ are joint positions and M is the number of joints in our model; $\mathbf{W} \in \mathbb{R}^{N \times M}$ is the matrix of skinning weights and $\theta \in \mathbb{R}^{M \times 4}$ are joint rotations represented by quaternions. In summary, $\tilde{\mathbf{V}}, \mathbf{J}$ and $\mathbf{W}$ are shape parameters (constant in time) and θ are pose parameters (varying in time). The deformed vertices can be computed as:

$$\mathbf{v}_i = D_i(\tilde{\mathbf{v}}_i, \mathbf{J}, \mathbf{W}, \theta) = \sum_{j=1}^{M} w_{i,j} T_j(\theta, \mathbf{J}) \tilde{\mathbf{v}}_i \qquad (2)$$

Note that here $\mathbf{v}_i, \tilde{\mathbf{v}}_i \in \mathbb{R}^{4 \times 1}$ are the deformed and rest pose vertex in homogeneous coordinates and $T_j(\theta, \mathbf{J}) \in \mathbb{R}^{4 \times 4}$ represents the transformation matrix of joint j. In the following we will use homogeneous coordinates interchangably with their 3D Cartesian counterparts.

Initialization. We initialize our body model by registering our corners to the STAR model [Osman et al. 2020]. We start by selecting a frame f_{init} in a rest-like pose where most corners are visible, and fit the STAR model to our labeled 3D points in f_{init} using a non-rigid ICP scheme which finds correspondences between our suit corners and the STAR model's mesh. The non-rigid ICP process is initialized by 10 to 20 hand picked correspondences between the STAR model and the 3D reconstructed corners. During the ICP procedure, We optimize both pose and shape parameters of the STAR model and iteratively update correspondences by projecting each of our 3D reconstructed points to the closest triangle of the STAR model (the actual closest point is represented using barycentric coordinates).

In this stage, we have registered most of our corners to the STAR model, but we still need to add corners that were unobserved in frame f_{init}. We can fit the STAR model to subsequent frames of our training motion using non-rigid ICP initialized by the registered corners instead of hand picked correspondences. These subsequent frames will reveal corners unobserved in the initial frame, which we register against the STAR mesh by closest-point projection as before. We use the corners registered to the STAR model's rest pose as the initial rest pose shape $\tilde{\mathbf{V}}^0$, and use barycentric interpolation to generate the initial skinning weights $\mathbf{W}^0$. Note that the number of vertices and mesh connectivity of our body model is different from the STAR model's mesh. We use each corner on the suit as a vertex of our model, and the rest pose vertex $\tilde{\mathbf{v}}_i$ corresponds to corner i in our suit. The meshing of our body model is discussed below. We use the STAR model's joints as the initial joint location $\mathbf{J}^0$. We removed the joints that controls head, neck, toes and palm

from the STAR model, resulting in $M = 16$ joints. We call this model our *initial body model*.

Model refinement. After the initialization, we further optimize the shape parameters to obtain our refined body model that more accurately fits a specific actor. Specifically, we optimize the skinning weights $\mathbf{W}$, the joint locations $\mathbf{J}$ and the rest pose vertex positions $\tilde{\mathbf{V}}$. Unlike *SMPL or STAR, we do not use pose-corrective blend shapes and instead correct the shape by interpolating non-articulated displacements, discussed in Section 5.2.

If $P^k, k = 1, 2, \ldots, K$ is the set of 3D points that were reconstructed from frame k and K is the number of frames in the training set, we refine the body model by minimizing:

$$\mathcal{L}_A(\tilde{\mathbf{V}}, \mathbf{J}, \mathbf{W}, \Theta) = \mathcal{L}_f(\tilde{\mathbf{V}}, \mathbf{J}, \mathbf{W}, \Theta) + \lambda_g \mathcal{L}_g(\mathbf{W}) + \lambda_J \mathcal{L}_J(\mathbf{J}) \quad (3)$$

where $\Theta = (\theta_1, \theta_2, \ldots, \theta_K)$ are the pose parameters of all the frames in the training set and $\mathcal{L}_f(\tilde{\mathbf{V}}, \mathbf{J}, \mathbf{W}, \Theta)$ is the fitting error term:

$$\mathcal{L}_f(\tilde{\mathbf{V}}, \mathbf{J}, \mathbf{W}, \Theta) = \frac{1}{\sum_{k=1}^{K} |P^k|} \sum_{k=1}^{K} \sum_{\mathbf{p}_i^k \in P^k} \|D_i(\tilde{\mathbf{v}}_i, \mathbf{J}, \mathbf{W}, \theta_k) - \mathbf{p}_i^k\|_2^2 \quad (4)$$

$\mathcal{L}_J(\mathbf{J})$ is an l_2 loss penalizing joint locations moving too far away from their initial positions and $\mathcal{L}_g(W)$ is a regularization term encouraging sparsity of the skinning weights:

$$\mathcal{L}_g(\mathbf{W}) = \sum_{i=1}^{N} \sum_{j=1}^{M} g_{i,j} w_{i,j}^2 \quad (5)$$

where $g_{i,j}$ is the geodesic distance from corner i to the closest vertex that has non-zero initial weight for joint $\mathbf{j}_j$ in the STAR model. The regularization weights were empirically set to $\lambda_g = 1000$ and $\lambda_J = 1$ when our spatial units are millimeters.

We optimize $\mathcal{L}_A$ with an alternating optimization scheme. Starting with the initial LBS model, we first calculate pose parameters θ_k for each frame. Then we optimize $\mathbf{W}$, $\mathbf{J}$, and $\tilde{\mathbf{V}}$ one by one, while keeping the other parameters fixed. We iterate this procedure until the error decrease becomes negligible; in our results we needed between 50-100 iterations.

After the optimization is finished, we mesh the rest pose vertices $\tilde{\mathbf{V}}$. From the unique ID of each corner, we know how they were connected into quads in the suit. We manually add vertices to close the holes which come from areas of the suit such as the zipper and the seams (see Fig. 9a). The result is a quad-dominant mesh (Fig. 9b).

5.2 Point Cloud Completion

After the optimization, the fitting error (Eq. 4) will drop from 13.5mm to 7.1mm on the test set; further results are reported in Section 6.4. The refined LBS body model is good for representing articulated skeletal motion of the actor's body, but it does not represent well effects such as breathing or flesh deformation. However, the non-articulated component of the motion that cannot be represented by LBS is relatively small. Therefore, we start by applying inverse skinning transformations (also known as "unposing") to our observed 3D reconstructed points $\mathbf{p_i}$, see Fig. 10d. We denote the inverse skinning of point i at pose k as $D_i^{-1}(\mathbf{p}_i, \mathbf{J}, \mathbf{W}, \theta_k)$. As can be seen in Fig. 10d, the $D_i^{-1}(\mathbf{p}_i, \mathbf{J}, \mathbf{W}, \theta_k)$ will not exactly match $\tilde{\mathbf{v}}_i$ due to the
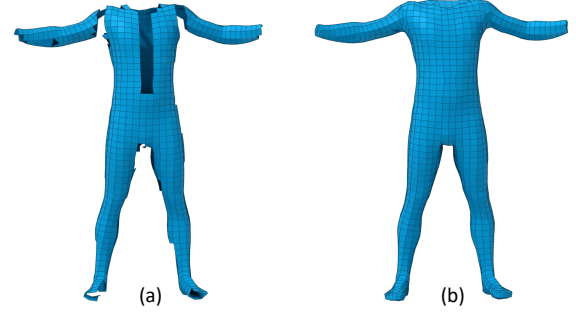
Fig. 9. (a) The quad structure corresponding to our suit has holes due to the zipper and the seams; (b) The completed rest pose mesh.
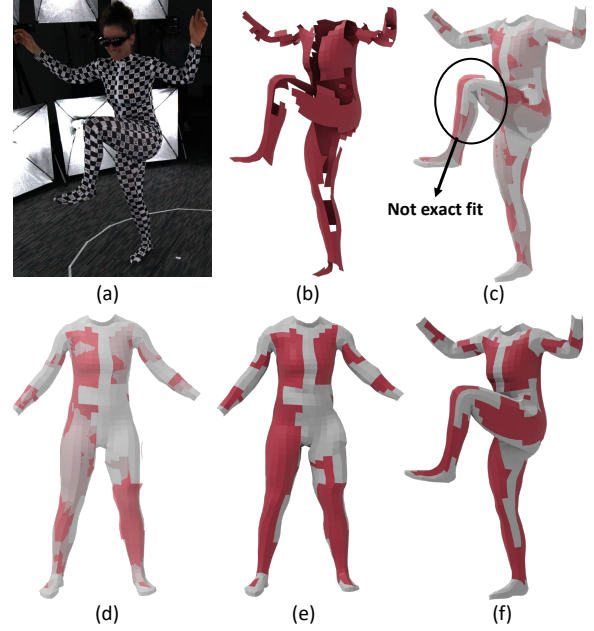

Fig. 10. Our hole-filling pipeline: (a) an input image; (b) 3D points P reconstructed from input images and connected with quads; (c) our refined body model (gray) does not fit P exactly (transparent rendering shows discrepancies); (d) inverse skinning, mapping P to the rest pose; (e) rest pose mesh interpolation, matching the inverse-skinned P exactly; (f) the final result obtained by forward skinning of the interpolated rest pose mesh.

non-articulated residuals. Formally, the non-articulated displacements $\Delta\tilde{\mathbf{v}}_i^k$ are defined as:

$$\mathbf{v}_i^k = D(\tilde{\mathbf{v}}_i + \Delta\tilde{\mathbf{v}}_i^k, \mathbf{J}, \mathbf{W}, \theta_k) \quad (6)$$

The key problem of our inpainting consists in interpolating the values of $\Delta\tilde{\mathbf{v}}_i^k$ from the observed points to the unobserved ones, in other words, predicting the unobserved non-articulated displacements, see Fig. 10e. The modified rest pose is then mapped back by (forward) skinning to produce the final mesh, see Fig. 10f.

Our method for predicting the unobserved non-articulated displacements in the rest pose is based on the assumption of spatio-temporal smoothness. We stack all of the rest pose displacements into a $KN \times 3$ matrix $\mathbf{X}$, where K is the number of frames and N

the number of vertices (all vertices, both unobserved and observed ones). We find $\mathbf{X}$ by solving the following constrained optimization problem:

$$\min \mathcal{L}_{\text{spat}}(\mathbf{X}) + w_T \mathcal{L}_{\text{temp}}(\mathbf{X})$$
$$\text{s.t. } \mathbf{CX} = \mathbf{D} \tag{7}$$

where $\mathcal{L}_{\text{spat}}$ is a spatial Laplacian term that penalizes non-smooth deformations of the mesh and $\mathcal{L}_{\text{temp}}$ is a temporal Laplacian term that penalizes non-smooth trajectories of the vertices. Both of the terms are positive semi-definite quadratic forms. The parameter w_T is a weight balancing these two terms which we empirically set to 100. The sparse selector matrix $\mathbf{C}$ represents the observed points (constraints) and $\mathbf{D}$ their unposed 3D positions for each frame (each frame may have a different set of observed points). Specifically, we define $\mathcal{L}_{\text{spat}}$ as:

$$\mathcal{L}_{\text{spat}} = \sum_{i=1}^{3} \mathbf{X}_i^T \mathbf{L} \mathbf{X}_i \tag{8}$$

where $\mathbf{L}$ is cotangent-weighted Laplacian of the rest pose and $\mathbf{X}_i$ is the i-th column of $\mathbf{X}$. We found this quadratic deformation energy to be sufficient because our non-articulated displacements in the rest pose are small, though in future work it would be possible to explore non-linear thin shell deformation energies. For $\mathcal{L}_{\text{temp}}$, we use 1D temporal Laplacian which corresponds to acceleration: $\|\Delta \tilde{\mathbf{v}}_i^{k-1} - 2\Delta \tilde{\mathbf{v}}_i^{k} + \Delta \tilde{\mathbf{v}}_i^{k+1}\|_2^2$. The $\mathcal{L}_{\text{spat}}$ operator is applied to all frames independently, and $\mathcal{L}_{\text{temp}}$ is applied to all vertices independently. However, their weighted combination in Eq. 7 introduces spatio-temporal coupling, allowing one observed point to affect unobserved points through space and time.

The optimization problem Eq. 7 is a convex quadratic problem subject to equality constraints, which we transform to a linear system (KKT matrix) and solve. The only complication is that when processing too many frames, the KKT system can become too large. For example, with $K = 5000$ frames, the KKT matrix becomes approximately $10^7 \times 10^7$. Even though the KKT matrix is sparse, the linear solve becomes costly. To avoid this problem, we observe that smoothing over too many frames is not necessary and introduce a windowing scheme, decomposing longer sequence into 150-frame windows and solve them independently. To avoid any non-smoothness when transitioning from one window to another, the 150-frame windows overlap by 50 frames. After solving the problem in Eq. 7 for each window separately, we smoothly blend the overlapping 50 frames to ensure smoothness when transitioning from one window to the next one.

In this section we considered only off-line hole filling, where we can infer information from future frames. This approach would not be applicable to real-time settings where future frames are not available.

6 RESULTS

We first discuss the details of our CNN training process and the results. To prepare the training data, we manually annotated 24 randomly selected images (4000×2160) of our three actors wearing the handwritten suits. Out of the 24, we withheld 4 images as test set A. To evaluate our system's ability to generalize to a different suit and person, we also captured a short sequence of a fourth actor

Table 1. Composition of our training data. The original training set is the training set before data augmentation from 20 annotated images. The test set A is from 4 annotated images of our 3 actors wearing the hand-drawn suit who appeared in the training set. The test set B contains images of fourth actor wearing a suit with printed font.

Type of Data	CornerdetNet	RejectorNet	RecogNet
Original Training Set	667320	21257	7402
Augmented Training Set	5678934	121060	118432
Synthetic Training Set	NA	NA	214471
Test set A	136500	3372	1245
Test set B (printed suit)	134641	3523	900

wearing the printed suit and annotated 4 images from this sequence as test set B. Note that our trained process has never seen the font in the printed suit. Table 1 shows the total numbers of images used for training our CNNs. As shown in the first row of Table 1, the original training set (without data augmentation) for *RecogNet* is much smaller compared to *CornerdetNet* and *RejectorNet*, because of the limited number of valid quads in each of our annotated images. To improve the classification performance of *RecogNet*, we used synthetically generated images to complement the real data, as discussed in Supplementary Material B.2. The synthetic data contain 214471 crops (104×104), which significantly improved the robustness of *RecogNet*, see Section 6.1.

We train our CNNs using Tensorflow [Abadi et al. 2015] using a single NVIDIA Titan RTX; for each of our CNNs, an overnight run is typically enough to converge to good results using the Adam optimizer. After our CNNs have been trained, we run inference on a PC with an i7-9700K CPU and an NVIDIA GTX 1080 GPU. With a 4000×2160 input image, an inference pass of *CornerdetNet* takes 300ms, generating candidate quads 10ms, *RejectorNet* takes 1-2s to classify all of the candidate quads (104×104) and *RecogNet* takes 5ms to recognize the valid quads. The computational bottleneck is the *RejectorNet* due to the large number of candidate quads; this could be improved in the future by a more aggressive culling of candidate quads. For each frame, the time for 3D reconstruction is negligible, taking less than 1ms for all points. Even though we used only one computer and processed our image sequences off-line, we would like to point out that our method for extracting 3D labeled points from multi-view images is embarrassingly parallel, because each frame and even each input crop for our CNNs can be processed independently. Coupling through time is introduced only in the final hole-filling step (Section 5.2). The time for solving the sparse linear system (Eq. 7) for a 150 frames window is about 10s.

We captured motion sequences of three actors, one male and two females. One of the female actor wears the small suit and the other two actors wear the medium suit. For each actor, we captured about 12,000 frames (at 30FPS) of raw image data consisting of 1) camera calibration, 2) 6000 frames of calisthenics-type sequence intended for body model refinement (also serving as a warm-up for the actor), 3) the main performance. Each frame consists of 16 images from our multicamera setup. It took about 300 hours to process all of the 576,000 images (4.6 TB) using one computer.

6.1 Evaluation of CNNs

CornerdetNet. There are two parts of the *CornerdetNet*'s output: 1) classification response that predicts whether there is a valid corner in the center 8×8 window of the input 20×20 image and 2) its subpixel coordinates (or arbitrary values of a corner is not present). In Table 2 we summarize the results for both classification and localization errors. The localization error is measured by the distance in pixels between the predicted corner location and the manually annotated corner location. The overall classification accuracy for *CornerdetNet* is 99.393% on the training set and 99.510% on the test set A. The fact that *CornerdetNet* works better on the test set A supports our hypothesis that more aggressive data augmentation results in worse performance on the training set but better performance on the test set. On the test set A, the average localization error is 0.21 pixels and 99% of the corner localizations achieve error 0.6361 pixels or less, which is remarkably low. Similar accuracy was also achieved on the test set B, which is from the printed suit, which means *CornerdetNet* can generalize to a suit that it has not been trained on. With our camera setup, 1 pixel error corresponds to approximately 1mm of 3D error for an actor 2 meters away from the camera. In practice, this means that our 3D reconstructed points are highly accurate, allowing us to capture minute motions such as muscle twitches or flesh jiggling.

Table 2. Results for *CornerdetNet* on training and test sets.

n=5678934	Actual True	Actual False
Prediction True	2.533%	0.587%
Prediction False	0.02%	96.86%

(a) Confusion matrix on training set

n=136500	Actual True	Actual False
Prediction True	1.7%	0.44%
Prediction False	0.051%	97.81%

(b) Confusion matrix on test set A

n=134641	Actual True	Actual False
Prediction True	1.214%	0.34%
Prediction False	0.021%	98.432%

(c) Confusion matrix on test set B (printed suit)

Data set	Max	Mean	Median
Training set	1.959	0.1793	0.1448
Test set A	0.9485	0.2121	0.1904
Test set B (printed suit)	1.081	0.2321	0.1866

(d) Max/mean/median of corner localization error in pixels

Data set	95%	99%	99.9%	99.99%
Training set	0.4613	0.6611	0.9115	1.166
Test set A	0.5151	0.6361	0.8912	0.9428
Test set B (printed suit)	0.5831	0.6331	0.9684	1.083

(e) Percentiles of corner localization error in pixels

RejectorNet. The confusion matrices of a trained *RejectorNet* network are reported in Table 3. The overall classification accuracy for *RejectorNet* is 99.723% on the training set, 99.704% on the test set A, and 99.31% on the test set B. From the confusion matrix, we
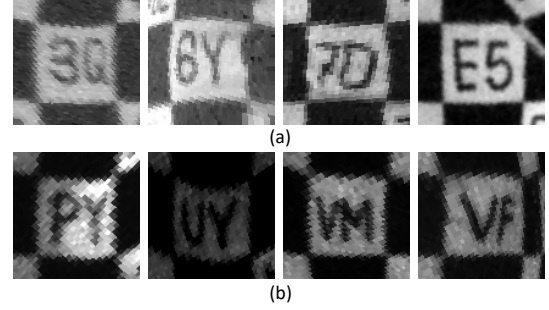
(a)

(b)

Fig. 11. Examples of errors made by *RejectorNet*. (a) False positives; even though the codes are legible, these samples were labeled negative due to slight image imperfections. (b) False negatives, labeled positive but close to the decision boundary.

can observe that we have more false positives than false negatives. The reason is that we intentionally annotated the training data conservatively. As shown in Fig. 11a, quads with even slight imperfections were labeled as negative examples. This results in *RecogNet* reporting more false positives, but the *RecogNet* actually inherits the conservative nature of the annotations; in practice, *RecogNet* only rarely accepts a low-quality quad image. This nature is well demonstrated by its performance on test set B. Because the *RejectorNet* have never seen the font used on the printed suit, due to its conservative nature it tends to reject a considerable number of valid quads (around 23% of valid quads are rejected). On the other hand, the *RejectorNet* returns no false positives, i.e., no invalid quads were falsely accepted.

Table 3. Confusion matrix for *RejectorNet*'s results on training and test set.

n=121060	Actual True	Actual False
Prediction True	13.133%	0.243%
Prediction False	0.034%	86.59%

(a) Confusion matrix on training set

n=3372	Actual True	Actual False
Prediction True	1.305%	0.297%
Prediction False	0.0%	98.399%

(b) Confusion matrix on the test set A

n=3523	Actual True	Actual False
Prediction True	2.286%	0%
Prediction False	0.6812%	97.03%

(c) Confusion matrix on the test set B (printed suit))

RecogNet. We compare the *RecogNet* trained with/without the synthetic training set in Table 4. Without using the synthetic training set, the *RecogNet* had prediction accuracy of 99.522% on the test set A. This accuracy was low and it was the main source of errors in our pipeline. Enhanced with the synthetic training set, the prediction accuracy on the test set A increased to 99.919% and significantly improved our results. Moreover, without using the synthetic training set, the *RecogNet* cannot generalize well to the test set B: the prediction accuracy on it is only 85.16%. That is because

RecogNet has never seen this new font on the printed suit. However, by enhancing training set with synthetic data which uses various fonts, the generalizability of *RecogNet* has improved significantly, achieving an accuracy of 100% on the test set B.

Table 4. Classification accuracy for *RecogNet* trained with/without synthetic training data.

With synthetic training set	Classification Accuracy			
	Real Training	Synthetic Training	Test A	Test B
No	99.940%	NA	99.522%	85.16%
Yes	99.967%	94.849%	99.919%	100%

Overall performance. In the previous sections we reported the results of each individual CNN. To evaluate our complete corner localization and labeling pipeline (Fig. 4), we use our 2 test sets of 8 manually annotated images (4000 × 2160) where we know the ground truth positions and labels of all corners. The images in the test set A and test set B contain 1702 and 1296 manually labeled corners, respectively. Our 2D pipeline detected 92% (1566 of 1702) of the ground truth corners from the test set A and 69% (896 of 1296) of the ground truth corners from the test set B. The discarded corners corresponded to low quality quads or quads with printed fonts that are significantly different from the hand-drawn ones, which were rejected by *RejectorNet*. Note that we intentionally trained the *RejectorNet* to be conservative, i.e., to reject all borderline cases. Missing observations on the test set A do not represent a big problem because they can be fixed by inpainting (Section 5.2); also, we observed that low-quality quads are often associated with inaccurate corner localization, increasing the noise in the 3D reconstruction. The missing observations on the test set B could be improved by enhancing the training of *RejectorNet* with a few more annotated images from the new suit, or training *RejectorNet* with synthetic data. The mean corner localization error in our is 0.4607 pixels test set A and 0.4678 on the test set B, and the maximum localization error is 1.854 on the test set A and 1.031 on the test set B. Due to our conservative rejection approach, the final CNN, *RecogNet*, made *zero* mistakes on the test sets, i.e., all of the labeled corners in the two test sets were assigned the correct label.

6.2 Comparison to previous methods

CornerdetNet. We compared our *CornerdetNet* with three alternative corner detectors: Shi-Tomasi [Shi et al. 1994], Harris [Harris et al. 1988], and deep learning based detector "SuperPoint" [DeTone et al. 2018]. For Shi-Tomasi [Shi et al. 1994] and Harris [Harris et al. 1988], we use OpenCV's [Bradski and Kaehler 2008] implementation and the recommended parameters. We used the method cv::cornerSubPix() [Förstner and Gülch 1987] to refine the detected corner locations to subpixel accuracy.

A qualitative comparison of corner detection results is shown in Fig. 12. We also quantitatively compare those corner detectors using the test set B in Table 5. To evaluate the classification accuracy, we match the detected corners to the annotated corners if the distance between them is less than 1.5 pixels. The classification accuracy comparison is shown (Table. 5a). As we can see from both
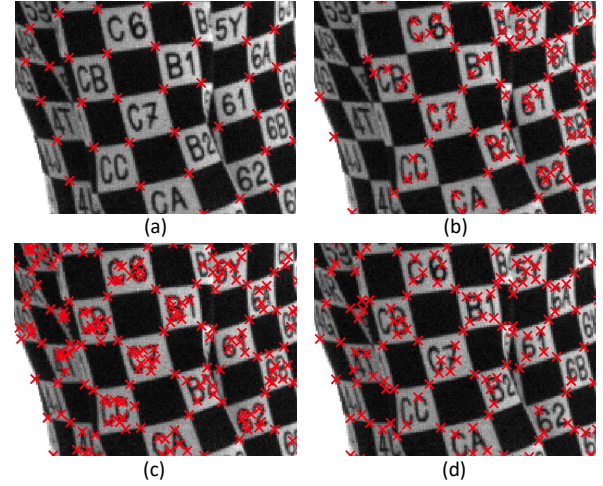

Fig. 12. Comparison between different corner detectors. (a) Our *Cornerdet-Net*; (b) Shi-Tomasi; (c) Harris; (d) SuperPoint.

Table 5. Comparison of different corner detectors using test set B. In (a), *False Negatives* means missing detection of annotated corners, *False Positives* means detector yields extra incorrect corners that are not annotated.

	CornerdetNet	Shi-Tomasi	Harris	SuperPoint
True Positives	928	400	867	901
False Negatives	11	540	73	39
False Positives	30	779	12413	3094

(a) Comparison of classification accuracies

	CornerdetNet	Shi-Tomasi	Harris	SuperPoint
Localization Error	0.2321	0.3135	0.3081	0.4937

(b) Comparison of mean localization errors

Fig. 12 and Table 5a, Shi-Tomasi, Harris and SuperPoints all have the same problem: they detect a lot of non-checkerboard corners (false positives), many of which are from the text on the suit. This is not surprising because those corner detectors are designed to detect general features that exists in nature, not checkerboard corners specifically. These wrong detections can slow down the processing of our pipeline because they lead to more invalid candidate quads. Also, they can confuse *RejectorNet*. On the other hand, alternative corner detectors are missing many valid corners in comparison to *CornerdetNet*, which will results in missing observations. From Table 5b, we can see that those corner detectors are also inferior to *CornerdetNet* in terms of localization accuracy.

RecogNet. We compared our *RecogNet* with Tesseract-OCR [Smith 2007]. The results are shown in Table 6. Tesseract-OCR produces significantly worse results than *RecogNet*, especially on the hand-drawn test set A. In practice, we find that Tesseract-OCR is very sensitive to stretching, distortion and noise. This is probably due to the fact that Tesseract-OCR was designed to recognize printed text with standard format rather than text on a suit worn by human which can have significant distortion due to stretching. Our dedicated *RecogNet* performs much better in this case.

Table 6. Comparison of text recognizers' prediction accuracies.

	RecogNet	Tesseract-OCR
test set A	99.919%	19.26%
test set B	100%	61.93%

(a) Reprojection error of 3D reconstruction [pixels]

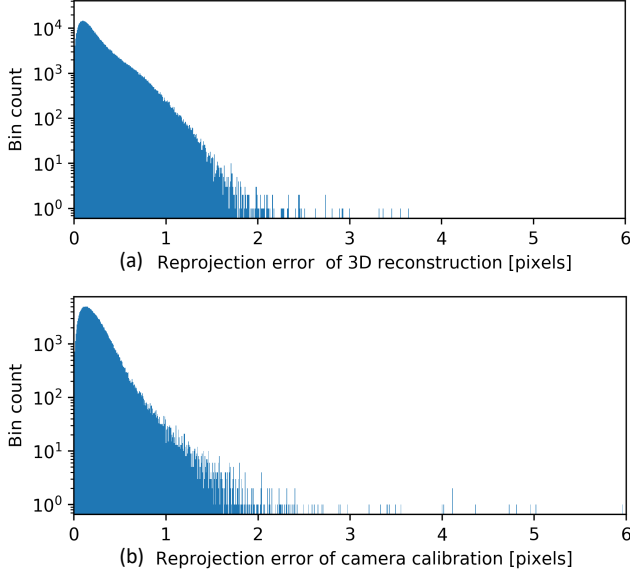(b) Reprojection error of camera calibration [pixels]

Fig. 13. Comparison of histograms of reprojection errors in 3D reconstruction and camera calibration. (a) Distribution of reprojection errors computed per camera for all the 3D reconstructed corners in 10000 consecutive frames. (b) The distribution of reprojection errors of camera calibration. We can see those two histograms look very similar.

6.3 3D Reconstruction Accuracy

Metrics. Evaluating 3D reconstruction accuracy is hard, because we do not have any ground truth measurements of a moving human body. To evaluate the accuracy of our 3D reconstructed corners, we compute their reprojection errors and we compare them to the reprojection errors obtained in our camera calibration process (Section 3.2). Using f^k to denote the projection function of camera k, if a reconstructed 3D point $\mathbf{p}_i$ is seen by camera k and $\mathbf{c}_i^k$ is the pixel location of the corresponding 2D corner i in camera k, the reprojection error for corner i in camera k is defined as:

$$\mathbf{e}_{i,k} = ||f^k(\mathbf{p}_i) - \mathbf{c}_i^k|| \tag{9}$$

The reprojection error for camera calibration is defined analogously, except that we use 3D calibration boards with perfect, rigid checkerboard corners and a standard OpenCV corner detector. In contrast, our corners are painted on an elastic suit worn by an actor.

Quantitative evaluation. We report the histograms of reprojection errors of 3D reconstruction and camera calibration in Fig. 13. The 3D reconstruction reprojection error is computed per camera for all the reconstructed points in a consecutive sequence of 10000 frames. The calibration reprojection error was computed on 448 frames that we use to calibrate the cameras, where we wave a 9×12 calibration board in front of our cameras. In 13, we can see the two
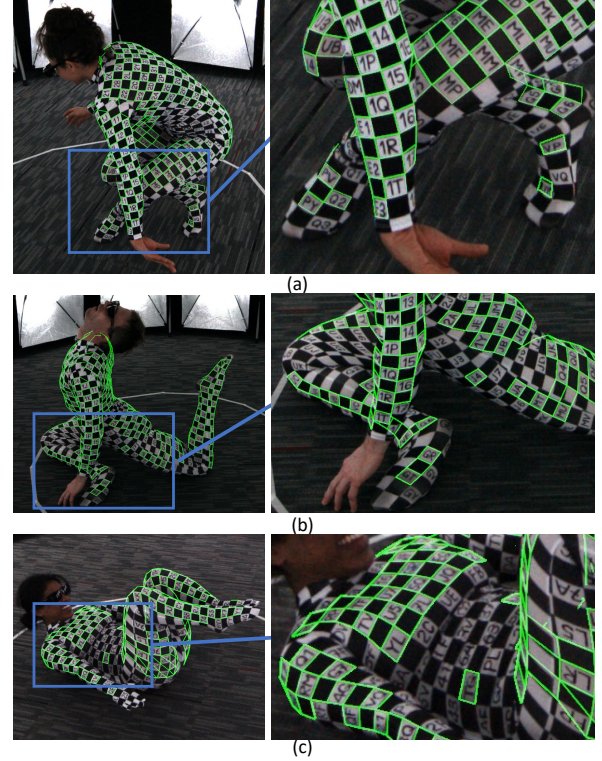
Fig. 14. Our results in challenging poses. Our reconstructed mesh (visualized as green wireframe) closely matches the checkerboard pattern in the original input images (background). Note the successful isolated code recognitions on the feet in (a) and (b).

error distributions look very similar, which means the reprojection errors of our 3D reconstruction results have similar statistics as the reprojection errors in camera calibration. We cannot expect to obtain lower reprojection errors than camera calibration.

Table 7 shows the percentiles of all the reprojections errors in 10000 frames that we use to evaluate the 3D reconstruction. 99% of the reprojection errors is less than 1.009 pixels, which is remarkably accurate given the high resolution of our images (4000×2160).

Table 7. Percentiles of reprojection errors computed per camera for all the reconstructed points in a consecutive sequence of 10000 frames.

95%	99%	99.9%	99.99%
0.6979	1.009	1.409	3.376

Qualitative evaluation. Fig. 14 shows challenging cases where there are significant self occlusions. We mesh the reconstructed point cloud using the rest pose mesh structure introduced in Section 5.1 by preserving the observed faces in the rest pose mesh (see Fig. 9b). Then we project the reconstructed mesh back to the image using the camera parameters, which gives us the green wireframe in Fig. 14. We can see that the mesh wireframe aligns very closely with the checkerboard pattern on the suit. Another important observation is that even despite large occlusions, our method can still obtain
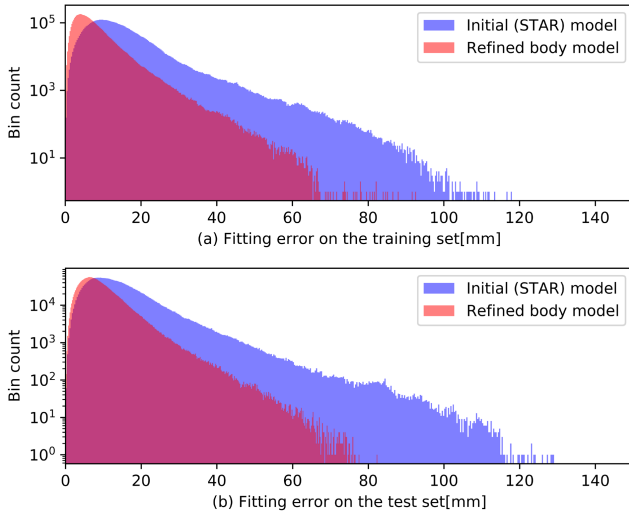
Fig. 15. Histograms of fitting errors of the initial body model and the refined body model on the training set and the test set.
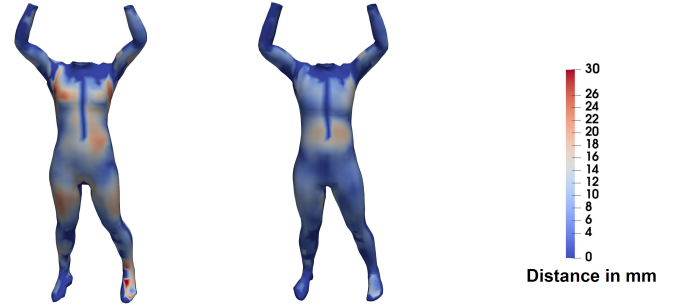

Fig. 16. The fitting errors on the body model; left: an initial body model which is just a re-meshing of the STAR model; right: our final refined body model.

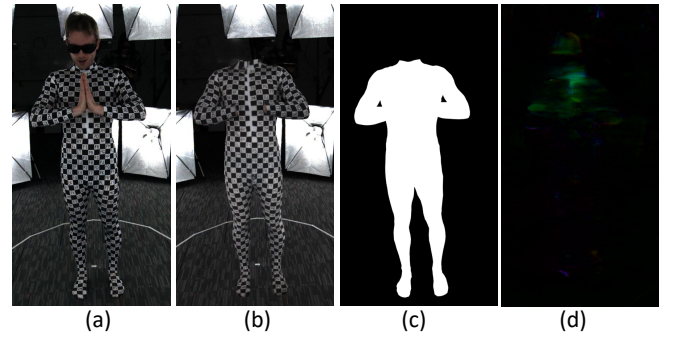

(a)   (b)   (c)   (d)

Fig. 17. (a) The input image. (b) The synthetic image rendered from our body mesh. (c) The foreground mask of our body mesh. (d) The optical flow between the synthetic image (b) and the real one (a). The angle of flow is visualized by hue and the magnitude of flow by value in HSV color model.

correctly labeled corners as long as the entire two-letter code is visible see, e.g., the foot and calf in Fig. 14a,b. In Fig. 14c, we can see that the conservative *RejectorNet* correctly rejects the wrinkled quads in the belly region, since reading the codes would be difficult or impossible.

6.4 Evaluation of Body Model Refinement

To refine our actor model, we record a 6000 frames training sequence. After the body model refinement we select another 3000 frames corresponding to motions different from the ones in the training set. The fitting error is defined as the distance between the vertices of the deformed body model and the actual 3D reconstructed corners. We compare the fitting errors between the initial model, which is just a remeshing of the STAR model (see Section 5.1), and the refined body model, which was optimized on the training set (see Section 5.2). Fig. 15 shows the distribution of fitting errors per vertex of the initial body model and the refined body model on the training set and the test set. We can see in both data sets, the refined body model is much more accurate. Specifically, the body model refinement reduces the average fitting error from 13.6mm to 5.2mm on the training set and from 13.5mm to 7.1mm on the test set. Fig. 16 visualizes the fitting errors on the body model before and after body model refinement in one example frame using a heat map.

6.5 Evaluation with Optical Flow

To quantify the accuracy of the 3D reconstruction of the entire body, we compare renderings of a textured mesh with the original images using optical flow [Bogo et al. 2014, 2017]. First, we need to create a suit-like texture for our body mesh (Fig. 9b). We create a standard UV parametrization for our mesh and generate the texture from 10 hand picked frames using a differentiable renderer [Ravi et al. 2020]; though this is just one possible way to generate the texture [Bogo et al. 2017]. We render the textured body mesh with back face culling enabled and overlay it over clean plates (i.e., images of the

capture volume without any actor). The virtual camera parameters are set to our calibration of the real cameras. The optical flow is computed from the synthetic images to the undistorted real images using FlowNet2 [Ilg et al. 2017] with the default settings. Because our mesh does not have the hands and the head, we first render a foreground mask of our body mesh (Fig. 17c). We only evaluate the optical flow on the region covered by the foreground mask to exclude the hands, the head and the background. The foreground mask cannot exclude the hands and the head when they are occluding the body (as in Fig. 17a, b) but, fortunately, the optical flow is robust to missing parts (see Fig. 17d). We use optical flow to compare the original images with two types of renders: 1) our low-dimensional refined body model (the gray mesh in Fig. 10c, which does not fit the reconstructed corners exactly), 2) our final result after adding non-articulated displacements (Fig. 10f). Fig. 18a plots the average optical flow norm for each frame for consecutive 2000 frames, including various challenging poses and fast motions. We can see that the result with non-articulated displacements is much more accurate than only our low-dimensional refined body model. This is mainly due to flesh deformation which is not well explained by the refined body model, especially in more extreme poses which correspond to the spikes in the blue curve in Fig. 18a. The red curve corresponds to our final result which exhibits consistently low optical flow errors.

We also plot the distribution of the optical flow norm for each pixel in the foreground mask in Fig. 18b. With our final animated
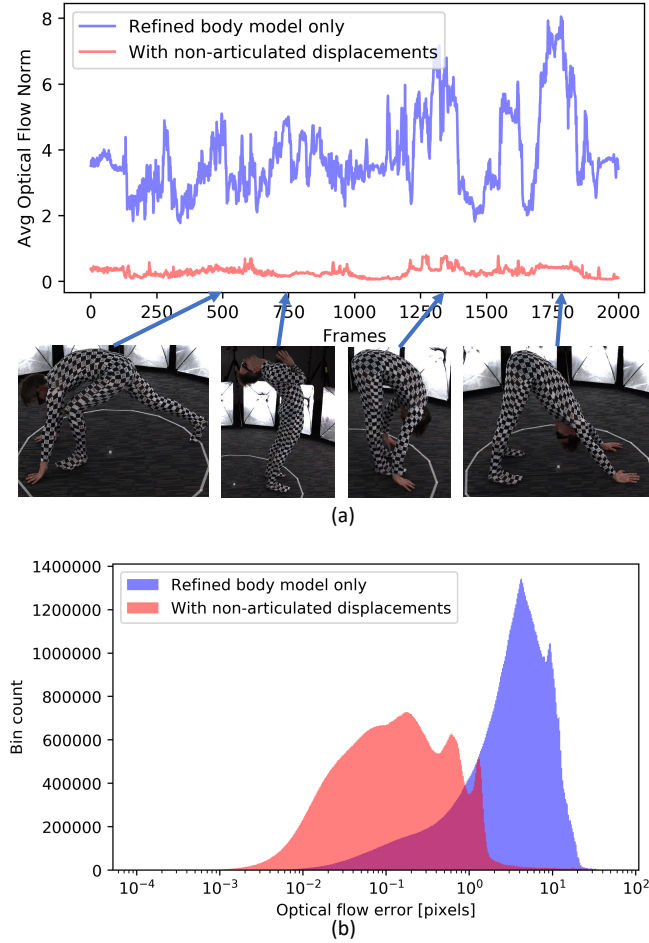
(a)

(b)

Fig. 18. Optical flow errors. The blue curves correspond to a low-dimensional refined body model only; the red curves correspond to our final results including non-articulated displacements. (a) The plot of average optical flow norm in a motion sequence of 2000 frames (we show four example frames below the graph). (b) Histograms of per-pixel optical flow norms for the same sequence.

mesh, 95% of pixels have optical flow norm less than 1.20 and 99% of pixels have optical flow norm less than 2.46.

7 LIMITATIONS AND FUTURE WORK

An obvious limitation of our method is the necessity of wearing a special motion capture suit. A suit can in principle slide over the skin, but we did not observe any significant sliding in our experiments because our suits are tightly fitting. If this became a problem in the future, we could increase adhesion with internal silicone patches as in sportswear, or even apply spirit gum or medical adhesives. The suit needs to be made in various sizes and fit may be a challenge for obese people. The holy grail of full-body capture is to get rid of suits and instead rely only on skin features such as the pores, similarly to facial performance capture. We tried imaging the bare skin, but with our current camera resolution (4000×2160) we were unable to get sufficient detail from the skin. We could obtain more

detail with narrower fields of view and more cameras to cover the capture volume, but then there are issues with the depth of field and hardware budgets. Additional complications of imaging bare skin are body hair and privacy concerns; our suit certainly has its disadvantages, but mitigates these issues. A significant advantage of our suit compared to traditional motion capture suits is that we do not need to attach any markers (reflective spheres, See Fig. 2a). Traditional motion capture markers can impede motion or even fall off, e.g., when the actor is rolling on the ground. An intriguing direction for future work would be to enhance our suit with additional sensors, in particular EMG, IMU or pressure sensors on the feet.

In this paper we focused on the body and ignored the motion of the face and the hands. Our actors are wearing sunglasses because our continuous passive lights are too bright; the perceived brightness could be reduced by lights which strobe in sync with camera shutters, but this would require significant investments in hardware. In future work, our method could be directly combined with modern methods that capture the motion of the face and the hands [Choutas et al. 2020; Joo et al. 2018; Pavlakos et al. 2019; Xiang et al. 2019]. We note that our current system captures the motion of the feet, but not the individual toes.

Our current data processing is off-line only. In the future, we believe it should be possible to create a real-time version of our system. This would require machine vision cameras tightly integrated with dedicated GPUs or tensor processors for real-time neural network inference. Each such hardware unit could emit small amounts of data: only information about the corner locations and their labels, avoiding the high bandwidth requirements typically associated with high-resolution video streams.

Another avenue for future work involves research of different types of fiducial markers that can be printed on the suit. In fact, we made initial experiments with printing on textile and sewing our own suits, which gives us much more flexibility than handwritten two-letter codes discussed in this paper. We postponed this line of research due to the Covid19 pandemic. Our pipeline for reconstructing labeled 3D points does not make any assumptions about the human body, which means that we could apply our method also for capturing the motion of clothing or even loose textiles such as a curtain.

8 CONCLUSIONS

We have presented a method for capturing more than 1000 uniquely labeled points on the surface of a moving human body. This technology was enabled by our new type of motion capture suit with checkerboard-type corners and two-letter codes enabling unique labeling of each corner. Our results were obtained with a multi-camera system built from off-the-shelf components at a fraction of the cost of a full-body 3DMD setup, while demonstrating a wider variety of motions than the DFAUST dataset [Bogo et al. 2017], including gymnastics, yoga poses and rolling on the ground. Our method for reconstructing labeled 3D points does not rely on temporal coherence, which makes it very robust to dis-occlusions and also invites parallel processing. We provide our code and data as supplementary materials and we will release an optimized version of our code as open source.
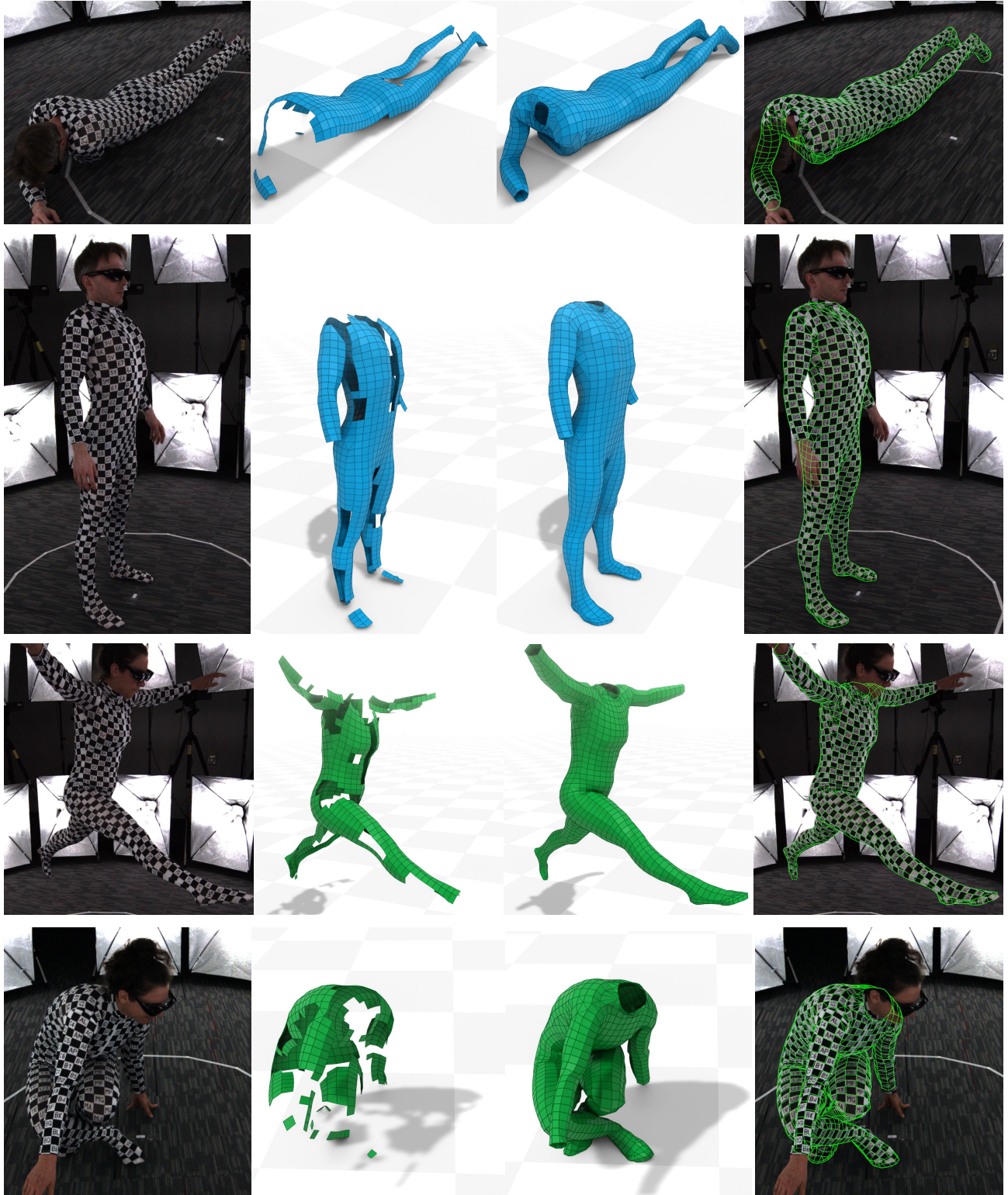
Fig. 19. Results in challenging poses. From the left to right: 1) input images, 2) raw 3D reconstructions with holes, 3) our final meshes after interpolating non-articulated displacements, and 4) wireframe rendering of the final meshes overlaid with the original images.

ACKNOWLEDGMENTS

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

Sameer Agarwal and Keir Mierle. 2012. Ceres solver: Tutorial & reference. *Google Inc 2* (2012), 72.

Benjamin Allain, Jean-Sébastien Franco, and Edmond Boyer. 2015. An efficient volumetric framework for shape tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 268–276.

Brett Allen, Brian Curless, Brian Curless, and Zoran Popović. 2003. The space of human body shapes: reconstruction and parameterization from range scans. In *ACM transactions on graphics (TOG)*, Vol. 22. ACM, 587–594.

Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. 2005. Scape: shape completion and animation of people. In *ACM Transactions on Graphics (TOG)*, Vol. 24. ACM, 408–416.

Andreas Aristidou, Daniel Cohen-Or, Jessica K Hodgins, and Ariel Shamir. 2018. Self-similarity analysis for motion capture cleaning. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 297–309.

Angelos Barmpoutis. 2013. Tensor body: Real-time reconstruction of the human body and avatar synthesis from RGB-D. *IEEE transactions on cybernetics* 43, 5 (2013), 1347–1356.

Stuart Bennett and Joan Lasenby. 2014. ChESS–Quick and robust detection of chessboard features. *Computer Vision and Image Understanding* 118 (2014), 197–210.

Federica Bogo, Michael J Black, Matthew Loper, and Javier Romero. 2015. Detailed full-body reconstructions of moving people from monocular RGB-D sequences. In *Proceedings of the IEEE International Conference on Computer Vision.* 2300–2308.

Federica Bogo, Javier Romero, Matthew Loper, and Michael J Black. 2014. FAUST: Dataset and evaluation for 3D mesh registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 3794–3801.

Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J Black. 2017. Dynamic FAUST: Registering human bodies in motion. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 6233–6242.

Adnane Boukhayma, Vagia Tsiminaki, Jean-Sébastien Franco, and Edmond Boyer. 2016. Eigen appearance maps of dynamic shapes. In *European Conference on Computer Vision.* Springer, 230–245.

G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

Gary Bradski and Adrian Kaehler. 2008. *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.".

Christoph Bregler, Jitendra Malik, and Katherine Pullen. 2004. Twist based acquisition and tracking of animal and human kinematics. *International Journal of Computer Vision* 56, 3 (2004), 179–194.

Thomas Brox, Bodo Rosenhahn, Juergen Gall, and Daniel Cremers. 2009. Combined region and motion-based 3D tracking of rigid and articulated objects. *IEEE transactions on pattern analysis and machine intelligence* 32, 3 (2009), 402–415.

Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2018. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. *arXiv preprint arXiv:1812.08008* (2018).

Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2017. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 7291–7299.

Dan Casas, Margara Tejera, Jean-Yves Guillemaut, and Adrian Hilton. 2012. 4D parametric motion graphs for interactive animation. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games.* 103–110.

Ben Chen, Caihua Xiong, and Qi Zhang. 2018. CCDN: Checkerboard corner detection network for robust camera calibration. In *International Conference on Intelligent Robotics and Applications.* Springer, 324–334.

Vasileios Choutas, Georgios Pavlakos, Timo Bolkart, Dimitrios Tzionas, and Michael J Black. 2020. Monocular expressive body regression through body-driven attention. *arXiv preprint arXiv:2008.09062* (2020).

Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. 2015. High-quality streamable free-viewpoint video. *ACM Transactions on Graphics (ToG)* 34, 4 (2015), 1–13.

Stefano Corazza, Lars Mündermann, Emiliano Gambaretto, Giancarlo Ferrigno, and Thomas P Andriacchi. 2010. Markerless motion capture through visual hull, articulated icp and subject specific model generation. *International journal of computer vision* 87, 1-2 (2010), 156–169.

Edilson De Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. 2008. Performance capture from sparse multi-view video. In *ACM SIGGRAPH 2008 papers.* 1–10.

Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. 2018. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops.* 224–236.

Simon Donné, Jonas De Vylder, Bart Goossens, and Wilfried Philips. 2016. MATE: Machine learning for adaptive calibration template detection. *Sensors* 16, 11 (2016), 1858.

Mingsong Dou, Jonathan Taylor, Henry Fuchs, Andrew Fitzgibbon, and Shahram Izadi. 2015. 3D scanning deformable objects with a single RGBD sensor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 493–501.

Mark Fiala. 2005. ARTag, a fiducial marker system using digital techniques. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 2. IEEE, 590–596.

Wolfgang Förstner and Eberhard Gülch. 1987. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In *Proc. ISPRS intercommission conference on fast processing of photogrammetric data.* Interlaken, 281–305.

Juergen Gall, Bodo Rosenhahn, Thomas Brox, and Hans-Peter Seidel. 2010. Optimization and filtering for human motion capture. *International journal of computer vision* 87, 1-2 (2010), 75.

Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. 2014. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition* 47, 6 (2014), 2280–2292.

D Gavrila and LS Davis. 1996. Tracking of humans in action: A 3-D model-based approach. In *ARPA Image Understanding Workshop.* (Palm Springs), 737–746.

Stevie Giovanni, Yeun Chul Choi, Jay Huang, Eng Tat Khoo, and KangKang Yin. 2012. Virtual try-on using kinect and HD camera. In *International Conference on Motion in Games.* Springer, 55–65.

Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. 2018. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 7297–7306.

Kaiwen Guo, Peter Lincoln, Philip Davidson, Jay Busch, Xueming Yu, Matt Whalen, Geoff Harvey, Sergio Orts-Escolano, Rohit Pandey, Jason Dourgarian, et al. 2019. The relightables: Volumetric performance capture of humans with realistic relighting. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–19.

Shangchen Han, Beibei Liu, Robert Wang, Yuting Ye, Christopher D Twigg, and Kenrick Kin. 2018. Online optical marker-based hand tracking with deep labels. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–10.

Christopher G Harris, Mike Stephens, et al. 1988. A combined corner and edge detector.. In *Alvey vision conference*, Vol. 15. Citeseer, 10–5244.

Richard I Hartley and Peter Sturm. 1997. Triangulation. *Computer vision and image understanding* 68, 2 (1997), 146–157.

Gines Hidalgo, Yaadhav Raaj, Haroon Idrees, Donglai Xiang, Hanbyul Joo, Tomas Simon, and Yaser Sheikh. 2019. Single-Network Whole-Body Pose Estimation. *arXiv preprint arXiv:1909.13423* (2019).

David A Hirshberg, Matthew Loper, Eric Rachlin, and Michael J Black. 2012. Coregistration: Simultaneous alignment and modeling of articulated 3D shape. In *European Conference on Computer Vision.* Springer, 242–255.

Daniel Holden. 2018. Robust solving of optical motion capture data by denoising. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.

Danying Hu, Daniel DeTone, and Tomasz Malisiewicz. 2019. Deep charuco: Dark charuco marker pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8436–8444.

Peng Huang, Chris Budd, and Adrian Hilton. 2011. Global temporal registration of multiple non-rigid surface sequences. In *CVPR 2011*. IEEE, 3473–3480.

Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. 2017. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2462–2470.

Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep features for text spotting. In *European conference on computer vision*. Springer, 512–528.

Hanbyul Joo, Tomas Simon, and Yaser Sheikh. 2018. Total capture: A 3d deformation model for tracking faces, hands, and bodies. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8320–8329.

Roland Kehl and Luc Van Gool. 2006. Markerless tracking of complex human motions from multiple views. *Computer Vision and Image Understanding* 104, 2-3 (2006), 190–209.

Hao Li, Bart Adams, Leonidas J Guibas, and Mark Pauly. 2009. Robust single-view geometry and motion reconstruction. *ACM Transactions on Graphics (ToG)* 28, 5 (2009), 1–10.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *European conference on computer vision*. Springer, 21–37.

Yebin Liu, Juergen Gall, Carsten Stoll, Qionghai Dai, Hans-Peter Seidel, and Christian Theobalt. 2013. Markerless motion capture of multiple characters using multiview image segmentation. *IEEE transactions on pattern analysis and machine intelligence* 35, 11 (2013), 2720–2735.

Stephen Lombardi, Jason Saragih, Tomas Simon, and Yaser Sheikh. 2018. Deep appearance models for face rendering. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–13.

Shangbang Long, Xin He, and Cong Yao. 2020. Scene text detection and recognition: The deep learning era. *International Journal of Computer Vision* (2020), 1–24.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. 2015. SMPL: A skinned multi-person linear model. *ACM transactions on graphics (TOG)* 34, 6 (2015), 248.

David G Lowe. 1999. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, Vol. 2. Ieee, 1150–1157.

Jianqi Ma, Weiyuan Shao, Hao Ye, Li Wang, Hong Wang, Yingbin Zheng, and Xiangyang Xue. 2018. Arbitrary-oriented scene text detection via rotation proposals. *IEEE Transactions on Multimedia* 20, 11 (2018), 3111–3122.

Qianli Ma, Jinlong Yang, Anurag Ranjan, Sergi Pujades, Gerard Pons-Moll, Siyu Tang, and Michael Black. 2020. Learning to Dress 3D People in Generative Clothing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Nadia Magnenat-Thalmann, Richard Laperrire, and Daniel Thalmann. 1988. Joint-dependent local deformations for hand animation and object grasping. In *In Proceedings on Graphics interface'88*. Citeseer.

Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. 2017. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 44.

Abhimitra Meka, Rohit Pandey, Christian Haene, Sergio Orts-Escolano, Peter Barnum, Philip Davidson, Daniel Erickson, Yinda Zhang, Jonathan Taylor, Sofien Bouaziz, Chloe Legendre, Wan-Chun Ma, Ryan Overbeck, Thabo Beeler, Paul Debevec, Shahram Izadi, Christian Theobalt, Christoph Rhemann, and Sean Fanello. 2020. Deep Relightable Textures - Volumetric Performance Capture with Neural Rendering. *ACM Transactions on Graphics (Proceedings SIGGRAPH Asia)* 39, 6. https://doi.org/10.1145/3414685.3417814

Alberto Menache. 2000. *Understanding motion capture for computer animation and video games*. Morgan kaufmann.

Richard A Newcombe, Dieter Fox, and Steven M Seitz. 2015. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 343–352.

Alejandro Newell, Kaiyu Yang, and Jia Deng. 2016. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*. Springer, 483–499.

Edwin Olson. 2011. AprilTag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*. IEEE, 3400–3407.

Ahmed A A Osman, Timo Bolkart, and Michael J. Black. 2020. STAR: A Spare Trained Articulated Human Body Regressor. In *European Conference on Computer Vision (ECCV)*. https://star.is.tue.mpg.de

Sang Il Park and Jessica K Hodgins. 2006. Capturing and animating skin deformation in human motion. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 881–889.

Sang Il Park and Jessica K Hodgins. 2008. Data-driven modeling of skin and muscle deformation. In *ACM SIGGRAPH 2008 papers*. 1–6.

Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed AA Osman, Dimitrios Tzionas, and Michael J Black. 2019. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 10975–10985.

Georgios Pavlakos, Xiaowei Zhou, Konstantinos G Derpanis, and Kostas Daniilidis. 2017. Harvesting multiple views for marker-less 3d human pose annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 6988–6997.

Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V Gehler, and Bernt Schiele. 2016. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4929–4937.

Gerard Pons-Moll, Javier Romero, Naureen Mahmood, and Michael J Black. 2015. Dyna: A model of dynamic human shape in motion. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 120.

Fabián Prada, Misha Kazhdan, Ming Chuang, Alvaro Collet, and Hugues Hoppe. 2016. Motion graphs for unstructured textured meshes. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–14.

Yaadhav Raaj, Haroon Idrees, Gines Hidalgo, and Yaser Sheikh. 2019. Efficient Online Multi-Person 2D Pose Tracking with Recurrent Spatio-Temporal Affinity Fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4620–4628.

Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. 2020. Accelerating 3D Deep Learning with Py-Torch3D. *arXiv:2007.08501* (2020).

Joseph Redmon and Ali Farhadi. 2017. YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7263–7271.

Kathleen M Robinette, Sherri Blackwell, Hein Daanen, Mark Boehmer, and Scott Fleming. 2002. *Civilian american and european surface anthropometry resource (caesar), final report. volume 1. summary*. Technical Report. SYTRONICS INC DAYTON OH.

Edward Rosten and Tom Drummond. 2006. Machine learning for high-speed corner detection. In *European conference on computer vision*. Springer, 430–443.

Peter Sand, Leonard McMillan, and Jovan Popović. 2003. Continuous capture of skin deformation. In *ACM SIGGRAPH 2003 Papers*. 578–586.

Volker Scholz, Timo Stich, Marcus Magnor, Michael Keckeisen, and Markus Wacker. 2005. Garment motion capture using color-coded patterns. In *ACM SIGGRAPH 2005 Sketches*. 38–es.

Jianbo Shi et al. 1994. Good features to track. In *1994 Proceedings of IEEE conference on computer vision and pattern recognition*. IEEE, 593–600.

Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, Vol. 2. IEEE, 629–633.

Min-Ho Song and Rolf Inge Godøy. 2016. How fast is your body motion? Determining a sufficient frame rate for an optical motion tracking system using passive markers. *PloS one* 11, 3 (2016), e0150993.

Jonathan Starck and Adrian Hilton. 2007. Surface capture for performance-based animation. *IEEE computer graphics and applications* 27, 3 (2007), 21–31.

Carsten Stoll, Nils Hasler, Juergen Gall, Hans-Peter Seidel, and Christian Theobalt. 2011. Fast articulated motion tracking using a sums of gaussians body model. In *2011 International Conference on Computer Vision*. IEEE, 951–958.

Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. 1999. Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*. Springer, 298–372.

Tony Tung and Takashi Matsuyama. 2010. Dynamic surface matching by geodesic mapping for 3d animation transfer. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 1402–1409.

Graham Upton and Ian Cook. 1996. *Understanding statistics*. Oxford University Press.

Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. 2008. Articulated mesh animation from multi-view silhouettes. In *ACM Transactions on Graphics (TOG)*, Vol. 27. ACM, 97.

John Wang and Edwin Olson. 2016. AprilTag 2: Efficient and robust fiducial detection. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 4193–4198.

Robert Y Wang and Jovan Popović. 2009. Real-time hand-tracking with a color glove. *ACM transactions on graphics (TOG)* 28, 3 (2009), 1–8.

Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. 2016. Convolutional pose machines. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 4724–4732.

Ryan White, Keenan Crane, and David A Forsyth. 2007. Capturing and animating occluded cloth. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 34–es.

Donglai Xiang, Hanbyul Joo, and Yaser Sheikh. 2019. Monocular total capture: Posing face, body, and hands in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 10965–10974.

Yuanlu Xu, Song-Chun Zhu, and Tony Tung. 2019. Denserac: Joint 3d pose and shape estimation by dense render-and-compare. In *Proceedings of the IEEE International Conference on Computer Vision*. 7760–7770.

Zhengyou Zhang. 2000. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence* 22, 11 (2000), 1330–1334.

Huiyu Zhou and Huosheng Hu. 2008. Human motion tracking for rehabilitation—A survey. *Biomedical signal processing and control* 3, 1 (2008), 1–18.