Real-Time Continuous Pose Recovery of Human Hands Using Convolutional Networks

Jonathan Tompson, Murphy Stein, Yann LeCun and Ken Perlin
New York University

We present a novel method for real-time continuous pose recovery of markerless complex articulable objects from a single depth image. Our method consists of the following stages: a randomized decision forest classifier for image segmentation, a robust method for labeled dataset generation, a convolutional network for dense feature extraction and finally an inverse kinematics stage for stable real-time pose recovery. As one possible application of this pipeline, we show state-of-the-art results for real-time puppeteering of a skinned hand-model.

Categories and Subject Descriptors: I.3.6 [**Computer Graphics**]: Methodology and Techniques—*Interaction Techniques*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Animation*

General Terms: Algorithms, Human Factors

Additional Key Words and Phrases: Hand Tracking, Neural Networks, Markerless Motion Capture, Analysis-By-Synthesis

ACM Reference Format:

Tompson, J., Stein, M., LeCun, Y., Perlin, K., 2014. Real-time Continuous Pose Recovery of Human Hands Using Convolutional Networks. ACM Trans. Graph.

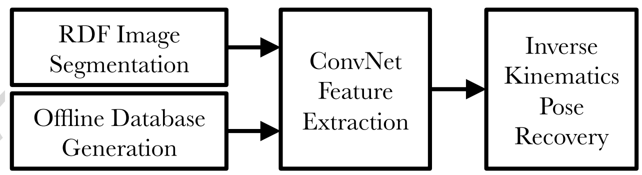


Fig. 1: Pose Recovery Pipeline Overview

1. INTRODUCTION

Inferring the pose of articulable objects from depth video data is a difficult problem in markerless motion capture. Requiring real-time inference with low-latency for real-time applications makes this even harder. The difficulty arises because articulable objects typically have many degrees of freedom (DOF), constrained parameter spaces, self-similar parts, and suffer from self-occlusion. All these factors make fitting a model directly to the depth data hard, and even undesirable in practice, unless the fitting process is able to account for such missing data.

One common approach to “fill in” missing data is to combine multiple simultaneous video streams; but this is a costly demand on the end-user and may prohibit widespread use of otherwise good solutions. A second common approach, called “supervised learning” in computer vision and machine learning, is to train a model on ground-truth data, which combines the full pose of the object in the frame with the depth image. The trained model can then use a priori information from known poses to make informed guesses about the likely pose in the current frame.

Large ground-truth datasets have been constructed for important articulable objects such as human bodies, and robust real-time pose inference systems have been trained on them using supervised learning. Unfortunately, most articulable objects, even common ones such as human hands, do not have publicly available datasets, or these datasets do not adequately cover the vast range of possible poses. Perhaps more importantly, it may be desirable to infer the real-time continuous pose of objects that do not yet have such datasets. The vast majority of objects seen in the world fall into this category, and a general method for dataset acquisition of articulable objects is an important contribution of this work.

The main difficulty with using supervised learning for training models to perform real-time pose inference of a human hand is in obtaining ground-truth data for hand pose. Typical models of the human hand have 25-50 degrees of freedom [Erol et al. 2007] and exclude important information such as joint angle constraints. Since real hands exhibit joint angle constraints that are pose dependent, faithfully expressing such limits is still difficult in practice. Unfortunately, without such constraints, most models are capable of poses which are anatomically incorrect. This means that sampling the space of possible parameters using a real hand is more desirable than exploring it with a model. With the advent of commodity depth sensors, it is possible to economically capture continuous traversals of this constrained low-dimensional parameter space in video, and then robustly fit hand models to the data to recover the pose parameters [Oikonomidis et al. 2011].

In this work, we present a solution to the difficult problem of inferring the continuous pose of a human hand by first constructing an accurate database of labeled ground-truth data in an automatic process, and then training a system capable of real-time inference. Since the human hand represents a particularly difficult kind of articulable object to track, we believe our solution is applicable to a wide range of articulable objects. Our method has a small latency equal to one frame of video, is robust to self-occlusion, requires no special markers, and can handle objects with self-similar parts, such as fingers. To allow a broad range of applications, our method works when the hand is smaller than 2% of the $640\times480 = 307$kpx image area.

Our method can be generalized to track any articulable object that satisfies three requirements: 1) the object to be tracked can be modeled as a 3D boned mesh, 2) a binary classifier can be made to label the pixels in the image belonging to the object, and 3) the projection from pose space (of the bones) to a projected 2D image in depth is approximately one-to-one. The model is used to automatically label depth video captured live from a user. This data is used to train a Randomized Decision Forest (RDF) architecture for image segmentation as well as a Convolutional Network (ConvNet) to infer the position of key model features in real-time. We also suggest a simple and robust inverse kinematics (IK) algorithm for real-time, high degree of freedom pose inference from the ConvNet output. The system can accommodate multiple commodity depth cameras for generating training data, but requires only a single depth camera for real-time tracking. We believe the key technical contribution of this work is the creation of a novel pipeline for fast pose inference, which is applicable to a wide variety of articulable objects. An overview of this pipeline is shown in Figure 1.

As a single example, training our system on an open-source linear-blend-skinning model of a hand with 42 degrees of freedom takes less than 10 minutes of human effort (18,000 frames at 30fps), followed by two days of autonomous computation time. Tracking and pose inference for a person's hand can then be performed in real-time using a single depth camera. Throughout our experiments, the camera is situated in front of the user at approximately eye-level height. The trained system can be readily used to puppeteer related objects such as alien hands, or real robot linkages, and as an input to 3D user interfaces [Stein et al. 2012].

2. RELATED WORK

A large body of literature is devoted to real-time recovery of pose for markerless articulable objects, such as human bodies, clothes, and man-made objects. As the primary contribution of our work is a fast pipeline for recovery of the pose of human hands in 3D, we will limit our discussion to the most relevant prior work.

Many groups have created their own dataset of ground-truth labels and images to enable real-time pose recovery of the human body. For example, Wang et al. [Wang et al. 2011] use the CyberGlove II Motion Capture system to construct a dataset of labeled hand poses from users, which are re-rendered as a colored glove with known-texture. A similar colored glove is worn by the user at run-time, and the pose is inferred in real-time by matching the imaged glove in RGB to their database of templates [Wang and Popović 2009]. In later work, the CyberGlove data was repurposed for pose inference using template matching on depth images, without a colored glove. Wang et al. have recently commercialized their hand-tracking system (which is now proprietary and is managed by 3Gear Systems [3Gear 2014]) and now uses a PrimeSense™ depth camera oriented above the table to recognize a large range of possible poses. This work differs from 3Gear's in a number of ways: 1) we attempt to perform continuous pose estimation rather than recognition by matching into a static and discrete database and 2) we orient the camera facing the user and so our system is optimized for a different set of hand gestures.

Also relevant to our work is that of Shotton et al. [Shotton et al. 2011], who used randomized decision forests to recover the pose of multiple bodies from a single frame by learning a per-pixel classification of the depth image into 38 different body parts. Their training examples were synthesized from combinations of known poses and body shapes. In similar work, Keskin et al. [Keskin et al. 2011] created a randomized decision forest classifier specialized for human hands. Lacking a dataset based on human motion capture, they synthesized a dataset from known poses in American Sign Language, and expanded the dataset by interpolating between poses. Owing to their prescribed goal of recognizing sign language signs themselves, this approach proved useful, but would not be feasible in our case as we require unrestricted hand poses to be recovered. In a follow on work [Keskin et al. 2012], Keskin et al. presented a novel shape classification forest architecture to perform per-pixel part classification.

Several other groups have used domain-knowledge and temporal coherence to construct methods that do not require any dataset for tracking the pose of complicated objects. For example, Wiese et al. [Weise et al. 2009] devise a real-time facial animation system for range sensors using salient points to deduce transformations on an underlying face model by framing it as energy minimization. In related work, Li et al. [Li et al. 2013] showed how to extend this technique to enable adaptation to the user's own facial expressions in an online fashion. Melax et al. [Melax et al. 2013] demonstrate a real-time system for tracking the full pose of a human hand by fitting convex polyhedra directly to range data using an approach inspired by constraint-based physics systems. Ballan et al. [Ballan et al. 2012] show how to fit high polygon hand models to multiple camera views of a pair of hands interacting with a small sphere, using a combination of feature-based tracking and energy minimization. In contrast to our method, their approach relies upon inter-frame correspondences to provide optical-flow and good starting poses for energy minimization.

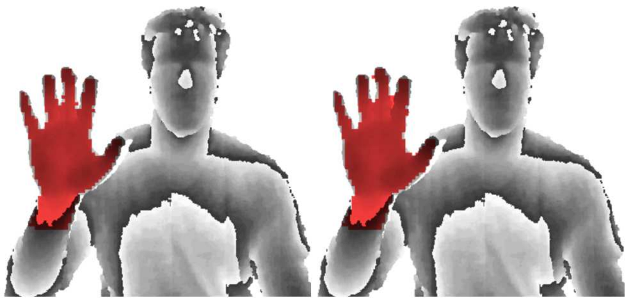
Early work by Rehg and Kanade [Rehg and Kanade 1994] demonstrated a model-based tracking system that fits a high-degree of freedom articulated hand model to greyscale image data using hand-designed 2D features. Zhao et al. [Zhao et al. 2012] use a combination of IR markers and RGBD capture to infer offline (at one frame per second) the pose of an articulated hand model. Similar to this work, Oikonomidis et al. [Oikonomidis et al. 2011] demonstrate the utility of Particle Swarm Optimization (PSO) for tracking single and interacting hands by searching for parameters of an explicit 3D model that reduce the reconstruction error of a z-buffer rendered model compared to an incoming depth image. Their work relies heavily on temporal coherence assumptions for efficient inference of the PSO optimizer, since the radius of convergence of their optimizer is finite. Unfortunately, temporal coherence cannot be relied on for robust real-time tracking since dropped frames and fast moving objects typically break this temporal coherency assumption. In contrast to their work, which used PSO directly for interactive tracking on the GPU at 4-15fps, our work shows that with relaxed temporal coherence assumptions in an offline setting, PSO is an invaluable *offline* tool for generating labeled data.

To our knowledge, there is no published prior work on using ConvNets to recover continuous 3D pose of human hands from depth data. However, several groups have shown ConvNets can recover the pose of rigid and non-rigid 3D objects such as plastic toys, faces and even human bodies. For example, LeCun et al. [LeCun et al. 2004] used ConvNets to deduce the 6DOF pose of 3D plastic toys by finding a low-dimensional embedding which maps RGB images to a six dimensional space. Osadchy et al. [Osadchy et al. 2005] use a similar formulation to perform pose detection of faces via a non-linear mapping to a low-dimensional manifold. Taylor et al. [Taylor et al. 2011] use crowd-sourcing to build a database of similar human poses from different subjects and then use ConvNets to perform dimensionality reduction to a low-dimensional manifold, where similarity between training examples is preserved. Lastly, Jiu et al. [Jiu et al. 2013] use ConvNets to perform per-pixel classifications of depth images (whose output is similar to [Shotton et al. 2011]) in order to infer human body pose, but they do not evaluate the performance of their approach on hand pose recognition.

Couprie et al. [Couprie et al. 2013] use ConvNets to perform image segmentation of indoor scenes using RGB-D data. The significance of their work is that it shows that ConvNets can perform high level reasoning from depth image features.

3. BINARY CLASSIFICATION

For the task of hand-background depth image segmentation we trained an RDF classifier to perform per-pixel binary segmentation on a single image. The output of this stage is shown in Figure 2. Decision forests are well-suited for discrete classification of



(a) Ground-Truth Labels (b) Labels Inferred by RDF

Fig. 2: Decision Forest Data: learned labels closely match target

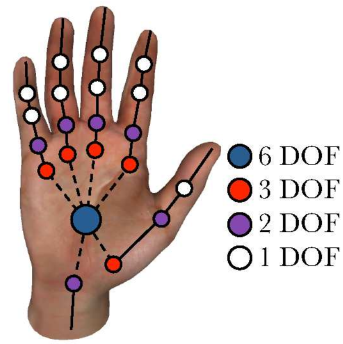


Fig. 3: Linear Blend Skinning (LBS) Model [Šarić 2011] with 42 DOF

body parts [Shotton et al. 2011]. Furthermore, since decision forest classification is trivially parallelizable, it is well-suited to real-time processing in multi-core environments.

After Shotton et al., our RDF is designed to classify each pixel in a depth image as belonging to a hand or background. Each tree in the RDF consists of a set of sequential deterministic decisions, called weak-learners (or nodes), that compare the relative depth of the current pixel to a nearby pixel located at a learned offset. The particular sequence of decisions a pixel satisfies induces a tentative classification into hand or background. Averaging the classification from all trees in the forest induces a final probability distribution for each pixel. As our implementation differs only slightly from that of Shotton et al., we refer interested readers to their past work, and focus on the innovations particular to our implementation.

While full body motion capture datasets are readily available [Allen et al. 2003], these datasets either lack articulation data for hands or else do not adequately cover the wide variety of poses that were planned for this system. Therefore, it was necessary to create a custom database of full body depth images with binary hand labeling for RDF training (Figure 2). We had one user paint their hands bright red with body paint and used a simple HSV-based distance metric to estimate a coarse hand labeling on the RGB image. The coarse labeling is then filtered using a median filter to remove outliers. Since commodity RGB+Depth (RGBD) cameras, typically exhibit imperfect alignment between depth and RGB, we used a combination of graph cut and depth-sensitive flood fill to further clean up the depth image labeling [Boykov et al. 2001].

In order to train the RDF we randomly sample weak-learners from a family of functions similar to [Shotton et al. 2011]. At a given pixel (u, v) on the depth image I each node in the decision tree evaluates:

$$I\left(u + \frac{\Delta u}{I\left(u, v\right)}, v + \frac{\Delta v}{I\left(u, v\right)}\right) - I\left(u, v\right) \geq d_t \qquad (1)$$

Where $I\left(u, v\right)$ is the depth pixel value in image I, Δu and Δv are learned pixel offsets, and d_t is a learned depth threshold. Experimentally, we found that (1) requires a large dynamic range of pixel offsets during training to achieve good classification performance. We suspect that this is because a given decision path needs to use both global and local geometry information to perform efficient hand-background segmentation. Since training time is limited, we define a discrete set of weak-learners that use offset and threshold values that are linear in log space and then we randomly sample weak-learners from this space during training.

4. DATASET CREATION

The goal of this stage is to create a database of RGBD sensor images representing a broad range of hand gestures with accurate ground-truth estimates (i.e. labels) of joint parameters in each image that may be used to train a ConvNet. The desired ground-truth label consists of a 42-dimensional vector describing the full degree of freedom pose for the hand in that frame. The DOF of each hand-joint is shown in Figure 3. After the hand has been segmented from the background using the RDF-based binary classification just described, we use a direct search method to deduce the pose parameters based on the approach of Oikonomidis et al. [Oikonomidis et al. 2011]. An important insight of our work is that we can capture the power of their direct search method in an offline fashion, and then use it to train ConvNets (or similar algorithms) that are better suited to fast computation. One advantage of this decoupling is that during offline training we are not penalized for using more complicated models, which are more expensive to render, and which better explain the incoming range data. A second advantage is that we can utilize multiple sensors for training, thereby mitigating problems of self-occlusion during real-time interaction with a single sensor.

The algorithm proposed by Oikonomidis et al. [Oikonomidis et al. 2011] and adopted with modifications for this work is as follows; starting with an approximate hand pose, a synthetic depth image is rendered and compared to the depth image using an scalar objective function. This depth image is rendered in an OpenGL-based framework, where the only render output is the distance from the camera origin and we use a camera with the same properties (e.g. focal length) as the PrimeSense$^{\text{TM}}$IR sensor. In practice the hand pose is estimated using the previous frame's pose when fitting a sequence of recorded frames. The pose is manually estimated using a simple UI for the first frame in a sequence. This results in a single scalar value representing the quality of the fit given the estimated pose coefficients. The particle swarm optimization with partial randomization (PrPSO) direct search method [Yasuda et al. 2010] is used to adjust the pose coefficient values to find the best fit pose that minimizes this objective function value. An overview of this algorithm is shown in Figure 4.

Since PSO convergence is slow once the swarm positions are close to the final solution (which is exacerbated when partial randomization is used to prevent premature swarm collapse on early local minima), we then use a robust variant of the Nelder-Mead optimization algorithm [Tseng 1995] after PSO has completed. The Nelder-Mead optimization algorithm is a simplex-based direct-search optimization algorithm for non-linear functions. We have found that for our optimization problem, it provides fast convergence when sufficiently close to local optima.

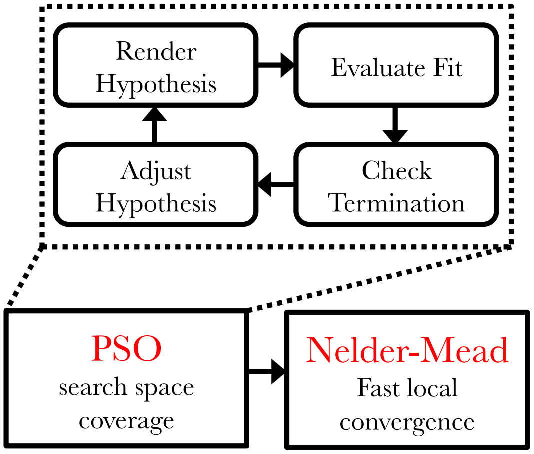


Fig. 4: Algorithm Pipeline For Dataset Creation

Since this dataset creation stage is performed offline, we do not require it to be fast enough for interactive frame rates. Therefore we used a high-quality, linear-blend-skinning (LBS) model [Šarić 2011] (shown in Figure 3) as an alternative to the simple ball-and-cylinder model of Oikonomidis et al. After reducing the LBS model's face count to increase render throughput, the model contains 1,722 vertices and 3,381 triangle faces, whereas the high density source model contained 67,606 faces. While LBS fails to accurately model effects such as muscle deformation and skin folding, it represents many geometric details that ball-and-stick models cannot.

To mitigate the effects of self-occlusion we used three sensors (at viewpoints separated by approximately 45 degrees surrounding the user from the front), with attached vibration motors to reduce IR-pattern interference [Butler et al. 2012] and whose relative positions and orientations were calibrated using a variant of the Iterative Closest Point (ICP) algorithm [Horn 1987]. While we use all three camera views to fit the LBS model using the algorithm described above, we only use depth images taken from the center camera to train the ConvNet. The contributions from each camera were accumulated into an overall fitness function $F(C)$, which includes two *a priori* terms ($\Phi(C)$ and $P(C)$) to maintain anatomically correct joint angles as well as a data-dependant term $\Delta(I_s, C)$ from each camera's contribution. The fitness function is as follows:

$$F(C) = \sum_{s=1}^{3} \Big(\Delta(I_s, C) \Big) + \Phi(C) + P(C) \qquad (2)$$

Where I_s is the s sensor's depth image and C is a 42-dimensional coefficient vector that represents the 6 DOF position and orientation of the hand as well as 36 internal joint angles (shown in Figure 3). $P(C)$ is an interpenetration term (for a given pose) used to invalidate anatomically incorrect hand poses and is calculated by accumulating the interpenetration distances of a series of bounding spheres attached to the bones of the 3D model. We define interpenetration distance as simply the sum of overlap between all pairs of interpenetrating bounding spheres. $\Phi(C)$ enforces a soft constraint that coefficient values stay within a predetermined range (C_{min} and C_{max}):

$$\Phi(C) = \sum_{k=1}^{n} w_k \left[\max\left(C_k - C_{k\text{max}}, 0\right) + \max\left(C_{k\text{min}} - C_k, 0\right) \right]$$

Where, w_k is a per-coefficient weighting term to normalize penalty contributions across different units (since we are including error terms for angle and distance in the same objective function). C_{min} and C_{max} were determined experimentally by fitting an unconstrained model to a discrete set of poses which represent the full range of motion for each joint. Lastly $\Delta(I_s, C)$ of Equation (2), measures the similarity between the depth image I_s and the synthetic pose rendered from the same viewpoint:

$$\Delta(I_s, C) = \sum_{u,v} \min\left(\left| I_s(u,v) - R_s(C,u,v) \right|, d_{\text{max}} \right)$$

Where, $I_s(u,v)$ is the depth at pixel (u,v) of sensor s, $R_s(C,u,v)$ is the synthetic depth given the pose coefficient C and d_{max} is a maximum depth constant. The result of this function is a clamped L1-norm pixel-wise comparison. It should be noted that we do not include energy terms that measure the silhouette similarity as proposed by Oikonomidis et al. since we found that when multiple range sensors are used these terms are not necessary.

5. FEATURE DETECTION

While Neural Networks have been used for pose detection of a limited set of discrete hand gestures (for instance discriminating between a closed fist and an open palm) [Nagi et al. 2011; Nowlan and Platt 1995], to our knowledge this is the first work that has attempted to use such networks to perform dense feature extraction of human hands in order to infer continuous pose. To do this we employ a multi-resolution, deep ConvNet architecture inspired by the work of Farabet et al. [Farabet et al. 2013] in order to perform feature extraction of 14 salient hand points from a segmented hand image. ConvNets are biologically inspired variants of multi-layered perceptrons, which exploit spatial correlation in natural images by extracting features generated by localized convolution kernels. Since depth images of hands tend to have many repeated local image features (for instance fingertips), ConvNets are well suited to perform feature extraction since multi-layered feature banks can share common features, thereby reducing the number of required free parameters.

We recast the full hand-pose recognition problem as an intermediate collection of easier individual hand-feature recognition problems, which can be more easily learned by ConvNets. In early experiments we found inferring mappings between depth image space and pose space directly (for instance measuring depth image geometry to extract a joint angle), yielded inferior results to learning with intermediate features. We hypothesize that one reason for this could be that learning intermediate features allows ConvNets to concentrate the capacity of the network on learning local features, and on differentiating between them. Using this framework the ConvNet is also better able to implicitly handle occlusions; by learning compound, high-level image features the ConvNet is able to infer the approximate position of an occluded and otherwise unseen feature (for instance, when making a fist, hidden finger-tip locations can be inferred by the knuckle locations).

We trained the ConvNet architecture to generate an output set of *heat-map* feature images (Figure 5). Each feature heat-map can be viewed as a 2D Gaussian (truncated to have finite support), where the pixel intensity represents the probability of that feature occurring in that spatial location. The Gaussian UV mean is centered at one of 14 feature points of the user's hand. These features represent key joint locations in the 3D model (e.g., knuckles) and were chosen such that the inverse kinematics (IK) algorithm described in Section 6 can recover a full 3D pose.

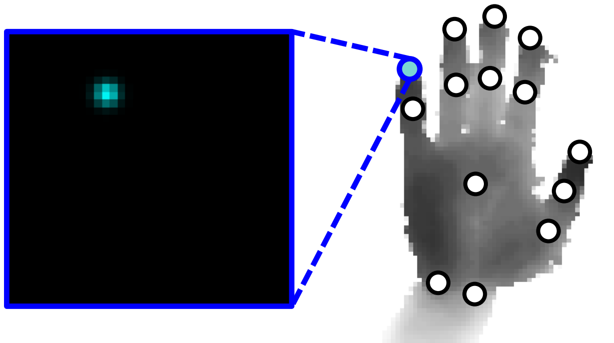


Fig. 5: Depth image overlaid with 14 feature locations and the heat-map for one fingertip feature.

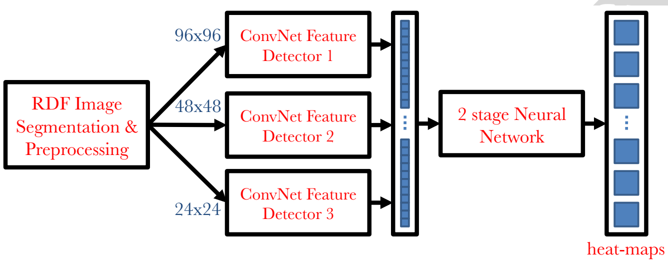
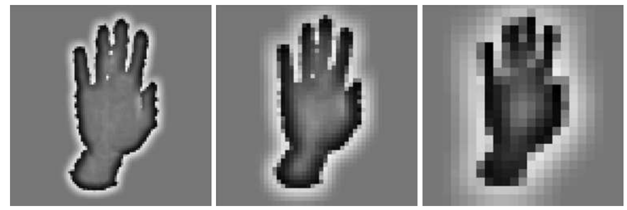


Fig. 6: Convolutional Network Architecture

We found that the intermediate heat-map representation not only reduces required learning capacity but also improves generalization performance since failure modes are often recoverable. Cases contributing to high test set error (where the input pose is vastly different from anything in the training set) are usually heat-maps that contain multiple hotspots. For instance, the heat-map for a fingertip feature might incorrectly contain multiple lobes corresponding to the other finger locations as the network failed to discriminate among fingers. When this situation occurs it is possible to recover a reasonable feature location by simple heuristics to decide which of these lobes corresponds to the desired feature (for instance if another heatmap shows higher probability in those same lobe regions then we can eliminate these as spurious outliers). Similarly, the intensity of the heat-map lobe gives a direct indication of the system's confidence for that feature, which is an extremely useful measure for practical applications.

Our multi-resolution ConvNet architecture is shown in Figure 6. The segmented depth image is initially pre-processed, whereby the image is cropped and scaled by a factor proportional to the mean depth value of the hand pixels, so that the hand is in the center and has size that is depth invariant. The depth values of each pixel are then normalized between 0 and 1 (with background pixels set to 1). The cropped and normalized image is shown in Figure 5.

The preprocessed image is then filtered using local contrast normalization [Jarrett et al. 2009], which acts as a high-pass filter to emphasize geometric discontinuities. The image is then downsampled twice (each time by a factor of 2) and the same filter is applied to each image. This produces a multi-resolution band-pass image pyramid with 3 banks (shown in Figure 7), whose total spectral density approximates the spectral density of the input depth image. Since experimentally we have found that hand-pose extraction requires knowledge of both local and global features, a single



(a) 96×96px (b) 48×48px (c) 24×24px

Fig. 7: Neural Network Input: Multi-Resolution Image Pyramid

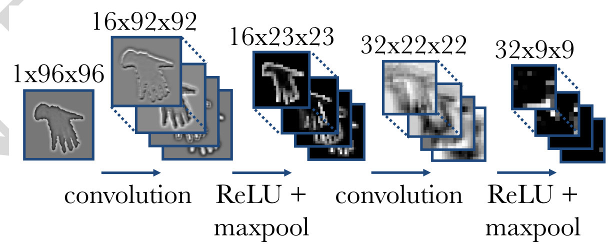


Fig. 8: High-Resolution Bank Feature Detector: each stage: (N_{features} × height × width)

resolution ConvNet would need to examine a large image window and thus would require a large learning capacity; as such a multi-resolution architecture is very useful for this application.

The pyramid images are propagated through a 2-stage ConvNet architecture. The highest resolution feature bank is shown in Figure 8. Each bank is comprised of 2 convolution modules, 2 piecewise non-linearity modules, and 2 max-pooling modules. Each convolution module uses a stack of learned convolution kernels with an additional learned output bias to create a set of output feature maps (please refer to [LeCun et al. 1998] for an in-depth discussion). The convolution window sizes range from 4x4 to 6x6 pixels. Each max-pooling [Nagi et al. 2011] module sub-samples it's input image by taking the maximum in a set of non-overlapping rectangular windows. We use max-pooling since it effectively reduces computational complexity at the cost of spatial precision. The max-pooling windows range from 2x2 to 4x4 pixels. The nonlinearity is a Rectify Linear Unit (ReLU), which has been shown to improve training speed and discrimination performance in comparison to the standard sigmoid units [Krizhevsky et al. 2012]. Each ReLU activation module computes the following per-pixel non-linear function:

$$f(x) = \max(0, x)$$

Lastly, the output of the ConvNet banks are fed into a 2-stage neural network shown in Figure 9. This network uses the high-level convolution features to create the final 14 heat-map images; it does so by learning a mapping from localized convolution feature activations to probability maps for each of the bone features. In practice these two large and fully-connected linear networks account for more than 80% of the total computational cost of the ConvNet. However, reducing the size of the network has a very strong impact on runtime performance. For this reason, it is important to find a good tradeoff between quality and speed. Another drawback of this method is that the neural network must implicitly learn a likelihood model for joint positions in order to infer anatomically correct output joints. Since we do not explicitly model joint connectivity in the network structure, the network requires a large amount of training data to perform this inference correctly.

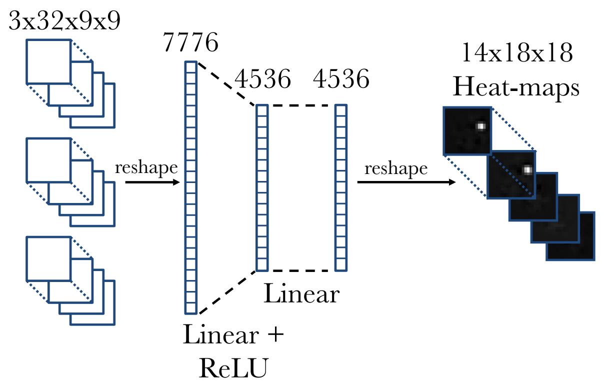


Fig. 9: 2-Stage Neural Network To Create The 14 Heat Maps (with sizing of each stage shown)

ConvNet training was performed using the open-source machine learning package Torch7 [Collobert et al. 2011], which provides access to an efficient GPU implementation of the back-propagation algorithm for training neural networks. During supervised training we use stochastic gradient descent with a standard L2-norm error function, batch size of 64 and the following learnable parameter update rule:

$$\Delta w_i = \gamma \Delta w_{i-1} - \lambda \left(\eta w_i - \frac{\partial L}{\partial w_i} \right)$$
$$w_{i+1} = w_i + \Delta w_i \tag{3}$$

Where w_i is a bias or weight parameter for each of the network modules for epoch i (with each epoch representing one pass over the entire training-set) and $\frac{\partial L}{\partial w_i}$ is the partial derivative of the error function L with respect to the learnable parameter w_i averaged over the current batch. We use a constant learning rate of $\lambda = 0.2$, and a momentum term $\gamma = 0.9$ to improve learning rate when close to the local minimum. Lastly, an L2 regularization factor of $\eta = 0.0005$ is used to help improve generalization.

During ConvNet training the pre-processed database images were randomly rotated, scaled and translated to improve generalization performance [Farabet et al. 2013]. Not only does this technique effectively increase the size of the training set (which improves test / validation set error), it also helps improve performance for other users whose hand size is not well represented in the original training set. We perform this image manipulation in a background thread during batch-training so the impact on training time is minimal.

6. POSE RECOVERY

We formulate the problem of pose estimation from the heat-map output as an optimization problem, similar to inverse kinematics (IK). We extract 2D and 3D feature positions from the 14 heat-maps and then minimize an appropriate objective function to align 3D model features to each heat-map position.

To infer the 3D position corresponding to a heat-map image, we need to determine the most likely UV position of the feature in the heatmap. Although the ConvNet architecture is trained to output heat-map images of 2D Gaussians with low-variance, in general, they output multimodal grayscale heat-maps which usually do not sum to 1. In practice, it is easy to deduce a correct UV position by finding the maximal peak in the heat-map (corresponding to the location of greatest confidence). Rather than use the most likely heat-map location as the final location, we fit a Gaussian model to the maximal lobe to obtain sub-pixel accuracy.

First we clamp heat-map pixels below a fixed threshold to get rid of spurious outliers. We then normalize the resulting image so it sums to 1, and we fit the best 2D Gaussian using Levenberg-Marquardt, and use the mean of the resulting Gaussian as the UV position. Once the UV position is found for each of the 14 heat-maps, we perform a lookup into the captured depth frame to obtain the depth component at the UV location. In case this UV location lies on a depth shadow where no depth is given in the original image, we store the computed 2D image for this point in the original image space. Otherwise, we store its 3D point.

We then perform unconstrained nonlinear optimization on the following objective function:

$$f(m) = \sum_{i=1}^{n} \left[\Delta_i (m) \right] + \Phi (C) \tag{4}$$
$$\Delta_i (m) = \begin{cases} \left\| (u,v,d)_i^t - (u,v,d)_i^m \right\|_2 & \text{If } d_i^t \neq 0 \\ \left\| (u,v)_i^t - (u,v)_i^m \right\|_2 & \text{otherwise} \end{cases}$$

Where $(u,v,d)_i^t$ is the target 3D heat-map position of feature i and $(u,v,d)_i^m$ is the model feature position for the current pose estimate. Equation (4) is an L2 error norm in 3D or 2D depending on whether or not the given feature has a valid depth component associated with it. We then use a simple linear accumulation of these feature-wise error terms as well as the same linear penalty constraint $(\Phi (C))$ used in Section 4. We use PrPSO to minimize Equation (4). Since function evaluations for each swarm particle can be parallelized, PrPSO is able to run in real-time at interactive frame rates for this stage. Furthermore, since a number of the 42 coefficients from Section 4 contribute only subtle behavior to the deformation of the LBS model at real time, we found that removing coefficients describing finger twist and coupling the last two knuckles of each finger into a single angle coefficient significantly reduces function evaluation time of (4) without noticeable loss in pose accuracy. Therefore, we reduce the complexity of the model to 23 DOF during this final stage. Fewer than 50 PrPSO iterations are required for adequate convergence.

This IK approach has one important limitation; the UVD target position may not be a good representation of the true feature position. For instance, when a feature is directly occluded by another feature, the two features will incorrectly share the same depth value (even though one is in front of the other). However, we found that for a broad range of gestures this limitation was not noticeable. In future work we hope to augment the ConvNet output with a learned depth offset to overcome this limitation.

7. RESULTS

For the results to follow, we test our system using the same experimental setup that was used to capture the training data; the camera is in front of the user (facing them) and is at approximately eye level height. We have not extensively evaluated the performance of our algorithm in other camera setups.

The RDF classifier described in Section 4 was trained using 6,500 images (with an additional 1,000 validation images held aside for tuning of the RDF meta-parameters) of a user performing typical one and two handed gestures (pinching, drawing, clapping, grasping, etc). Training was performed on a 24 core machine for approximately 12 hours. For each node in the tree, 10,000 weak-learners were sampled. The error ratio of the number of incorrect

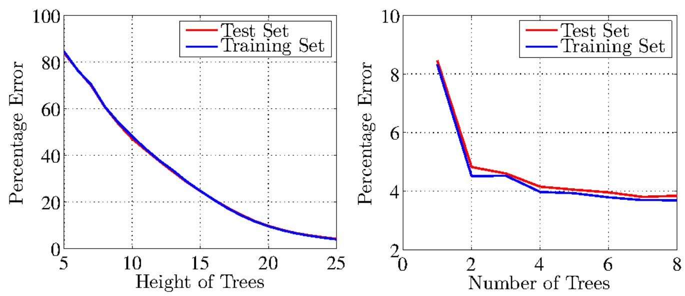


Fig. 10: RDF Error

(a) Sensor Depth (b) Synthetic Depth (c) Per-Pixel Error

Fig. 11: Dataset Creation: Objective Function Data (with Libhand model [Šarić 2011])

pixel labels to total number of hand pixels in the dataset for varying tree counts and tree heights is shown in Figure 10.

We found that 4 trees with a height of 25 was a good tradeoff of classification accuracy versus speed. The validation set classification error for 4 trees of depth 25 was 4.1%. Of the classification errors, 76.3% were false positives and 23.7% were false negatives. We found that in practice small clusters of false positive pixel labels can be easily removed using median filtering and blob detection. The common classification failure cases occur when the hand is occluded by another body-part (causing false positives), or when the elbow is much closer to the camera than the hand (causing false positives on the elbow). We believe this inaccuracy results from the training set not containing any frames with these poses. A more comprehensive dataset, containing examples of these poses, should improve performance in future.

Since we do not have a ground-truth measure for the 42 DOF hand model fitting, quantitative evaluation of this stage is difficult. *Qualitatively*, the fitting accuracy was visually consistent with the underlying point cloud. An example of a fitted frame is shown in Figure 11. Only a very small number of poses failed to fit correctly; for these difficult poses, manual intervention was required.

One limitation of this system was that the frame rate of the PrimeSense™camera (30fps) was not enough to ensure sufficient temporal coherence for correct convergence of the PSO optimizer. To overcome this, we had each user move their hands slowly during training data capture. Using a workstation with an Nvidia GTX 580 GPU and 4 core Intel processor, fitting each frame required 3 to 6 seconds. The final database consisted of 76,712 training set images, 2,421 validation set images and 2,000 test set images with their corresponding heat-maps, collected from multiple participants. A small sample of the test set images is shown in Figure 12.

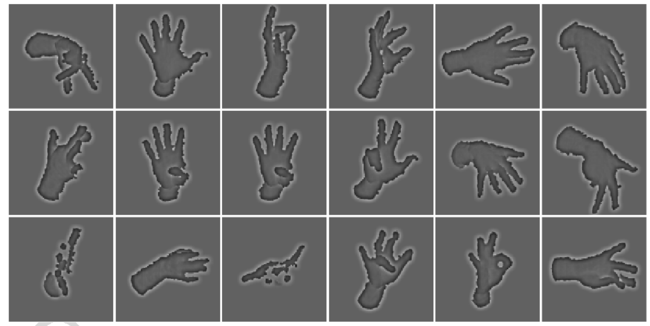


Fig. 12: Sample ConvNet Test Set images

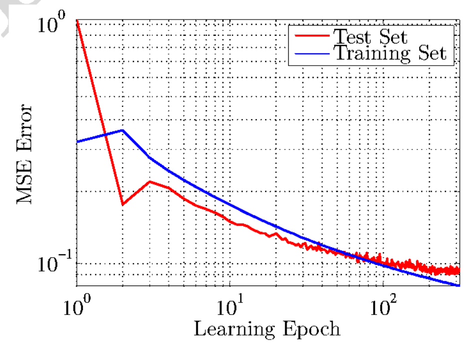


Fig. 13: ConvNet Learning Curve

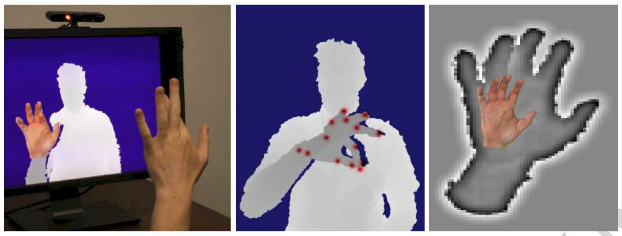
The ConvNet training took approximately 24 hours where early stopping is performed after 350 epochs to prevent overfitting. ConvNet hyper parameters, such as learning rate, momentum, L2 regularization, and architectural parameters (e.g., max-pooling window size or number of stages) were chosen by coarse meta-optimization to minimize a validation-set error. 2 stages of convolution (at each resolution level) and 2 full-connected neural network stages were chosen as a tradeoff between numerous performance characteristics: generalization performance, evaluation time, and model complexity (or ability to infer complex poses). Figure 13 shows the mean squared error (MSE) after each epoch. The MSE was calculated by taking the mean of sum-of-squared differences between the calculated 14 feature maps and the corresponding target feature maps.

The mean UV error of the ConvNet heat-map output on the test set data was 0.41px (with standard deviation of 0.35px) on the 18x18 resolution heat-map image[1]. After each heat-map feature was translated to the 640x480 depth image, the mean UV error was 5.8px (with standard deviation of 4.9px). Since the heat-map downsampling ratio is depth dependent, the UV error improves as the hand approaches the sensor. For applications that require greater

[1]To calculate this error we used the technique described in Section 6 to calculate the heat-map UV feature location and then calculated the error distance between the target and ConvNet output locations

Table I. : Heat-Map UV Error by Feature Type

Feature Type	Mean (px)	STD (px)
Palm	0.33	0.30
Thumb Base & Knuckle	0.33	0.43
Thumb Tip	0.39	0.55
Finger Knuckle	0.38	0.27
Finger Tip	0.54	0.33



(a) (b) (c)

Fig. 14: Real-Time Tracking Results, a) Typical Hardware Setup, b) Depth with Heat-Map Features, c) ConvNet Input and Pose Output

accuracy, the heat-map resolution can be increased for better spatial accuracy at the cost of increased latency and reduced throughput.

Table I shows the UV accuracy for each feature type. Unsurprisingly, we found that the ConvNet architecture had the most difficulty learning fingertip positions, where the mean error is 61% higher than the accuracy of the palm features. The likely cause for this inaccuracy is twofold. Firstly, the fingertip positions undergo a large range of motion between various hand-poses and therefore the ConvNet must learn a more difficult mapping between local features and fingertip positions. Secondly, the PrimeSense^{TM}Carmine 1.09 depth camera cannot always recover depth of small surfaces such as fingertips. The ConvNet is able to learn this noise behavior, and is actually able to approximate fingertip location in the presence of missing data, however the accuracy for these poses is low.

The computation time of the entire pipeline is 24.9ms, which is within our 30fps performance target. Within this period: decision forest evaluation takes 3.4ms, depth image preprocessing takes 4.7ms, ConvNet evaluation takes 5.6ms and pose estimation takes 11.2ms. The entire pipeline introduces approximately one frame of latency. For an example of the entire pipeline running in real-time as well as puppeteering of the LBS hand model, please refer to the supplementary video (screenshots from this video are shown in Figure 14).

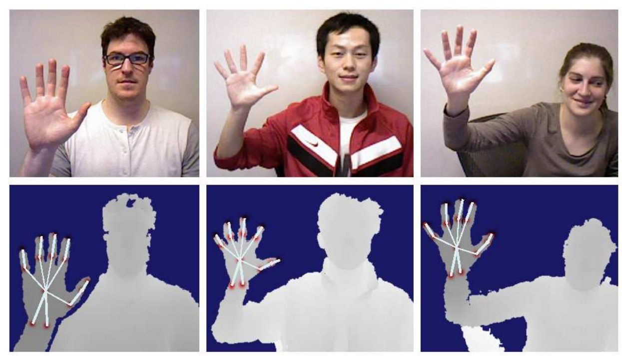
Figure 15 shows three typical fail cases of our system. In 15a) finite spatial precision of the ConvNet heat-map results in finger tip positions that are not quite touching. In 15b) no similar pose exists in the database used to train the ConvNet, and for this example the network generalization performance was poor. In 15c) the PrimeSense^{TM}depth camera fails to detect the ring finger (the surface area of the finger tip presented to the camera is too small and the angle of incidence in the camera plane is too shallow), and the ConvNet has difficulty inferring the finger tip position without adequate support in the depth image, which results in an incorrect inferred position.

Figure 16 shows that the ConvNet output is tolerant for hand shapes and sizes that are not well represented in the ConvNet training set. The ConvNet and RDF training sets did not include any images for user b) and user c) (only user a)). We have only evaluated the system's performance on adult subjects. We found that

a) b) c)

Fig. 15: Fail Cases: RGB ground-truth (top row), inferred model [Šarić 2011] pose (bottom row)

adding a single per-user scale parameter to approximately adjust the size of the LBS model to a user's hand, helped the real-time IK stage better fit to the ConvNet output.



a) b) c)

Fig. 16: Hand Shape/Size Tolerance: RGB ground-truth (top row), depth with annotated ConvNet output positions (bottom row)

Comparison of the relative real-time performance of this work with relevant prior art, such as that of [3Gear 2014] and [Melax et al. 2013], is difficult for a number of reasons. Firstly, [Melax et al. 2013] uses a different capture device, which prevents fair comparison as it is impossible (without degrading sensor performance by using mirrors) for multiple devices to see the hand from the same viewpoint simultaneously. Secondly, no third-party ground truth database of poses with depth frames exists for human hands, so comparing the quantitative accuracy of numerous methods against a known baseline is not possible. More importantly however, is that the technique utilized by [3Gear 2014] is optimized for an entirely different use case and so fair comparison with their work is very difficult. [3Gear 2014] utilizes a vertically mounted camera, can track multiple hands simultaneously, and is computationally less expensive than the method presented in our work.

Figure 17 examines the performance of this work with the proprietary system of [3Gear 2014] (using the fixed-database version of the library), for 4 poses chosen to highlight the relative difference between the two techniques (images used with permission from 3Gear). We captured this data by streaming the output of both systems simultaneously (using the same RGBD camera). We mounted the camera vertically, as this is required for [3Gear 2014], however our training set did not include any poses from this orientation. Therefore, we expect our system to perform sub-optimally for this very different use case.

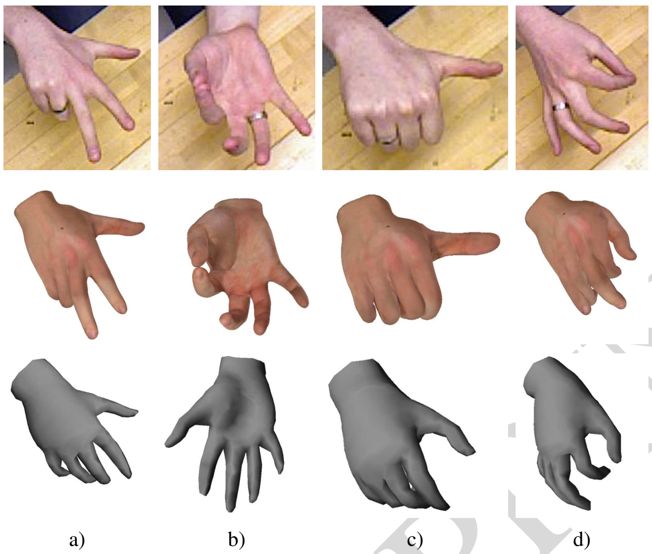


Fig. 17: Comparison with state-of-the-art commercial system: RGB ground-truth (top row), this work inferred model [Šarić 2011] pose (middle row), [3Gear 2014] inferred model pose (bottom row) (images used with permission from 3Gear).

8. FUTURE WORK

As indicated in Figure 16, qualitatively we have found that the ConvNet generalization performance to varying hand shapes is acceptable but could be improved. We are confident we can make improvements by adding more training data from users with different hand sizes to the training set.

For this work, only the ConvNet forward propagation stage was implemented on the GPU. We are currently working on implementing the entire pipeline on the GPU, which should improve performance of the other pipeline stages significantly. For example, the GPU ConvNet implementation requires 5.6ms, while the same network executed on the CPU (using optimized multi-threaded C++ code) requires 139ms.

The current implementation of our system can track two hands only if they are not interacting. While we have determined that the dataset generation system can fit multiple strongly interacting hand poses with sufficient accuracy, it is future work to evaluate the neural network recognition performance on these poses. Likewise, we hope to evaluate the recognition performance on hand poses involving interactions with non-hand objects (such as pens and other man-made devices).

While the pose recovery implementation presented in this work is fast, we hope to augment this stage by including a model-based fitting step that trades convergence radius for fit quality. Specifically, we suspect that replacing our final IK stage with an energy-based local optimization method, inspired by the work of Li et al. [Li et al. 2008] could allow our method to recover second-order surface effects such as skin folding and skin-muscle coupling from very limited data, and still with low-latency. In addition to inference, such a localized energy-minimizing stage would enable improvements to the underlying model itself. Since these localized methods typically require good registration, our method, which gives correspondence from a single image could advance the state-of-the-art in non-rigid model capture.

Finally, we hope to augment our final IK stage with some form of temporal pose prior to reduce jitter; for instance, using an extended Kalman filter as a post-processing step to clean up the ConvNet feature output.

9. CONCLUSION

We have presented a novel pipeline for tracking the instantaneous pose of articulable objects from a single depth image. As an application of this pipeline we showed state-of-the-art results for tracking human hands in real-time using commodity hardware. This pipeline leverages the accuracy of offline model-based dataset generation routines in support of a robust real-time convolutional network architecture for feature extraction. We showed that it is possible to use intermediate heat-map features to extract accurate and reliable 3D pose information at interactive frame-rates using inverse kinematics.

REFERENCES

3GEAR. 2014. 3gear sytems hand-tracking development platform, http://www.threegear.com/.

ALLEN, B., CURLESS, B., AND POPOVIĆ, Z. 2003. The space of human body shapes: reconstruction and parameterization from range scans. *ACM Trans. Graph. 22,* 3, 587–594.

BALLAN, L., TANEJA, A., GALL, J., VAN GOOL, L., AND POLLEFEYS, M. 2012. Motion capture of hands in action using discriminative salient points. In *Proceedings of the 12th European conference on Computer Vision*. 640–653.

BOYKOV, Y., VEKSLER, O., AND ZABIH, R. 2001. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell. 23,* 11 (Nov.), 1222–1239.

BUTLER, D. A., IZADI, S., HILLIGES, O., MOLYNEAUX, D., HODGES, S., AND KIM, D. 2012. Shake'n'sense: reducing interference for overlapping structured light depth cameras. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*. 1933–1936.

COLLOBERT, R., KAVUKCUOGLU, K., AND FARABET, C. 2011. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*.

COUPRIE, C., FARABET, C., NAJMAN, L., AND LECUN, Y. 2013. Indoor semantic segmentation using depth information. In *International Conference on Learning Representations*.

EROL, A., BEBIS, G., NICOLESCU, M., BOYLE, R. D., AND TWOMBLY, X. 2007. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*.

FARABET, C., COUPRIE, C., NAJMAN, L., AND LECUN, Y. 2013. Learning hierarchical features for scene labeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 35,* 8, 1915–1929.

HORN, B. K. P. 1987. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America 4,* 4, 629–642.

JARRETT, K., KAVUKCUOGLU, K., RANZATO, M., AND LECUN, Y. 2009. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*. 2146–2153.

JIU, M., WOLF, C., TAYLOR, G. W., AND BASKURT, A. 2013. Human body part estimation from depth images via spatially-constrained deep learning. *Pattern Recognition Letters*.

KESKIN, C., KIRAC, F., KARA, Y., AND AKARUN, L. 2011. Real time hand pose estimation using depth sensors. In *Computer Vision Workshops, 2011 IEEE International Conference on*. 1228–1234.

KESKIN, C., KIRAC, F., KARA, Y. E., AND AKARUN, L. 2012. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part VI*. ECCV'12. Springer-Verlag, Berlin, Heidelberg, 852–863.

KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. 1106–1114.

LECUN, Y., BOTTOU, L., BENGIO, Y., AND HAFFNER, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86,* 11, 2278–2324.

LECUN, Y., HUANG, F. J., AND BOTTOU, L. 2004. Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition. Proceedings of the 2004 IEEE Computer Society Conference on*. Vol. 2. II–97–104.

LI, H., SUMNER, R. W., AND PAULY, M. 2008. Global correspondence optimization for non-rigid registration of depth scans. *Computer Graphics Forum (Proc. SGP'08) 27,* 5 (July).

LI, H., YU, J., YE, Y., AND BREGLER, C. 2013. Realtime facial animation with on-the-fly correctives. *ACM Transactions on Graphics 32,* 4 (July).

MELAX, S., KESELMAN, L., AND ORSTEN, S. 2013. Dynamics based 3d skeletal hand tracking. In *Proceedings of the ACM Symposium on Interactive 3D Graphics and Games*.

NAGI, J., DUCATELLE, F., DI CARO, G., CIRESAN, D., MEIER, U., GIUSTI, A., NAGI, F., SCHMIDHUBER, J., AND GAMBARDELLA, L. 2011. Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *Signal and Image Processing Applications, 2011 IEEE International Conference on*. 342–347.

NOWLAN, S. J. AND PLATT, J. C. 1995. A convolutional neural network hand tracker. In *Advances in Neural Information Processing Systems 7*. 901–908.

OIKONOMIDIS, I., KYRIAZIS, N., AND ARGYROS, A. 2011. Efficient model-based 3d tracking of hand articulations using kinect. Proceedings of the British Machine Vision Conference.

OSADCHY, M., LECUN, Y., MILLER, M. L., AND PERONA, P. 2005. Synergistic face detection and pose estimation with energy-based model. In *In Advances in Neural Information Processing Systems (NIPS*. 1017–1024.

REHG, J. M. AND KANADE, T. 1994. Visual tracking of high dof articulated structures: an application to human hand tracking. In *Proceedings of the Third European Conference on Computer Vision*. 35–46.

SHOTTON, J., FITZGIBBON, A., COOK, M., SHARP, T., FINOCCHIO, M., MOORE, R., KIPMAN, A., AND BLAKE, A. 2011. Real-time human pose recognition in parts from single depth images. In *Computer Vision and Pattern Recognition. Proceedings of the 2011 IEEE Conference on*. 1297–1304.

STEIN, M., TOMPSON, J., XIAO, X., HENDEEE, C., ISHII, H., AND PERLIN, K. 2012. Arcade: a system for augmenting gesture-based computer graphic presentations. In *ACM SIGGRAPH 2012 Computer Animation Festival*. 77–77.

TAYLOR, G. W., SPIRO, I., BREGLER, C., AND FERGUS, R. 2011. Learning invariance through imitation. In *Computer Vision and Pattern Recognition. Proceedings of the 2011 IEEE Computer Society Conference on*. 2729–2736.

TSENG, P. 1995. Fortified-descent simplicial search method: A general approach. Tech. rep., SIAM Journal of Optimization.

ŠARIĆ, M. 2011. Libhand: A library for hand articulation. Version 0.9.

WANG, R., PARIS, S., AND POPOVIĆ, J. 2011. 6d hands: markerless hand-tracking for computer aided design. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. 549–558.

WANG, R. Y. AND POPOVIĆ, J. 2009. Real-time hand-tracking with a color glove. In *ACM SIGGRAPH 2009 papers*. 63:1–63:8.

WEISE, T., LI, H., GOOL, L. V., AND PAULY, M. 2009. Face/off: Live facial puppetry. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer animation*.

YASUDA, T., OHKURA, K., AND MATSUMURA, Y. 2010. Extended pso with partial randomization for large scale multimodal problems. In *World Automation Congress, 2010*. 1–6.

ZHAO, W., CHAI, J., AND XU, Y.-Q. 2012. Combining marker-based mocap and rgb-d camera for acquiring high-fidelity hand motion data. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '12. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 33–42.

Received ; accepted