First-Person Pose Recognition using Egocentric Workspaces

Grégory Rogez[1,2], James S. Supančič III[1], Deva Ramanan[1]
[1]Dept of Computer Science, University of California, Irvine, CA, USA
[2]Universidad de Zaragoza, Zaragoza, Spain
grogez@unizar.es grogez,jsupanci,dramanan@ics.uci.edu

Abstract

We tackle the problem of estimating the 3D pose of an individual's upper limbs (arms+hands) from a chest mounted depth-camera. Importantly, we consider pose estimation during everyday interactions with objects. Past work shows that strong pose+viewpoint priors and depth-based features are crucial for robust performance. In egocentric views, hands and arms are observable within a well defined volume in front of the camera. We call this volume an egocentric workspace. A notable property is that hand appearance correlates with workspace location. To exploit this correlation, we classify arm+hand configurations in a global egocentric coordinate frame, rather than a local scanning window. This greatly simplify the architecture and improves performance. We propose an efficient pipeline which 1) generates synthetic workspace exemplars for training using a virtual chest-mounted camera whose intrinsic parameters match our physical camera, 2) computes perspective-aware depth features on this entire volume and 3) recognizes discrete arm+hand pose classes through a sparse multi-class SVM. We achieve state-of-the-art hand pose recognition performance from egocentric RGB-D images in real-time.

1. Introduction

Understanding hand poses and hand-object manipulations from a wearable camera has potential applications in assisted living [23], augmented reality [6] and life logging [19]. As opposed to hand-pose recognition from third-person views, egocentric views may be more difficult due to additional occlusions (from manipulated objects, or self-occlusions of fingers by the palm) and the fact that hands interact with the environment and often leave the field-of-view. The latter necessitates constant re-initialization, precluding the use of a large body of hand trackers which typically perform well given manual initialization.

Previous work for egocentric hand analysis tends to rely on local 2D features, such as pixel-level skin classification [17, 18] or gradient-based processing of depth maps with scanning-window templates [25]. Our approach follows in the tradition of [25], who argue that near-field depth measures obtained from a egocentric-depth sensor considerably simplifies hand analysis. Interestingly, egocentric-depth is not "cheating" in the sense that humans make use of stereoscopic depth cues for near-field manipulations [7]. We extend this observation by building an explicit 3D map of the observable near-field workspace.

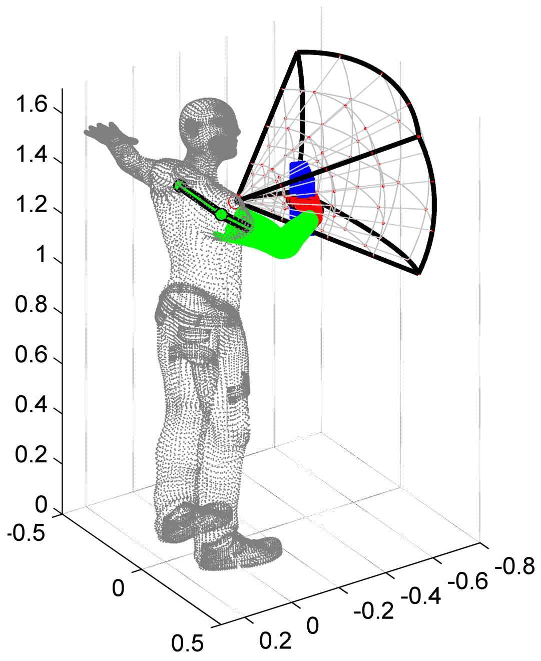

Figure 1. **Egocentric workspaces**. We directly model the observable egocentric workspace in front of a human with a 3D volumetric descriptor, extracted from a 2.5D egocentric depth sensor. In this example, this volume is discretized into $4 \times 3 \times 4$ bins. This feature can be used to accurately predict shoulder, arm, hand poses, even when interacting with objects. We describe models learned from synthetic examples of observable egocentric workspaces obtained by placing a virtual Intel Creative camera on the chest of an animated character.

Contributions: In this work, we describe a new computational architecture that makes use of **global** egocentric views, **volumetric** representations, and **contextual** models of interacting objects and human-bodies. Rather than detecting hands with a local (translation-invariant) scanning-window classifier, we process the entire global egocentric view (or *work-space*) in front of the observer (Fig. 1). Hand appearance is not translation-invariant due to perspective effects and kinematic constraints with the arm. To capture such effects, we build a library of synthetic 3D egocentric workspaces generated using real capture conditions (see examples in Fig. 2). We animate a 3D human character model inside virtual scenes with objects, and render such animations with a chest-mounted camera whose intrinsics match our physical camera . We simultaneously recognize arm and hand poses while interacting with objects by classifying the whole 3D volume using a multi-class Support Vector Machine (SVM) classifier. Recognition is simple and fast enough to be implemented in 4 lines of code.

1.1. Related work

Hand-object pose estimation: While there is a large body of work on hand-tracking [14, 13, 12, 1, 22, 31, 34], we focus on hand pose estimation during object manipulations. Object interactions both complicate analysis due to additional occlusions, but also provide additional contextual constraints (hands cannot penetrate object geometry, for example). [10] describe articulated tracker with soft anti-penetration constraints, increasing robustness to occlusion. Hamer *et al.* describe contextual priors for hands in relation to objects [9], and demonstrate their effectiveness for increasing tracking accuracy. Objects are easier to animate than hands because they have fewer joint parameters. With this intuition, object motion can be used as an input signal for estimating hand motions [8]. [26] use a large synthetic dataset of hands manipulating objects, similar to us. We differ in our focus on single-image and egocentric analysis.

Egocentric Vision: Previous studies have focused on activities of daily living [23, 5]. Long-scale temporal structure was used to handle complex hand object interactions, exploiting the fact that objects look different when they are manipulated (active) versus not manipulated (passive) [23]. Much previous work on egocentric hand recognition make exclusive use of RGB cues [18, 16], while we focus on volumetric depth cues. Notable exceptions include [3], who employ egocentric RGB-D sensors for personal workspace monitoring in industrial environments and [20], who employ such sensors to assist blind users in navigation.

Depth features: Previous work has shown the efficacy of depth cues [28, 35]. We compute volumetric depth features from point clouds. Previous work has examined point-cloud processing of depth-images [36, 29, 35]. A common technique estimates local surface orientations and normals [36, 35], but this may be sensitive to noise since it requires

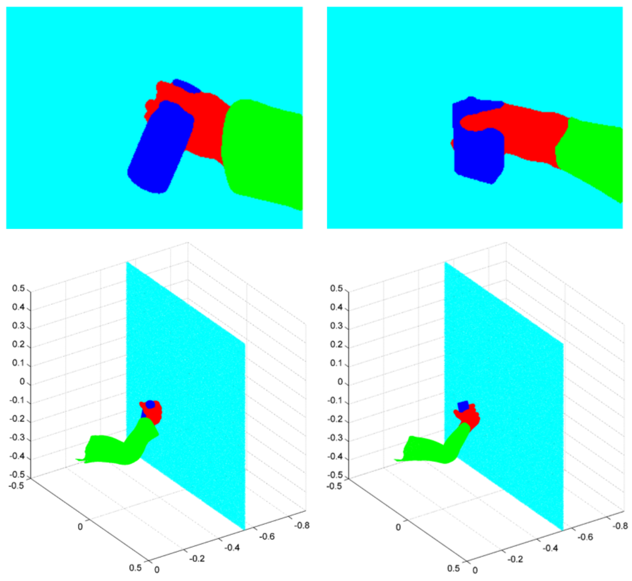

Figure 2. **Synthesis:** We generate a training set by sampling different dimensions of a workspace model, yielding a total number of $N_{arm} \times N_{hand} \times N_{object} \times N_{background}$ samples. We sample N_{arm} arm poses, a fixed set of hand-object configurations ($N_{hand} \times N_{object} = 100$) and a fixed set of $N_{background}$ background scenes captured with an Intel Creative depth camera. For each hand-object model, we randomly perturb shoulder, arm and hand joint angles to generate physically possible arm+hand+object configurations. We show 2 examples of a bottle-grasp (left) and a juice-box-grasp (right) rendered in front of a flat wall.

derivative computations. We use simpler volumetric features, similar to [30] except that we use a spherical coordinate frame that does not slide along a scanning window (we want to measure depth in an egocentric coordinate frame).

Non-parametric recognition: Our work is inspired by non-parametric techniques that make use of synthetic training data [26, 27, 10, 2, 33]. [27] make use of pose-sensitive hashing techniques for efficient matching of synthetic RGB images rendered with Poser. We generate synthetic depth images, mimicking capture conditions of our actual camera.

2. Training data

We begin by generating a training set of realistic 3D egocentric workspaces. Specifically, we render synthetic 3D hand-object data (generated from a 3D animation system) on top of real 3D background scenes, making use of the test camera projection matrix. Because egocentric scenes involve objects that lie close to the camera, we found it useful to model camera-specific perspective effects.

Poser models. Our in-house grasp database is constructed by modifying the commercial Everyday hands Grasp Poser library [4]. We vary the objects being interacted with, as well as the clothing of the character, i.e., with and without sleeves. We use more than 200 grasp-

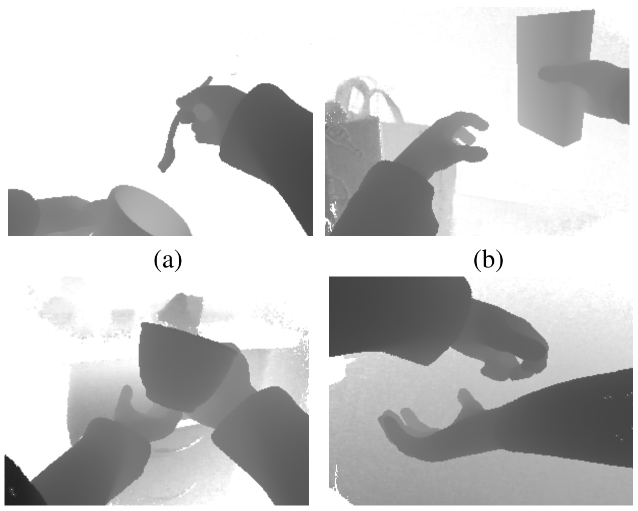

Figure 3. **Examples of synthetic training images**. Our rendering pipeline produces realistic depth maps consisting of multiple hands manipulating household objects in front of everyday backgrounds.

ing postures and 49 objects, including kitchen utensils, personal bathroom items, office/classroom objects, fruits, etc. Additionally we use 6 models of empty hands: wave, fist, thumbs-up, point, open/close fingers. Some objects can be handled with different canonical grasps. For example, one can grip a bottle by its body or by its lid when opening it. We manually add such variants.

Kinematic model. Let θ be a vector of arm joint angles, and let ϕ be a vector of grasp-specific hand joint angles, obtained from the above set of Poser models. We use a standard forward kinematic chain to convert the location of finger joints $\mathbf{u}$ (in a local coordinate system) to image coordinates:

$$\mathbf{p} = C\prod_i T(\theta_i)\prod_j T(\phi_j)\mathbf{u}, \quad \text{where} \quad T, C \in \mathbb{R}^{4\times 4},$$
$$\mathbf{u} = \begin{bmatrix} u_x & u_y & u_z & 1\end{bmatrix}^T, \quad (x,y) = (f\frac{p_x}{p_z}, f\frac{p_y}{p_z}), \quad (1)$$

where T specifies rigid-body transformations (rotation and translation) along the kinematic chain and C specifies the extrinsic camera parameters. Here $\mathbf{p}$ represents the 3D position of point $\mathbf{u}$ in the camera coordinate system. To generate the corresponding image point, we assume camera intrinsics are given by identity scale factors and a focal length f (though it is straightforward to use more complex intrinsic parameterizations). We found it important to use the f corresponding to our physical camera, as it is crucial to correctly model perspective effects for our near-field workspaces.

Pose synthesis: We wish to generate a large set of postured hands. However, building a generative model of grasps is not trivial. One option is to take a data-driven approach and collect training samples using motion capture [24]. Instead, we take a model-driven approach that perturbs a small set of manually-defined canonical postures. To ensure that physically plausible perturbations are generated, we take a simple *rejection sampling* approach. We fix ϕ parameters to respect the hand grasps from Poser, and add small Gaussian perturbations to arm joint angles

$$\theta'_i = \theta_i + \epsilon \quad \text{where} \quad \epsilon \sim N(0, \sigma^2).$$

Importantly, this generates hand joints $\mathbf{p}$ at different translations and viewpoints, correctly modeling the dependencies between both. For each perturbed pose, we render hand joints using (1) and keep exemplars that are 90% visible (e.g., their projected (x, y) coordinates lie within the image boundaries). We show examples in Fig. 2.

Depth maps. Associated with each rendered set of keypoints, we would also like a depth map. To construct a depth map, we represent each rigid limb with a dense cloud of 3D vertices $\{\mathbf{u}_i\}$. We produce this cloud by (over) sampling the 3D meshes defining each rigid-body shape. We render this dense cloud using forward kinematics (1), producing a set of points $\{\mathbf{p}_i\} = \{(p_{x,i}, p_{y,i}, p_{z,i})\}$. We define a 2D depth map $z[u, v]$ by ray-tracing. Specifically, we cast a ray from the origin, in the direction of each image (or depth sensor) pixel location (u, v) and find the closest point:

$$z[u, v] = \min_{k \in \text{Ray}(u,v)} ||\mathbf{p}_k|| \qquad (2)$$

where $\text{Ray}(u, v)$ denotes the set of points on (or near) the ray passing through pixel (u, v). We found the above approach simpler to implement than hidden surface removal, so long as we projected a sufficiently dense cloud of points.

Multiple hands: Some object interactions require multiple hands interacting with a single object. Additionally, many views contain the second hand in the "background". For example, two hands are visible in roughly 25% of the frames in our benchmark videos. We would like our training dataset to have similar statistics. Our existing Poser library contains mostly single-hand grasps. To generate additional multi-arm egocentric views, we randomly pair 25% of the arm poses with a mirrored copy of another randomly-chosen pose. We then add noise to the arm joint angles, as described above. Such a procedure may generate unnatural or self-intersecting poses. To efficiently remove such cases, we separately generate depth maps for the left and right arms, and only keep pairings that produce compatible depth maps:

$$|z_{left}[u, v] - z_{right}[u, v]| > \delta \quad \forall u, v \qquad (3)$$

We find this simple procedure produces surprisingly realistic multi-arm configurations (Fig. 3). Finally we add background clutter from depth maps of real egocentric scenes

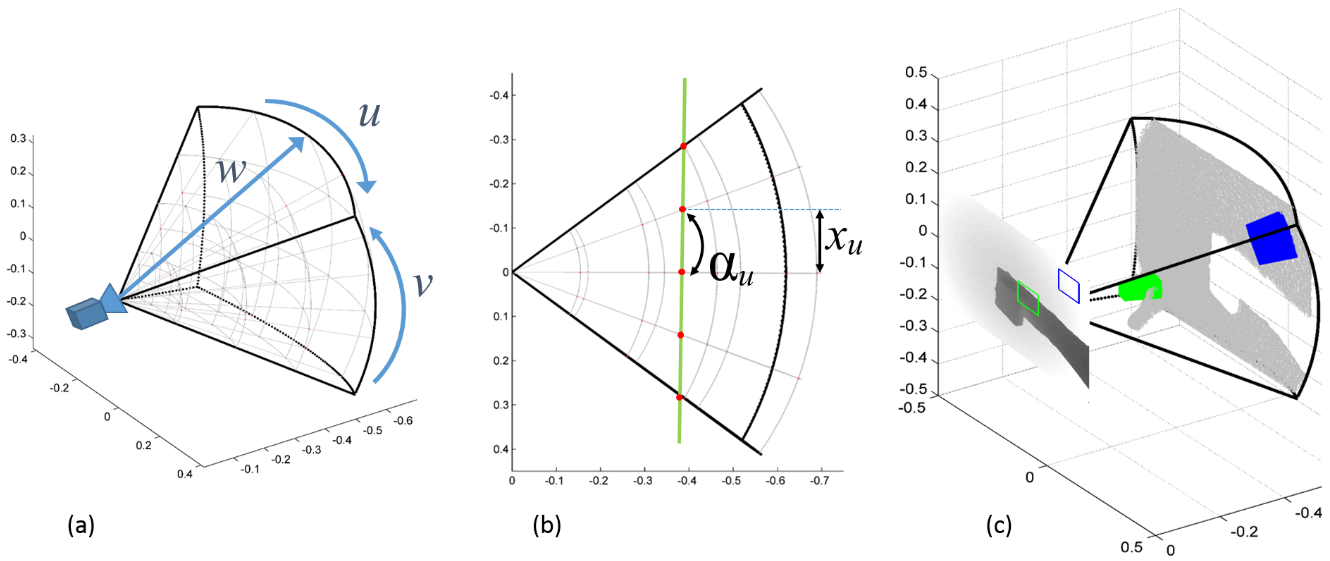
Figure 4. **Volume quantization.** We quantize those points that fall within the egocentric workspace (observable volume within z_{max} = 70cm) into a binary spherical voxel grid of $N_u \times N_v \times N_w$ voxels (a). We vary the azimuth angle α to generate equal-size projections on the image plane (b). Spherical bins ensure that voxels at different distances project to same image area (c). This allows for efficient feature computation and occlusion handling, since occluded voxels along the same line-of-sight can easily be identified.

(not from our benchmark data). We use the above approach to generate over 100,000 multi-hand(+arm+objects) configurations and associated depth-maps.

3. Formulation

3.1. Perspective-aware depth features

It may seem attractive to work in orthographic (or scaled orthographic) coordinates, as this simplifies much of 3D analysis. Instead, we posit that perspective distortion is useful in egocentric settings and should be exploited: objects of interest (hands, arms, and manipulated things) tend to lie near the body and exhibit perspective effects. Specifically, parts of objects that are closer to the camera project to a larger image size. To model such effects, we construct a spherical bin histogram by gridding up the egocentric workspace volume by varying azimuth and elevation angles (Fig. 4). We demonstrate that this feature outperforms orthographic counterparts, and is also faster to compute.

Binarized volumetric features: Much past work processes depth maps as 2D rasterized sensor data. Though convenient for applying efficient image processing routines such as gradient computations (e.g., [32]), rasterization may not fully capture the 3D nature of the data. Alternatively, one can convert depth maps to a full 3D point cloud [15], but the result is orderless making operations such as correspondence-estimation difficult. We propose encoding depth data in a 3D volumetric representation, similar to [30]. To do so, we can back-project the depth map from (2) into a cloud of visible 3D points $\{\mathbf{p}_k\}$, visualized in Fig. 5-(b). They are a subset of the original cloud of 3D points $\{\mathbf{p}_i\}$ in Fig. 5-(a). We now bin those visible points that fall within the egocentric workspace in front of the camera (observable volume within z_{max} = 70cm) into a binary voxel grid of $N_u \times N_v \times N_w$ voxels:

$$b[u, v, w] = \begin{cases} 1 & \text{if} \quad \exists k \text{ s.t. } \quad \mathbf{p}_k \in F(u, v, w) \\ 0 & \quad \text{otherwise.} \end{cases} \tag{4}$$

where $F(u, v, w)$ denotes the set of points within a voxel centered at coordinate (u, v, w).

Spherical voxels: Past work tends to use rectilinear voxels [30, 15]. Instead, we use a spherical binning structure, centering the sphere at the camera origin (Fig. 4). At first glance, this might seem strange because voxels now vary in size – those further away from the camera are larger. The main advantage of a "perspective-aware" binning scheme is that all voxels now project to the same image area in pixels (Fig. 4-(c)). We will show that this both increases accuracy (because one can better reason about occlusions) and speed (because volumetric computations are sparse).

Efficient quantization: Let us choose spherical bins $F(u, v, w)$ such that they project to a single pixel (u, v) in the depth map. This allows one to compute the binary voxel grid $b[u, v, w]$ by simply "reading off" the depth value for each $z(u, v)$ coordinates, quantizing it to z', and assigning 1 to the corresponding voxel:

$$b[u, v, w] = \begin{cases} 1 & \text{if} \quad w = z'[u, v] \\ 0 & \quad \text{otherwise} \end{cases} \tag{5}$$

This results in a sparse volumetric voxel features visualized in Fig. 5-(c). Crucially, a spherical parameterization al-

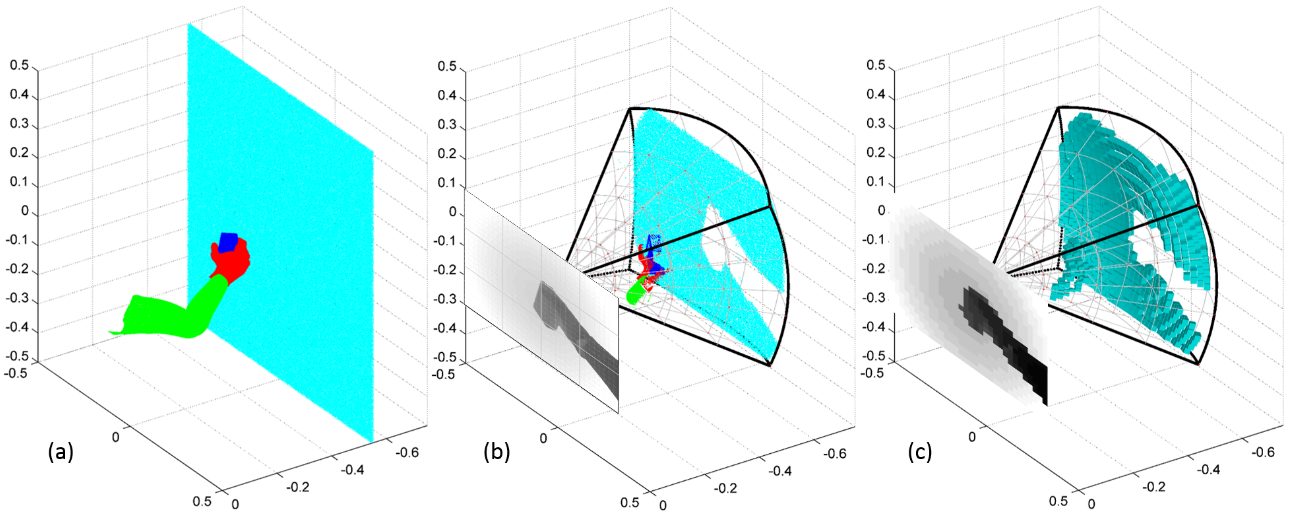
Figure 5. **Binarized volumetric feature.** We synthesize training examples by randomly perturbing shoulder, arm and hand joint angles in a physically possible manner (a). For each example, a synthetic depth map is created by projecting the visible set of dense 3D points using a real-world camera projection matrix (b). The resulting 2D depth map is then quantized with a regular grid in x-y directions and binned in the viewing direction to compute our new binarized volumetric feature (c). In this example, we use a $32 \times 24 \times 35$ grid. Note that for clarity we only show the sparse version of our 3D binary feature. We also show the quantized depth map $z[u, v]$ as a gray scale image (c).

lows one to efficient reason about occlusions: once a depth measurement is observed at position $b[u', v', w'] = 1$, all voxels behind it are occluded for $w \geq w'$. This arises from the fact that single camera depth measurements are, in fact, 2.5D. By convention, we define occluded voxels to be "1". Note that such occlusion reasoning is difficult with orthographic parameterizations because voxels are not arranged along line-of-sight rays.

In practice, we consider a coarse discretization of the volume to make the problem more tractable. The depth map $z[x, y]$ is resized to $N_u \times N_v$ (smaller than depth map size) and quantized in the z-direction. To minimize the effect of noise when counting the points which fall in the different voxels, we quantize the depth measurements by applying a median filter on the pixel values within each image region:

$$\begin{gathered}\forall u, v \in [1, N_u] \times [1, N_v], \\ z'[u, v] = \frac{N_w}{z_{max}} \text{median}(z[x, y] : (x, y) \in P(u, v)),\end{gathered} \tag{6}$$

where $P(u, v)$ is the set of pixel coordinates in the original depth map corresponding to pixel coordinate (u, v) coordinates in the resized depth map.

3.2. Global pose classification

We quantize the set of poses from our synthetic database into K coarse classes for each limb, and train a K-way pose-classifier for pose-estimation. The classifier is linear and makes use of our sparse volumetric features, making it quite simple and efficient to implement.

Pose space quantization: For each training exemplar, we generate the set of 3D keypoints: 17 joints (elbow + wrist + 15 finger joints) and the 5 finger tips. Since we want to recognize coarse limb (arm+hand) configurations, we cluster the resulting training set by applying K-means to the elbow+wrist+knuckle 3D joints. We usually represent each of the K resulting clusters using the average 3D/2D keypoint locations of both arm+hand (See examples in Fig. 6). Note that K can be chosen as a compromise between accuracy and speed.

Global classification: We use a linear SVM for a multi-class classification of upper-limb poses. However, instead of classifying local scanning-windows, we classify global depth maps quantized into our binarized depth feature $b[u, v, w]$ from (5). Global depth maps allow the classifier to exploit contextual interactions between multiple hands, arms and objects. In particular, we find that modeling arms is particularly helpful for detecting hands. For each class $k \in \{1, 2, ...K\}$, we train a one-vs-all SVM classifier obtaining weight vector which can be re-arranged into a $N_u \times N_v \times N_w$ tensor $\beta_k[u, v, w]$. The score for class k is then obtained by a simple dot product of this weight and our binarized feature $b[u, v, w]$:

$$\text{score}[k] = \sum_u \sum_v \sum_w \beta_k[u, v, w] \cdot b[u, v, w]. \tag{7}$$

We visualize projections of the learned weight tensor $\beta_k[u, v, w]$ in Fig. 6 and slices of the tensor in Fig. 7.

3.3. Joint feature extraction and classification

To increase run-time efficiency, we exploit the sparsity of our binarized volumetric feature and jointly implement

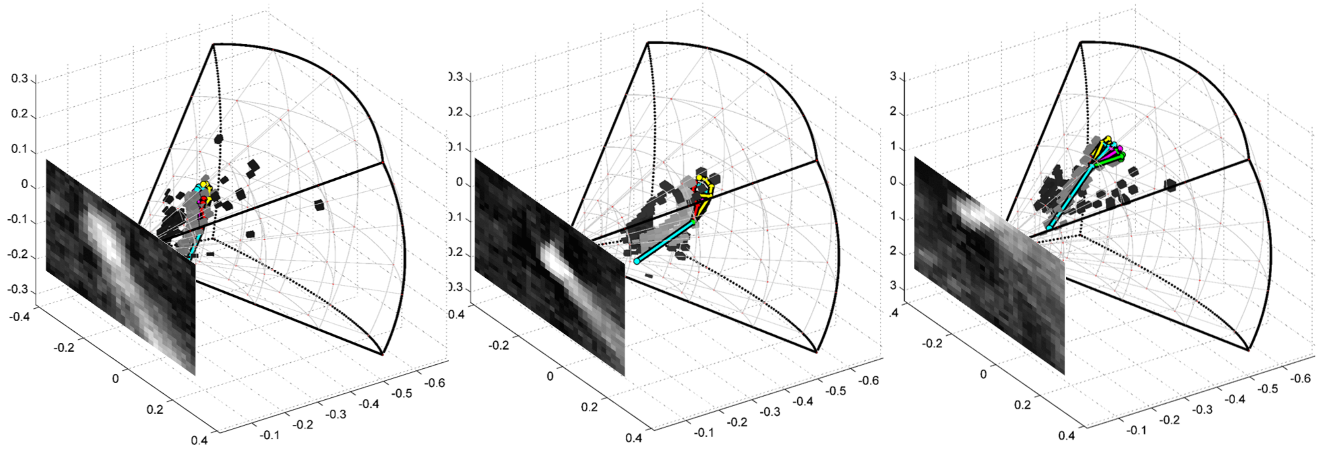

Figure 6. **Pose classifiers.** We visualize the linear weight tensor $\beta_k[u, v, w]$ learnt by the SVM for a $32 \times 24 \times 35$ grid of binary features for 3 different pose clusters. We plot a 2D (u, v) visualization obtained by computing the max along w. We also visualize the corresponding average 3D pose in the egocentric volume together with the top 500 positive (light gray) and negative weights (dark gray) within $\beta_k[u, v, w]$.

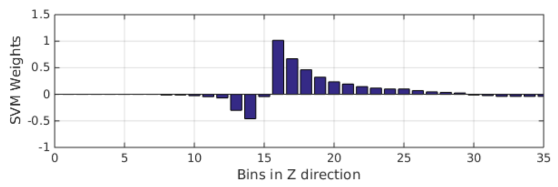

Figure 7. **Weights along w**. We visualize the SVM weights $\beta_k[u, v, w]$ for a particular (u, v) location. Our histogram encoding allows us to learn smooth nonlinear functions of depth values. For example, the above weights respond positively to depth values midway into the egocentric volume, but negatively to those closer.

feature extraction and SVM scoring. Since the final score is a simple dot product with binary features, one can readily extract the feature and update the score on the fly. Because all voxels behind the first measurement are backfilled, the SVM score for each class k from (7) can be written as:

$$\text{score}[k] = \sum_u \sum_v \beta'_k[u, v, z'[u, v]], \tag{8}$$

where $z'[u, v]$ is the quantized depth map and tensor $\beta'_k[u, v, w]$ is the cumulative sum of the weight tensor along dimension w:

$$\beta'_k[u, v, w] = \sum_{d>=w} \beta_k[u, v, d]. \tag{9}$$

Note that the above cumulative-sum tensors can be precomputed. This makes test-time classification quite efficient (8). Feature extraction and SVM classification can be computed jointly following the algorithm presented in Alg. 1. Our implementation runs at 275 frames per second.

```
input : Quantized depth map $z'[u, v]$.
        Cumsum'ed weights $\{\beta'_k[u, v, w]\}$.
        output: score$[k]$
1 for $u \in \{0, 1, ...N_u\}$ do
2     for $v \in \{0, 1, ...N_v\}$ do
3         for $k \in \{0, 1, ...K\}$ do
4             score$[k] += \beta'_k[u, v, z'[u, v]]$
5         end
6     end
7 end
```

Algorithm 1: Joint feature extraction & classification. We jointly extract binarized depth features and evaluate linear classifiers for all quantized poses k. We precompute a "cumsum" β'_k of our SVM weights. At each location (u, v), we add all the weights corresponding to the voxels behind $z[u, v]$, i.e. such that $w \geq z[u, v]$.

4. Experiments

For evaluation, we use the recently released UCI Egocentric dataset [25] and score hand pose detection as a proxy for limb pose recognition (following the benchmark criteria used in [25]) . The dataset consists of 4 video sequences (around 1000 frames each) of everyday egocentric scenes with hand annotations every 10 frames.

Feature evaluation: We first compare hand detection accuracy for different K-way SVM classifiers trained on HOG on depth (as in [25]) and HOG on RGB-D, thus exploiting the stereo-views provided by RGB and depth sensors. To evaluate our voxel encoding, we also trained a SVM directly on the quantized depth map $z[u, v]$ (without constructing a sparse binary feature). To evaluate our

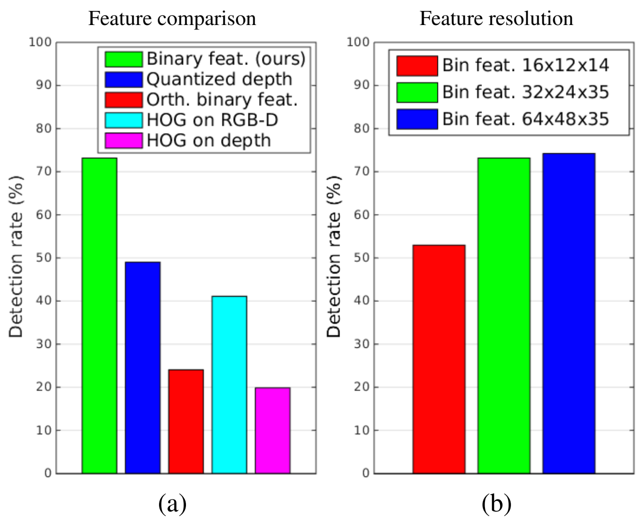

Figure 8. **Feature evaluation**. We compare our feature encoding to different variants (for $K = 750$ classes) in **(a)**. Our feature outperforms HOG-on-depth and HOG-on-RGBD. Our feature also outperforms orthographic voxels and the raw quantized depth map, which surprisingly itself outperforms all other baselines. When combined with a linear classifier, our sparse encoding can learn nonlinear functions of depth (see Fig. 7), while the raw depth map can only learn linear functions. We also vary the resolution of our feature in **(b)**, again for $K = 750$. A size of $32 \times 24 \times 35$ is a good trade-off between size and performance. Doubling the resolution in u, v marginally improves accuracy.

perspective voxels, we compare to an orthographic version of our binarized volumetric feature (similar to past work [30, 15]). In that case, we quantize those points that fall within a 64x48x70 cm^3 egocentric workspace in front of the camera into a binary grid of square voxels:

$$b_{\perp}[u,v,w] = \begin{cases} 1 & \text{if} \quad \exists i \text{ s.t.} \quad (x_i, y_i, z_i) \in N(u,v,w) \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where $N(u, v, w)$ specifies a $2 \times 2 \times 2cm$ cube centered at voxel (u, v, w). Note that this feature is considerably more involved to calculate, since it requires an explicit backprojection and explicit geometric computations for binning. Moreover, identifying occluded voxels is difficult because they are not arranged along line-of-sight rays.

The results obtained with $K = 750$ pose classes are reported in Fig. 8-(a). Our perspective binary features clearly outperforms other types of features. We reach 72% detection accuracy while the state of the art [25] reports 60% accuracy. Our volumetric feature has empirically strong performance in egocentric settings. One reason is that it is robust to small intra-cluster misalignment and deformations because all voxels behind the first measurement are backfilled. Second, it is sensitive to variations in apparent size induced by perspective effects (because voxels have consis-

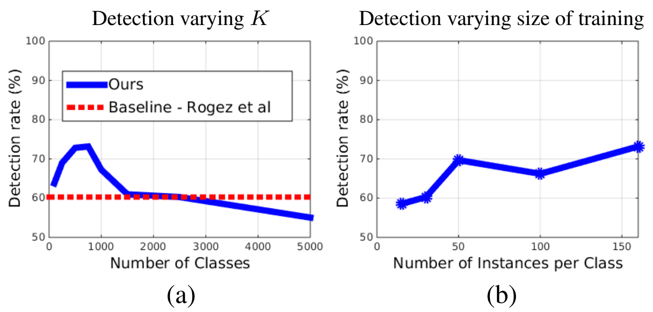

Figure 9. **Clustering and size of training set.** In **(a)**, we plot performance as a function of K (the number of discretized pose classes) for a fixed-size training set. For reference, we also plot the state-of-the-art method from [25]. In **(b)**, we plot performance as we increase the amount of training data for K. Both results suggest that our system may perform better with more training data and more quantized poses. Please see text for further discussion.

tent perspective projections). In Fig. 8-(b), we also show results varying the resolution of the grid. Our choice of $32 \times 24 \times 35$ is a good trade-off between feature dimensionality and performance.

We compare primarily to [25], as that method was already shown to outperform commercial (Intel PXC [11]) and fully-featured tracking systems [21]. Such systems perform poorly due to occlusions inherent in egocentric viewpoints. Notably, [25] use local part templates in a scanning window fashion. Our global approach captures correlations between pose and spatial location, and better deals with occlusion where local appearance can be misleading.

Training data and clustering: We evaluated the performance of our algorithm when varying the number of quantized pose classes K and the amount of training data. Fig. 9-(a) varies K for a fixed training set of 120,000 training images. Performance maxes out relatively quickly at $K = 750$, suggesting that our model may be overfitting due to lack of training data. Fig. 9-(a) fixes $K = 750$ and increases the amount of training data per quantized class. Here, we see a more consistent increase in accuracy. These results suggest that a massive training set and larger K may produce better results.

Qualitative results: We illustrate successes in difficult scenarios in Fig. 10 and analyze common failure modes in Fig. 11. Please see the figures for additional discussion.

5. Conclusions

We have proposed a new approach to the problem of egocentric 3D hand pose recognition during interactions with objects. Instead of classifying local depth image regions through a typical translation-invariant scanning window, we have shown that classifying the global arm+hand+object configurations within the "whole" egocentric workspace in front of the camera allows for fast and accurate results. We

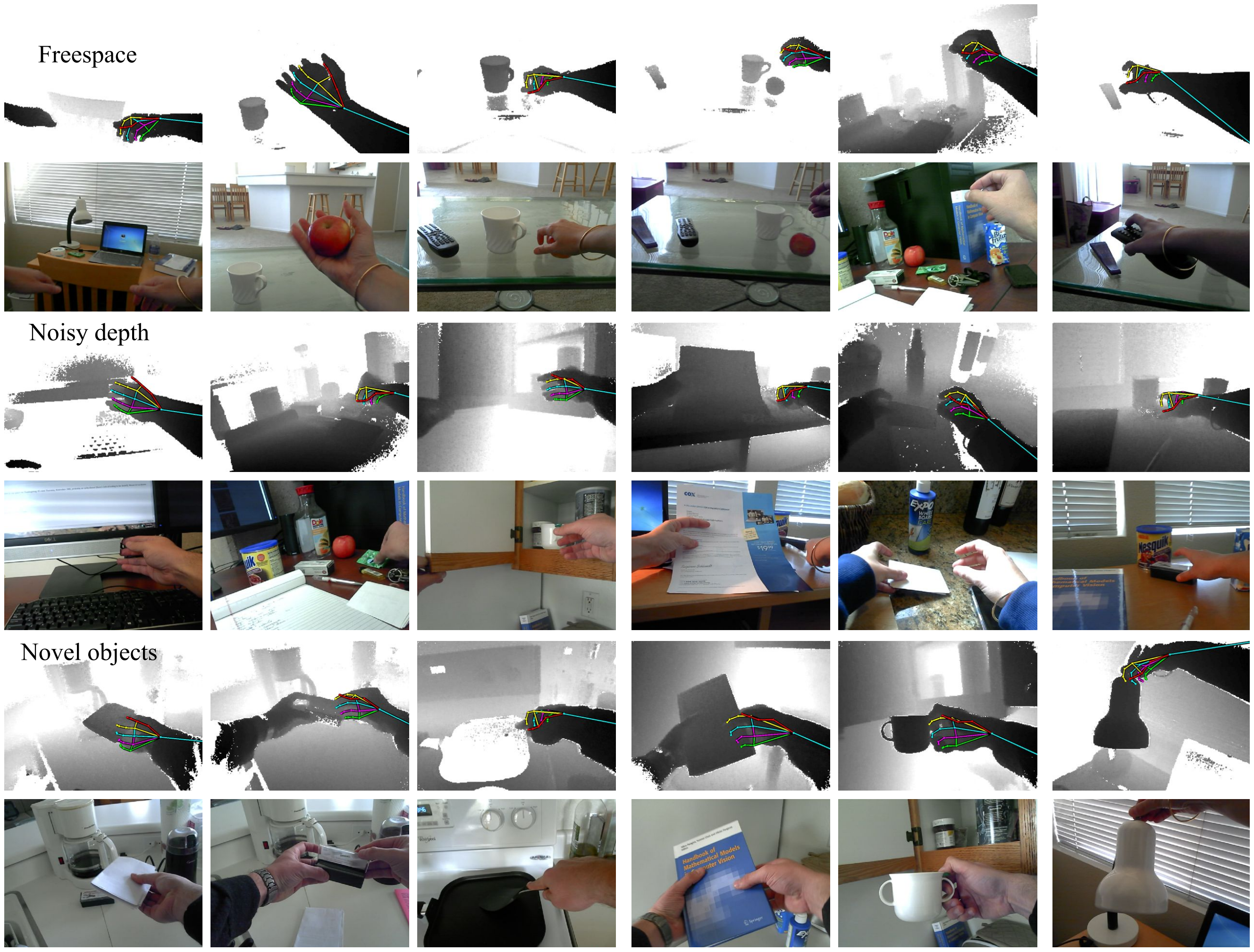

Figure 10. **Good detections**. We show frames where arm and hand are correctly detected. First, we present some easy cases of hands in free-space (**top row**). Noisy depth data and cluttered background cases (**middle row**) showcases the robustness of our system while novel objects (**bottom row:** envelope, staple box, pan, double-handed cup and lamp) require generalization to unseen objects at train-time.

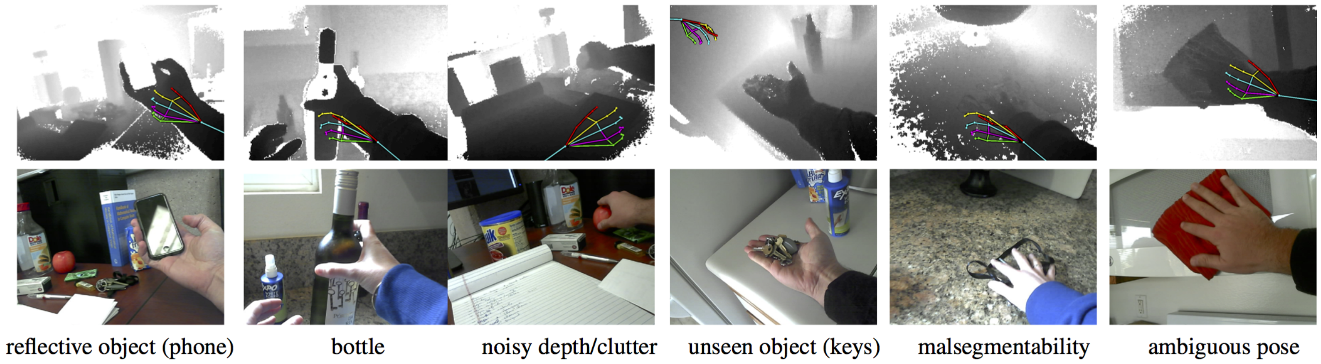

Figure 11. **Hard cases**. We show cases where the pose is not correctly recognized (sometimes not even detected): excessively-noisy depth data, hands manipulating reflective material (phone or bottle of wine) or malsegmentability cases of hands touching background.

train our model by synthesizing workspace exemplars consisting of hands, arms, objects and backgrounds. Our model explicitly reasons about perspective occlusions while being both conceptually and practically simple to implement (4 lines of code). We produce state-of-the-art real-time results for egocentric pose estimation in real-time.

Aknowledgement. GR was supported by the European Commission under FP7 Marie Curie IOF grant "Egovision4Health" (PIOF-GA-2012-328288). JS and DR were supported by NSF Grant 0954083, ONR-MURI Grant N00014- 10-1-0933, and the Intel Science and Technology Center - Visual Computing.

References

[1] V. Athitsos and S. Sclaroff. Estimating 3d hand pose from a cluttered image. In *CVPR (2)*, pages 432–442, 2003. 2

[2] T. Y. D. Tang and T.-K. Kim. Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *ICCV*, pages 1–8, 2013. 2

[3] D. Damen, A. P. Gee, W. W. Mayol-Cuevas, and A. Calway. Egocentric real-time workspace monitoring using an rgb-d camera. In *IROS*, 2012. 2

[4] Daz3D. Every-hands pose library. `http://www.daz3d.com/everyday-hands-poses-for-v4-and-m4`, 2013. 2

[5] A. Fathi, A. Farhadi, and J. Rehg. Understanding egocentric activities. In *ICCV*, 2011. 2

[6] A. Fathi, J. K. Hodgins, and J. M. Rehg. Social interactions: A first-person perspective. In *CVPR*, pages 1226–1233, 2012. 1

[7] A. R. Fielder and M. J. Moseley. Does stereopsis matter in humans? *Eye*, 10(2):233–238, 1996. 1

[8] H. Hamer, J. Gall, R. Urtasun, and L. Van Gool. Data-driven animation of hand-object interactions. In *2011 IEEE International Conference on Automatic Face Gesture Recognition and Workshops (FG 2011)*, pages 360–367. 2

[9] H. Hamer, J. Gall, T. Weise, and L. Van Gool. An object-dependent hand pose prior from sparse training data. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 671–678. 2

[10] H. Hamer, K. Schindler, E. Koller-Meier, and L. Van Gool. Tracking a hand manipulating an object. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1475–1482. 2

[11] Intel. Perceptual computing sdk, 2013. 7

[12] M. Kölsch. An appearance-based prior for hand tracking. In *ACIVS (2)*, pages 292–303, 2010. 2

[13] M. Kölsch and M. Turk. Hand tracking with flocks of features. In *CVPR (2)*, page 1187, 2005. 2

[14] T. Kurata, T. Kato, M. Kourogi, K. Jung, and K. Endo. A functionally-distributed hand tracking method for wearable visual interfaces and its applications. In *MVA*, pages 84–89, 2002. 2

[15] K. Lai, L. Bo, and D. Fox. Unsupervised feature learning for 3d scene labeling. In *ICRA*, 2014. 4, 7

[16] C. Li and K. M. Kitani. Model recommendation with virtual probes for egocentric hand detection. In *ICCV*, 2013. 2

[17] C. Li and K. M. Kitani. Model recommendation with virtual probes for egocentric hand detection. In *ICCV*, 2013. 1

[18] C. Li and K. M. Kitani. Pixel-level hand detection in egocentric videos. In *CVPR*, 2013. 1, 2

[19] Z. Lu and K. Grauman. Story-driven summarization for egocentric video. In *CVPR*, 2013. 1

[20] S. Mann, J. Huang, R. Janzen, R. Lo, V. Rampersad, A. Chen, and T. Doha. Blind navigation with a wearable range camera and vibrotactile helmet. In *ACM International Conf. on Multimedia*, MM '11, 2011. 2

[21] I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BMVC*, 2011. 7

[22] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Tracking the Articulated Motion of Two Strongly Interacting Hands. In *CVPR*, 2012. 2

[23] H. Pirsiavash and D. Ramanan. Detecting activities of daily living in first-person camera views. In *CVPR*, 2012. 1, 2

[24] G. Pons-Moll, A. Baak, T. Helten, M. Müller, H. Seidel, and B. Rosenhahn. Multisensor-fusion for 3d full-body human motion capture. In *CVPR*, pages 663–670, 2010. 3

[25] G. Rogez, M. Khademi, J. Supancic, J. Montiel, and D. Ramanan. 3d hand pose detection in egocentric rgbd images. In *ECCV Workshop on Consuper Depth Camera for Vision (CDC4V)*, pages 1–11, 2014. 1, 6, 7

[26] J. Romero, H. Kjellstrom, C. H. Ek, and D. Kragic. Non-parametric hand pose estimation with object context. *Im. and Vision Comp.*, 31(8):555 – 564, 2013. 2

[27] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 750–757. IEEE, 2003. 2

[28] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake. Efficient human pose estimation from single depth images. 35(12):2821–2840. 2

[29] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013. 2

[30] S. Song and J. Xiao. Sliding shapes for 3d object detection in rgb-d images. In *European Conference on Computer Vision*, 2014. 2, 4, 7

[31] B. Stenger, A. Thayananthan, P. Torr, and R. Cipolla. Model-based hand tracking using a hierarchical bayesian filter. *PAMI*, 28(9):1372–1384,, 2006. 2

[32] S. Tang, X. Wang, X. Lv, T. X. Han, J. Keller, Z. He, M. Skubic, and S. Lao. Histogram of oriented normal vectors for object recognition with a depth sensor. In *ACCV 2012*, pages 525–538. 2013. 4

[33] D. Tzionas and J. Gall. A comparison of directional distances for hand pose estimation. In J. Weickert, M. Hein, and B. Schiele, editors, *Pattern Recognition*, number 8142 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 2013. 2

[34] R. Wang and J. Popovic. Real-time hand-tracking with a color glove. *ACM Trans on Graphics*, 28(3), 2009. 2

[35] C. Xu and L. Cheng. Efficient hand pose estimation from a single depth image. In *ICCV*, 2013. 2

[36] M. Ye, X. Wang, R. Yang, L. Ren, and M. Pollefeys. Accurate 3d pose estimation from a single depth image. In *ICCV*, pages 731–738, 2011. 2