Efficient Human Pose Estimation from Single Depth Images

Jamie Shotton, *Member, IEEE,* Ross Girshick, Andrew Fitzgibbon, *Senior Member, IEEE,* Toby Sharp, *Senior Member, IEEE,* Mat Cook, Mark Finocchio, Richard Moore, *Member, IEEE,* Pushmeet Kohli, Antonio Criminisi, Alex Kipman, Andrew Blake, *Fellow, IEEE*

(Invited Paper - CVPR 2011 special issue)

Abstract—We describe two new approaches to human pose estimation. Both can quickly and accurately predict the 3D positions of body joints from a single depth image, without using any temporal information. The key to both approaches is the use of a large, realistic, and highly varied synthetic set of training images. This allows us to learn models that are largely invariant to factors such as pose, body shape, field-of-view cropping, and clothing. Our first approach employs an intermediate body parts representation, designed so that an accurate per-pixel classification of the parts will localize the joints of the body. The second approach instead directly regresses the positions of body joints. By using simple depth pixel comparison features, and parallelizable decision forests, both approaches can run super-realtime on consumer hardware. Our evaluation investigates many aspects of our methods, and compares the approaches to each other and to the state of the art. Results on silhouettes suggest broader applicability to other imaging modalities.

Index Terms—Computer vision, Machine learning, Pixel classification, Depth cues, Range data, Games.

1 Introduction

The fast and reliable estimation of the pose of the human body from images has been a goal of computer vision for decades. Robust interactive pose estimation has applications including gaming, human-computer interaction, security, telepresence, and even health-care. The recent availability of high-speed depth sensors has greatly simplified the task [1], [2], [3], [4], [5], [6]. However, until the launch of the Microsoft Kinect camera and gaming platform [7] in November 2010, even the best systems exhibited failures when faced with unusual poses, occlusion, sensor noise, and the constraints of super-realtime operation (*i.e.* with a budget of a fraction of the total processor cycles).

This paper describes some of the research behind a core component of the skeletal tracking pipeline that ships in with Kinect [7]. The aim to ship a consumer product necessitated two important design goals: robustness and computational efficiency. We wanted to build something that could work without calibration for any human body shape, so that anyone could start using the system immediately. The system also had to be able to run for hours at a time without failing catastrophically. Our final requirements came in the form of tight budgets for compute and memory usage.

Towards these goals, this paper presents two related approaches for estimating human pose, illustrated in Fig. 1. We will refer to these as *body part classification* (BPC) and *offset joint regression* (OJR). The BPC and OJR algorithms output high-quality shortlists of confidence-weighted proposals for the 3D locations of the skeletal body joints. These proposals are computed at each frame and for each joint independently.

Traditional human body tracking algorithms [8], [9], [1], [10], [6], [11] infer a complete skeleton by exploiting kinematic constraints and can achieve high frame-rates by using temporal coherence from frame-to-frame. However, without regular re-initialization, tracking algorithms are prone to catastrophic loss of track. Our original design was for our 3D body joint proposals to provide initialization and per-frame recovery to complement any appropriate tracking algorithm. However, our per-frame, per-joint proposals have proven remarkably accurate, and might well be usable without tracking a full body model.

Both BPC and OJR use an efficient decision forest that is applied at each pixel in the image. Evaluating the contribution of each pixel to each joint separately avoids any combinatorial search over body joints. The forest uses simple yet discriminative depth comparison image features that give 3D translation invariance while maintaining high computational efficiency. In an optimized implementation, these features and the classifier itself can be evaluated in parallel across each pixel on a GPU [12] or multi-core CPU. Both algorithms can run at super-realtime rates on consumer hardware, leaving sufficient computational resources to allow complex game logic and graphics to run in parallel.

The two methods also share their use of a very large, realistic, synthetic training corpus, generated by rendering depth images of humans. Each render is assigned

- *This work was undertaken at Microsoft Research, Cambridge, in collaboration with Xbox. See http://research.microsoft.com/vision/.*
- *R. Girshick is currently a postdoctoral fellow at UC Berkeley.*
- *R. Moore is currently working at ST-Ericsson.*

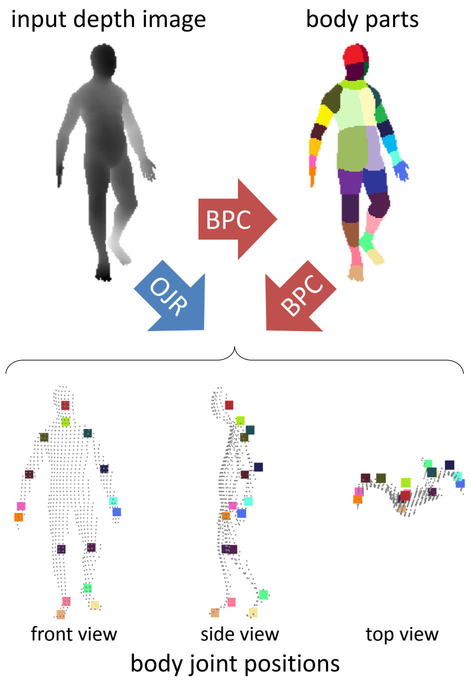

Fig. 1. **Method overview on ground truth example.** Body part classification (BPC) first predicts a (color-coded) body part label at each pixel, and then uses these inferred labels to localize the body joints. Offset joint regression (OJR) instead more directly regresses the positions of the joints. The input depth point cloud is shown overlaid on the body joint positions for reference.

randomly sampled parameters including body shape, size, pose, scene position, *etc.* We can thus quickly and cheaply generate hundreds of thousands of varied images *with associated ground-truth* (the body part label images and the set of 3D body joint positions). This allows us to train deep forests, without the risk of overfitting, that can naturally handle a full range of human body shapes undergoing general body motions [13], self-occlusions, and poses cropped by the image frame.

Body part classification, originally published in [14], was inspired by recent object recognition work that divides objects into parts (*e.g.* [15], [16], [17], [5]). BPC uses a randomized *classification* forest to densely predict discrete body part labels across the image. Given the strong depth image signal, no pairwise terms or CRF have proved necessary for accurate labeling. The pattern of these labels is designed such that the parts are spatially localized near skeletal joints of interest. Given the depth image and the known calibration of the depth camera, the inferred per-pixel label probabilities can be reprojected to define a density over 3D world space. Offset joint regression [18] instead employs a randomized *regression* forest to directly cast a set of 3D offset votes from each pixel to the body joints. These votes are used to again define a world space density. Modes of these density functions can be found using mean shift [19] to give the final set of 3D body joint proposals. Optimized implementations of our algorithms can run at around 200 frames per second on consumer hardware, at least one order of magnitude faster than existing approaches.

To validate our algorithms, we evaluate on both real and synthetic depth images, containing challenging poses of a varied set of subjects. Even without exploiting temporal or kinematic constraints, the 3D body joint proposals are both accurate and stable. We investigate the effect of several training parameters and show a substantial improvement over the state of the art. Further, preliminary results on silhouette images suggest more general applicability of our approach to scenarios where depth cameras are not available.

1.1 Contributions

Our main contribution are as follows.

- We demonstrate that using efficient machine learning approaches, trained with a large-scale, highly varied, synthetic training set, allows one to accurately predict the positions of the human body joints in super-realtime.
- We show how a carefully designed pattern of body parts can transform the hard problem of pose estimation into an easier problem of per-pixel semantic segmentation.
- We examine both classification and regression objective functions for training the decision forests, and obtain slightly surprising results that suggest a limitation of the standard regression objective.
- We employ regression models that compactly summarize the pixel-to-joint offset distributions at leaf nodes. We show that these make our method both faster and more accurate than Hough Forests [20]. We will refer to this as 'vote compression'.

This paper builds on our earlier publications [14], [18]. It unifies the notation, explains the approaches in more detail, and includes a considerably more thorough experimental validation.

1.2 Depth imaging

Depth imaging technology has advanced dramatically over the last few years, and has finally reached a consumer price point [7]. Pixels in a depth image indicate the calibrated distance in meters of 3D points in the world from the imaging plane, rather than a measure of intensity or color. We employ the Kinect depth camera (and simulations thereof) to provide our input data. Kinect uses structured infra-red light and can infer depth images with high spatial and depth resolution at 30 frames per second.

Using a depth camera gives several advantages for human pose estimation. Depth cameras work in low light conditions (even in the dark), help remove ambiguity in scale, are largely color and texture invariant, and resolve silhouette ambiguities. They also greatly simplify the task of background subtraction which we assume in this work as a pre-processing step. Most importantly for our approach, since variations in color and texture are not imaged, it is much easier to synthesize realistic depth images of people and thus cheaply build a large training dataset.

1.3 Related Work

Human pose estimation has generated a vast literature, surveyed in [21], [22]. We briefly review some of the recent advances.

1.3.1 Recognition in parts

Several methods have investigated using some notion of distinguished body parts. One popular technique, pictorial structures [23], was applied by Felzenszwalb & Huttenlocher [24] to efficiently estimate human pose by representing the body by a collection of parts arranged in a deformable configuration. Springs are used between parts to model the deformations. Ioffe & Forsyth [25] group parallel edges as candidate body segments and prune combinations of segments using a projected classifier. Ramanan & Forsyth [26] find candidate body segments as pairs of parallel lines and cluster their appearances across frames, connecting up a skeleton based on kinematic constraints. Sigal *et al.* [9] use eigen-appearance template detectors for head, upper arms and lower legs proposals. Non-parametric belief propagation was then used to infer whole body pose. Tu's 'auto-context' was used in [27] to obtain a coarse body part labeling. These labels were not defined to localize joints, and classifying each frame took about 40 seconds. 'Poselets' that form tight clusters in both 3D pose and 2D image appearance, detectable using SVMs, were presented by Bourdev & Malik [17]. Wang & Popović [10] proposed a related approach to track a hand clothed in a colored glove; our BPC system could be viewed as automatically inferring the colors of a virtual colored suit from a depth image. As detailed below, our BPC algorithm [14], extends the above techniques by using parts that densely cover the body and directly localize body joints.

1.3.2 Pose from depth

Recent work has exploited improvements in depth imaging and 3D input data. Anguelov *et al.* [28] segment puppets in 3D range scan data into head, limbs, torso, and background using spin images and a MRF. Grest *et al.* [1] use Iterated Closest Point (ICP) to track a skeleton of a known size and starting position from depth images. In [3], Zhu & Fujimura build heuristic detectors for coarse upper body parts (head, torso, arms) using a linear programming relaxation, but require a T-pose initialization to calibrate the model shape. Siddiqui & Medioni [4] hand-craft head, hand, and forearm detectors, and show that data-driven MCMC model fitting outperforms the iterated closest point algorithm. Kalogerakis *et al.* [29] classify and segment vertices in a full closed 3D mesh into different parts, but do not deal with occlusions and are sensitive to mesh topology. Plagemann *et al.* [5] build a 3D mesh to find geodesic extrema interest points which are classified into 3 parts: head, hand, and foot. This method provides both a location and orientation estimate of these parts, but does not distinguish left from right, and the use of interest points limits the choice of parts.

1.3.3 Regression

Regression has been a staple of monocular 2D human pose estimation [30], [31], [32], [13]. Several methods have explored matching exemplars or regressing from a small set of nearest neighbors. The shape context descriptor was used by Mori & Malik [33] to retrieve exemplars. Shakhnarovich *et al.* [34] estimate upper body pose, interpolating k-NN poses efficiently indexed by parameter sensitive hashing. Agarwal & Triggs [30] learn a regression from kernelized image silhouette features to pose. Navaratnam *et al.* [32] use the marginal statistics of unlabeled data to improve pose estimation. Local mixtures of Gaussian Processes were used by Urtasun & Darrell [13] to regress human pose. Our OJR approach combines some ideas from these approaches with the tools of high-speed object recognition based on decision trees.

1.3.4 Other approaches

An alternative random forest based method for pose estimation was proposed by [35]. Their approach quantizes the space of rotations and gait cycle, though does not directly produce a detailed pose estimate.

A related technique to our OJR algorithm is used in object localization. For example, in the implicit shape model (ISM) [36], visual words are used to learn voting offsets to predict 2D object centers. ISM has been extended in two pertinent ways. Müller *et al.* [37] apply ISM to body tracking by learning separate offsets for each body joint. Gall and Lempitsky [20] replace the visual word codebook of ISM by learning a random forest in which each tree assigns every image pixel to a decision-tree leaf node at which is stored a potentially large collection of votes. This removes the dependence of ISM on repeatable feature extraction and quantization, as well as the somewhat arbitrary intermediate codebook representation. Associating a collection of 'vote offsets' with each leaf node/visual word, these methods then accumulate votes to determine the object centers/joint positions. Our OJR method builds on these techniques by compactly summarizing the offset distributions at the leaf nodes, learning the model hyper-parameters, and using a continuous test-time voting space.

1.4 Outline

The remainder of the paper is organized as follows. Sec. 2 explains how we generate the large, varied training set that is the key to our approach. Following that, Sec. 3 describes the two algorithms in a unified framework. Our experimental evaluation is detailed in Sec. 4, and we conclude in Sec. 5.

2 Data

Many techniques for pose estimation require training images with high quality ground truth labels, such as joint positions. For real images, these labels can be very expensive to obtain. Much research has thus focused on techniques to overcome lack of training data by using computer graphics [34], [38], [39], but there are two potential problems:

1) Rendering realistic intensity images is hampered by the huge color and texture variability induced by clothing, hair, and skin, often meaning that the data are reduced to 2D silhouettes [30]. While depth cameras significantly reduce this difficulty, considerable variation in body and clothing *shape* remains.
2) Synthetic body pose renderers use, out of necessity, real motion capture (mocap) data. Although techniques exist to simulate human motion (*e.g.* [40]) they do not yet produce a full range of volitional motions of a human subject.

In this section we describe how we overcome these problems. We take real mocap data, retarget this to a variety of base character models, and then synthesize a large, varied dataset. We believe the resulting dataset to considerably advance the state of the art in both scale and variety, and will demonstrate the importance of such a large dataset in our evaluation.

2.1 Motion capture data

As noted above, simulating human pose data is an unsolved problem. Instead, we obtain ground truth pose data using marker-based motion capture of real human actors. The human body is capable of an enormous range of poses. Modeled jointly, the number of possible poses is exponential in the number of articulated joints. We cannot thus record all possible poses. However, there is hope. As will be seen in Sec. 3, our algorithms, based on sliding window decision forests, were designed to only look at a local neighborhood of a pixel. By looking at local windows, we factor whole body poses into combinations of local poses, and can thus expect the forests to *generalize* somewhat to unseen poses. In practice, even a limited corpus of mocap data where for example each limb separately moves through a wide range of poses has proven sufficient. Further, we need not record mocap with variation in rotation about the vertical axis, mirroring left-right, scene position, body shape and size, or camera pose, all of which can be simulated. Given our core entertainment scenario, we recorded 500K frames in a few hundred sequences of driving, dancing, kicking, running, navigating menus, *etc.*

To create training data we render single, static depth images because, as motivated above, our algorithms deliberately eschew temporal information. Often, changes in pose from one mocap frame to the next are so small as to be insignificant. We can thus discard many similar, redundant poses using 'furthest neighbor' clustering [41]. We represent a pose P as a collection $P = (\mathbf{p}_1, \ldots, \mathbf{p}_J)$ of J joints where each $\mathbf{p}_j$ is a 3D position vector. Starting with set $\mathcal{P}_{\text{all}}$ of all the recorded mocap poses, we choose an initial pose at random and then greedily grow a set $\mathcal{P}$ as

$$\mathcal{P} \;:=\; \mathcal{P} \cup \{ \operatorname*{argmax}_{P \in \mathcal{P}_{\text{all}} \setminus \mathcal{P}} \min_{P' \in \mathcal{P}} d_{\text{pose}}(P, P') \} \;, \tag{1}$$

where as the distance between poses we use the maximum Euclidean distance over body joints j

$$d_{\text{pose}}(P, P') \;=\; \max_{j \in \{1,\ldots,J\}} \|\mathbf{p}_j - \mathbf{p}'_j\|_2 \;. \tag{2}$$

We stop growing set $\mathcal{P}$ when there exists no unchosen pose P which has $d_{\text{pose}}(P, P') > D_{\text{pose}}$ for any chosen pose P'. We set $D_{\text{pose}} = 5\text{cm}$. This results in a final subset $\mathcal{P} \subset \mathcal{P}_{\text{all}}$ containing approximately 100K most dissimilar poses.

We found it necessary to iterate the process of motion capture, rendering synthetic training data, training the classifier, and testing joint prediction accuracy. This allowed us to refine the mocap database with regions of pose space that had been previously missed out. Our early experiments employed the CMU mocap database [42] which gave acceptable results though covers far less of pose space.

2.2 Rendering synthetic data

We build a randomized rendering pipeline. This can be viewed as a generative model from which we can sample fully labeled training images of people. Our goals in building this pipeline were twofold: *realism* – we want the samples to closely resemble real images so that the learned model can work well on live camera input; and *variety* – the dataset must contain a good coverage of the appearance variations we hope to recognize at test time. Fig. 3 illustrates the huge space of possible appearance variations we need to deal with for just one body part, even when restricted to a pixel's local neighborhood as discussed above.

Our features achieve 3D translation invariance by design (see below). However, other invariances such as pose and shape cannot be designed so easily or efficiently, and must instead be encoded implicitly through the training data. The rendering pipeline thus randomly samples a set of parameters, using the best approximations we could reasonably achieve to the variations we expected to observe in the real world. While we cannot hope to sample all possible combinations of variations, if samples contain somewhat independent variations (in particular, excluding artificial correlations such as thin people always wear a hat), we can expect the classifier

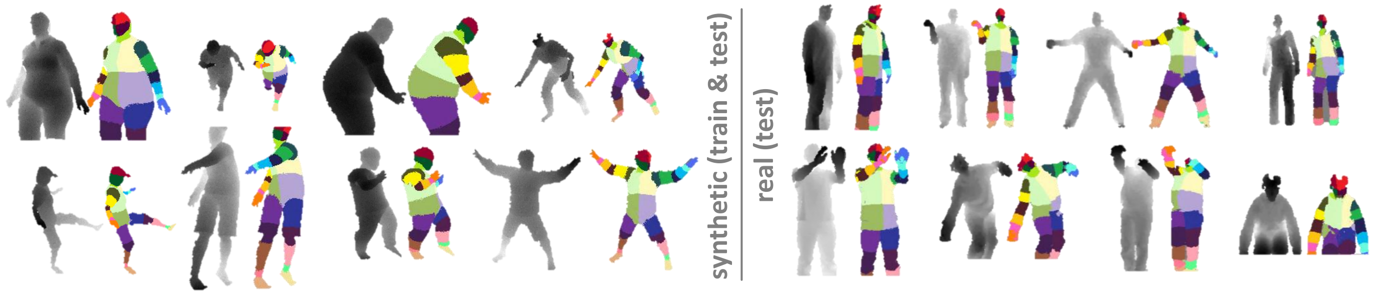

Fig. 2. **Synthetic *vs.* real data.** Pairs of depth images and corresponding color-coded ground truth body part label images. The 3D body joint positions are also known (but not shown). Note wide variety in pose, shape, clothing, and crop. The synthetic images look remarkably similar to the real images, lacking primarily just the high-frequency texture.

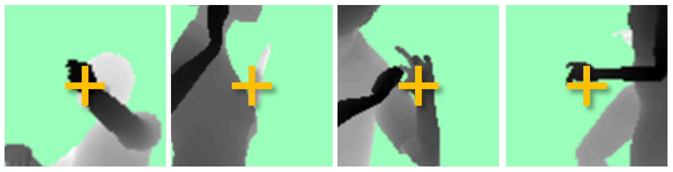

Fig. 3. **A single body part varies widely in its context.**

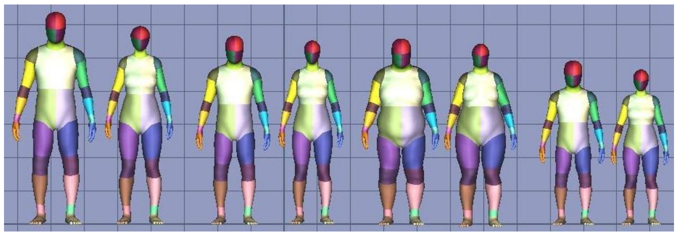

Fig. 4. **Example base character models.**

to learn a large degree of invariance. Let us run through the variations we simulate:

Base Character. We use 3D models of 15 varied base characters, both male and female, from child to adult, short to tall, and thin to fat. Some examples are shown in Fig. 4. A given render will pick uniformly at random from the characters.

Pose. Having discarded redundant poses from the mocap data, we retarget the remaining poses $P \in \mathcal{P}$ to each base character using [43]. A pose is selected uniformly at random and mirrored left-right with probability $\frac{1}{2}$ to prevent a left or right bias.

Rotation & Translation. The character is rotated about the vertical axis and translated in the scene, uniformly at random. Translation ensures we obtain cropped training examples where the character is only partly in-frame.

Hair & Clothing. We add mesh models of several hair styles and items of clothing chosen at random. A slight gender bias is used, so that, for instance, long hair is chosen more often for the female models, and beards are only chosen for the male models.

Weight & Height Variation. The base characters already include a wide variety of weights and heights. To add further variety we add an extra variation in height ($\pm 10\%$) and weight ($\pm 10\%$). For rendering efficiency, we assume this variation does not affect the pose retargeting.

Camera Position & Orientation. The camera height, pitch and roll are chosen uniformly at random within a range believed to be representative of an entertainment scenario in a home living room.

Camera Noise. While depth camera technology has improved rapidly in the last few years, real depth cameras exhibit noise, largely due to non-IR-reflecting materials (*e.g.* glass, hair), surfaces that are almost perpendicular to the sensor, and ambient illumination. To ensure high realism in our dataset, we thus add artificial noise to the clean computer graphics renders to simulate the depth imaging process: dropped-out pixels, depth shadows, spot noise and disparity quantization.

We use standard linear skinning techniques from computer graphics to animate the chosen 3D mesh model given the chosen pose, and a custom pixel shader is used to render the depth images. Fig. 2 compares the varied output of the pipeline to hand-labeled real depth images. The synthetic data is used both as fully labeled training data, and, alongside real hand-labeled depth images, as test data in our evaluation.

In building this randomized rendering pipeline, we attempted to fit as much variety in as many ways as we could, given the time constraints we were under. Investigating the precise effects of the choice and amounts of variation would be fascinating, but lies beyond the scope of this work.

2.3 Training data labeling

A major advantage of using synthetic training images is that the ground truth labels can be generated almost for free, allowing one to scale up supervised learning to very large scales. The complete rendering pipeline allows us to rapidly sample hundreds of thousands of unique images of people. The particular tasks we address in this work, BPC and OJR, require different types of label, described next.

2.3.1 Body part classification labels

Our first algorithm, BPC, aims to predict a discrete body part label at each pixel. At training time, these labels are required for all pixels, and we thus represent the labels as a color-coded body part label image that accompanies each depth image (see Figs. 1 and 2).

The use of an intermediate body part representation that can localize 3D body joints is a key contribution of this work. It transforms the pose estimation problem into one that can readily be solved by efficient classification algorithms. The particular pattern of body parts used was designed by hand to balance these desiderata:

- the parts must densely cover the body, as a prediction is made for every pixel in the foreground;
- the parts should not be so small and numerous as to waste capacity of the classifier; and
- the parts must be small enough to well localize a region of the body.

By centering and localizing some of the parts around body joints of interest, accurate body *part* predictions will necessarily spatially localize those body *joints*, and, because we have calibrated depth images, this localization will implicitly be in 3D.

The parts definition can be specified in a texture map and retargetted to the various 3D base character meshes for rendering. For our experiments, we define 31 body parts: LU/RU/LW/RW head, neck, L/R shoulder, LU/RU/LW/RW arm, L/R elbow, L/R wrist, L/R hand, LU/RU/LW/RW torso, LU/RU/LW/RW leg, L/R knee, L/R ankle, and L/R foot (Left, Right, Upper, loWer). Distinct parts for left and right allow the classifier to learn to disambiguate the left and right sides of the body. The precise definition of these parts might be changed to suit a particular application. For example, in an upper body tracking scenario, all the lower body parts could be merged into a single part.

2.3.2 Offset joint regression labels

Our second algorithm, OJR, instead aims to estimate the 3D joint positions more directly. As such, the ground truth labels it requires are simply the ground truth 3D joint positions. These are trivially recorded during the standard mesh skinning process. In our experiments, we use 16 body joints: head, neck, L/R shoulder, L/R elbow, L/R wrist, L/R hand, L/R knee, L/R ankle, and L/R foot. This selection allows us to directly compare the BPC and OJR approaches on a common set of predicted joints.

3 Method

Our algorithms cast votes for the position of the body joints by evaluating a sliding window decision forest at each pixel. These votes are then aggregated to infer reliable 3D body joint position proposals. In this section we describe: (i) the features we employ to extract discriminative information from the image; (ii) the structure of a random forest, and how it combines multiple such features to achieve an accurate set of votes; (iii) the different leaf node prediction models used for BPC and OJR; (iv) how the pixel votes are aggregated into a set of joint position predictions at test time; and (iv) how the forests are learned.

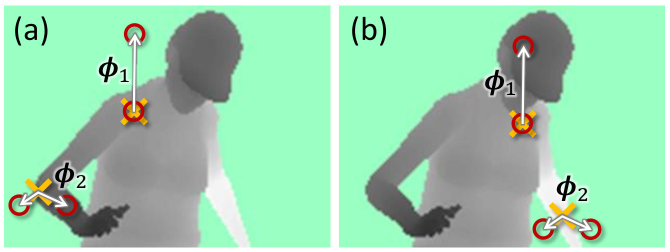

Fig. 5. **Depth image features.** The yellow crosses indicate the image pixel $\mathbf{u}$ being classified. The red circles indicate the offset pixels as defined in Eq. 3. In (a), the two example features give a large depth difference response, *i.e.* $|f(\mathbf{u}|\phi)|$ is large. In (b), the same two features at new image locations give a much smaller response. In practice, many such features combined in a decision forest give a strong discriminative signal.

3.1 Depth image features

We employ simple depth comparison features, inspired by those in [44]. Individually these features provide only a weak discriminative signal, but combined in a decision forest they prove sufficient to accurately disambiguate different appearances and regions of the body. At a given pixel $\mathbf{u}$, the feature response is computed as

$$f(\mathbf{u}|\phi) = z\left(\mathbf{u} + \frac{\boldsymbol{\delta}_1}{z(\mathbf{u})}\right) - z\left(\mathbf{u} + \frac{\boldsymbol{\delta}_2}{z(\mathbf{u})}\right) , \qquad (3)$$

where feature parameters $\boldsymbol{\phi} = (\boldsymbol{\delta}_1, \boldsymbol{\delta}_2)$ describe 2D pixel offsets $\boldsymbol{\delta}$, and function $z(\mathbf{u})$ looks up the depth at pixel $\mathbf{u} = (u, v)^\top$ in a particular image. Each feature therefore performs two offset 'depth probes' in the image and takes their difference. The normalization of the offsets by $\frac{1}{z(\mathbf{u})}$ ensures that the feature response is depth invariant: at a given point on the body, a fixed world space offset will result whether the depth pixel is close or far from the camera. The features are thus 3D translation invariant, modulo perspective effects. If an offset pixel $\mathbf{u}'$ lies on the background or outside the bounds of the image, the depth probe $z(\mathbf{u}')$ is assigned a large positive constant value.

During training of the tree structure, offsets $\boldsymbol{\delta}$ are sampled at random within a box of fixed size. We investigate sampling strategies in Sec. 3.5.1, and evaluate the effect of this maximum depth probe offset in Fig. 11(c). We further set $\boldsymbol{\delta}_2 = \mathbf{0}$ with probability $\frac{1}{2}$. This means that roughly half the features evaluated are 'unary' (look at only one offset pixel) and half are 'binary' (look at two offset pixels). In practice the results appear to be fairly insensitive to this parameter.

Fig. 5 illustrates two different features. The unary feature with parameters ϕ_1 looks upwards: Eq. 3 will

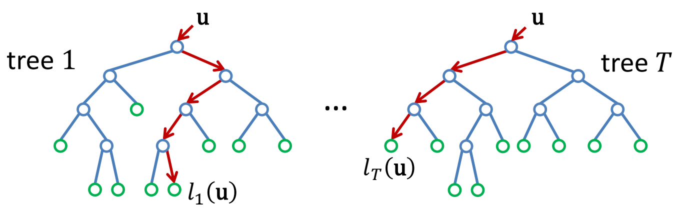

Fig. 6. **Randomized Decision Forests.** A forest is an ensemble of T decision trees. Each tree consists of split nodes (blue) and leaf nodes (green). The red arrows indicate the different paths that might be taken by different trees for a particular input.

give a large positive response for pixels $\mathbf{u}$ near the top of the body, but a value close to zero for pixels $\mathbf{u}$ lower down the body. By similar reasoning, the binary feature (ϕ_2) may be seen instead to help find thin vertical structures such as the arm.

The design of these features was strongly motivated by their computational efficiency: no preprocessing is needed; each feature need only read at most 3 image pixels and perform at most 5 arithmetic operations. Further, these features can be straightforwardly implemented on the GPU. Given a larger computational budget, one could employ potentially more powerful features based on, for example, depth integrals over regions, curvature, or more complex local descriptors *e.g.* [45].

3.2 Randomized forests

Randomized decision trees and forests [46], [47], [48], [49], [50] have proven fast and effective multi-class classifiers for many tasks [44], [51], [52], [50], and can be implemented efficiently on the GPU [12]. As illustrated in Fig. 6, a forest is an ensemble of T decision trees, each consisting of split and leaf nodes. We will use n to denote any node in the tree, and l to denote a leaf node specifically. Each split node contains a 'weak learner' represented by its parameters $\boldsymbol{\theta} = (\boldsymbol{\phi}, \tau)$: the 2D offsets $\boldsymbol{\phi} = (\boldsymbol{\delta}_1, \boldsymbol{\delta}_2)$ used for feature evaluation above, and a scalar threshold τ. To make a prediction for pixel $\mathbf{u}$ in a particular image, one starts at the root and traverses a path to a leaf by repeated evaluating the weak learner function

$$h(\mathbf{u}; \boldsymbol{\theta}_n) = [f(\mathbf{u}; \boldsymbol{\phi}_n) \geq \tau_n] \ , \tag{4}$$

where $[\cdot]$ is the 0-1 indicator. If $h(\mathbf{u}; \boldsymbol{\theta}_n)$ evaluates to 0, the path branches to the left child of n, otherwise it branches to the right child. This repeats until a leaf node l is reached. We will use $l(\mathbf{u})$ to indicate the particular leaf node reached for pixel $\mathbf{u}$. The same algorithm is applied at each pixel for each tree t, resulting in the set of leaf nodes reached $\mathcal{L}(\mathbf{u}) = \{l_t(\mathbf{u})\}_{t=1}^{T}$. More details can be found in [50], a tutorial on decision forests.

3.3 Leaf node prediction models

At each leaf node l in each tree is stored a learned *prediction* model. In this work we use two types of prediction model. For BPC, where a classification forest is used, the prediction model is a probability mass function $p_l(c)$ over body parts c. For OJR, where a regression forest is used, the prediction model is instead a set of weighted relative votes $\mathcal{V}_{lj}$ for each joint j. In this section we describe these two models, and show how both algorithms can be viewed as casting a set of weighted world space *votes* for the 3D positions of the each joint in the body. Sec. 3.4 will then show how these votes are aggregated in an efficient smoothing and clustering step based on mean shift to produce the final 3D body joint proposals.

3.3.1 *Body part classification* (BPC)

BPC predicts a body part label at each pixel as an intermediate step towards predicting joint positions. The classification forest approach achieves this by storing a distribution $p_l(c)$ over the discrete body parts c at each leaf l. For a given input pixel $\mathbf{u}$, the tree is descended to reach leaf $l = l(\mathbf{u})$ and the distribution $p_l(c)$ is retrieved. The distributions are averaged together for all trees in the forest to give the final classification as

$$p(c|\mathbf{u}) = \frac{1}{T} \sum_{l \in \mathcal{L}(\mathbf{u})} p_l(c) \ . \tag{5}$$

One can visualize the most likely body part inferred at each pixel as an image, and examples of this are given in Fig. 10. One might consider smoothing this signal in the image domain. For example, one might use probabilities $p(c|\mathbf{u})$ as the unary term in a conditional random field with a pairwise smoothness prior [53]. However, since the per-pixel signal is already very strong and such smoothing would likely be expensive to compute, we do not use such a prior.

The image space predictions are next re-projected into world space. We denote the re-projection function as $\mathbf{x}(\mathbf{u}) = (x(\mathbf{u}), y(\mathbf{u}), z(\mathbf{u}))^\top$. Conveniently, the known $z(\mathbf{u})$ from the calibrated depth camera allows us to compute $x(\mathbf{u})$ and $y(\mathbf{u})$ trivially.

Next, we must decide how to map from surface body *parts* to interior body *joints*. In Sec. 2 we defined many, though not all, body part labels c to spatially align with the body joints j, and conversely most joints j have a specific part label c. We will thus use $c(j)$ to denote the body part associated with joint j.

Algorithm 1 Body part classification voting

```
1: initialize $\mathcal{X}_j^{\text{BPC}} = \emptyset$ for all joints $j$
2: for all foreground pixels $\mathbf{u}$ in the test image do
3:    evaluate forest to reach leaf nodes $\mathcal{L}(\mathbf{u})$
4:    evaluate distribution $p(c|\mathbf{u})$ using Eq. 5
5:    compute 3D pixel position $\mathbf{x}(\mathbf{u}) = (x(\mathbf{u}), y(\mathbf{u}), z(\mathbf{u}))^\top$
6:    for all joints $j$ do
7:       compute pushed-back position $\mathbf{x}_j(\mathbf{u})$
8:       lookup relevant body part $c(j)$
9:       compute weight $w$ as $p(c = c(j)|\mathbf{u}) \cdot z^2(\mathbf{u})$
10:      add vote $(\mathbf{x}_j(\mathbf{u}), w)$ to set $\mathcal{X}_j^{\text{BPC}}$
11: return set of votes $\mathcal{X}_j^{\text{BPC}}$ for each joint $j$
```

Now, no matter how well aligned in the x and y directions, the body parts inherently lie on the surface of the body. They thus cannot align in the z direction with the *interior* body joint position we are after. (See Fig. 1). We therefore use a learned per-joint vector $\boldsymbol{\zeta}_j = (0, 0, \zeta_j)^\top$ that pushes back the re-projected pixel surface positions into the world to better align with the interior joint position: $\mathbf{x}_j(\mathbf{u}) = \mathbf{x}(\mathbf{u}) + \boldsymbol{\zeta}_j$. This simple approach effectively assumes each joint is spherical, and works well and efficiently in practice. As an indication, the mean across the different joints of the learned push-backs ζ is 0.04m.

We finally create the set $\mathcal{X}_j^{\text{BPC}}$ of weighted world space votes using Algorithm 1. These votes will be used in the aggregation step below. As you see, the position of each vote is given by the pushed-back world space pixel position $\mathbf{x}_j(\mathbf{u})$. The vote weight w is given by the probability mass for a particular body part, multiplied by the squared pixel depth. This depth-weighting compensates for observing fewer pixels when imaging a person standing further from the camera, and ensures the aggregation step is depth invariant. In practice this gave a small but consistent improvement in joint prediction accuracy.

Note that each pixel produces exactly one vote for each body joint, and these votes all share the same world space position. In practice many of the votes will have zero probability mass and can be ignored. This contrasts with the OJR prediction model, described next, where each pixel can cast several votes for each joint.

3.3.2 Offset joint regression (OJR)

The OJR approach aims to predict the set of weighted votes directly, without going through an intermediate representation. The forest used here is a regression forest [54], [50] since the leaves make continuous predictions. At each leaf node l we store a distribution over the *relative* 3D offset from the re-projected pixel coordinate $\mathbf{x}(\mathbf{u})$ to each body joint j of interest. Each pixel can thus potentially cast votes to all joints in the body, and unlike BPC, these votes may differ in all world space coordinates and thus directly predict interior rather than surface positions.

Algorithm 2 Offset joint regression voting

```
1:  initialize $\mathcal{X}_j^{\text{OJR}} = \emptyset$ for all joints $j$
2:  **for all** foreground pixels $\mathbf{u}$ in the test image **do**
3:      evaluate forest to reach leaf nodes $\mathcal{L}(\mathbf{u})$
4:      compute 3D pixel position $\mathbf{x}(\mathbf{u}) = (x(\mathbf{u}), y(\mathbf{u}), z(\mathbf{u}))^\top$
5:      **for all** leaves $l \in \mathcal{L}(\mathbf{u})$ **do**
6:          **for all** joints $j$ **do**
7:              lookup weighted *relative* vote set $\mathcal{V}_{lj}$
8:              **for all** $(\boldsymbol{\Delta}_{ljk}, w_{ljk}) \in \mathcal{V}_{lj}$ **do**
9:                  compute *absolute* position $\mathbf{x} = \mathbf{x}(\mathbf{u}) + \boldsymbol{\Delta}_{ljk}$
10:                 compute weight $w$ as $w_{ljk} \cdot z^2(\mathbf{u})$
11:                 add vote $(\mathbf{x}, w)$ to set $\mathcal{X}_j^{\text{OJR}}$
12: sub-sample $\mathcal{X}_j^{\text{OJR}}$ to contain at most $N_{\text{sub}}$ votes
13: **return** sub-sampled vote set $\mathcal{X}_j^{\text{OJR}}$ for each joint $j$
```

Ideally one would like to make use of a *distribution* of such offsets. Even for fairly deep trees, we have observed highly multi-modal empirical offset distributions at the leaves. Thus for many nodes and joints, approximating the distribution over offsets as a Gaussian would be inappropriate. One alternative, Hough forests [20], is to represent the distribution as the set of *all* offsets seen at training time. However, Hough forests trained on our large training sets would require vast amounts of memory and be prohibitively slow for a realtime system.

We therefore, in contrast to [36], [20], represent the distribution using a *small* set of 3D *relative vote* vectors $\boldsymbol{\Delta}_{ljk} \in \mathbb{R}^3$. The subscript l denotes the tree leaf node (as before), j denotes a body joint, and $k \in \{1, \ldots, K\}$ denotes a cluster index.[1] We have found $K = 1$ or 2 has given good results, and while the main reason for keeping K small is efficiency, we also empirically observed (Sec. 4.5.4) that increasing K beyond 1 gives only a very small increase in accuracy. As described below, these relative votes are obtained by clustering an unbiased sample of all offsets seen at training time using mean shift (see Sec. 3.5.2). Unlike [37], a corresponding confidence weight w_{ljk} is assigned to each vote, given by the size of its cluster, and our experiments in Sec. 4.5.6 show these weights are critical for high accuracy. We will refer below to the *set* of relative votes for joint j at node l as $\mathcal{V}_{lj} = \{(\boldsymbol{\Delta}_{ljk}, w_{ljk})\}_{k=1}^K$.

We detail the test-time voting approach for OJR in Algorithm 2, whereby the set $\mathcal{X}_j^{\text{OJR}}$ of *absolute* votes cast by all pixels for each body joint j is collected. As with BPC, the vote weights are multiplied by the squared depth to compensate for differing surface areas of pixels. Optionally, the set $\mathcal{X}_j^{\text{OJR}}$ can be sub-sampled by taking either the top N_{sub} weighted votes or instead N_{sub} randomly sampled votes. Our results show that this can dramatically improve speed while maintaining high accuracy (Fig. 13(c)).

Compared to BPC, OJR more directly predicts joints that lie behind the depth surface, and can cope with joints that are occluded or outside the image frame. Fig. 7 illustrates the voting process for OJR.

3.4 Aggregating predictions

We have seen above how at test time both BPC and OJR can be seen as casting a set of weighted votes in world space for the location of the body joints. These votes must now be aggregated to generate reliable proposals for the positions of the 3D skeletal joints. Producing multiple proposals for each joint allows us to capture the inherent uncertainty in the data. These proposals are the final output of our algorithm. As we will see in our experiments, these proposals can accurately localize the positions of body joints from a single image. Given a whole sequence, the proposals could also be used by

1. We use K to indicate the *maximum* number of relative votes allowed. In practice we allow some leaf nodes to store fewer than K votes for some joints.

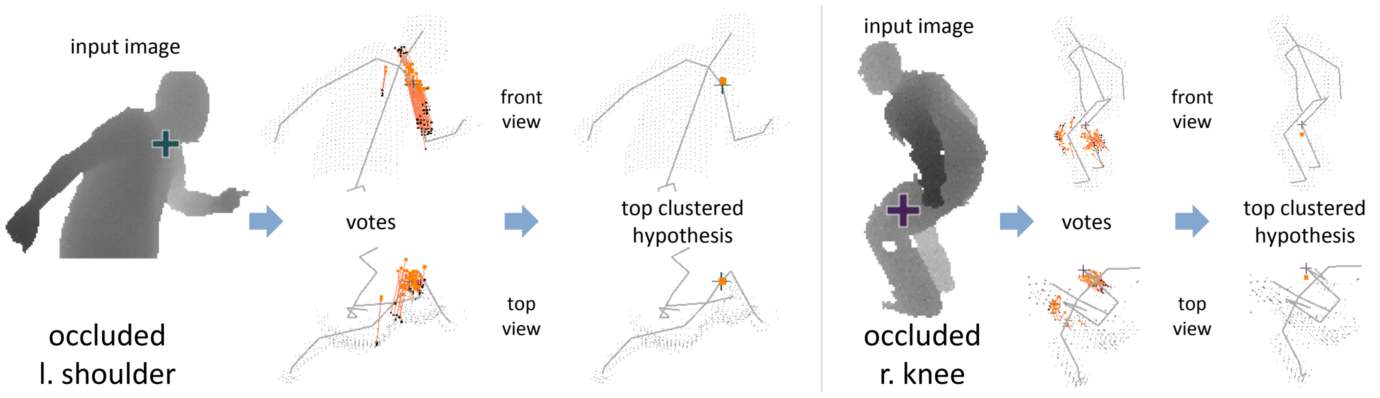

Fig. 7. **Offset joint regression voting at test time.** Each pixel (black square) casts a 3D vote (orange line) for each joint. Mean shift is used to aggregate these votes and produce a final set of 3D predictions for each joint. The highest confidence prediction for each joint is shown. Note accurate prediction of internal body joints even when occluded.

a tracking algorithm to self-initialize and recover from failure.

A simple option might be to accumulate the global centroid of the votes for each joint. However, the votes are typically highly multi-modal, and so such a global estimate is inappropriate. Instead we employ a local mode finding approach based on mean shift [55].

We first define a Gaussian Parzen density estimator per joint j as

$$p_j^m(\mathbf{x}') \propto \sum_{(\mathbf{x},w)\in\mathcal{X}_j^m} w \cdot \exp\left(-\left\|\frac{\mathbf{x}'-\mathbf{x}}{b_j^m}\right\|^2\right) , \qquad (6)$$

where $\mathbf{x}'$ is a coordinate in 3D world space, $m \in \{\text{BPC}, \text{OJR}\}$ indicates the approach, and b_j^m is a learned per-joint bandwidth.

Mean shift is then used to find modes in this density efficiently. The algorithm starts at a subset $\hat{\mathcal{X}}_j^m \subseteq \mathcal{X}_j^m$ of the votes, and iteratively walks up the density by computing the mean shift vector [55] until convergence. Votes that converge to the same 3D position within some tolerance are grouped together, and each group forms a body joint proposal, the final output of our system. A confidence weight is assigned to each proposal as the sum of the weights w of the votes in the corresponding group. For both BPC and OJR this proved considerably more reliable than taking the modal density estimate (*i.e.* the value $p_j(\mathbf{x}')$). For BPC the starting point subset $\hat{\mathcal{X}}_j^{\text{BPC}}$ is defined as all votes for which the original body part probability was above a learned probability threshold $\alpha_{c(j)}$. For OJR, all votes are used as starting points, i.e. $\hat{\mathcal{X}}_j^{\text{OJR}} = \mathcal{X}_j^{\text{OJR}}$.

3.5 Training

Each tree in the decision forest is trained on a set of images randomly synthesized using the method described in Sec. 2. Because we can synthesize training data cheaply, we use a different set of training images for each tree in the forest. As described above, each image is fully labeled: for BPC there is one body part label c per foreground pixel $\mathbf{u}$, and for OJR there is instead one pose $P = (\mathbf{p}_1, \ldots, \mathbf{p}_J)$ of 3D joint position vectors $\mathbf{p}_j$ per training image. For notational simplicity, we will assume that $\mathbf{u}$ uniquely encodes a 2D pixel location in a *particular* image, and thus can range across all pixels in all training images. A random subset of $N_{\text{ex}} = 2000$ example pixels from each image is used. Using a subset of pixels reduces training time and ensures a roughly even contribution from each training image.

The following sections describe training the structure of the trees, the leaf node prediction models, and the hyper-parameters. Note that we can decouple the training of the tree structure from the training of the leaf predictors; more details are given below.

3.5.1 *Tree structure training*

To train the tree structure, and thereby the weak learner parameters used at the split nodes, we use the standard greedy decision tree training algorithm. At each node, a set $\mathcal{T}$ of many candidate weak learner parameters $\boldsymbol{\theta} \in \mathcal{T}$ is sampled (these $\boldsymbol{\theta}$ parameters are those used in Eq. 4). Each candidate is then evaluated against an objective function I. Each sampled $\boldsymbol{\theta}$ induces a partition of the set $\mathcal{S} = \{\mathbf{u}\}$ of all training pixels that reached the node, into left $\mathcal{S}^{\text{L}}(\boldsymbol{\theta})$ and right $\mathcal{S}^{\text{R}}(\boldsymbol{\theta})$ subsets, according to the evaluation of the weak learner function (Eq. 4). The best $\boldsymbol{\theta}$ is selected according to

$$\boldsymbol{\theta}^\star = \underset{\boldsymbol{\theta}\in\mathcal{T}}{\text{argmin}} \sum_{d\in\{\text{L},\text{R}\}} \frac{|\mathcal{S}^d(\boldsymbol{\theta})|}{|\mathcal{S}|} I(\mathcal{S}^d(\boldsymbol{\theta})) \qquad (7)$$

which minimizes objective function I while balancing the sizes of the left and right partitions. We investigate both classification and regression objective functions, as described below. If the tree is not too deep, the algorithm then recurses on the example sets $\mathcal{S}^{\text{L}}(\boldsymbol{\theta}^\star)$ and $\mathcal{S}^{\text{R}}(\boldsymbol{\theta}^\star)$ for the left and right child nodes respectively.

Training the tree structure is by far the most expensive part of the training process, since many candidate parameters must be tried at an exponentially growing number of tree nodes as the depth increases. To keep the training times practical we employ a distributed implementation. At the high end of our experiments, training 3 trees to depth 20 from 1 million images takes about a day on a 1000 core cluster. (GPU based implementations are also possible and might be considerably cheaper). The resulting trees each have roughly 500K nodes, suggesting fairly balanced trees.

We next describe the two objective functions investigated in this work.

Classification. The standard classification objective $I^{\text{cls}}(\mathcal{S})$ minimizes the Shannon entropy of the distribution of the known ground truth labels corresponding to the pixels in $\mathcal{S}$. Entropy is computed as

$$I^{\text{cls}}(\mathcal{S}) = -\sum_c p(c|\mathcal{S}) \log p(c|\mathcal{S}) \ , \tag{8}$$

where $p(c|\mathcal{S})$ is the normalized histogram of the set of body part labels $c(\mathbf{u})$ for all $\mathbf{u} \in \mathcal{S}$.

Regression. Here, the objective is to partition the examples to give nodes with minimal uncertainty in the joint offset distributions at the leaves [56], [20]. In our problem, the offset distribution for a given tree node is likely to be highly multi-modal (see examples in Fig. 9). One approach might be to fit a Gaussian mixture model (GMM) to the offsets and use the negative log likelihood of the offsets under this model as the objective. However, GMM fitting would need to be repeated at each node for thousands of candidate weak learners, making this prohibitively expensive. Another possibility might be to use non-parametric entropy estimation [57], but again this would increase the cost of training considerably.

Following existing work [20], we instead employ the much cheaper sum-of-squared-differences objective:

$$I^{\text{reg}}(\mathcal{S}) = \sum_j \sum_{\mathbf{u} \in \mathcal{S}_j} ||\boldsymbol{\Delta}_{\mathbf{u}\to j} - \boldsymbol{\mu}_j||_2^2 \ , \tag{9}$$

where offset vector $\boldsymbol{\Delta}_{\mathbf{u}\to j} = \mathbf{p}_j - \mathbf{x}(\mathbf{u})$, and

$$\boldsymbol{\mu}_j = \frac{1}{|\mathcal{S}_j|} \sum_{\mathbf{u} \in \mathcal{S}_j} \boldsymbol{\Delta}_{\mathbf{u}\to j} \ , \tag{10}$$

$$\mathcal{S}_j = \{ \ \mathbf{u} \in \mathcal{S} \ | \ \|\boldsymbol{\Delta}_{\mathbf{u}\to j}\|_2 < \rho \ \} \ . \tag{11}$$

Unlike [20], we introduce an offset vector length threshold ρ to remove offsets that are large and thus likely to be outliers (results in Sec. 4.5.1 highlight importance of ρ). While this model implicitly assumes a uni-modal Gaussian, which we know to be unrealistic, for learning the tree structure, this assumption is tractable and can still produce satisfactory results.

Discussion. Recall that the two objective functions above are used for training the tree *structure*. We are then at liberty to fit the leaf prediction models in a different fashion (see next section). Perhaps counter-intuitively, we observed in our experiments that optimizing with the classification objective I^{cls} works well for the OJR task. Training for classification will result in image patches reaching the leaf nodes that tend to have both similar appearances and local body joint configurations. This means that for nearby joints, the leaf node offsets are likely to be small and tightly clustered. The classification objective further avoids the assumption of the offset vectors being Gaussian distributed.

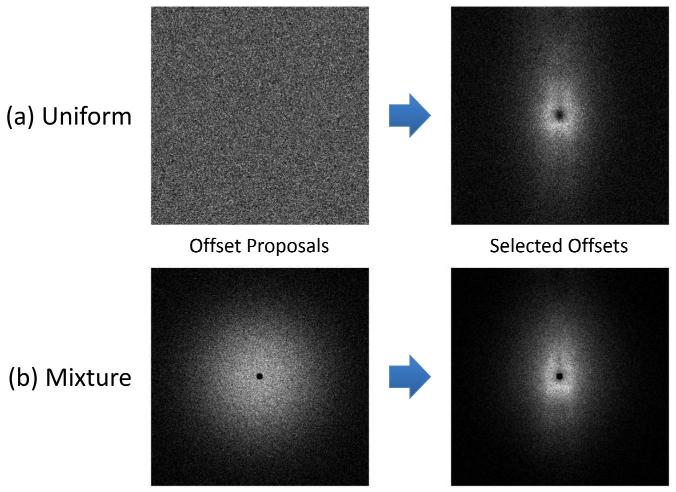

Fig. 8. **Sampling strategies for $\boldsymbol{\delta}$.** (a) A uniform proposal distribution is used to sample the 2D feature offsets $\boldsymbol{\delta}$ (see Eq. 3) during tree structure training. After training, a 2D histogram of the selected $\boldsymbol{\delta}$ values across all split nodes in the forest is plotted. The resulting distribution is far from uniform. (b) Building a mixture distribution to approximate these selected offsets, the tree structure training selects a similar distribution of offsets. However, as seen in Fig. 11(e,f), this can have a substantial impact on training efficiency.

We did investigate further node splitting objectives, including various forms of mixing body part classification and regression (as used in [20]), as well as variants such as separate regression forests for each joint. However, none proved better than either the standard classification or regression objectives defined above.

Sampling $\boldsymbol{\theta}$. The mechanism for proposing $\mathcal{T}$, the set of candidate weak learner parameters $\boldsymbol{\theta}$, merits further discussion, especially as the search space of all possible $\boldsymbol{\theta}$ is large. The simplest strategy is to sample $|\mathcal{T}|$ values of $\boldsymbol{\theta}$ from a *uniform* proposal distribution $p(\boldsymbol{\theta})$, defined here over some range of offsets $\boldsymbol{\phi} = (\boldsymbol{\delta}_1, \boldsymbol{\delta}_2)$ and over some range of thresholds τ. If the forest is trained using this proposal distribution, one finds that the empirical distribution $p(\boldsymbol{\theta}^\star)$ (computed over the chosen $\boldsymbol{\theta}^\star$ across all nodes in the forest) ends up far from uniform.

This suggests an iterative strategy: start from a uniform proposal distribution $p(\boldsymbol{\theta})$, train the forest, examine the distribution $p(\boldsymbol{\theta}^\star)$ of the chosen $\boldsymbol{\theta}^\star$s, design an improved *non*-uniform proposal distribution $p'(\boldsymbol{\theta})$ that approximates $p(\boldsymbol{\theta}^\star)$, and repeat. The intuition is that if you show the training algorithm more features that are likely to be picked, it will not need to see so many to find a good one. To make this procedure 'safe' the new proposal distribution $p'(\boldsymbol{\theta})$ can include a mixture with a uniform distribution with a small mixture coefficient (*e.g.* 10%). In practice we observed a small but consistent improvement in accuracy when iterating this process once (see Fig. 11(e,f)), though further iterations did not help. See Fig. 8 for an illustration. This idea is explored further in [58].

3.5.2 Leaf node prediction models

Given the learned tree structure, we must now train the prediction models at the leaf nodes. It is possible to first train the tree structure as described in the previous section, and then 'retro-fit' the leaf predictors by passing all the training examples down the trained tree to find the set of training examples that reach each individual leaf node. This allows us to investigate the use of different tree structure objectives for a given type of prediction model; see results below in Sec. 4.5.1.

For the BPC task, we simply take $p_l(c) = p(c|\mathcal{S})$, the normalized histogram of the set of body part labels $c(\mathbf{u})$ for all pixels $\mathbf{u} \in \mathcal{S}$ that reached leaf node l.

For OJR, we must instead build the weighted relative vote sets $\mathcal{V}_{lj} = \{(\mathbf{\Delta}_{ljk}, w_{ljk})\}_{k=1}^{K}$ for each leaf and joint. To do this, we employ a clustering step using mean shift, detailed in Algorithm 3. This algorithm describes how each training pixel induces a relative offset to all ground truth joint positions[2], and once aggregated across all training images, these are clustered using mean shift. To maintain practical training times and keep memory consumption reasonable we use reservoir sampling [59] to sample N_{res} offsets. Reservoir sampling is an algorithm that allows one to maintain a fixed-size unbiased sample from a potentially infinite stream of incoming samples; see [59] for more details. In our case, it allows us to uniformly sample N_{res} offsets at each node from which to learn the prediction models, without having to store the much larger set of offsets being seen.

Mean shift mode detection is again used for clustering, this time on the following density:

$$p_{lj}(\mathbf{\Delta}') \propto \sum_{\mathbf{\Delta} \in \mathcal{R}_{lj}} \exp\left(-\left\|\frac{\mathbf{\Delta}' - \mathbf{\Delta}}{b^\star}\right\|^2\right) . \qquad (12)$$

This is similar to Eq. 6, though now defined over *relative* offsets, without weighting, and using a learned bandwidth $b^\star$. Fig. 9 visualizes a few examples sets $\mathcal{R}_{lj}$ that are clustered. The positions of the modes form the relative votes $\mathbf{\Delta}_{ljk}$ and the numbers of offsets that reached each mode form the vote weights w_{ljk}. To prune out long range predictions which are unlikely to be reliable, only those relative votes that fulfil a per joint distance threshold λ_j are stored; this threshold could equivalently be applied at test time though would waste memory in the tree. In Sec. 4.5.4, we show that there is little or no benefit in storing more than $K = 2$ relative votes per leaf.

We discuss the effect of varying the reservoir capacity in Sec. 4.5.7. In our unoptimized implementation, learning these relative votes for 16 joints in 3 trees trained with 10K images took approximately 45 minutes on a single 8-core machine. The vast majority of that time is spent traversing the tree; the use of reservoir sampling ensures the time spent running mean shift totals only about 2 minutes.

3.5.3 Learning the hyper-parameters

Some of the hyper-parameters used in our methods are the focus of our experiments below in Sec. 4.4 and Sec. 4.5. Others are optimized by grid search to maximize our mean average precision over a 5K image validation set. These parameters include the probability thresholds α_c (the chosen values were between 0.05 and 0.3), the surface push-backs ζ_j (between 0.013m to 0.08m), the test-time aggregation bandwidths b_j^m (between 0.03m and 0.1m), the shared training-time bandwidth $b^\star$ (0.05m).

2. Recall that for notational simplicity we are assuming $\mathbf{u}$ defines a pixel in a particular image; the ground truth joint positions P used will therefore correspond for each particular image.

Algorithm 3 Learning relative votes

```
1: // Collect relative offsets
2: initialize $\mathcal{R}_{lj} = \emptyset$ for all leaf nodes $l$ and joints $j$
3: for all training pixels $\mathbf{u} \in \mathcal{S}$ do
4:     descend tree to reach leaf node $l = l(\mathbf{u})$
5:     compute 3D pixel position $\mathbf{x}(\mathbf{u})$
6:     for all joints $j$ do
7:         lookup ground truth joint positions $P = \{\mathbf{p}_j\}$
8:         compute relative offset $\mathbf{\Delta}_{\mathbf{u}\to j} = \mathbf{p}_j - \mathbf{x}(\mathbf{u})$
9:         store $\mathbf{\Delta}_{\mathbf{u}\to j}$ in $\mathcal{R}_{lj}$ with reservoir sampling
10: // Cluster
11: for all leaf nodes $l$ and joints $j$ do
12:     cluster offsets $\mathcal{R}_{lj}$ using mean shift
13:     discard modes for which $\|\mathbf{\Delta}_{ljk}\|_2 >$ threshold $\lambda_j$
14:     take top $K$ weighted modes as $\mathcal{V}_{lj}$
15: return relative votes $\mathcal{V}_{lj}$ for all nodes and joints
```

4 Experiments

In this section we describe the experiments performed to evaluate our method on several challenging datasets. We begin by describing the test data sets and error metrics, before giving some qualitative results. Following that, we examine in detail the effect of various hyper-parameters on BPC and then OJR. We finally compare the two methods, both to each other and to alternative approaches.

4.1 Test data

We use both synthetic and real depth images to evaluate our approach. For the synthetic test set ('MSRC-5000'), we synthesize 5000 test depth images, together with the ground truth body part labels and body joint positions, using the pipeline described in Sec. 2. However, to ensure a fair and distinct test set, the original mocap *poses* used to generate these test images are held out

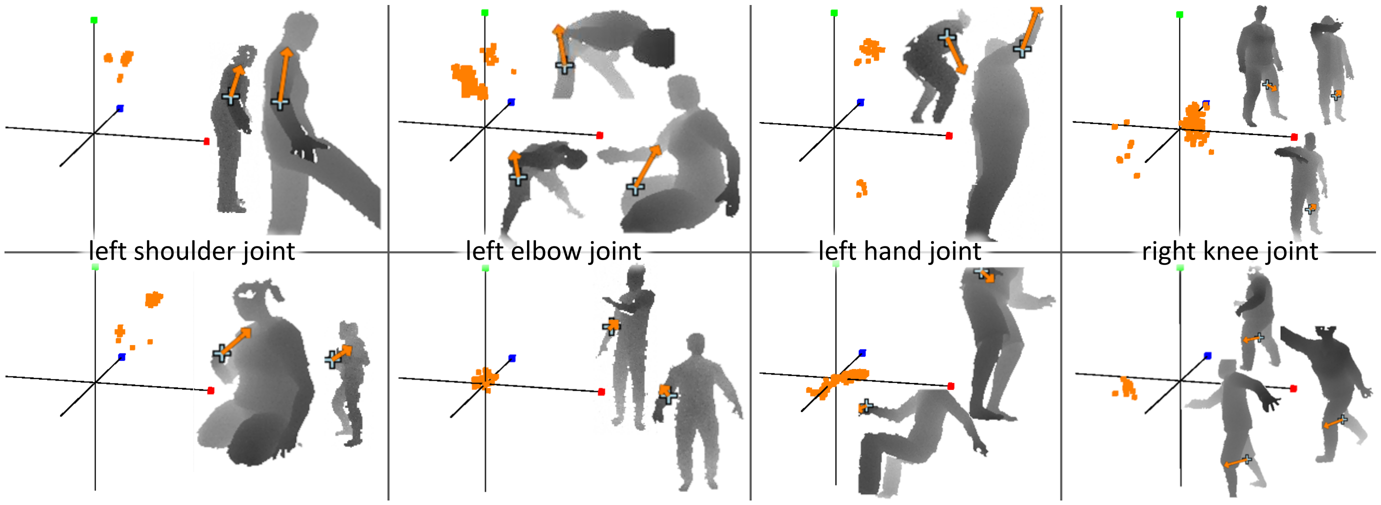

Fig. 9. **Empirical offset distributions for offset joint regression.** We visualize the set R_{lj} of 3D relative offset vectors $\mathbf{\Delta}_{\mathbf{u}\to j}$. Each set of axis represents a different leaf node, and the orange squares plot the vectors $\mathbf{\Delta}_{\mathbf{u}\to j} \in R_{lj}$ at that leaf. (The red, green, and blue squares indicate respectively the positive x, y, and z axes; each half-axis represents 0.5m in world space). We also show training images for each node illustrating the pixel that reached the leaf node as a cyan cross, and the offset vector as an orange arrow. Note how the decision trees tend to cluster pixels with similar local appearance at the leaves, but the inherent remaining ambiguity results in multi-modal offset distributions. The OJR algorithm compresses these distributions to a very small number of modes while maintaining high test accuracy.

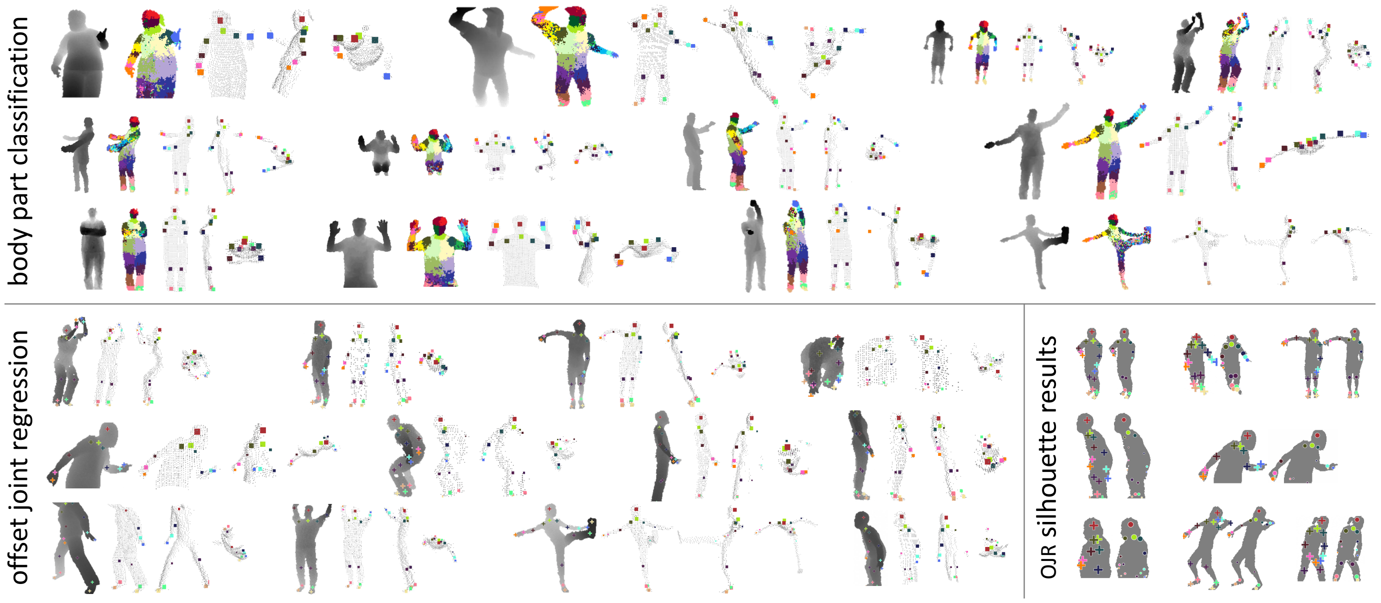

Fig. 10. **Example inferences on both synthetic and real test images.** In each example we see the input depth image, the inferred most likely body part labels (for BPC only), and the inferred body joint proposals shown as front, right, and top views overlaid on a depth point cloud. Only the most confident proposal for each joint above a fixed, shared threshold is shown, though the algorithms predict multiple proposals per joint. Both algorithms achieve accurate prediction of body joints for varied body sizes, poses, and clothing. We show failure modes in the bottom rows of the two larger panels. There is little qualitatively to tell between the two algorithms, though the middle row of the OJR results shows accurate prediction of even occluded joints (not possible with BPC), and further results in Sec. 4.6 compare quantitatively. Best viewed digitally at high zoom.

from the training data. Our real test set consists of 8808 frames of real depth images over 15 different subjects, hand-labeled with dense body parts and 7 upper body joint positions. We also evaluate on the real test depth data from [6].

As we will see, the results are highly correlated between the synthetic and real data. Furthermore, our synthetic test set appears to be far 'harder' than either of the real test sets due to its extreme variability in pose and body shape. After some initial experiments we thus focus our evaluation on the harder synthetic test set.

In most of the experiments below, we limit the rotation of the user to $\pm 120^\circ$ in both training and synthetic test data since the user is facing the camera (0°) in our main entertainment scenario. However, we do also investigate the full 360° scenario.

4.2 Error metrics

We quantify accuracy using (i) a classification metric (for BPC only) and (ii) a joint prediction metric (for both BPC and OJR). As the classification metric, we report the average per-class segmentation accuracy. This metric is computed as the mean of the diagonal elements of the confusion matrix between the ground truth body part label and the most likely inferred label. This metric weights each body part equally despite their varying sizes.

As the joint prediction metric, we generate recall-precision curves as a function of the predicted confidence threshold, as follows. All proposals below a given confidence threshold are first discarded; varying this threshold gives rise to the full recall-precision curve. Then, the first body joint proposal within a threshold D_{tp} meters of the ground truth position is taken as a true positive, while any other proposals that are also within D_{tp} meters count as false positives. This penalizes multiple spurious detections near the correct position which might slow a downstream tracking algorithm. Any proposals outside D_{tp} meters also count as false positives. Any joint for which there is no proposal of sufficient confidence within D_{tp} is counted as a false negative. However, we choose not to penalize joints that are invisible in the image as false negatives.

Given the full recall-precision curve, we finally quantify accuracy as average precision (the area under the curve) per joint, or mean average precision (mAP) over all joints. Note that, for example, a mean squared error (MSE) metric is inappropriate to evaluate our approach. Our algorithms aim to provide a strong signal to initialize and re-initialize a subsequent tracking algorithm. As such, evaluating our approach on MSE would fail to measure joints for which there are zero or more than one proposal, and would fail to measure how reliable the joint proposal confidence measures are. Our mAP metric effectively measures *all* proposals (not just the most confident): the only way to achieve a perfect score of 1 is to predict exactly one proposal for each joint that lies within D_{tp} of the ground truth position. For most results below we set $D_{\text{tp}} = 0.1\text{m}$ as the threshold, though we investigate the effect of this threshold below in Fig. 14c.

For BPC we observe a strong correlation of classification and joint prediction accuracy (*cf.* the blue curves in Fig. 11(a) and Fig. 15(b)). This suggests the trends observed below for one also apply for the other. For brevity we thus present results below for only the more interesting combinations of methods and metrics.

4.3 Qualitative results

Fig. 10 shows example inferences for both the BPC and OJR algorithms. Note high accuracy of both classification and joint prediction, across large variations in body and camera pose, depth in scene, cropping, and body size and shape (*e.g.* small child *vs.* heavy adult). Note that no temporal or kinematic constraints (other than those implicitly encoded in the training data) are used for any of our results. When tested on video sequences (not shown), most joints can be accurately predicted in most frames with remarkably little jitter.

A few failure modes are evident: (i) difficulty in distinguishing subtle changes in depth such as the crossed arms; (ii) for BPC, the most likely inferred part may be incorrect, although often there is still sufficient correct probability mass in distribution $p(c|\mathbf{u})$ that an accurate proposal can still result during clustering; and (iii) failure to generalize well to poses not present in training. However, the inferred confidence values can be used to gate bad proposals, maintaining high precision at the expense of recall.

In these and other results below, unless otherwise specified, the following training parameters were used. We trained 3 trees in the forest. Each was trained to depth 20, on 300K images per tree, using $N_{\text{ex}} = 2000$ training example pixels per image. At each node we tested 2000 candidate offset pairs ϕ and 50 candidate thresholds τ per offset pair, *i.e.* $|\mathcal{T}| = 2000 \times 50$. Below, unless specified, the number of images used refers to the total number used by the whole forest; each tree will be trained on a subset of these images.

4.4 Body part classification (BPC) experiments

We now investigate the effect of several training parameters on the BPC algorithm, using the classification accuracy metric. The following sections refer to Fig. 11.

4.4.1 *Number of training images*

In Fig. 11(a) we show how test accuracy increases approximately logarithmically with the number of randomly generated training images, though starts to tail off around 100K images. This saturation could be for several reasons: (i) the model capacity of the tree has been reached; (ii) the error metric does not accurately capture the continued improvement in this portion of the graph (*e.g.* the underlying probability distribution is

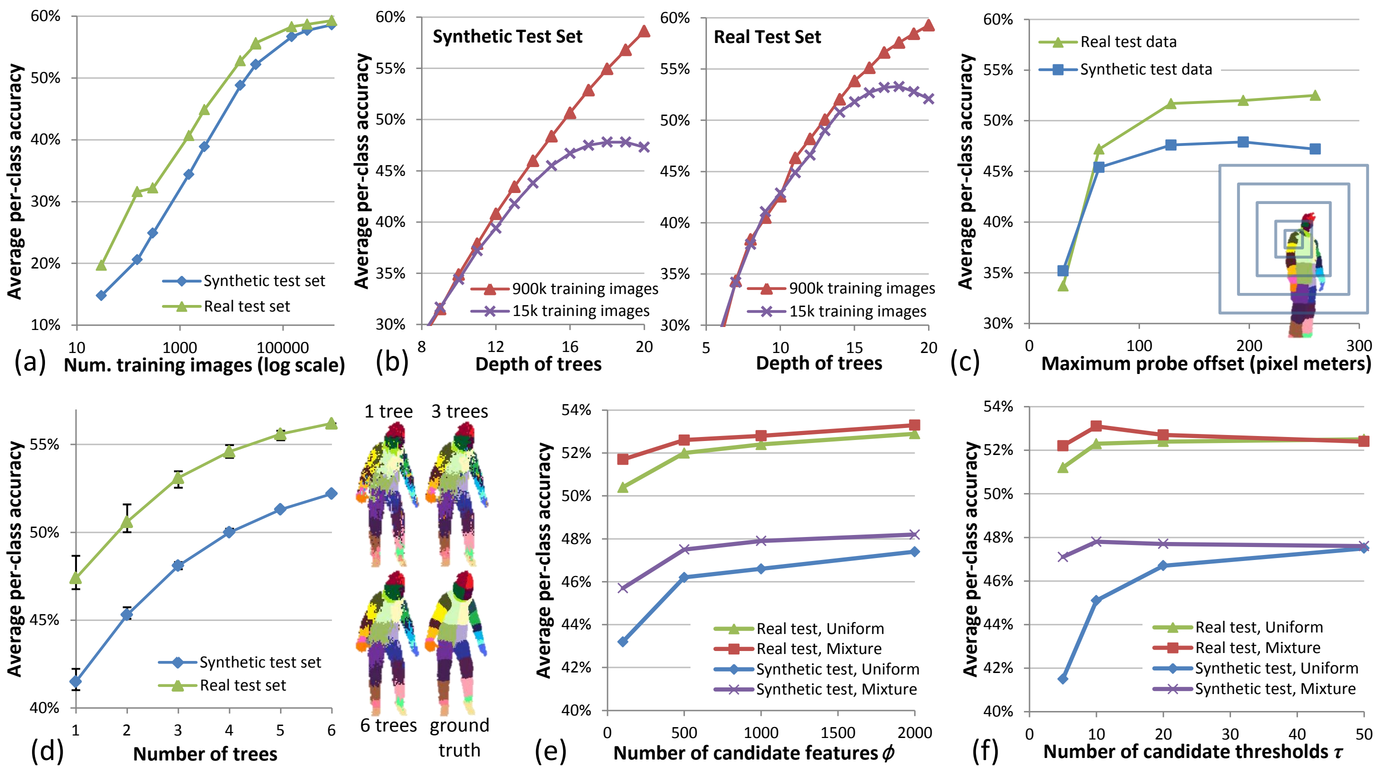

Fig. 11. **Training parameters *vs.* classification accuracy of the body part classification (BPC) algorithm.** (a) Number of training images. (b) Depth of trees. (c) Maximum depth probe offset. (d) Number of trees. (e,f) Number of candidate features ϕ and thresholds τ evaluated during training, for both real and synthetic test data, and using a uniform and mixture proposal distribution during tree structure training (see Sec. 3.5.1).

improving but the MAP label is constant); or (iii) the training images are rendered using a fixed set of only 100K poses from motion capture (though with randomly chosen body shapes, global rotations and translations). Given the following result, the first of these possibilities is quite likely.

4.4.2 *Depth of trees*

Fig. 11(b) shows how the depth of trees affects test accuracy using either 15K or 900K images. Of all the training parameters, depth appears to have the most significant effect as it directly impacts the model capacity of the classifier. Using only 15K images we observe overfitting beginning around depth 17, but the enlarged 900K training set avoids this. The high accuracy gradient at depth 20 suggests even better results can be achieved by training still deeper trees, at a small extra run-time computational cost and a large extra memory penalty.

4.4.3 *Maximum probe offset*

The range of depth probe offsets ϕ allowed during training has a large effect on accuracy. We show this in Fig. 11(c) for 5K training images, where 'maximum probe offset' means the maximum absolute value proposed for both x and y coordinates of $\boldsymbol{\delta}_1$ and $\boldsymbol{\delta}_2$ in Eq. 3. The concentric boxes on the right show the five tested maximum offsets, calibrated for a left shoulder pixel in that image (recall that the offsets scale with the world depth of the pixel). The largest maximum offset tested covers almost all the body. As the maximum probe offset is increased, the classifier is able to use more spatial context to make its decisions. (Of course, because the search space of features is enlarged, one may need a larger set $\mathcal{T}$ of candidate features during training). Accuracy increases with the maximum probe offset, though levels off around 129 pixel meters, perhaps because a larger context makes overfitting more likely.

4.4.4 *Number of trees*

We show in Fig. 11(d) test accuracy as the number of trees is increased, using 5K images for each depth 18 tree. The improvement starts to saturate around 4 or 5 trees, and is considerably less pronounced than when making the trees deeper. The error bars give an indication of the remarkably small variability between trees. The qualitative results illustrate that more trees tend to reduce noise, though even a single tree can get the overall structure fairly well.

4.4.5 *Number of features and thresholds*

Fig. 11(e,f) shows the effect of the number of candidate features ϕ and thresholds τ evaluated during tree training. Using the mixture proposal distributions for sampling ϕ and τ (see Sec. 3.5.1) allows for potentially

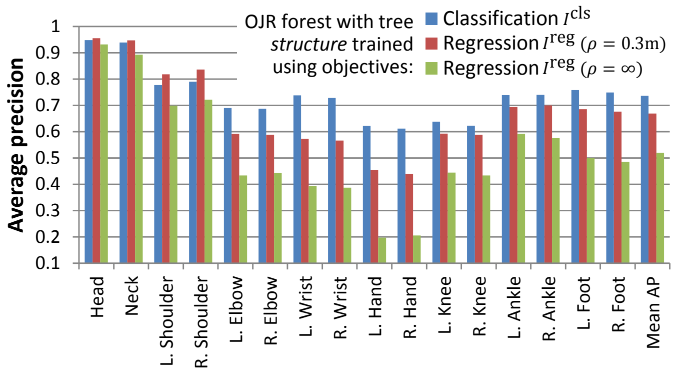

Fig. 12. **Comparison of tree structure objectives used to train the offset joint regression forest.** In all cases, after the tree structure has been trained, the same regression model is fit for each leaf node, as described in Sec. 3.5.2.

much higher training efficiency. Most of the gain occurs up to 500 features and 20 thresholds per feature. On the easier real test set the effects are less pronounced. These results used 5K images for each of 3 trees to depth 18. The slight peaks on the mixture proposal curves are likely down to overfitting.

4.4.6 Discussion

The trends observed above on the synthetic and real test sets appear highly correlated. The real test set appears consistently 'easier' than the synthetic test set, probably due to the less varied poses present. For the remaining experiments, we thus use the harder synthetic test set.

We now switch our attention to the joint prediction accuracy metric. We have observed (for example, *cf.* the blue curves in Fig. 11(a) and Fig. 15(b)) a strong correlation between the classification and joint prediction metrics. We therefore expect that the trends observed above also apply to joint prediction.

4.5 Offset joint recognition (OJR) experiments

The previous section investigated the effect of many of the system parameters for BPC. We now turn to the OJR algorithm and perform a similar set of experiments. The results in this section all make use of the average precision metric on joint prediction accuracy (see Sec. 4.2).

4.5.1 Tree structure objectives

The task of predicting continuous joint locations from depth pixels is fundamentally a regression problem. Intuitively, we might expect a regression-style objective function to produce the best trees for our approach. To investigate if this is indeed the case, we evaluated several objective functions for training the structure of the decision trees, using forests trained with 5K images. The results, comparing average precision on all joints, are summarized in Fig. 12.

Surprisingly, for all joints except head, neck, and shoulders, trees trained using the classification objective I^{cls} (*i.e.* training the tree *structure* for BPC using Eq. 8, but then retro-fitting the leaf prediction models for OJR; see Sec. 3.5.2) gave the highest accuracy. We believe the uni-modal assumption implicit in the regression objective (Eq. 9) may be causing this, and that classification of body parts is a reasonable proxy for a regression objective that correctly accounts for multi-modality. A further observation from Fig. 12 is that the ρ threshold parameter (used in Eq. 11 to remove outliers) does improve the regression objective, but not enough to beat the classification objective.

Another possible problem with Eq. 9 could be the summation over joints j. To investigate this, we experimented with training separate regression forests, each tasked with predicting the location of just a single joint. A full forest was trained with 5K images for each of four representative joints: head, l. elbow, l. wrist, and l. hand. With $\rho = \infty$, they achieved AP scores of 0.95, 0.564, 0.508, and 0.329 respectively (*cf.* the green bars in Fig. 12: 0.923, 0.403, 0.359, and 0.198 respectively). As expected, due to greater model capacity (*i.e.* one forest for each joint *vs.* one forest shared for all joints), the per-joint forests produce better results. However, these results are still considerably worse than the regression forests trained with the classification objective.

Given these findings, the following experiments all use the classification objective.

4.5.2 Tree depth and number of trees

Fig. 13(a) shows that mean average precision (mAP) rapidly improves as the tree depth increases, though it starts to level off around depth 18. As with BPC, the tree depth is much more important than the number of trees in the forest: with just one tree, we obtain a mAP of 0.730, with two trees 0.759, and with three trees 0.770.

4.5.3 Vote length threshold

We obtain our best results when using a separate voting length threshold λ_j for each joint (see Algorithm 3). These thresholds are optimized by grid search on a 5K validation data set, using a step size of 0.05m in the range $[0.05, 0.60]$m. In Fig. 13(b) we compare accuracy obtained using a single learned threshold shared by all joints (the blue curve), against the mAP obtained with per-joint thresholds (the dashed red line). When using a shared threshold it appears critical to include votes from pixels at least 10cm away from the target joints. This is likely because the joints are typically over 10cm away from the surface where the pixels lie.

We next investigate the effect of the metric used to optimize these thresholds. Interestingly, the optimized length thresholds λ_j turn out quite differently, according to whether the failure to predict an occluded joint is counted as a false negative or simply ignored. In Table 1, we see that longer range votes are chosen to maximize mAP when the model *is* penalized for missing occluded joints. In some cases, such as head, feet, and ankles, the difference is quite large. This makes sense: occluded

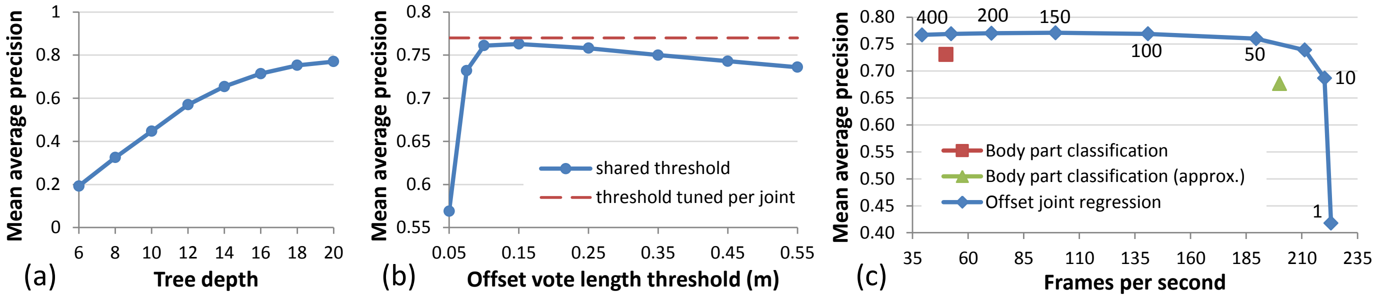

Fig. 13. **Effect of various system parameters on the offset joint regression (OJR) algorithm.** (a) mAP *vs.* tree depth. (b) mAP *vs.* a single, shared vote length threshold for all joints. (c) mAP *vs.* fps. The blue curve is generated by varying N_{sub}, the number of votes retained before running mean shift at test time.

joints tend to be further away from visible depth pixels than non-occluded joints, and predicting them will thus require longer range votes. This experiment used 30K training images.

4.5.4 Number of votes per leaf K

Increasing K, the maximum number of votes stored at each leaf, from 1 to 2 boosted mAP slightly from 0.763 to 0.770. However, for $K \in [2, 10]$ there was no appreciable difference in accuracy, and so for all other experiments presented we used $K = 2$. We hypothesize that aggregating over many image pixels that reach a diverse set of leaf nodes makes storing multiple local modes in each leaf node somewhat redundant. The comparison in Sec. 4.7.3 illustrates how accuracy may change as K gets much larger.

4.5.5 Using mean offsets

We tried using a single relative vote $\mathbf{\Delta}_{lj1}$ chosen to be the *mean* of the offsets reaching each leaf for each joint, rather than the top local mode. (Note that our model learned using mean shift with $K = 1$ is *not* the same as taking the mean of all the data). To achieve a sensible result, we found the mean vote's weight w_{ij1} to be very important. The best result obtained took w_{ij1} as the number of offsets within 5cm of the mean. Performance decreased from 0.763 (top *local* mode with $K = 1$) to 0.739 (mean of all offsets). Significant degradation was observed in the arm joints which exhibit much more multi-modality in the offsets: computed over elbows, wrists, and hands, the mAP dropped from 0.726 to 0.639. For robust results, using the top local mode thus appears better than the mean.

	errors on occluded joints:	
	not penalized λ_j	**penalized** λ_j
Head	0.20	0.50
Neck	0.20	0.35
L. Shoulder	0.30	0.45
R. Shoulder	0.35	0.40
L. Elbow	0.15	0.15
R. Elbow	0.15	0.15
L. Wrist	0.10	0.10
R. Wrist	0.10	0.10
L. Hand	0.15	0.10
R. Hand	0.10	0.15
L. Knee	0.35	0.30
R. Knee	0.45	0.30
L. Ankle	0.15	0.45
R. Ankle	0.15	0.55
L. Foot	0.10	0.45
R. Foot	0.10	0.55

TABLE 1
Optimized values for the test-time vote length thresholds λ_j under two different error metrics.

4.5.6 Learned relative vote weights w_{ljk}

To quantify the role of the relative vote weights, we tested our system with $w_{ljk} = 1, \forall l, j, k$. This uniform weight assignment decreased mAP dramatically from 0.770 to 0.542, underscoring the importance of learning the vote weights.

4.5.7 Training-time reservoir capacity N_{res}

The size of the reservoir had relatively little effect on accuracy. Reducing the reservoir capacity from 100 to 50 led to a small decrease in accuracy from mAP 0.770 to 0.766. Interestingly, increasing the reservoir capacity to 200 and 300 also caused a small drop (0.755 and 0.747, respectively). These results suggest that even a small sample of offsets is sufficient to characterize their distribution well for clustering.

4.5.8 Test-time vote sub-sampling N_{sub}

Even with the learned vote length thresholds λ_j, an average of about 1000 votes are cast per joint when processing a test image. As described in Algorithm 2, prior to aggregating votes with mean shift, we optionally sub-sample the voting space to at most N_{sub} votes. First, using fixed $N_{\text{sub}} = 200$, we experimented with different sub-sampling strategies: top N_{sub} weighted votes; uniformly random sampling; random sampling weighted by vote weight. These three methods achieved mAP scores of 0.770, 0.727, and 0.753, respectively.

Second, using the top N_{sub} strategy, we found that accuracy varies slowly with N_{sub}. We illustrate the substantial improvement in runtime speed this allows in

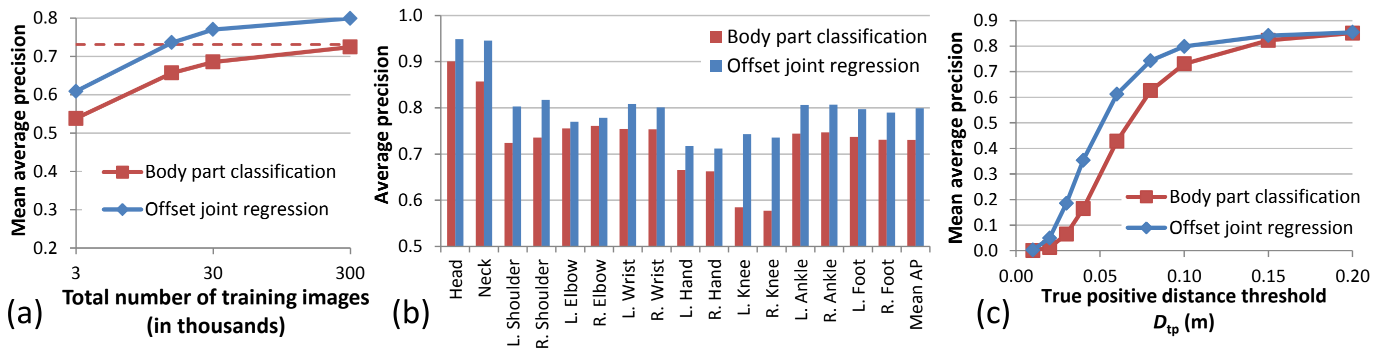

Fig. 14. **Comparing body part classification (BPC) with offset joint regression (OJR).** (a) Effect of total number of training images. The dashed line shows the best result of BPC trained with 900K images. (b) Average precision on each of the 16 test body joints. (c) Effect of the true positive threshold $D_{\rm tp}$ on the metric.

Fig. 13(c). This graph plots mAP against fps as a function of $N_{\rm sub}$, and compares with the BPC algorithm (and BPC running an approximate but faster implementation). Representative values of $N_{\rm sub}$ from 1 to 400 are overlaid on the plot. The best tradeoff between prediction accuracy and prediction speed is at about $N_{\rm sub} = 50$. All timings were measured on an 8-core machine taking advantage of CPU parallelism. Using sub-sampling to achieve speedups, while maintaining acuracy, is only possible if the vote weights are well correlated with predictive strength, further underscoring the importance of vote weight learning.

4.6 Comparison between BPC and OJR

Having investigated the BPC and OJR algorithms separately above, we now compare these algorithms to each other. Fig. 14(a) compares mean average precision for different training set sizes. In all cases we observe OJR performing more accurately than BPC. In Fig. 14(b) we show a per-joint breakdown of these results, using the best results obtained for each method (900K and 300K training images for BPC and OJR respectively).[3]

There are several possible reasons for OJR giving a more accurate result than BPC. One possibility is that OJR can directly regress the positions of joints inside the body. The joints showing the most substantial improvements (head, neck, shoulders, and knees) are also those where surface body parts cover a large area and are furthest from the joint center. Another ability of OJR is predicting occluded joints. When the mAP metric is changed to penalize failure to predict occluded joints, the improvement of OJR over BPC is even more apparent: 0.663 *vs.* 0.560 (both methods trained with 30K images). Example inferences showing localization of occluded joints are presented in Fig. 10 (OJR panel, middle row). As previously observed in Fig. 13(c), the OJR algorithm can also make predictions faster than BPC.

3. The results for OJR at 300K images were already better than the equivalent BPC forest trained with 900K images (see Fig. 14a), and so we chose not to expend the considerable energy in training a directly comparable 900K forest.

4.6.1 *True positive threshold*

A value of $D_{\rm tp} = 0.1$m is used as the true positive radius for most of the results presented here. We quantify the effect of this threshold on the mAP score in Fig. 14(c). The OJR algorithm maintains considerably higher mAP score as the radius shrinks in comparison to BPC.

4.6.2 *Full rotations*

To evaluate the scenario of full $360°$ rotation of the user, we trained BPC and OJR forests on images containing full rotations, and tested on 5K synthetic full rotation images. Despite the massive increase in left-right ambiguity, both approaches do remarkably well. Trained on 900K images, the BPC forest achieved an mAP of 0.655, while, trained on 300K images, the OJR forest achieved an mAP of 0.711. These results indicate that the forests can accurately learn the subtle visual cues that distinguish front and back facing poses. The residual left-right ambiguity might be handled by a tracking algorithm that propagates multiple hypotheses through time.

4.6.3 *Multiple people*

Our approach can propose joint positions for multiple people in the image: neither the forest evaluation at each pixel nor the aggregation step are limited to a single person. The forest could be explicitly trained with images containing multiple people, but in practice generalizes well without such explicit training. Preliminary results of this are given in the supplementary video.

4.6.4 *Silhouette images*

Although we focus on depth images in this paper, our methods can be applied without modification to 2D silhouette images. Silhouettes might readily be obtained using color segmentation or static background subtraction from RGB images. To prototype this scenario, we flattened both the training and test images to a fixed canonical depth, 2m, producing silhouettes in which the body size is unknown. To compute average precision for this 2D prediction task, we modified the true positive

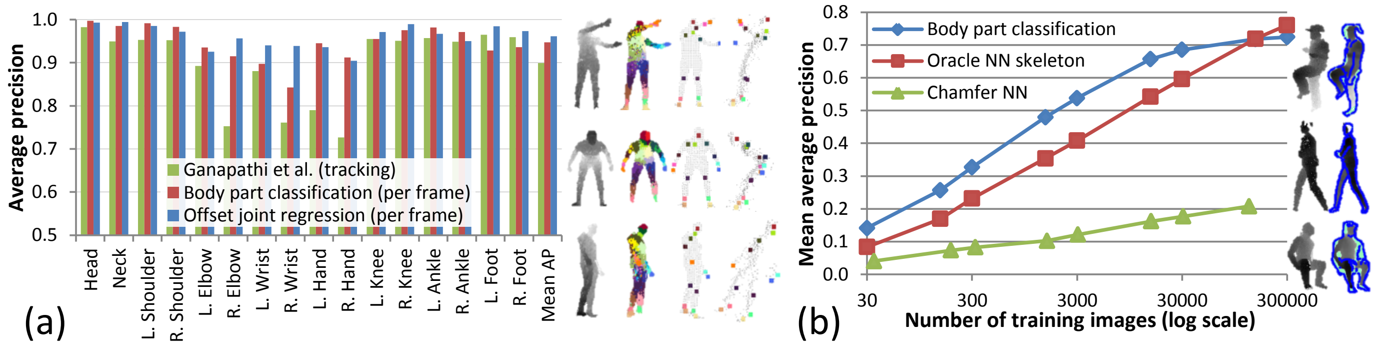

Fig. 15. **Comparisons to the state of the art.** (a) Comparison with [6]. Even without the kinematic and temporal constraints exploited by [6], both our approaches are able to more accurately localize body joints. (b) Comparison between BPC and nearest neighbor matching. Example chamfer matches shown right.

radius to an absolute *pixel* distance, $\hat{D}_{\text{tp}} = 10$ pixels. BPC achieves 0.465 mAP, and OJR achieves 0.596 mAP. Example inferences of OJR from silhouettes appear in the bottom right panel of Fig. 10, where the crosses show the ground truth, and the circles show the inferred joint predictions. Note how the proposal confidences (shown as the circle radii) correlate well with the ambiguity in the signal. The most probable reason for the significant drop in accuracy is due to missing information. For example, whenever a hand crosses in front of the body, the silhouettes will likely not contain enough information to reliably detect the hand.

4.7 Comparison to the state of the art

In this final evaluation section, we compare our algorithms to other methods.

4.7.1 *Comparison to Ganapathi et al. [6]*

The authors of [6] kindly provided their test data and results for direct comparison. Their algorithm uses sparse body part proposals from [5] and further tracks the skeleton with kinematic and temporal information. Their real data comes from a time-of-flight depth camera with very different noise characteristics to our structured light sensor. Without any changes to our training data or algorithm, Fig. 15(a) shows considerably improved joint prediction average precision for both BPC and OJR. Our algorithms also run at least 10 times faster, though we do of course require the large training corpus.

4.7.2 *Whole pose nearest neighbor matching*

Both of our methods attempt to find the body joints independently of each other. One alternative is to attempt to match a whole pose at once. Whole pose matching has the benefit of an extremely informative raw signal, but unfortunately has a search space that is exponential in the number of articulated joints. We compare our BPC algorithm with two variants of whole pose matching in Fig. 15(b). The first, idealized, variant matches the ground truth *test skeleton* to a set of training exemplar skeletons with optimal rigid translational alignment in 3D world space. Of course, in practice one has no access to the test skeleton. While this oracle is thus not an achievable algorithm, it does give us an upper bound on whole pose matching accuracy. As an example of a realizable system, the second variant uses chamfer matching [60] to compare the test image to the training exemplars. This is computed using depth edges and 12 orientation bins. To make the chamfer task easier, we throw out any cropped training or test images. We align images using the 3D center of mass, and found that further local rigid translation only reduced accuracy.

Our BPC algorithm, recognizing in parts, generalizes better than even the idealized skeleton matching until about 150K training images are reached. The speed of nearest neighbor chamfer matching is also drastically slower (2 fps) than our algorithm. While hierarchical matching [60] might be faster, one would still need a massive exemplar set to achieve comparable accuracy.

4.7.3 *Comparison between* OJR *and Hough forests [20]*

We compare OJR and Hough forests [20], using an identical tree structure for both. There are two main algorithmic differences. First, OJR clusters the offsets during training. This contrasts with Hough forests where all offset vectors are stored. To compare, we re-train the leaf nodes of our OJR forest, storing up to 400 offset votes for each joint, uniformly sampled (using all votes would have been prohibitive). The second difference is that we use a continuous voting space at test time, while Hough forests instead discretize the voting volume. Unfortunately, the inherent 3D nature of our problem makes discrete voting much less attractive than for 2D prediction. Our test data covers a large voting volume of 4m $\times$ 4m $\times$ 5m. To allow accurate localization we used a voxel resolution of 2cm per side, resulting in a voting volume with 10 million bins. At test time, we smooth the voting volume using a Gaussian of fixed standard deviation 1.3cm.

Due to memory and runtime constraints we compare on two representative joints (an 'easy' joint, head, and a 'hard' joint, left hand) in Fig. 16. Interestingly, even with discrete voting, using two votes per leaf performs slightly better than voting with a large number of stored offsets. This is likely due to the clustering removing many outliers making the final density much peakier. The results also show the clear improvement obtained by using the classification objective to train the tree structure instead of the regression objective as used in [20]. (This reiterates the findings from Fig. 12).

Our implementation of the Hough voting ran at approximately 0.5 fps for only 2 body joints, compared to around 200 fps for OJR which predicts all 16 joints. We experimented with a few voxel resolutions and smoothing kernel sizes, though the slow runtime speed prohibited selection of per joint smoothing kernel widths by grid search.

5 Conclusions

We have presented two algorithms, body part classification and offset joint regression. The two algorithms have much in common. They both use decision forests and simple depth-invariant image features. Both methods exploit a large, highly varied, synthetic training set, allowing us to train very deep trees. We have shown that the forests can learn invariance to both pose and shape while avoiding overfitting. The BPC approach introduces a set of surface body parts that the forest tries to infer. These body parts are aligned with the joints of interest, so that an accurate classification will localize the joints of the body. The OJR approach instead casts votes that try to directly predict the positions of interior body joints. In both methods, mean shift is used to aggregate votes to produce a final set of confidence-weighted 3D body joint proposals.

Our experiments have demonstrated that both algorithms can accurately predict the 3D locations of body joints in super-realtime from single depth or silhouette images. We have further shown state of the art accuracy and speed against several competing approaches. The body joint proposals might be used as an output in their own right, or used for initialization and re-initialization of a subsequent tracking and model fitting algorithm that exploits temporal and kinematic constraints.

Of our two approaches, which should you use? The numerical results show that OJR will give considerably higher accuracy than BPC. OJR also seems intuitively a 'cleaner' solution: it does not need the intermediate definition of body parts, and the offsets vote directly for interior joint positions. This also means that OJR is capable of predicting occluded joints, while BPC will instead give no proposals for an occluded joint. However, OJR proved considerably more complex: the obvious regression objective does not work well, and many hyperparameters had to be optimized against a validation set. A promising direction for future work is to investigate alternative efficient tree-structure learning objectives that handle multi-modal problems effectively.

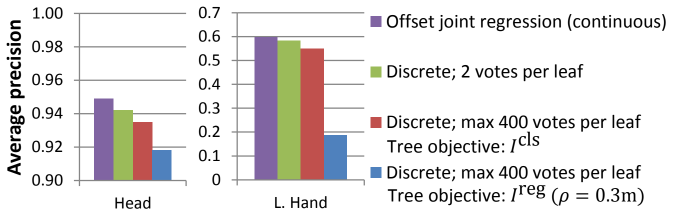

Fig. 16. **Comparison with Hough forest voting [20].** The Hough forest regression objective performs poorly for articulated joint such as hands. Vote compression and continuous voting improve accuracy slightly, while running 3200x faster (see text).

Acknowledgments

The authors would like to thank the many skilled engineers in Microsoft who built the Kinect tracking system on top of this research. In particular: Robert Craig, Matt Bronder, Craig Peeper, Momin Al-Ghosien, Sam Mann, and Ryan Geiss. We also thank John Winn, Duncan Robertson, Shahram Izadi, Ollie Williams, and Mihai Budiu for all their help and discussions. Finally thanks to Varun Ganapathi and Christian Plagemann for providing their test data.

References

[1] D. Grest, J. Woetzel, and R. Koch, "Nonlinear body pose estimation from depth images," in *In Proc. DAGM*, 2005.
[2] S. Knoop, S. Vacek, and R. Dillmann, "Sensor fusion for 3D human body tracking with an articulated 3D body model," in *Proc. ICRA*, 2006.
[3] Y. Zhu and K. Fujimura, "Constrained optimization for human pose estimation from depth sequences," in *Proc. ACCV*, 2007.
[4] M. Siddiqui and G. Medioni, "Human pose estimation from a single view point, real-time range sensor," in *CVCG at CVPR*, 2010.
[5] C. Plagemann, V. Ganapathi, D. Koller, and S. Thrun, "Real-time identification and localization of body parts from depth images," in *Proc. ICRA*, 2010.
[6] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun, "Real time motion capture using a single time-of-flight camera," in *Proc. CVPR*, 2010.
[7] Microsoft Corp. Redmond WA, "Kinect."
[8] C. Bregler and J. Malik, "Tracking people with twists and exponential maps," in *Proc. CVPR*, 1998.
[9] L. Sigal, S. Bhatia, S. Roth, M. Black, and M. Isard, "Tracking loose-limbed people," in *Proc. CVPR*, 2004.
[10] R. Wang and J. Popović, "Real-time hand-tracking with a color glove," in *Proc. ACM SIGGRAPH*, 2009.
[11] M. Brubaker, D. Fleet, and A. Hertzmann, "Physics-based person tracking using the anthropomorphic walker," *IJCV*, 2010.
[12] T. Sharp, "Implementing decision trees and forests on a GPU," in *Proc. ECCV*, 2008.
[13] R. Urtasun and T. Darrell, "Sparse probabilistic regression for activity-independent human pose inference," in *Proc. CVPR*, 2008.
[14] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from a single depth image," in *Proc. CVPR*. IEEE, 2011.
[15] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," in *Proc. CVPR*, 2003.
[16] J. Winn and J. Shotton, "The layout consistent random field for recognizing and segmenting partially occluded objects," in *Proc. CVPR*, 2006.
[17] L. Bourdev and J. Malik, "Poselets: Body part detectors trained using 3D human pose annotations," in *Proc. ICCV*, 2009.

[18] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon, "Efficient regression of general-activity human poses from depth images," in *Proc. ICCV*, 2011.
[19] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE TPAMI*, vol. 24, no. 5, pp. 603–619, 2002.
[20] J. Gall and V. Lempitsky, "Class-specific Hough forests for object detection," in *IEEE Trans. PAMI*, 2009.
[21] T. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *CVIU*, 2006.
[22] R. Poppe, "Vision-based human motion analysis: An overview," *CVIU*, vol. 108, 2007.
[23] M. Fischler and R. Elschlager, "The representation and matching of pictorial structures," *IEEE Trans. on Computers*, vol. 22, no. 1, pp. 67–92, Jan. 1973.
[24] P. Felzenszwalb and D. Huttenlocher, "Pictorial structures for object recognition," *IJCV*, vol. 61, no. 1, pp. 55–79, Jan. 2005.
[25] S. Ioffe and D. Forsyth, "Probabilistic methods for finding people," *IJCV*, vol. 43, no. 1, pp. 45–68, 2001.
[26] D. Ramanan and D. Forsyth, "Finding and tracking people from the bottom up," in *Proc. CVPR*, 2003.
[27] Z. Tu, "Auto-context and its application to high-level vision tasks," in *Proc. CVPR*, 2008.
[28] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, and A. Ng, "Discriminative learning of markov random fields for segmentation of 3D scan data," in *Proc. CVPR*, 2005.
[29] E. Kalogerakis, A. Hertzmann, and K. Singh, "Learning 3D mesh segmentation and labeling," *ACM Trans. Graphics*, vol. 29, no. 3, 2010.
[30] A. Agarwal and B. Triggs, "3D human pose from silhouettes by relevance vector regression," in *Proc. CVPR*, 2004.
[31] A. Kanaujia, C. Sminchisescu, and D. Metaxas, "Semi-supervised hierarchical models for 3D human pose reconstruction," in *Proc. CVPR*, 2007.
[32] R. Navaratnam, A. W. Fitzgibbon, and R. Cipolla, "The joint manifold model for semi-supervised multi-valued regression," in *Proc. ICCV*, 2007.
[33] G. Mori and J. Malik, "Estimating human body configurations using shape context matching," in *Proc. ICCV*, 2003.
[34] G. Shakhnarovich, P. Viola, and T. Darrell, "Fast pose estimation with parameter sensitive hashing," in *Proc. ICCV*, 2003.
[35] G. Rogez, J. Rihan, S. Ramalingam, C. Orrite, and P. Torr, "Randomized trees for human pose detection," in *Proc. CVPR*, 2008.
[36] B. Leibe, A. Leonardis, and B. Schiele, "Robust object detection with interleaved categorization and segmentation," *IJCV*, vol. 77, no. 1-3, pp. 259–289, 2008.
[37] J. Müller and M. Arens, "Human pose estimation with implicit shape models," in *ARTEMIS*, 2010.
[38] R. Okada and S. Soatto, "Relevant feature selection for human pose estimation and localization in cluttered images," in *Proc. ECCV*, 2008.
[39] H. Ning, W. Xu, Y. Gong, and T. S. Huang, "Discriminative learning of visual words for 3D human pose estimation," in *Proc. CVPR*, 2008.
[40] H. Sidenbladh, M. Black, and L. Sigal, "Implicit probabilistic models of human motion for synthesis and tracking," in *ECCV*, 2002.
[41] T. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theor. Comp. Sci.*, vol. 38, 1985.
[42] CMU Mocap Database, "http://mocap.cs.cmu.edu/."
[43] Autodesk MotionBuilder.
[44] V. Lepetit, P. Lagger, and P. Fua, "Randomized trees for real-time keypoint recognition," in *Proc. CVPR*, 2005, pp. 2:775–781.
[45] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. PAMI*, vol. 24, 2002.
[46] B. Shepherd, "An appraisal of a decision tree approach to image classification," in *IJCAI*, 1983.
[47] J. R. Quinlan, "Induction of decision trees," *Mach. Learn*, 1986.
[48] Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees," *Neural Computation*, vol. 9, no. 7, pp. 1545–1588, 1997.
[49] L. Breiman, "Random forests," *Mach. Learning*, vol. 45, no. 1, pp. 5–32, 2001.
[50] A. Criminisi, J. Shotton, and E. Konukoglu, "Decision forests: A unified framework," *NOW Publishers, To Appear*, 2012.
[51] F. Moosmann, B. Triggs, and F. Jurie, "Fast discriminative visual codebooks using randomized clustering forests," in *NIPS*, 2006.
[52] J. Shotton, M. Johnson, and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in *Proc. CVPR*, 2008.
[53] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "*TextonBoost*: Joint appearance, shape and context modeling for multi-class object recognition and segmentation," in *Proc. ECCV*, 2006.
[54] A. Criminisi, J. Shotton, D. Robertson, and E. Konukoglu, "Regression forests for efficient anatomy detection and localization in CT studies," in *MCV, MICCAI workshop*, 2010.
[55] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. PAMI*, vol. 24, no. 5, 2002.
[56] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin, "The elements of statistical learning: data mining, inference and prediction," *The Mathematical Intelligencer*, vol. 27, no. 2, pp. 83–85, 2005.
[57] S. Nowozin, "Improved information gain estimates for decision tree induction," in *Proc. Int. Conf. on Machine Learning*, 2012.
[58] A. Montillo, J. Shotton, J. Winn, J. Iglesias, D. Metaxas, and A. Criminisi, "Entangled decision forests and their application for semantic segmentation of CT images," in *Proc. IPMI*, 2011.
[59] J. S. Vitter, "Random sampling with a reservoir," *ACM Trans. Math. Software*, vol. 11, no. 1, pp. 37–57, 1985.
[60] D. Gavrila, "Pedestrian detection from a moving vehicle," in *Proc. ECCV*, Jun. 2000.

Jamie Shotton is a Senior Researcher in the Machine Learning & Perception Group at Microsoft Research Cambridge, where he has worked since 2008. Prior to this he graduated with a BA in computer science and a PhD in computer vision from the University of Cambridge, and worked as a post-doctoral Toshiba Fellow at Toshiba's Corporate Research and Development Center in Japan. His recent work has focused on human pose estimation for Microsoft's Kinect camera. This work was awarded the Best Paper Award at CVPR 2011, and the Royal Academy of Engineering's MacRobert Award 2011. His broader on-going research includes gesture and action recognition, object recognition, medical imaging, and machine learning.

Ross Girshick is a postdoctoral fellow working with Jitendra Malik at UC Berkeley. He recently received his PhD in Computer Science from the University of Chicago, where he was advised by Pedro Felzenszwalb. His primary research interests are in developing representations and machine learning techniques for object recognition. While completing his PhD studies, Ross lead winning entries in the PASCAL VOC object detection challenge. He also interned as a researcher at Microsoft Research Cambridge. Ross holds an MS in Computer Science from the University of Chicago and a BS in Computer Science from Brandeis University.

Andrew Fitzgibbon is a Principal Researcher at Microsoft Research, Cambridge, UK. His research interests are in the intersection of computer vision and computer graphics, with occasional excursions into machine learning and neuroscience. Recent papers have been on models of nonrigid motion, human 3D perception, large-scale image search, and nonlinear optimization. He has co-authored "best papers" at ICCV, CVPR, ISMAR, and BMVC, and software he wrote won an Engineering Emmy Award in 2002 for significant contributions to the creation of complex visual effects. He studied at University College Cork, at Heriot-Watt university, and received his PhD from Edinburgh University in 1997. Until June 2005 he held a Royal Society University Research Fellowship at Oxford University's Department of Engineering Science.

Toby Sharp is a Principal Research Software Development Engineer in the Machine Learning and Perception group at Microsoft Research, Cambridge, where he has worked since 2005. He has transferred several research projects into Microsoft technologies and products, and has contributed computer vision components to Microsoft's Kinect, Office and LifeCam software. His other work includes computer graphics, image editing and GPU programming. He holds a BSc in Mathematics from the University of York. Prior to joining Microsoft, he developed consumer photo and video editing software for Serif in the UK. In 2010 he received the Excellence in Design award from RSNA for real-time virtual colonoscopy software. In 2011 he and colleagues received the Best Paper prize at CVPR and the Royal Academy of Engineering MacRobert Award for the machine learning contribution to Kinect's body tracking.

Mat Cook works as a developer at Microsoft Research. He specialises in graphics and synthetic ground truth generation. He played a key role on the team that built Kinect's body part recognition algorithm. This won the Best Paper award at CVPR 2011 and the Royal Academy of Engineering's MacRobert Award in 2011. He holds a BSc in Physics from Imperial College, London.

Mark Finocchio is a Principal Software Engineer in the Xbox Incubation group at Microsoft, where he has worked since 1997. Prior to this he graduated from New Mexico State University with a BS in Electrical Engineering and an MS in Computer Science. He is a recipient of the Microsoft Technical Recognition Award for Outstanding Technical Achievement for his work on real-time human pose estimation for Kinect.

Richard Moore is a director of research and development at ST-Ericsson. He graduated from the University of California, San Diego and has a Master's degree from Pepperdine University, in management science and business administration, respectively. Prior to ST-Ericsson he worked in the Paris labs of Roland Moreno, the inventor of the smart card, and in various start-ups in communications and imaging, before joining Microsoft's Xbox incubation team responsible for Kinect.

Pushmeet Kohli is a research scientist in the Machine Learning and Perception group at Microsoft Research Cambridge, and an associate of the Psychometric Centre, University of Cambridge. His PhD thesis written at Oxford Brookes University was the winner of the British Machine Vision Association's Sullivan Doctoral Thesis Award, and a runner-up for the British Computer Society's Distinguished Dissertation Award. Pushmeet's research revolves around Intelligent Systems and Computational Sciences with a particular emphasis on algorithms and models for scene understanding and human pose estimation. His papers have appeared in SIGGRAPH, NIPS, ICCV, AAAI, CVPR, PAMI, IJCV, CVIU, ICML, AISTATS, AAMAS, UAI, ECCV, and ICVGIP and have won best paper awards in ECCV 2010, ISMAR 2011 and ICVGIP 2006, 2010.

Antonio Criminisi joined Microsoft Research Cambridge as a Visiting Researcher in 2000. In 2001 he moved to the Interactive Visual Media Group in Redmond (WA, USA) as a Post-Doctoral Researcher. In 2002 he moved back to Cambridge UK to join the Vision Group as a Researcher. In September 2011 he became Senior Researcher and now leads the medical image analysis team. Antonio's research interests include medical image analysis, object recognition, image and video analysis and editing, teleconferencing, 3D reconstruction with application to virtual reality, forensic science, and history of art.

In July 1996 he received a degree in Electronics Engineering from the University of Palermo, and in December 1999, he obtained the Doctor of Philosophy degree in Computer Vision at the University of Oxford. His thesis 'Accurate Visual Metrology from Single and Multiple Uncalibrated Images' won the British Computer Society Distinguished Dissertation Award 2000. He has won a number of best paper prizes in top computer vision conferences.

Alex Kipman is the General Manager of Incubation for the Interactive Entertainment Business at Microsoft. An innovator and a visionary, Kipman hand-selected and led the team that developed Kinect. Born in Brazil, he is the primary inventor of Kinect and is named on 60 patents directly related to Kinect. He graduated from Rochester Institute of Technology with a degree in software engineering. Among other honors, Kipman was named one of TIME Magazine's 2011 100 People of the Year.

Andrew Blake is a Microsoft Distinguished Scientist and has been the Laboratory Director of Microsoft Research Cambridge, UK, since 2010. Prior to joining Microsoft, Andrew trained in mathematics and electrical engineering in Cambridge, and studied for a doctorate in Artificial Intelligence in Edinburgh. He has been a pioneer in the development of the theory and algorithms that allow computers to behave as seeing machines. He has published several books, has twice won the prize of the European Conference on Computer Vision, and was awarded the IEEE David Marr Prize in 2001. In 2006 the Royal Academy of Engineering awarded him its Silver Medal, and in 2007 the Institution of Engineering and Technology presented him with the Mountbatten Medal. He was elected Fellow of the Royal Academy of Engineering in 1998, of the IEEE in 2008, and of the Royal Society in 2005. In 2010, Andrew was elected to the council of the Royal Society. In 2011, he and colleagues received the Royal Academy of Engineering MacRobert Award for their machine learning contribution to Microsoft Kinect.