# Efficient Regression of General-Activity Human Poses from Depth Images

Ross Girshick[†⋆] Jamie Shotton[†] Pushmeet Kohli[†] Antonio Criminisi[†] Andrew Fitzgibbon[†]
[†]Microsoft Research Cambridge [⋆]University of Chicago

## Abstract

*We present a new approach to general-activity human pose estimation from depth images, building on Hough forests. We extend existing techniques in several ways: real time prediction of multiple 3D joints, explicit learning of voting weights, vote compression to allow larger training sets, and a comparison of several decision-tree training objectives. Key aspects of our work include: regression directly from the raw depth image, without the use of an arbitrary intermediate representation; applicability to general motions (not constrained to particular activities) and the ability to localize occluded as well as visible body joints. Experimental results demonstrate that our method produces state of the art results on several data sets including the challenging MSRC-5000 pose estimation test set, at a speed of about 200 frames per second. Results on silhouettes suggest broader applicability to other imaging modalities.*

## 1. Introduction

Estimation of the pose of the human body from images has been a goal of computer vision for decades, and the recent availability of high-speed depth sensors has made realtime body tracking a reality [13]. However, even with depth images, the best existing systems exhibit 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). In this paper we combine some ideas from the regression-based approaches that have been a staple of monocular 2D human pose estimation [1, 19, 10, 15] with the tools of high-speed object recognition based on decision trees.

In particular, we address the task of general-activity pose estimation [19], where the subjects are assumed capable of any motion, rather than introducing a prior restricting the range of motions. We assume a system architecture where a fast discriminative process predicts a set of candidate joint positions from each image independently, which is fed to a 'downstream' kinematic tracker, for example a generative model such as [4], in order to produce final pose estimates. Our focus in this paper is on the first phase: designing a fast algorithm which generates a high-quality shortlist of candidates for each joint position.

A common theme in many recent approaches for human pose estimation has been the focus on localizing different body parts [2, 16, 18]. For example, Shotton *et al.* [18], which appears to represent the state of the art as implemented in the Kinect system, works by segmenting the different human body parts using a random forest classifier. The resulting segmentation is used to localize the parts' spatial modes, which in turn are used for predicting particular joint locations. Such classification approaches are extremely efficient and have been shown to obtain good results, but suffer from a number of problems:

- the methods require an arbitrary definition of body parts that roughly align with the body joints;
- the surface segmentation modes lie on the *surface*, whereas joints are *inside* the body; and
- the location of a joint cannot be estimated when the associated part is occluded.

In contrast, the regression-based approaches cited above provide a direct prediction of the joint positions, even under occlusion. However, even with an appropriate image descriptor, few of the approaches can run at very high speed, because the regressors used typically require comparison of the descriptors with a potentially large collection of relevance vectors (even after sparsification) for activity-independent estimation [19].

A related problem is solved in object localization. For example, in the implicit shape model (ISM) [11], visual words are used to learn voting offsets to predict 2D object centers. ISM has been extended in two ways relevant to this paper's work. Müller *et al.* [14] apply ISM to body tracking by learning separate offsets for each body joint. Gall and Lempitsky [7] 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. Advantages over the body-part-based methods include: pixels which belong to different body parts or at different locations in the image can combine their votes; and a single depth pixel can vote for the location of any subset of joints. Another, somewhat different, random forest based method for pose estimation was proposed by [17]. Their method quantizes the space of rotations and gait cycle and using classification forests to approximate the gait/rotation regression problem, but does not directly produce a detailed pose estimate.

Our method may be seen as a novel combination of [7] and [14], with the following additional contributions:

- We examine several regression and classification objective functions for decision tree learning, showing that a pure classification objective based on body parts for learning tree *structure* combined with leaf-node regression models for predicting continuous outputs yields best results.
- We employ regression models that compactly summarize the offset distributions at leaf nodes ('vote compression'), which not only make our method much faster but also much more accurate compared to [7].
- We learn the model hyper-parameters that weight votes, yielding a significant accuracy increase.
- Our novel use of reservoir sampling, vote compression, and test-time subsampling enables training from large data sets and permits super-realtime test performance.

## 2. Data

The Kinect sensor [13] comprises both a traditional RGB camera feed and a structured light depth feed. The depth readings are calibrated, giving a per-pixel scene distance in meters at high resolution. Depth cameras bring many advantages to pose estimation including easy background subtraction, color and texture invariance, and ease of synthesizing realistic training data. In this work we use only the depth information and assume foreground/background separation. Using just depth allows tracking in the dark and keeps computation cost low.

Despite the benefits of depth imaging, humans come in a huge range of shapes, sizes and poses. We employ the training data from [18] where the use of a highly varied synthetic training data set was suggested, such that a classifier might learn invariance to shape, size and pose. Driven by a large corpus of general-activity motion capture data retargetted to diverse body shapes and sizes, this data set contains hundreds of thousands of distinct rendered depth images with ground truth body joint positions. Artificial noise, calibrated to match the Kinect sensor, was added. Since we address the problem of static body pose recognition, our training and testing algorithms operate on many single frames rather than sequences. For the experiments in this paper we aim to predict the 3D locations of 16 body joints: head, neck, shoulders, elbows, wrists, hands, knees, ankles, feet. We also evaluate on silhouette data, where the depth images are flattened to a fixed depth.

## 3. Joint Position Regression

Our algorithm infers the 3D position of several body joints by aggregating votes cast by a regression forest. In this section we describe regression forests: first how they are used at test time, and then how they are trained.

A regression forest is an ensemble of decision trees [3] that predicts continuous outputs [6]. Due to space limitations, we assume the reader is acquainted with the well-known classification random forest, *e.g.* [12]. Each binary decision tree consists of split nodes and leaf nodes: the split nodes contain tests which evaluate image features to decide whether to branch to the left or right child; the leaf nodes contain some prediction (either categorical for classification, or continuous for regression).

At the split nodes, we employ the features from [18] which compare the depth at nearby pixels to a threshold. Building on those used in [12], these features are extremely fast to evaluate, and additionally are depth-invariant and have been shown to discriminate human appearance in depth images well.

At each leaf node $l$ we store a distribution over the *relative* 3D offset to each body joint $j$ of interest (potentially to all joints), *i.e.* a continuous regression output. The use of large training sets means that storing all the offsets seen at training time [7] would be prohibitive, and we must instead use a compact representation of the distribution. Furthermore, even for fairly deep trees, we observe highly multimodal offset distributions (see Fig. 2). For many nodes and joints, approximating the distribution over offsets as a Gaussian would thus be inappropriate. We instead represent the distribution using a few 3D *relative vote* vectors $\mathbf{\Delta}_{ljk} \in \mathbb{R}^3$, obtained by taking the centers of the $K$ largest modes (indexed by $k$) found by mean shift.[1] Unlike [14], we assign a confidence weight $w_{ljk}$ to each vote, given by the size of its cluster. Our experiments in Sec. 4.3 highlight the importance of this. We refer below to the *set* of relative votes for joint $j$ at node $l$ as $V_{lj} = \{(\mathbf{\Delta}_{ljk}, w_{ljk})\}_{k=1}^{K}$.

In our approach, and in contrast to [11, 7], we store very few relative votes at each leaf node, *e.g.* $K = 1$ or 2. While the main reason for keeping $K$ small is for efficiency, we also empirically observe (Sec. 4.3) that increasing $K$ beyond 1 gives only a very small increase in accuracy.

### 3.1. Inference

We detail the test time inference in Algorithm 1, whereby the set $Z_j$ of absolute votes cast by all pixels for each body joint $j$ is aggregated using mean shift.

Note that only those relative votes that fulfil a per joint distance threshold $\lambda_j$ are used.[2] This threshold prunes out long range predictions which are unlikely to be reliable, and was found to improve accuracy considerably. Following [18], we re-weight the confidence of each relative vote to compensate for observing fewer pixels when imaging a person standing further from the camera. Optionally, the set $Z_j$ can be sub-sampled to dramatically improve speed while maintaining high accuracy (Fig. 7(c)). We include results below evaluating both taking the top $N$ weighted votes or instead taking $N$ randomly sampled votes.

To aggregate the absolute votes $Z_j$, we define per joint a continuous distribution over world space $\mathbf{z}'$ using a Gaus-

[1] We use $K$ to indicate the *maximum* number of relative votes for each joint. Some leaf nodes may store fewer than $K$ votes for some joints.

[2] The threshold could equivalently be applied at training time.

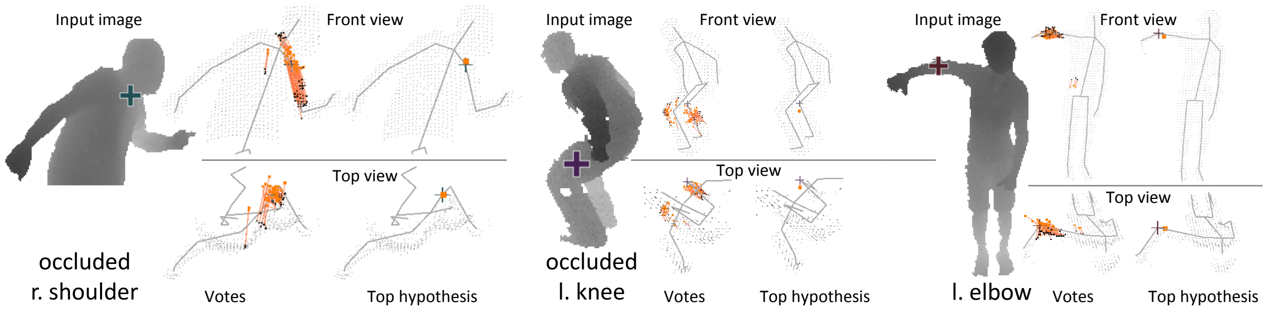


Figure 1. **Aggregation of pixel votes at test time.** Each pixel (black square) casts a 3D vote (orange line) for each joint. Mean shift is used to aggregate these votes and produce a final set of hypotheses for each joint. Note accurate predictions of internal body joints even when occluded. The highest confidence hypothesis for each joint is shown. NB 'left' refers to the user's left as if looking in a mirror.

sian Parzen density estimator as

$$p_j(\mathbf{z}') \propto \sum_{(\mathbf{z},w)\in Z_j} w \cdot \exp\left(-\left\|\frac{\mathbf{z}'-\mathbf{z}}{b_j}\right\|_2^2\right) , \qquad (1)$$

where $b_j$ is a learned per-joint bandwidth (see Sec. 3.2.2). We employ mean shift [5] to efficiently find the modes of this distribution[3], which form our 3D position hypotheses for each joint. We get a final confidence for each joint hypothesis by summing up the depth-adjusted weights $w$ of all votes that reached each mode. This was found to work considerably better than using the inferred density at the mode.

The proposal of multiple hypotheses for each joint allows us to capture the inherent uncertainty in the data. Note how our approach can, in contrast to [18], predict joints that lie behind the depth surface or are occluded or outside the image frame. To illustrate inference, each pane of Fig. 1 shows an input depth image, the top 50 votes cast for a target joint, and the highest scoring hypothesis.

[3]For extra speed we fix the set of neighbors used to compute the mean shift vectors after the first iteration. This has little effect on accuracy in practice.

**Algorithm 1** Inferring joint position hypotheses

```
1:  // Collect absolute votes
2:  initialize $Z_j = \emptyset$ for all joints $j$
3:  for all pixels $q$ in the test image do
4:      lookup 3D pixel position $\mathbf{x}_q = (x_q, y_q, z_q)^\top$
5:      for all trees in forest do
6:          descend tree to reach leaf node $l$
7:          for all joints $j$ do
8:              lookup weighted relative vote set $V_{lj}$
9:              for all $(\boldsymbol{\Delta}_{ljk}, w_{ljk}) \in V_{lj}$ do
10:                 if $\|\boldsymbol{\Delta}_{ljk}\|_2 \leq$ distance threshold $\lambda_j$ then
11:                     compute absolute vote $\mathbf{z} = \boldsymbol{\Delta}_{ljk} + \mathbf{x}_q$
12:                     adapt confidence weight $w = w_{ljk} \cdot z_q^2$
13:                     $Z_j := Z_j \cup \{(\mathbf{z}, w)\}$
14: // Aggregate weighted votes
15: sub-sample $Z_j$ to contain $N$ votes
16: aggregate $Z_j$ using mean shift on Eq. 1
17: return weighted modes as final hypotheses
```

## 3.2. Training

Training consists of estimating the structure and features of the trees, the set of relative votes $V_{lj}$ to each joint at each leaf, and the hyper-parameters of the model. An ideal learning algorithm would jointly optimize all these components to maximize our final accuracy metric (mean average precision) on the training data, which we do for several hyper-parameters. However, for the scale of our problem, this is sadly infeasible to do for all model parameters, and so we employ approximations whereby the tree structure is learned separately from the relative votes at the leaves and from the hyper-parameters. Despite these approximations, our evaluation in Sec. 4 demonstrates state of the art accuracy. We now describe the three stages in our training procedure, describing first how to learn the relative votes given an arbitrary binary decision tree structure, then the optimization of hyper-parameters, and finally how to train the tree structure.

### 3.2.1 Learning the leaf node regression models

Algorithm 2 describes how the set of relative votes is learned, given a known tree structure: each training pixel induces a relative offset to all ground truth joint positions and these are clustered using mean shift.

The problem addressed in this work requires a large number of training pixels (10s or 100s of millions). Voting with all training offsets for all body joints at test time (as done in Hough forests) is prohibitively slow for a real time system. Furthermore, storing all offsets in a Hough forest to predict 16 joint locations, using 10k images × 2k pixels per image requires roughly 10GB.

We thus summarize the offset distributions observed in the training set to reduce memory usage and improve test time speed. Looking at the empirical offset distributions (see Fig. 2), we typically observe scattered outliers and a few denser clumps. Simple options for summarization, such as sub-sampling the offsets uniformly or fitting a single Gaussian model, are inappropriate due to these outliers and multi-modality. We instead cluster the offsets per leaf per joint to obtain the set of relative votes $V_{lj}$. Mean shift is used again for clustering, on a density estimator similar

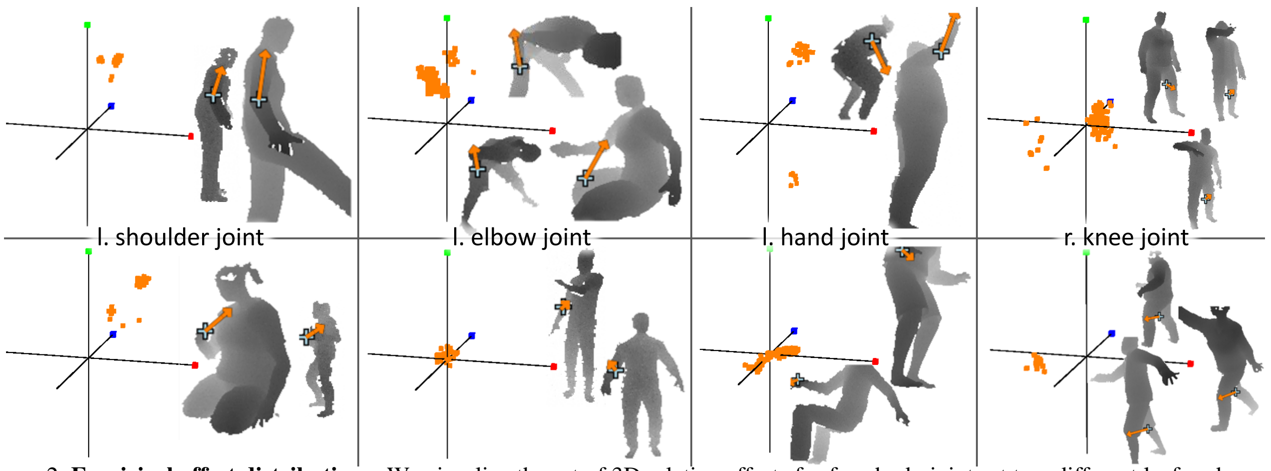


Figure 2. **Empirical offset distributions.** We visualize the set of 3D relative offsets for four body joints at two different leaf nodes each. For each set of axes we plot an orange square at the offset from each training pixel reaching a particular node to the target joint. (The red, green, and blue squares indicate respectively the positive x, y, and z axes; each half-axis represents 0.5m in world space). We 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. Our algorithm compresses these distributions to a very small number of modes while maintaining high test accuracy.

to Eq. 1, though here defined over *relative* offsets, without weighting, and using a learned bandwidth $b^\star$. 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 $w_{ljk}$. In Sec. 4.3, we show that there is no benefit to storing more than $K = 2$ relative votes per leaf.

To maintain practical training times and keep memory consumption reasonable we use reservoir sampling [20] to maintain a fixed-size unbiased sample of $C$ offsets. We discuss the effect of varying the reservoir capacity in Sec. 4.3. 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.

The trained model can be thought of as a Gaussian mixture model (GMM) with shared isotropic variance $b_j^2$. Note however that taking the top $K$ modes found by mean shift gives a very different output to learning a GMM using EM with a fixed $K$. In particular, fitting a model with $K = 1$ is *not* the same as fitting a single Gaussian to all the data.

**Algorithm 2** Learning relative votes
```
 1: // Collect relative offsets
 2: initialize $R_{lj} = \emptyset$ for all leaf nodes $l$ and joints $j$
 3: for all pixels $q$ in all training images $i$ do
 4:     lookup ground truth joint positions $\mathbf{z}_{ij}$
 5:     lookup 3D pixel position $\mathbf{x}_{iq}$
 6:     compute relative offset $\mathbf{\Delta}_{iq\to j} = \mathbf{z}_{ij} - \mathbf{x}_{iq}$
 7:     descend tree to reach leaf node $l$
 8:     store $\mathbf{\Delta}_{iq\to j}$ in $R_{lj}$ with reservoir sampling
 9: // Cluster
10: for all leaf nodes $l$ and joints $j$ do
11:     cluster offsets $R_{lj}$ using mean shift
12:     take top $K$ weighted modes as $V_{lj}$
13: return relative votes $V_{lj}$ for all nodes and joints
```

### 3.2.2 Learning the hyper-parameters

The training-time clustering bandwidth $b^\star$ and length threshold $\rho$, and the test-time per-joint aggregation bandwidth $b_j$ and vote length thresholds $\lambda_j$ were optimized by grid search to maximize mean average precision over a 5000 image validation set. The optimal bandwidth $b^\star$ was 0.05m. The optimized length thresholds $\lambda_j$ fall between 0.1m and 0.55m, indicating that some joints benefit from fairly long range offsets, while other joints require shorter range predictions. For comparison with [18], the main error metric does not penalize failures to predict occluded joints. Interestingly, when the error metric is changed to penalize such failures, the optimized $\lambda_j$ are typically larger (in some cases dramatically so) than when optimizing with the main error metric. We include a table of the $\lambda_j$ found under both error metrics in the supplementary material.

### 3.2.3 Learning the tree structure

To train the tree structure and image feature tests used at the split nodes, we use the standard greedy decision tree training algorithm. The set of all training pixels $Q = \{(i, q)\}$ (pixels $q$ in images $i$) is recursively partitioned into left $Q_\mathrm{l}(\phi)$ and right $Q_\mathrm{r}(\phi)$ subsets by evaluating many splitting function candidates $\phi$ under an error function $E(Q)$ (see below). The best splitting function is selected according to

$$\phi^* = \operatorname*{argmin}_{\phi} \sum_{s\in\{\mathrm{l},\mathrm{r}\}} \frac{|Q_s(\phi)|}{|Q|} E(Q_s(\phi)) \qquad (2)$$

which minimizes the error while balancing the sizes of the left and right partitions. If the tree is not too deep, the algorithm recurses on the examples $Q_\text{l}(\phi^*)$ and $Q_\text{r}(\phi^*)$ for the left and right children respectively. See *e.g.* [12, 18] for more details. As proxies to our ideal objective of maximizing joint detection accuracy on the training set, we investigated both regression and classification objective functions, as described below.

**Regression.** Here, the objective is to partition the examples to give nodes with minimal uncertainty in their joint offset distributions [9, 7]. In our problem, the offset distribution for a given tree node is likely to be highly multi-modal. A good approach might be to fit a GMM to the offsets (perhaps similarly to how the *leaf* regression models are learned) 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 splitting candidates, making this prohibitively expensive.

Following existing work [7], we employ the much cheaper sum-of-squared-differences objective:

$$E^{\text{reg}}(Q) = \sum_j \sum_{(i,q)\in Q_j} ||\mathbf{\Delta}_{iq\to j} - \boldsymbol{\mu}_j||_2^2 \,, \tag{3}$$

$$\boldsymbol{\mu}_j = \frac{1}{|Q_j|} \sum_{(i,q)\in Q_j} \mathbf{\Delta}_{iq\to j} \text{ , where} \tag{4}$$

$$Q_j = \{\, (i,q) \in Q \mid \|\mathbf{\Delta}_{iq\to j}\|_2 < \rho \,\} \tag{5}$$

Unlike [7], we introduce an offset vector length threshold $\rho$ to remove offsets that are large and thus likely to be outliers (results in Sec. 4.1 highlight its importance). While this model implicitly assumes a uni-modal Gaussian, which we know to be unrealistic, for learning the tree structure, this assumption can still produce satisfactory results.

**Classification.** Since the tree *structure* is learned independently from the regression models at the leaves, we explored a second objective $E^{\text{cls}}(Q)$ that minimizes the Shannon entropy for a classification task. Entropy is computed over the normalized histogram of a set of ground truth body part labels $c_{iq}$ for all $(i,q) \in Q$. We re-use the parts proposed in [18], a set of 31 body parts defined to localize particular regions of the body close to joints of interest.

Optimizing for body part classification is a good proxy for the regression task, as image patches reaching the resulting leaf nodes 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 objective further avoids the assumption of the offset vectors being Gaussian distributed.

We experimented with other node splitting objectives, including various forms of mixing body part classification and regression (as used in [7]), as well as separate regression forests for each joint. As we discuss in Sec. 4.1, we found that trees trained for the body part classification task outperformed trees trained with all other tested methods, including single joint regression.

## 4. Experimental Results and Discussion

In this section we evaluate several aspects of our work and compare with existing techniques. We evaluate on several datasets including the MSRC dataset of 5000 synthetic depth images [18] and the Stanford dataset of real depth images [8], obtaining state of the art results. We follow the protocol from [18] as follows. Joint prediction is cast as a detection problem, and average precision (AP) and its mean across joints (mAP) is used to measure accuracy. The most confident joint hypothesis within distance $D = 0.1$m of the ground truth counts as a true positive; any others count as false positives. Missing predictions for occluded joints do not count as false negatives, except in one experiment below.

### 4.1. Tree structure training objectives

We evaluated several objective functions for training the structure of the decision trees, using forests of 3 trees each trained to depth 20 with 5000 images. The results, comparing average precision on all joints, are summarized in Fig. 3. 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. Perhaps surprisingly then, for all joints except head, neck, and shoulders, trees trained using the body part classification objective from [18] gave the highest accuracy. We believe the uni-modal assumption implicit in the regression objective may be causing this, and that body part classification is a reasonable proxy for a regression objective that correctly accounts for multi-modality. Investigating efficient methods for fitting multi-modal distributions in a regression objective remains future work. In the remainder of this section we base all of our experiments on 3 trees trained to depth 20 using the body part classification objective.

### 4.2. Comparisons

**Hough forests [7].** We compare our use of offset clustering during training and a continuous voting space for testing, with the approach taken by [7] where all offset vectors are stored during training and a discretized voting volume is used for testing, given a fixed tree structure. The diverse MSRC-5000 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. Note that the inherent 3D nature of the problem makes discrete voting much less attractive than for 2D prediction. At test time, we smooth the voting volume using a Gaussian of fixed standard deviation 1.3cm.

We re-trained the leaf nodes, storing up to 400 offset votes for each joint, uniformly sampled (using all votes would have been prohibitive). Due to memory and runtime constraints we compare on two representative joints (head and left hand) in Fig. 4. Interestingly, even with discrete

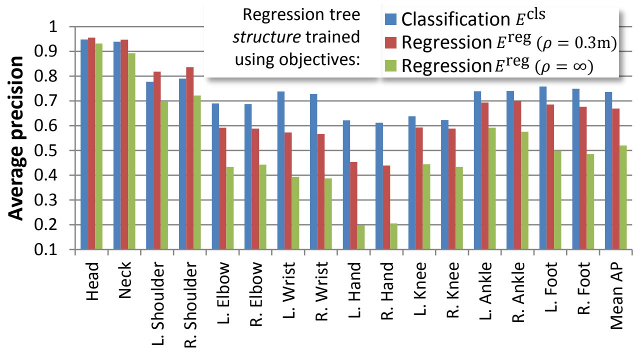


Figure 3. **Comparison of tree structure training objectives.** In all cases, after the tree structure has been trained, the same regression model is fit for each leaf node, as described in Sec. 3.2.1.

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 for training tree structure compared to the regression objective used in [7]. (This finding is also borne out in Fig. 3, especially for articulated joints.)

Our implementation of the Hough voting ran at approximately 0.5 fps for only 2 body joints, compared to 200 fps for our algorithm which predicts all 16 joints (both implementations unoptimized). 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.

**Shotton *et al.* [18].** We compared directly on the MSRC-5000 test set. We give some example inferences in Fig. 5. In Fig. 6(a) we compare mean average precision for different training set sizes. In all cases we observe significant improvements over [18]. Even with only 15k training images, our algorithm obtains a mAP of 0.736, edging out the best result in [18] of 0.731 which used 60 times more data. We obtain our best result, a mAP of 0.799, using 300k images.

In Fig. 6(b) we show a per-joint breakdown of our improvement over [18]. One source of improvement is likely to be the ability of algorithm to directly regress the positions of joints inside the body: the joints showing the most substantial improvements (head, neck, shoulder, and knee joints) are also those where surface body parts cover a large area and are furthest from the joint center.

Another ability of our algorithm is to predict occluded joints. When the mean average precision metric is changed to penalize failure to predict occluded joints, the improvement of our method over [18] is even more apparent: 0.663 *vs*. 0.560, yielding a difference of 10.3% compared to 8.5% when the error metric is unchanged (both methods trained with 30k images). Example inferences showing localization of some occluded joints are presented in Fig. 1 and Fig. 5 (middle row).

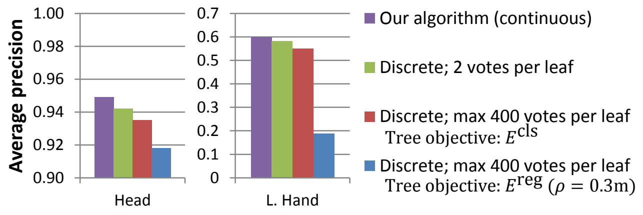


Figure 4. **Comparison with Hough forest voting [7].** The Hough forest regression objective performs poorly for articulated joint such as hands. Vote compression and continuous voting improve accuracy slightly, while running 3200x faster (see text).

Beyond predicting joint positions more accurately, our algorithm also makes predictions faster, running at 200 fps compared to the 50 fps achieved by [18]. See also Fig. 7c.

**Ganapathi *et al.* [8].** We also compared on the Stanford data set of real depth images. Our algorithm, predicting 3D joint poses for each frame independently, obtains a mAP of 0.961 on this data set. This closely matches the results of [18] (0.947), and surpasses the result of [8] (0.898) which additionally exploited temporal and kinematic constraints.

### 4.3. System parameters

We investigated the effects of various system parameters on prediction accuracy and runtime speed. We used 10k training images in the following experiments.

**Tree depth.** Fig. 7(a) shows that mean average precision rapidly improves as the tree depth increases, though it begins to level off around depth 18. The effect of tree depth is much more significant than the effect of varying 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.

**Vote length threshold.** We obtain our best results when tuning a separate voting length threshold $\lambda_j$ for each joint using a validation data set. In Fig. 7(b) we compare accuracy obtained using a single threshold shared by all joints, against the mAP obtained with per-joint thresholds, 0.770, which is plotted as a dashed red line. This experiment shows that it is critical to include votes from pixels at least 10cm away from the target joints, since the joints are typically over 10cm away from the surface where the pixels lie. After reaching a global threshold of 15cm, with mAP of 0.763, accuracy slowly decreases.

**Number of votes per leaf $K$.** Increasing $K$ from 1 to 2 boosted mAP slightly from 0.763 to 0.770. For the range of $K \in [2, 10]$, 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 reaching a diverse set of leaf nodes makes storing multiple local modes in each leaf node somewhat redundant. The comparison with Hough forests above illustrates how accuracy may change as $K$ gets much larger.

**Using the mean of offsets.** We also tried using a single relative vote $\mathbf{\Delta}_{lj1}$ chosen to be the *mean* of the offsets

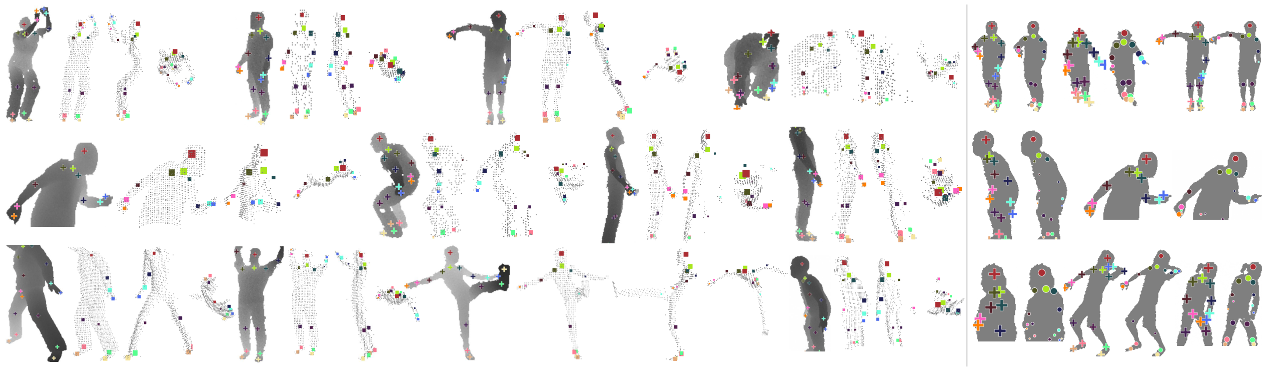
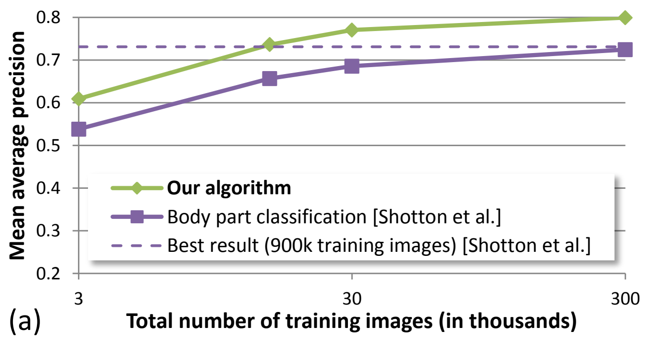


Figure 5. **Inferred joint positions.** (Left) Each example shows an input depth image with color-coded ground truth joint positions overlaid, and then inferred joint positions from front, right, and top views. The size of the boxes indicates the inferred confidence. Our algorithm achieves accurate prediction of internal body joints for varied body sizes, poses, and clothing. The middle row shows accurate prediction of even occluded joints, and the bottom row shows some failure cases. (Right) Example inference results on flattened 2D silhouettes. Ground truth joint positions are plotted as crosses and the highest scoring hypothesis for each joint appears as a color-coded circle, with size indicating confidence. Despite substantially more visual ambiguity, our algorithm is able to predict many joint positions accurately.



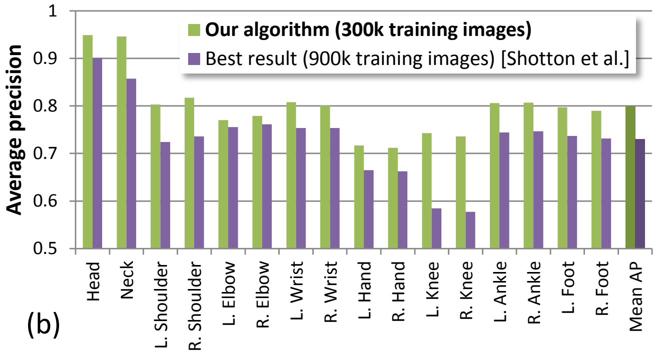


Figure 6. **Results on the MSRC-5000 test set compared to [18].** (a) Mean average precision versus total number of training images. (b) Average precision on each of the 16 test body joints. Our algorithm achieves substantially better accuracy with fewer training images.

reaching each leaf for each joint, rather than the top local mode. 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. Here, the mAP computed over elbows, wrists, and hands dropped from 0.726 to 0.639. For robust results, using the top local mode thus appears better than the mean.

**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 our strategy of learning vote weights.

**Reservoir capacity** $C$**.** The size of the reservoir had relatively little effect on accuracy. Reducing the reservoir capacity at training time 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.

**Test time vote sub-sampling** $N$**.** Even with the learned vote length thresholds $\lambda_j$, an average of about 1000 votes are cast per joint when processing a test image. Prior to aggregating votes with mean shift, we optionally sub-sample the voting space to at most $N$ votes. First, using fixed $N = 200$ we experimented with different sub-sampling strategies: top $N$ weighted votes; uniform sampling; sampling weighted by vote weight. The three methods achieved mAP scores of 0.770, 0.727, and 0.753, respectively. Using the top $N$ strategy, we find that accuracy varies slowly with $N$. We illustrate the substantial improvement in runtime speed this allows in Fig. 7(c), where mAP is plotted against fps as a function of $N$, and compare with [18] on similar hardware. Representative values of $N$ from 1 to 400 are overlaid on the plot. The best tradeoff between prediction accuracy and prediction speed is at about $N = 50$. All timings were measured on an 8-core machine taking advantage of CPU parallelism.

### 4.4. Predictions from 2D images

Though we focus on depth images in this paper, our method applies without modification to 2D silhouette im-

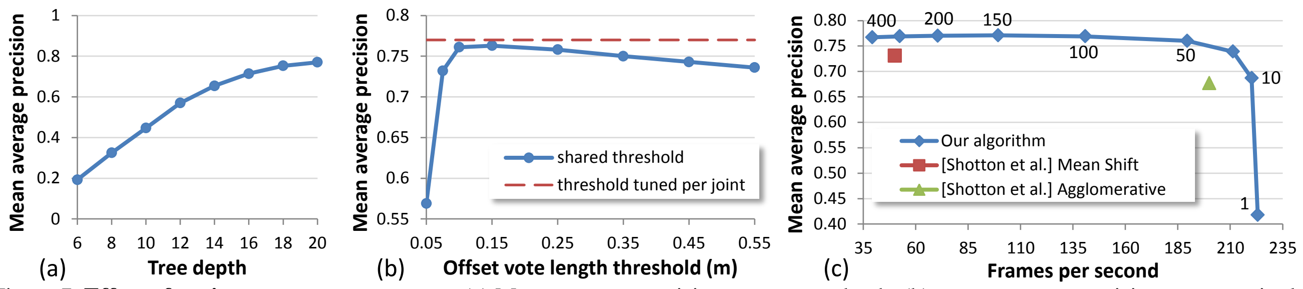


Figure 7. **Effect of various system parameters.** (a) Mean average precision versus tree depth; (b) mean average precision versus a single, shared vote length threshold for all joints; and (c) mean average precision versus frames per second as a function of the number of votes retained before running mean shift in the voting space at test time.

ages. To test 2D prediction accuracy, we flattened our training and test images to a fixed depth, 2m, producing silhouettes in which the pose scale is unknown. To compute average precision for this 2D prediction task, we modified the true positive radius to an absolute *pixel* distance, $D = 10$ pixels. Our algorithm compares very favorably to [18] on this task, achieving mAP 0.596 *vs*. 0.465. Example inferences from silhouettes appear in the right column of Fig. 5. Note how the hypothesis confidences correlate well with the ambiguity in the signal.

## 5. Discussion

We have presented a new, extremely efficient regression forest based method for human pose estimation in single depth or silhouette images. Unlike previous work [18], our method does not predict segmentations of the surface of the body, but instead directly predicts the positions of interior body joints even under occlusion. Our method extends Hough forests [7] by using compact models for storing and aggregating relative offset votes. We investigate different objectives for learning the structure of the forest and choices of vote representation and aggregation. Due to our emphasis on efficiency of training, these models can be automatically learned from large training sets. Results on several challenging real and synthetic datasets show considerable improvement over the state of the art while maintaining super-realtime speeds.

Our experimental evaluation reveals that, perhaps surprisingly, for learning the *structure* of the regression forest on our difficult multiple body joint prediction problem, the Hough forest objective function performs poorly relative to the body part classification objective used in [18]. We believe that this effect is due to the errors introduced by the implicit modeling of the inherently multi-modal offset distributions at the interior tree nodes as uni-modal Gaussians in the tree structure regression objective. This also may be compounded by the greedy nature of the tree learning procedure. Further investigation of alternative efficient tree structure learning regression objectives is a promising direction for future work.

**Acknowledgements.** We thank Toby Sharp, Mat Cook, John Winn, and Duncan Robertson for their help and interesting discussions.

## References

[1] A. Agarwal and B. Triggs. 3D human pose from silhouettes by relevance vector regression. In *Proc. CVPR*, 2004. 1

[2] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3D human pose annotations. In *Proc. ICCV*, 2009. 1

[3] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984. 2

[4] M. Brubaker, D. Fleet, and A. Hertzmann. Physics-based person tracking using the anthropomorphic walker. *IJCV*, 2010. 1

[5] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE TPAMI*, 24(5):603–619, 2002. 3

[6] 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. 2

[7] J. Gall and V. Lempitsky. Class-specific Hough forests for object detection. In *PAMI*. IEEE, 2009. 1, 2, 5, 6, 8

[8] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. Real time motion capture using a single time-of-flight camera. In *Proc. CVPR*, pages 755–762. IEEE, 2010. 5, 6

[9] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005. 5

[10] A. Kanaujia, C. Sminchisescu, and D. Metaxas. Semi-supervised hierarchical models for 3D human pose reconstruction. In *Proc. CVPR*, 2007. 1

[11] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *IJCV*, 77(1-3):259–289, 2008. 1, 2

[12] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *Proc. CVPR*, pages 2:775–781, 2005. 2, 5

[13] Microsoft Corp. Redmond WA. Kinect for Xbox 360. 1, 2

[14] J. Müller and M. Arens. Human pose estimation with implicit shape models. In *ARTEMIS*, 2010. 1, 2

[15] R. Navaratnam, A. W. Fitzgibbon, and R. Cipolla. The joint manifold model for semi-supervised multi-valued regression. In *Proc. ICCV*, 2007. 1

[16] C. Plagemann, V. Ganapathi, D. Koller, and S. Thrun. Real-time identification and localization of body parts from depth images. In *Proc. ICRA*, 2010. 1

[17] G. Rogez, J. Rihan, S. Ramalingam, C. Orrite, and P. Torr. Randomized trees for human pose detection. In *Proc. CVPR*, 2008. 1

[18] 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. 1, 2, 3, 4, 5, 6, 7, 8

[19] R. Urtasun and T. Darrell. Local probabilistic regression for activity-independent human pose inference. In *Proc. CVPR*, 2008. 1

[20] J. S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Software*, 11(1):37–57, 1985. 4