Latent Regression Forest: Structured Estimation of 3D Articulated Hand Posture

Danhang Tang Hyung Jin Chang* Alykhan Tejani* Tae-Kyun Kim
Imperial College London, London, UK
{d.tang11, hj.chang, alykhan.tejani06, tk.kim}@imperial.ac.uk

Abstract

In this paper we present the Latent Regression Forest (LRF), a novel framework for real-time, 3D hand pose estimation from a single depth image. In contrast to prior forest-based methods, which take dense pixels as input, classify them independently and then estimate joint positions afterwards; our method can be considered as a structured coarse-to-fine search, starting from the centre of mass of a point cloud until locating all the skeletal joints. The searching process is guided by a learnt Latent Tree Model which reflects the hierarchical topology of the hand. Our main contributions can be summarised as follows: (i) Learning the topology of the hand in an unsupervised, data-driven manner. (ii) A new forest-based, discriminative framework for structured search in images, as well as an error regression step to avoid error accumulation. (iii) A new multi-view hand pose dataset containing 180K annotated images from 10 different subjects. Our experiments show that the LRF out-performs state-of-the-art methods in both accuracy and efficiency.

1. Introduction

Since the widespread success of real-time human body pose estimation [10], the area of hand pose estimation has received much attention within the computer vision community. Accurate and efficient hand pose estimation is beneficial to many higher level tasks such as human computer interaction, gesture understanding and augmented reality. In this paper we introduce a method for real-time 3D hand pose estimation from a single depth image.

State-of-the-art human body pose estimation techniques consist mainly of data-driven, bottom-up approaches, in which pixels are independently assigned body part labels [15] or vote for joint locations [5, 18, 13]. However, in comparison to the human body, the hand has far more complex articulations, self-occlusions and multiple viewpoints; thus, these approaches require exponentially more data to capture this variation, making their direct application difficult.

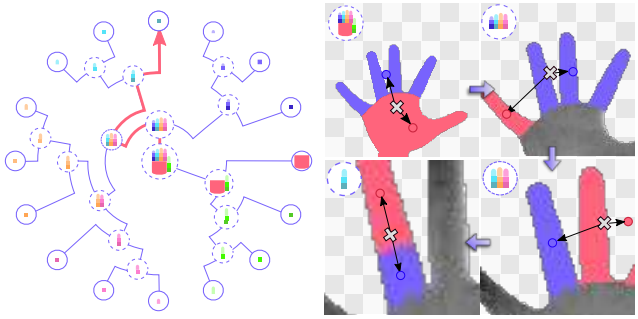


Figure 1: Our method can be viewed as a search process, guided by a binary Latent Tree Model(LTM); starting from root of the LTM, we minimise the offset to its children at each level until reaching a leaf node which corresponds to a skeletal joint position. For simplicity, we only show the searching process for one joint.(All figures best viewed in colour and high-definition)

Furthermore, few bottom-up approaches use a global refinement step, such as enforcing dependency between local outputs [16] or kinematic constraints [17]. Without such procedures, highly unlikely or even impossible poses can be produced as output.

In contrast, other approaches to hand-pose estimation have used a more top-down, global approach in which hypotheses are generated from a 3D hand model and poses are tracked by fitting the model to the test data [3, 6, 1, 12]. Whilst these model-based approaches inherently deal with the kinematic constraints, joint articulations and viewpoint changes, their performance heavily relies upon accurate pose initialisation and structural correlation between the synthetic model and testing subject (i.e. hand width and height).

In this paper, we present the *Latent Regression Forest (LRF)* for real-time 3D hand pose estimation from a single depth image. We formulate the problem as a dichotomous divide-and-conquer search for skeletal joints which

*These authors contributed equally to this work.

is conducted as a structured coarse-to-fine search, guided by a learnt topological model of the hand (see Figure 1). Furthermore, the topological model is used to enforce implicitly learnt global kinematic constraints on the output. Additionally, by training in a discriminative manner using our new diverse hand pose dataset, our approach is able to generalise to hands of various shapes and sizes as demonstrated in our experiments. Our experiments show that the LRF outperforms state-of-the-art methods in both accuracy and efficiency. The main contributions of our work can be summarised as follows:

1) Unsupervised learning of the hand topology: We represent the topology of the hand by a Latent Tree Model[2] which is learnt in an unsupervised fashion. This topological model is used while training the Latent Regression Forest to enable a structured coarse-to-fine approach.

2) Latent Regression Forest: We introduce a framework for structured coarse-to-fine search in depth images. Guided by the learnt Latent Tree Model, we learn binary decision trees that iteratively divide the input image into sub-regions until each sub-region corresponds to a single skeletal joint. Furthermore, an error regressor is embedded into each stage of the framework, in order to avoid error accumulation.

3) A new multi-view hand pose dataset: We present a new hand pose dataset containing 180K fully 3D annotated depth images from 10 different subjects.

2. Related Work

Hand pose estimation has a long and diversified history in the computer vision community. With the recent introduction of low cost real-time depth sensors, this field, as well as the closely related field of human body pose estimation, has received much attention. In this section we will discuss some of the more recent related works to these problems, however, we refer the reader to [4] for a detailed survey of earlier hand pose estimation algorithms.

Many works for human body pose estimation use large synthetic datasets in training and use either pixel-wise classification [15], or joint regression techniques [5, 18, 13, 16] for pose estimation. However, in comparison to the human body, the hand has far more complex articulations, self-occlusions and multiple viewpoints; thus, these approaches require exponentially more data to capture this variation, making their direct application difficult.

Keskin *et al*. [8] propose a solution to the data-explosion problem, by first clustering the training data followed by training multiple experts on each cluster using the method of [15]. Furthermore, due to the increased variation in the hand, capturing ground-truth annotated real data is a problem in its own right. Tang *et al*. [17] investigate semi-supervised learning for hand pose estimation using annotated synthetic data and unlabelled real data.

Recently many tracking-based methods have also been proposed for hand pose estimation. Oikonomidis *et al*. [12] introduce a tracking based method for hand pose estimation in depth images using particle swarm optimisation. De La Gorce *et al*. [3] incorporate shading and texture information into a model-based tracker, whereas Ballan *et al*. [1] use salient points on finger-tips for pose estimation. Very recently, Melax *et al*. [9] proposed a tracker based on physical simulation which achieves state-of-the art performance in real-time.

Graphical models, especially tree-based models have recently been used for estimating human body pose. Tian *et al*. [19] build a hierarchical tree models to examine spatial relationships between body parts. Whereas Wang and Li [20] use a Latent Tree Model to approximate the joint distributions of body part locations. Latent Tree Models, in particular, are interesting as they are able to represent complex relationships in the data [11] and, furthermore, recent methods for constructing these models ([2, 7]) enable us to learn consistent and minimal latent tree models in a computationally efficient manner.

3. Methodology

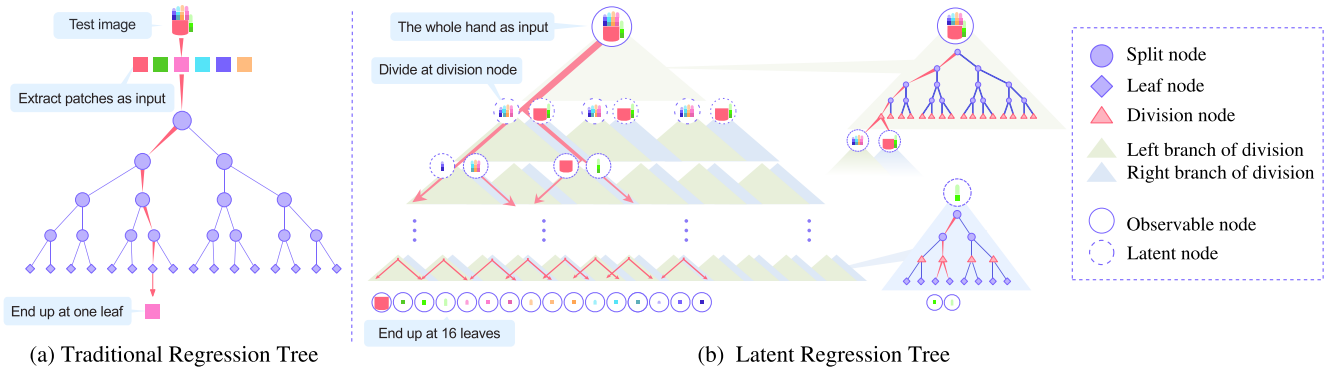
The hand pose estimation problem can be decomposed into estimating the location of a discrete set of joints on the hand skeleton model. We formulate this as a dichotomous divide-and-conquer search problem, in which the input image is recursively divided in to two cohesive sub-regions, until each sub-region contains only one skeletal joint. To attain robustness to the complex articulations of the hand, the search is carried out in a structured, coarse-to-fine manner, where the granularity of each search stage is defined by a learnt topological model of the hand.

In Section 3.1 we discuss how we can learn the hand topology in an unsupervised fashion. Following this, in Section 3.2, we discuss how this topology is used to build a Latent Regression Forest (LRF) to perform a structured, coarse-to-fine search in the image space. Finally, in Section 3.3 we discuss a strategy to reduce error propagation within the LRF.

3.1. Learning the hand topology

To guide the search process, we desire to define a coarse-to-fine, hierarchical topology of the hand, where the most coarse level of the hierarchy is defined by the input to the search, the entire hand, and the most fine level by the outputs, the skeletal joints.

Given training images annotated with the 3D positions of all skeletal joints, we additionally define the 3D position of the entire hand as the centre of mass of all points in the depth image. Thus, our objective is to learn a hierarchical topology starting from the centre of mass and ending at the skeletal joints. This is achieved by modelling the topology



(a) Traditional Regression Tree (b) Latent Regression Tree

Figure 2: An comparison between (a) a traditional Regression Tree, where each patch sample is extracted from a depth image, propagated down the tree and ends up at one leaf node; and (b) an Latent Regression Tree, where the whole point cloud is propagated down the tree and keep dividing until ending up at 16 leaf nodes.

as a *Latent Tree Model* and making use of the recently proposed Chow-Liu Neighbour-Joining (CLNJ) method [2] to construct it.

A Latent Tree Model is a tree-structured graphical model, $\mathcal{M} = (O \cup L, E)$, where the vertexes are composed of *observable vertexes* (skeletal joints and the hand position), O, and *latent vertexes*, $L = \{l\}$, where $l \subseteq O$; and E denotes edges(see Figure 3(c)(d)). In our work we only consider binary tree models as to easily integrate them within the Latent Regression Forest, which itself is composed of binary trees.

The CLNJ method takes as input a pairwise distance matrix, $\mathcal{D}$, of all observable vertexes. Using the training set, $\mathcal{S}$, the distance $\mathcal{D}$ between two observable vertexes, x and y, is defined as:

$$\mathcal{D}_{xy} = \frac{\sum_{I \in \mathcal{S}} \delta(I, x, y)}{|\mathcal{S}|} \quad (1)$$

where $\delta(I, x, y)$ is a function measuring the distance between vertexes x and y in image I. In this work we compare two different distance functions, the first being the standard Euclidean distance between the 3D positions of x and y, and the second being the geodesic distance.

To calculate the geodesic distance between vertexes x and y in image I, we first construct a fully connected, undirected graph of all observable vertexes in I. Edges in this graph are then removed if there is a large depth discontinuity along this edge in image space. What remains is a graph in which the edges all lie along a smoothly transitioning depth path. The geodesic distance between two vertexes can then be calculated as the shortest path connecting them in this graph.

In Figures 3(a) and (b) we demonstrate the difference between Euclidean and geodesic distance on a toy hand model. Through different poses the Euclidean distance between two joints can change drastically while the geodesic one remains largely unchanged; this robustness to pose is preferable to the capture the topology of human hand.

In Figures 3(c) and (d) two Latent Tree Models generated using the Euclidean and geodesic metrics respectfully are shown; as highlighted by the arrows, the Euclidean-generated model groups sub-parts of fingers with different fingers e.g. root of middle finger is grouped with index finger, whereas in the geodesic-generated model the fingers are all separated.

3.2. Latent Regression Forest

The aim of a Latent Regression Forest (LRF) is to perform a search of an input image for several sub-regions, each corresponding to a particular skeletal joint. Searching is preformed in a dichotomous divide-and-conquer fashion where each division is guided by the learnt Latent Tree Model representing the topology of the hand.

A LRF is an ensemble of randomised binary decision trees, each trained on a bootstrap sample of the original training data. Each Latent Regression Tree contains three types of nodes: *split*, *division* and *leaf* (see Fig. 2). Split nodes perform a test function on input data and decides to route them either left or right. Division nodes divide the current search objective into two disjoint objectives and propagate input data down both paths in parallel. Finally, leaf nodes are terminating nodes representing a single skeletal joint and store votes for the location of this joint in 3D space.

In Section 3.2.1 we discuss how to build the LRF followed by a discussion of the testing procedure in Section 3.2.2.

3.2.1 Training

Given a Latent Tree Model (LTM) of the hand topology, $\mathcal{M}$, for each vertex $i \in \mathcal{M}$, $i = 0...|\mathcal{M}|$, its parent is defined by $p(i)$ and its 2 children by $l(i)$ and $r(i)$. For each training

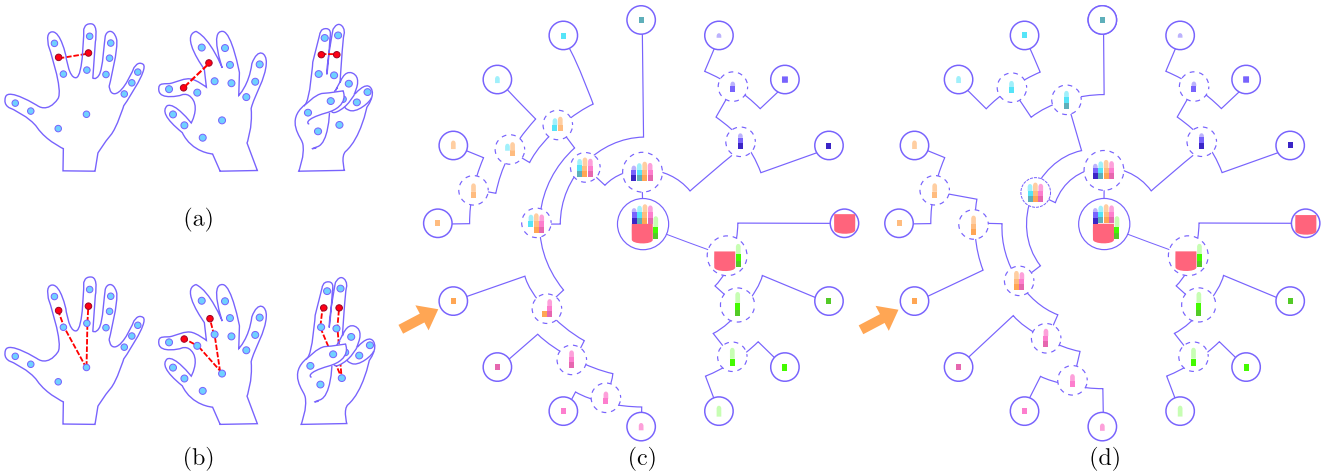


Figure 3: (a) Euclidean distance between two joints. (b) Geodesic distance between two joints. (c) LTM generated using the Euclidean distance metric. (d) LTM generated using the geodesic distance metric. In (c) and (d) solid circles represent observable vertexes and dashed ones latent vertexes. This figure is best viewed in colour.

depth image, I, a 3D position, $\boldsymbol{\rho}_i^I$, is associated with i. For each observable vertex, $i \in O$, this is simply the position of the associated joint; for each latent vertex, $i \in L$, the position is represented by the mean position of the observable nodes they are composed of. Therefore, each training sample can be represented as a tuple $\left(I, \boldsymbol{\rho}_i^I\right)$, where $\boldsymbol{\rho}_i^I$ are the 3D positions of the associated vertex, i, in the image I.

Each Latent Regression Tree (LRT) in the Latent Regression Forest is trained as follows: the LRT is trained in stages, where each stage corresponds to a non-leaf vertex in the LTM, $\mathcal{M}$. Starting with the root vertex, $i = 0$, of $\mathcal{M}$ we grow the LRT with the objective of separating the image into two cohesive sub-regions which correspond to the vertexes, $l(i)$ and $r(i)$, which are the children of the root node.

This separation is achieved by growing a few layers of Latent Regression Tree. At each node, we randomly generate splitting candidates, $\Phi = \{(f_i, \tau_i)\}$, consisting of a function, f_i, and threshold, τ_i, which splits the input data, $\mathcal{S}$, into two subsets, $\mathcal{S}^l$ & $\mathcal{S}^r$, s.t. $\mathcal{S}^l = \{I | f_i(I) < \tau_i\}$ and $\mathcal{S}^r = \mathcal{S} \setminus \mathcal{S}^l$. A function, f_i, for a splitting candidate, whilst at the stage represented by the LTM vertex i is defined as:

$$f_i(I) = d_I\left(\boldsymbol{\rho}_i^I + \frac{\mathbf{u}}{d_I(\boldsymbol{\rho}_0^I)}\right) - d_I\left(\boldsymbol{\rho}_i^I + \frac{\mathbf{v}}{d_I(\boldsymbol{\rho}_0^I)}\right), \quad (2)$$

where $d_I(\cdot)$ is the depth at an image position, $\boldsymbol{\rho}_i^I$ is the position of the LTM vertex, i, in the image, I and vectors $\mathbf{u}$ and $\mathbf{v}$ are random offsets. Similarly to [15], the offsets are normalised to make them depth-invariant. However, in order to avoid error accumulation in depth values, the normalisation factor is always the centre of mass, $\frac{1}{d_I(\boldsymbol{\rho}_0^I)}$.

The splitting candidate, ϕ_i^*, that gives the largest information gain is stored at the LRT node, which is a *split node* as in the standard Random Forest. The information gain whilst at the stage represented by the LTM vertex i is defined as:

$$IG_i(\mathcal{S}) = \sum_{m}^{l(i),r(i)} \mathrm{tr}(\Sigma_{im}^{\mathcal{S}}) - \sum_{k}^{\{l,r\}} \frac{\mathcal{S}^k}{|\mathcal{S}|} \left(\sum_{m}^{l(i),r(i)} \mathrm{tr}\left(\Sigma_{im}^{\mathcal{S}^k}\right) \right)$$

where $\Sigma_{im}^{\mathcal{X}}$ is the sample covariance matrix of the set of offset vectors $\left\{\left(\boldsymbol{\rho}_m^I - \boldsymbol{\rho}_i^I\right) | I \in \mathcal{X}\right\}$ and $\mathrm{tr}(\cdot)$ is the trace function. The offset vectors indicate the offsets from the current centre to each centre of the two subregions.

This process is then repeated recursively on each split of the data, $\mathcal{S}^l$ & $\mathcal{S}^r$, until the information gain falls below a threshold.

At this point we introduce a *division node* which divides the current search objective into two finer ones and enters the next search stage. The division node duplicates the training data and continues to grow the tree along two separate paths, each corresponding to to one of the two children of the current LTM vertex, i. Additionally, for each training image, I, reaching this division node we store the vectors $\boldsymbol{\theta}_m = \left(\boldsymbol{\rho}_m^I - \boldsymbol{\rho}_i^I\right)$ corresponding to the 3D offsets of i and its children $m \in \{l(i), r(i)\}$.

This process of split followed by division is then repeated until the LTM vertex, i, to be considered is a leaf; at which point we create a leaf node in the LRT corresponding to the skeletal joint represented by i. The leaf node stores information about the 3D offset of i from its parent $p(i)$, that being $\left(\boldsymbol{\rho}_i^I - \boldsymbol{\rho}_{p(i)}^I\right)$.

As previously mentioned, the hand has many complex articulations and self-occlusions, thus, in order to fully capture this variation the training set used is extremely large.

To retain training efficiency we make use of the fact that we train in coarse-to-fine stages based on the learnt LTM. An intuition is that coarse stages require less training data than the fine ones, therefore we can gradually add more training data at each stage of the training procedure.

For an LTM of maximum depth D, we split the training data, $\mathcal{S}$, into D equally sized random, disjoint subsets $\mathcal{S}_0, ...\mathcal{S}_{D-1}$. We start training an LRT with $\mathcal{S}_0$ for the first stage, and for each stage after we add an additional subset to the training data. That is, for stage d the training set is composed of $\mathcal{S}_d \cup \mathcal{S}_{d-1}$. The training procedure to grow a single LRT is described in Algorithm 1.

Algorithm 1 Growing a Latent Regression Tree

Input: A set of training samples $\mathcal{S}$; a pre-learned LTM $\mathcal{M} = (O \cup L, E)$ with maximum depth D.
Output: A LRT T

procedure GROW($\mathcal{S}$, M)
  Equally divide $\mathcal{S}$ into random subsets $\mathcal{S}_0, ...\mathcal{S}_D$
  Let $i \leftarrow 0$, $j \leftarrow 0$ ▷ Initialise ith node of LTM and jth node of LRT
  Let $d \leftarrow 0$ ▷ First stage of training
  SPLIT($i, j, \mathcal{S}_0, d$)

function SPLIT(i, j, $\mathcal{S}$, d)
  Randomly propose a set of split candidates Φ.
  for all $\phi \in \Phi$ **do**
    Partition $\mathcal{S}$ into $\mathcal{S}^l$ and S^r by ϕ with Eq. 2.
  Use the entropy in Eq. 3.2.1 to find the optimal $\phi*$
  if $IG_i(\mathcal{S})$ is sufficient **then**
    Save j as a *split node* into T.
    SPLIT(i, $l(j)$, $\mathcal{S}^l$, d)
    SPLIT(i, $r(j)$, $\mathcal{S}^r$, d)
  else if $i \in L$ **then**
    Save j as a *division node* into T
    Let $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_{d+1}$
    SPLIT($l(i)$, $l(j)$, $\mathcal{S}$, $d+1$)
    SPLIT($r(i)$, $r(j)$, $\mathcal{S}$, $d+1$)
  else
    Save j as a *leaf node* into T.
  Return

3.2.2 Testing

At test time, pose estimation is performed on an image I as follows; we define the starting position for the search, $\rho^I_{i=0}$ as the centre of mass of the depth image, which corresponds to the root vertex of the LTM. Starting at the root of the Latent Regression Forest, the image traverses the tree, branching left or right according to the split-node function, until reaching a division node. For each offset, $\boldsymbol{\theta}_j$ stored at the division node, 3D votes are accumulated in two Hough spaces, H^l and H^r, where the votes for H^l are defined as $\left\{\rho^I_i + \frac{\boldsymbol{\theta}_j}{\rho^I_0} | \boldsymbol{\theta}_j \in \boldsymbol{\theta}_l\right\}$ and similarly for H^r. The modes of these two Hough spaces now represent the two new positions, $\rho^I_{l(i)}$ and $\rho^I_{r(i)}$, from which the next search stage begins. This process is then repeated recursively until each path terminates at a leaf node.

This process will result in the image reaching *multiple leaf nodes*, one for each terminating node in the LTM. Using the stored offsets at the leaf nodes, each leaf node votes for its corresponding skeletal joint in a corresponding 3D Hough space. Aggregating votes of all trees, we locate the final positions of the joints by a structured search in the Hough space, for which the structure is dictated by the learnt LTM as follows. For each skeletal joint, we assign to it a dependent observable vertex in the LTM which corresponds to the vertex with the smallest geodesic distance as calculated in the matrix, $\mathcal{D}$ (Eq. 1). The location of each joint in the Hough space is then defined as the maxima which is closest to the location of its dependent vertex.

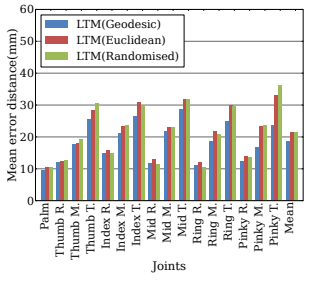
Efficiency In contrast to the state-of-the-art bottom-up approaches that take dense pixels as input [8] our algorithm takes the whole image as input. Thus, while both methods are constrained in complexity by the depth of the trees, d, i.e. $\mathcal{O}(d)$, ours has a much lower constant factor. This is because the number of pixels to be evaluated in bottom-up approaches are usually in the order of thousands for a standard VGA image; whereas, in contrast, we only evaluate one sample per image.

3.3. Cascaded Error Regressor

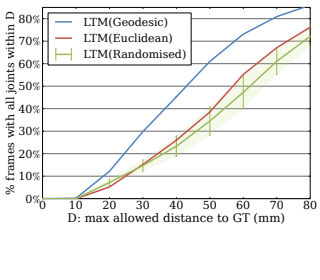
As explained in previous sections, a multi-stage coarse-to-fine structured search is efficient. However, an underlying risk is that the dependency between stages can lead to error accumulation throughout the search. To compensate for this, we embed an *error regressor* inspired by [14] into each stage of Latent Regression Forest. After training stage d with set $\mathcal{S}_d$ and before creating a *division node* , we use S_{d+1} to validate the trained forest so far. For each sample $s_i \in \mathcal{S}_{d+1}$, an error offset $\Delta\boldsymbol{\theta}$ between the ground truth and the estimation is measured. Similar to the previously described method of splitting, the forest is further grown for a few layers in order to minimise the variance of $\Delta\boldsymbol{\theta}$. Once the information gain falls below a threshold a *division node* is generated and the forest training enters next stage, $d+1$.

4. Experiments

Dataset In this paper, we use Intel®'s *Creative Interactive Gesture Camera* [9] as a depth sensor for capturing training and testing data. As the state-of-the-art consumer time-of-flight sensor, it captures depth images at a lower noise level than structured-light sensors making them ideal for hand pose estimation. For labelling, we utilise [9] to obtain a



(a)



(b)

Figure 5: Effect of different LTMs.(R:root, M:middle, T:tip)

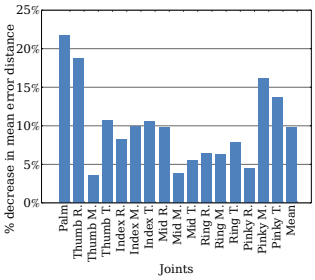


Figure 6: Error regression.

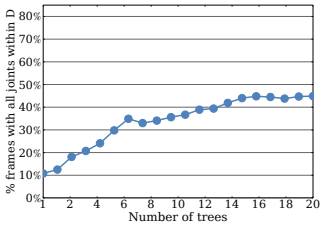
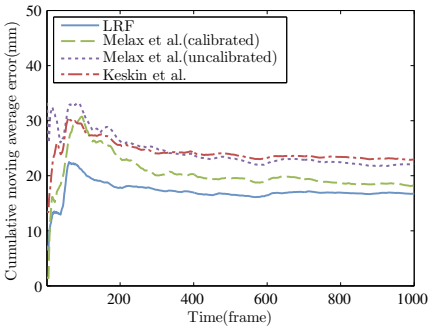
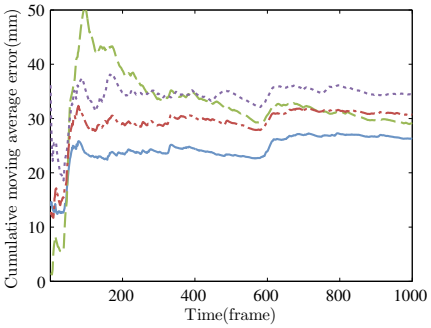


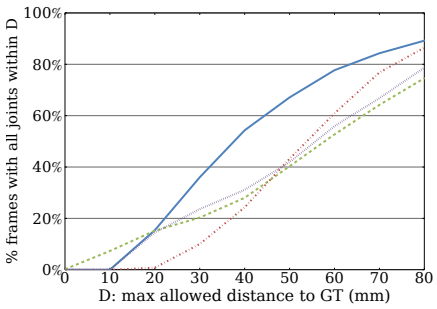
Figure 7: Number of trees.



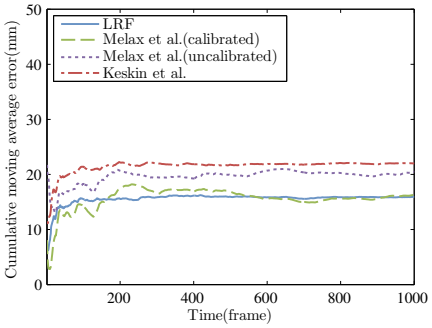
(a) Test sequence A (average error)



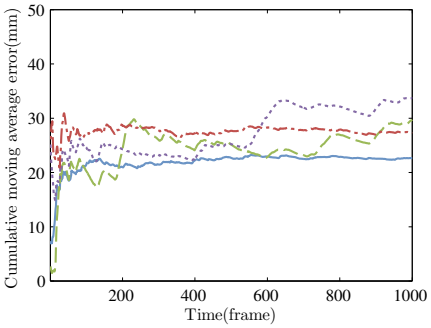
(b) Test sequence A(index tip)



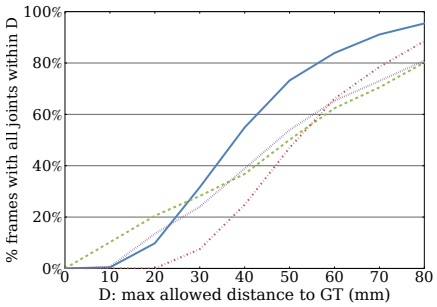
(c) Worst case accuracy [18] of sequence A



(d) Test sequence B(average error)



(e) Test sequence B(index tip)



(f) Worst case accuracy [18] of sequence B

Figure 8: Quantitative comparison against state-of-the-art methods.

preliminary pose for each frame, and then manually refine.

For training, we have collected sequences from 10 different subjects with varying hand sizes by asking each subject to make various hand poses with an illustration of 26 different postures shown as aid. Each sequence was then samples at 3fps producing a total of 20K images and by additionally applying in-plane rotations to this set, the final dataset contains 180K ground truth annotated training images. For testing, we have collected two sequences (denoted sequence A and B) each containing 1000 frames capturing a vast array of different poses with severe scale and viewpoint changes. Furthermore, as [9] is tracking based and requires initialisation (frontal view of an open hand), in order to do a fair comparison both test sequences start in this way.

In all experiments we train each Latent Regression Tree by evaluating 2000 splitting candidates at each node and the threshold used to stop growing the tree at a particular stage is chosen based on the size of a finger joint, which was set to $(10mm)^2$.

In Section 4.1 we conduct a self comparison of the different components in the Latent Regression Forest. Following this, In Section 4.2 we do a thorough evaluation against other state-of-the-art methods. Finally, in Figure 9 we present some qualitative results.

4.1. Self Comparisons

To evaluate the impact of different distance metrics used when constructing the LTM we quantitatively measure the impact of the different topologies on performance. We compare LTMs generated using the Euclidean and geodesic distance as well as 5 randomly generated LTMs. For each of these 7 topologies, an Latent Regression Forest is trained on a subset of the training data and evaluated on sequence A.

Figure 5(a) shows the standard evaluation metric of mean error, in mm, for each joint across the sequence. As shown, the Euclidean-generated LTM performs slightly better than the random ones, whereas the geodesic-generated LTM achieves the best performance on all joints except for two. In addition to this, we also employ more challenging metric, the proportion of test images that have *all* predicted joints within a certain maximum distance from the ground truth, which was recently proposed in [18]. The results using this metric can be seen in Figure 5(b). As shown, the Euclidean-generated LTM achieves the same performance as the upper-bound of performance from the random LTMs, whereas the geodesic-generated LTM significantly outperforms all of them showing a 20% improvement at a threshold of $40mm$.

Additionally, we evaluate the impact of the cascaded error regressor. In Figure 6 we show the decrease in mean error distance for each joint across the whole sequence. As can be seen, we achieve up to a 22% reduction in mean error for one joint and and improvement of 10% on average.

In principle, since each tree generates much less votes comparing to traditional regression tree, more trees are needed in order to produce robust results. Figure 7 shows the accuracy impact from different number of trees. A reasonable choice considering the trade-off between accuracy and efficiency is 16 trees, which is the setting we use in all experiments.

4.2. Comparison with State-of-the-arts

We compare a 16-tree Latent Regression Forest with two state-of-the-art methods. The first is a regression version of Keskin *et al*. [8], for which we use our own implementation using the training parameters as described in [8]. The second method we compare to is the model-based tracker of Melax *et al*. [9], for which we use a compiled binary version provided by the authors. As this method is model based it requires calibration of the hand structure (width and height). Therefore, in order to do a fair comparison we compare to two versions of this method, one which has been calibrated and one which has not.

In Figures 8 (a) and (d) we show the cumulative moving average of the mean joint error. As can be seen, our approach maintains a low average error throughout both sequences, and as expected the tracking based approaches reduce in error over time. In Figures 8 (b) and (e) we show the cumulative moving average of the index fingertip error, a relatively unstable joint. Notice, that after approximately the 500th frame the tracking based methods continuously decrease in accuracy for this joint, indicating the tracking has failed and could not recover. This further highlights the benefit of using frame-based approaches.

Additionally, in Figures 8 (c) and (f), we compare all methods using the more challenging metric proposed in [18]. As can be seen our method largely outperforms the other state-of-the-arts. Furthermore, our method runs in real-time at 62.5fps which is comparable to [9] (60 fps) and much faster than [8] (8.6fps). Note that here both our method and [8] are unoptimised—single threaded, without any CPU/GPU parallelism.

5. Conclusion & Future Work

In this paper we presented the Latent Regression Forest, a method for real-time estimation of 3D articulated hand pose. We formulated the problem as a structured coarse-to-fine search for skeletal joints, in which we learnt the granularity of each search stage using a Latent Tree Model. Furthermore, compared to other forest-based methods that take dense pixels as input, our method is applied on the whole image as opposed to individual pixels, greatly increasing the run-time speed. To the best of our knowledge this is the first work combining Latent Tree Models and Random Forests, allowing us to apply the Latent Regression Forest to many existing applications of the Latent Tree Model. As future work, we plan to investigate the application of the Latent Regression Forest to many other structured problems, either spatially or temporally.

Acknowledgement
This project was supported by the Samsung Advanced Institute of Technology(SAIT).

References

[1] L. Ballan, A. Taneja, J. Gall, L. V. Gool, and M. Pollefeys. Motion capture of hands in action using discriminative salient points. In *ECCV*, 2012.

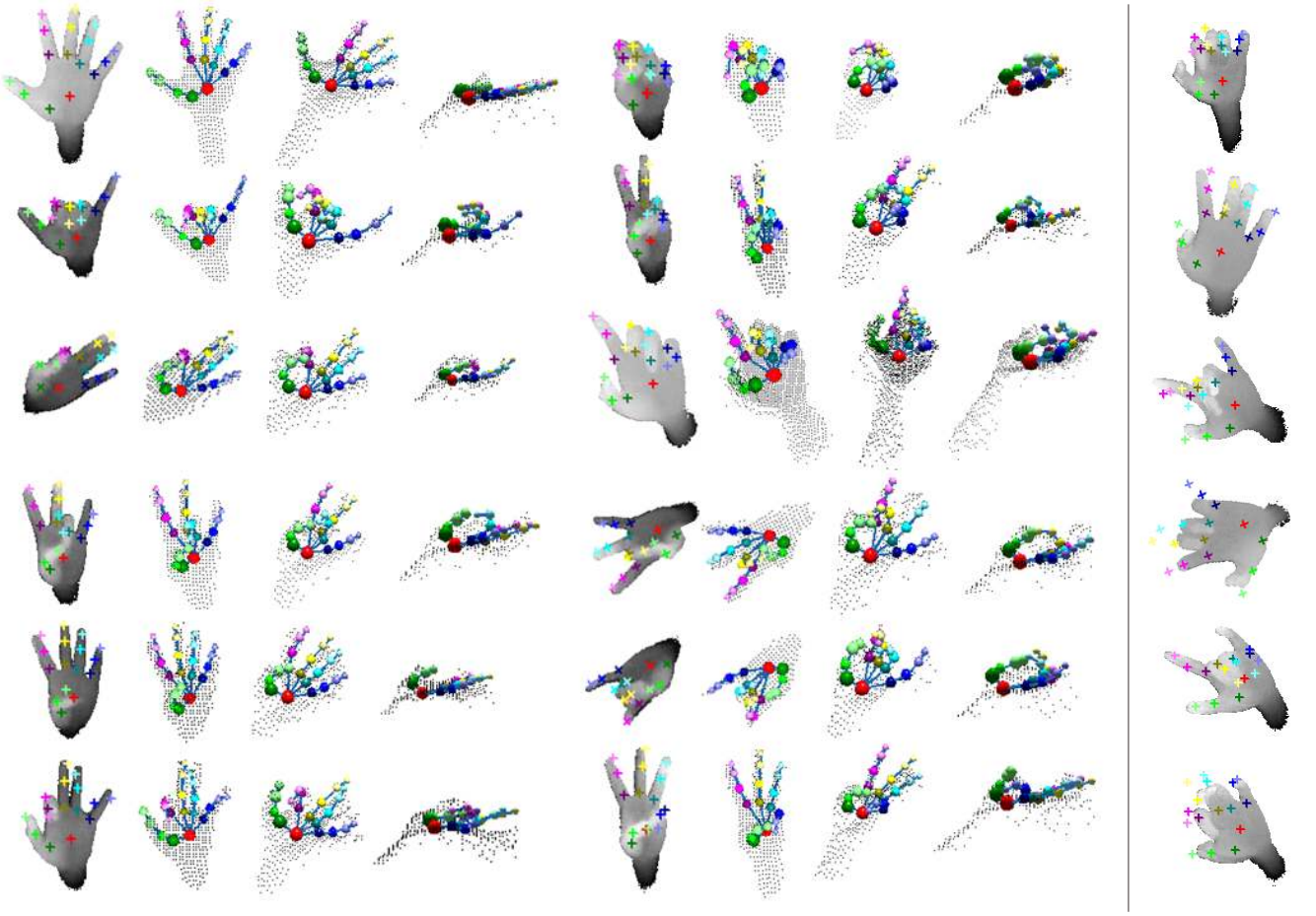
[2] M. J. Choi, V. Y. F. Tan, A. Anandkumar, and A. S. Willsky. Learning latent tree graphical models. *JMLR*, 12:1771–1812, 2011.

[3] M. de La Gorce, D. Fleet, and N. Paragios. Model-based 3d hand pose estimation from monocular video. *TPAMI*, 33(9):1793–1805, 2011.

[4] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly. Vision-based hand pose estimation: A review. *CVIU*, 108(1):52–73, 2007.

[5] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon. Efficient regression of general-activity human poses from depth images. *ICCV*, 2011.

[6] H. Hamer, K. Schindler, E. Koller-Meier, and L. V. Gool. Tracking a hand manipulating an object. In *ICCV*, 2009.



(a) Success case (b) Failure cases

Figure 9: In (a) we show success cases; we show the localisation on the depth image followed by a visualisation of the estimated 3D joint locations from multiple angles. In (b) we show some failure cases, note however that the structure of the output is still in line with the hand topology. This image is best viewed in colour.

[7] S. Harmeling and C. K. I. Williams. Greedy learning of binary latent trees. *TPAMI*, 33(6):1087–1097, 2011.

[8] C. Keskin, F. Kıraç, Y. E. Kara, and L. Akarun. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *ECCV*, 2012.

[9] S. Melax, L. Keselman, and S. Orsten. Dynamics based 3d skeletal hand tracking. In *Interactive 3D Graphics and Games*, 2013.

[10] Microsoft Corp. Redmond WA. Kinect for xbox 360.

[11] R. Mourad, C. Sinoquet, N. L. Zhang, T. Liu, and P. Leray. A survey on latent tree models and applications. *JAIR*, 47:157–203, 2013.

[12] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *ICCV*, 2011.

[13] G. Pons-Moll, J. Taylor, J. Shotton, A. Hertzmann, and A. Fitzgibbon. Metric regression forests for human pose estimation. In *BMVC*, 2013.

[14] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line random forests. In *ICCV Workshops*, 2009.

[15] 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 single depth images. In *CVPR*, 2011.

[16] M. Sun, P. Kohli, and J. Shotton. Conditional regression forests for human pose estimation. In *CVPR*, 2012.

[17] D. Tang, T.-H. Yu, and T.-K. Kim. Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *ICCV*, 2013.

[18] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon. The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *CVPR*, 2012.

[19] Y. Tian, C. Zitnick, and S. Narasimhan. Exploring the spatial hierarchy of mixture models for human pose estimation. In *ECCV*, 2012.

[20] F. Wang and Y. Li. Beyond physical connections: Tree models in human pose estimation. In *CVPR*, 2013.