Entropy Driven Hierarchical Search for 3D Human Pose Estimation

Ben Daubney
B.Daubney@Swansea.ac.uk
Xianghua Xie
X.Xie@Swansea.ac.uk

Department of Computer Science
Swansea University
UK, SA2 8PP

3D Human pose estimation from a single monocular image is an extremely difficult problem. Currently there are two main approaches to solving this problem, the first is to learn a direct mapping from image features to 3D pose [1], the second is to first extract 2D pose as an intermediate stage and then 'lift' this to a 3D pose [2]. The limitation with both of these approaches is that they are only applicable to poses that are similar to those represented in the original training set, e.g. walking. It is unlikely they will scale to extract arbitrary 3D poses. Contrary to this, in the domain of 2D pose estimation current state-of-the-art methods have been shown capable of detecting poses that are much more varied [3]. This has been achieved using generative models built around the Pictorial Structures representation that decomposes pose estimation into a search across individual parts [4].

In this paper we present a generative method to extract 3D pose from single images using a part based representation. The method is stochastic, though in contrast to methods used for 3D tracking (e.g. the particle filter), where the search space in each frame is tightly constrained by previous observations, in single image pose estimation the search space is much larger. To permit a search over this space a generative prior model is learnt from motion capture data. Stochastic samples are used to approximate this prior and to facilitate its update. In effect, the initial prior is iteratively deformed to the posterior distribution.

The body is represented by a set of ten parts, each part has a fixed length and connected parts are forced to join at fixed locations. The conditional distribution between two connected parts is modeled by first learning a joint distribution using a GMM $p(\mathbf{x}_i, \mathbf{x}_j|\theta_{ij})$, where $\mathbf{x}_i$ and $\mathbf{x}_j$ is the state of the ith and jth part respectively and θ_{ij} is the set of model parameters. As each model is represented using a GMM the model parameters are defined as $\theta_{ij} = \{\lambda_{ij}^k, \mu_{ij}^k, \Sigma_{ij}^k\}_{k=1}^K$, where K is the number of components in the model and $\lambda_{ij}^k, \mu_{ij}^k, \Sigma_{ij}^k$ represent the kth component's weight, mean and covariance respectively. For efficiency all covariances used to represent limb conditionals are diagonal and can be partitioned such that $\Sigma_{ij}^k = diag(\Lambda_{ii}^k, \Lambda_{jj}^k)$ and likewise $\mu_{ij}^k = \left(\mu_i^k, \mu_j^k\right)$.

Given a value for $\mathbf{x}_j$ (e.g. a sample) the conditional distribution $p(\mathbf{x}_i|\mathbf{x}_j, \theta_{ij}^k)$ is first calculated from the joint distribution $p(\mathbf{x}_i, \mathbf{x}_j|\theta_{ij})$, following which a sample $\mathbf{x}_i$ can be drawn from it. The conditional distribution, $p(\mathbf{x}_i|\mathbf{x}_j, \theta_{ij}^k)$, is also a GMM with parameters $\{\lambda_i^k, \mu_i^k, \Lambda_{ii}^k\}_{k=1}^K$. The component weights are proportional to the marginal distribution $\lambda_i^k \propto p(\mathbf{x}_j|\theta_{ij}^k)$, which is calculated from the normal distribution $p(\mathbf{x}_j|\theta_{ij}^k) = \lambda_{ij}^k \mathcal{N}(\mathbf{x}_j; \mu_j^k, \Lambda_{jj}^k)$. Note this conditional model is different to typical approximations used, when the conditional model is approximated by $p(\mathbf{x}_{ij}|\theta_{ij})$, where $\mathbf{x}_{ij}$ is the value of $\mathbf{x}_i$ in the ***local frame*** of reference of $\mathbf{x}_j$ [3]. A benefit of learning a full conditional model between neighboring parts is that different GMM components learnt in quaternion space correspond to different spatial locations in $\mathbb{R}^3$. This is illustrated in Fig. 1 where it can clearly be seen that this representation can clearly capture multiple modes.

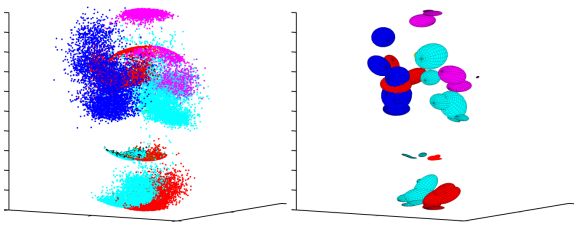

Figure 1: A visualization of the conditional distributions $p(\mathbf{x}_i|\mathbf{x}_j, \theta_{ij}^k)$ for each part. (left) individual samples for each part. (right) fitting a covariance to the samples generated from each covariance.

Given a fixed set of particles, T, a sampling scheme is employed to distribute the particles across the different parts of the model. Intuitively parts that have greater uncertainty require more particles since a larger space must be searched. A measure of the uncertainty is provided by the Entropy of the distribution. Whilst there is not an exact analytical expression to calculate the Entropy of a GMM, an approximation can be used to express the GMM as a single Gaussian. The Entropy can then be calculated as $h(\theta_{ij}) = \ln\sqrt{(2\pi e)^d|\hat{\Lambda}_{ii}|}$, where d is the dimension of $\mathbf{x}_i$ and $\hat{\Lambda}_{ii}$ is the covariance used to approximate the GMM. The number of samples assigned to each part N_i is then given by $N_i \propto e^{h(\theta_{ij})}$, which is equivalent to distributing the particles to each part proportional to the hypervolume encompassed by the covariance.

It is assumed the root node (Torso) is fixed. Samples for each part are then grown out from this using importance resampling. The weights for each sample are provided using local appearance cues. Two image cues are used, a crude foreground segmentation and edge gradients using HOG features. Once samples have been generated for all parts, information is back propagated towards the root node using message passing. Following this each GMM component is updated using the set of weighted samples generated by it. A problem with this approach is that if the samples poorly represent the distribution they are approximating the updated model is likely to be poor. This could happen if too few samples are used and over fitting could occur. To prevent this the KL divergence between the sample set and the distribution they are approximating is calculated. This can then be used to estimate a weighted update between the updated and the original prior distribution.

The joint distribution for each part was learnt using the Train partition of the HumanEva dataset using data taken across all 3 subjects and all 5 different actions (Box, Gesture, ThrowCatch, Walk, Jog) contained in the set. Hence the solution space for correct pose is not constrained to a single action. For quantitative evaluation a test set of 500 examples was created from the Validation partition of the HumanEva dataset. The root node and orientation was set using the pelvis marker data and the scale was set as the maximum distance between the head and the feet using the ground truth provided. Typical pose estimation results showing particle convergence are given in Fig. 2.

To estimate pose when the root node location is unknown we propose to apply the presented method at multiple positions in the image. In a current Matlab implementation 15,000 different root node locations can be searched in ≈ 2 mins per iteration.

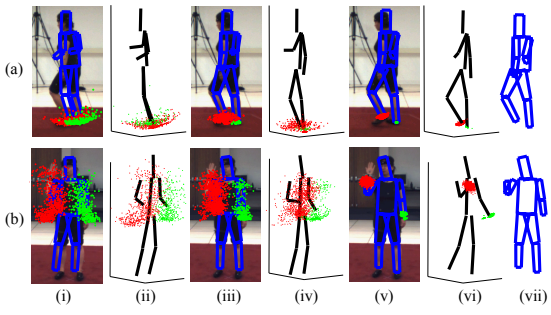

Figure 2: Example frames after iterations 1,4 and 10 showing the expected pose after each iteration in 3D and projected onto the image. Red samples show samples for the right foot (a) and right wrist (b), blue points depict the samples for the opposing parts.

[1] Ankur Agarwal and Bill Triggs. Recovering 3d human pose from monocular images. *PAMI*, 28(1):44–58, 2006.

[2] M. Andriluka, S. Roth, and B. Schiele. Monocular 3d pose estimation and tracking by detection. In *CVPR*, 2010.

[3] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61:55–79, 2005.

[4] M. A. Fischler and R. A. Elsclager. The representation and matching of pictorial structures. In *IEEE Transactions on Computer, 22(1)*, pages 67–92, 1973.