Structural SVM for Visual Localization and Continuous State Estimation

Catalin Ionescu
University of Bonn
catalin.ionescu@ins.uni-bonn.de

Liefeng Bo
TTI-Chicago
blf0218@tti-c.org

Cristian Sminchisescu
University of Bonn
sminchisescu.ins.uni-bonn.de

Abstract

We present an integrated model for visual object localization and continuous state estimation in a discriminative structured prediction framework. While existing discriminative 'prediction through time' methods have showed remarkable versatility for visual reconstruction and tracking problems, they tend to assume that the input is known (or the object is segmented) a condition that can rarely be accommodated in images of real scenes. Our structural Support Vector Machine (structSVM) framework offers an end-to-end training and inference framework that overcomes these limitations by consistently searching both in the space of possible inputs (effectively an efficient form of object localization) and in the space of possible structured outputs, given those inputs. We demonstrate the potential of this methodology for 3d human pose reconstruction in monocular images both in the HumanEva benchmark, where 3d ground truth is available, and qualitatively, in un-instrumented images of real scenes.[1]

1. Introduction

We study the integrated problem of localization and continuous state estimation for structured vision problems, using discriminative learning methods. A variety of visual scene understanding scenarios require both the localization of scene elements, objects or people in images, *e.g*. identifying their bounding box, and the reconstruction of their internal representation/state with respect to the task (orientation, human joint angles, object part locations in 2d or 3d, *etc*.) from images or video.[2] It is well understood that both inputs and outputs are structured and exhibit strong internal correlations, see also fig. 3. Originally, the problem has been approached using generative methods based on Kalman filters, for Gaussian models [3], or particle filters, in the non-Gaussian case [16]. In principle, generative frameworks provide dynamic constraints and support a unified treatement of detection and state estimation: marginal distributions can be obtained for both the image location of an object and for its other state components.[3] While this made it possible to achieve promising results in problems with state spaces of moderate dimension, the setting faces important computational challenges. Sampling in high-dimensions remains problematic, particularly at run-time, although good methods exist [10, 7, 28]. In addition, current models often rely on naïve Bayes assumptions in order to achieve tractability. This tends to produce unrealistically spiky likelihoods that complicate inference, and are difficult to mold into a state-space model that reflects the distribution of correct object locations in images.

The difficulties encountered with generative models motivated a flux of research into discriminative techniques. In this case, continuous state estimation is formulated as learning mappings from image features to observations based on training sets of object state instances and their images. Methods range from nearest neighbor or regression, to mixture of experts [24, 1, 23, 27, 26]. Temporal constraints can also be modeled. Together these have produced density propagation algorithms, effectively 'predictors through time'[27, 1] that have overcome some of the limitations of previous methods in terms of generality, ease of construction, speed, and the ability to automatically initialize and track complex motions.

However, a weak spot of existing discriminative time-series frameworks is their very origin in functional approximation from data. While regressive methods, no matter how complex, assume the input is known at test-time (run-time), this is rarely the case for visual detection and tracking problems, where one needs to search not only over possible outputs but also rank/select/classify in the large space of possible inputs—a problem that is often separately addressed as object detection or localization. This led to methods that dominantly worked with clean, segmented objects, or com-

[1] All authors contributed equally to this research.

[2] The separation into detection/localization and state estimation is used for specificity and in order to put existing work in perspective. Naturally, the location of an object is best viewed as a component of its state. The very product of this paper is a uniform discriminative model.

[3] The methods offer a joint distribution on all state variables. This decomposes as two marginals, one on variables controlling image location, the other on the remaining (*e.g*. 2d or 3d) state dimensions.

bined a separately trained detector and a predictor [27, 26]. Here we search for an alternative, integrated end-to-end learning and inference framework. We exploit large-margin ideas and tensor product kernels for structured data in conjunction with cutting plane algorithms, in order to incorporate object detection and continuous prediction-through-time in a single discriminative model. We report encouraging results for human detection and 3d pose estimation in monocular images, including both evaluation in the HumanEva benchmark and in images of complex real-world scenes.

1.1. Related Work

This research links to structured prediction methods and their applications for tracking and human pose estimation While the literature is too large to fully review, it is worth pointing out that integrated methods for detection and continuous state estimation have not yet been proposed for visual tracking in a discriminative setting. Predictive methods [1, 23, 27, 26] have shown good potential for automatic scene understanding but they often assume that the outputs are independent and the input is given or easy to obtain, *e.g*. features are extracted inside a bounding box obtained from background subtraction. When visually analyzing real-world scenes, this requirement is problematic. This led researchers into studying the joint use of state predictors and separately trained (sliding window) detectors or object localizers [27, 26, 11].[4]

Structured data can be modeled either by including sophisticated constraints into regression methods (linear or non-linear manifold assumptions [9, 27, 18, 8]), or by designing new cost functions. There is a choice of modeling correlations as part of the loss function, or as a form of regularization.

Another approach to structured prediction relies on max-margin formulations in conjunction with kernels defined over multivariate input and output spaces. Structural support vector machines, initially introduced for discrete state spaces [29], can be generalized to continuous outputs by learning a scoring function so that the pair corresponding to the given input-output training example ranks higher than a pair formed by the given input and any other output [32, 21]. Structural SVMs [4] have been applied to localization, in order to predict the discrete coordinates of the bounding box of an object (a subroutine within an object detector) by defining a special histogram kernel over image regions contained within the bounding box. Our work can be viewed as a generalization of [29, 32] to integrated localization and state estimation problems with continuous structured inputs and outputs. The extension to continuous high-dimensional state spaces is non-trivial in terms of both models/kernels and optimization, and we will show that structural SVM can work even in cases where exact inference is intractable, see also [13]. According to that taxonomy [13] our inference method falls within the class of greedy undergenerating methods (this stands in contrast to overgenerating methods, *e.g*. relaxations), but also notice that discrete experiments in [13] find the performance of both types of methods similar in practice (naturally both the models and the benchmarks in [13] are different from ours). A significant amount of research focuses on structured prediction in a probabilistic, maximum likelihood setting, *e.g*. dependency networks, CRF [14, 19]. Such methods can be effective but do not immediately lead to efficient algorithms for the *continuous* case—in particular training requires the calculation of high-dimensional integrals. It also remains to be seen to what extent normalization is desirable—in particular the presence of multiple objects in an image would automatically downgrade all peaks, no matter how strong the object signal, making it difficult to reliably produce a set of discrete answers in a variety of imaging conditions.

[4] While efficient 2d tree-based localization methods exist [22, 12], they are applicable, typically, to discrete state spaces of moderate dimension.

2. Joint Localization and State Estimation

In supervised learning, we are given a set of the input-output pairs $\{\mathbf{r}_i, \mathbf{z}_i\}_{i=1}^N$, where N is the size of training set and $\mathbf{z} \in Z$ are interdependent outputs. We aim to learn a function that best represents the relationship between inputs and outputs. In structural learning, the discriminative function is a linear combination of joint features

$$g(\mathbf{r}) = \operatorname{argmin}_{\mathbf{z}} f_{\mathbf{w}}(\mathbf{r}, \mathbf{z}) = \mathbf{w}^\top \Psi(\mathbf{r}, \mathbf{z}) \tag{1}$$

where $\mathbf{w}$ is a parameter vector and $\Psi(\mathbf{r}, \mathbf{z})$ is a feature vector induced by a joint kernel $K(\mathbf{r}, \mathbf{z}, \mathbf{r}', \mathbf{z}') = \Psi(\mathbf{r}, \mathbf{z})^\top \Psi(\mathbf{r}', \mathbf{z}')$. The specific form of joint features $\Psi(\mathbf{r}, \mathbf{z})$ is problem-dependent, an aspect discussed in detail in §2.1. The scoring function $f_{\mathbf{w}}(\mathbf{r}, \mathbf{z})$ can be interpreted as a compatibility that measures how well the output $\mathbf{z}$ matches the input $\mathbf{r}$.

To learn the discriminative function, $f_{\mathbf{w}}(\mathbf{r}, \mathbf{z})$, the structural SVM (structSVM) maximizes the generalized maximum margin loss:

$$\min_{\mathbf{w}, \xi} \frac{1}{2}\mathbf{w}^\top \mathbf{w} + C\xi \tag{2}$$

$$s.t. \quad \forall(\overline{\mathbf{z}}_1, \ldots, \overline{\mathbf{z}}_N) \in Z^N$$

$$\frac{1}{N}\mathbf{w}^\top \sum_{j=1}^{N} [\Psi(\mathbf{r}_j, \mathbf{z}_j) - \Psi(\mathbf{r}_j, \overline{\mathbf{z}}_j)] \geq \frac{1}{N}\sum_{j=1}^{N} \Delta(\mathbf{z}_j, \overline{\mathbf{z}}_j) - \xi$$

where $\Delta(\mathbf{z}_j, \overline{\mathbf{z}}_j)$ is a loss function that should decrease as the output $\overline{\mathbf{z}}_j$ approaches the ground truth $\mathbf{z}_j$. We use the so-called '*1*-slack formulation', which is equivalent to the '*n*-slack' analogue, but is more efficient in conjunction with cutting plane algorithms (we use) due to a significantly smaller dual problem [17].

Under infinitely many constraints, standard duality does not apply. However, for any small δ we can assume a finite δ-cover of our data domain, where the constraints are locally uniform. This allows to recast the problem into one with finite (yet large) number of constraints. In this case, the primal/dual theory implies that the parameter $\mathbf{w}$ has the form:

$$\mathbf{w} = \frac{1}{N} \sum_{\overline{\mathbf{Z}} \in Z^N} \alpha_{\overline{\mathbf{Z}}} \sum_{j=1}^{N} [\Phi(\mathbf{r}_j, \mathbf{z}_j) - \Phi(\mathbf{r}_j, \overline{\mathbf{z}}_j)] \quad (3)$$

where $\overline{\mathbf{Z}} = (\overline{\mathbf{z}}_\mathbf{1}, \ldots, \overline{\mathbf{z}}_\mathbf{N})$.

2.1. Joint Kernel for Location and State Estimation

For joint localization and state (pose) estimation, the input is an image, $\mathbf{r}$, and the output is the bounding box of the object *together* with its corresponding continuous state (*e.g*. 2d or 3d pose): $\mathbf{z} = (\mathbf{y}, \mathbf{x})$. We use $\mathbf{r}|\mathbf{y}$ to denote the feature vector of image regions restricted within the bounding box instantiated by $\mathbf{y}$. Here, we consider a joint kernel where the combined feature vector can be written as a tensor product over the two corresponding subspaces

$$\Psi(\mathbf{r}, \mathbf{z}) = \phi(\mathbf{r}|\mathbf{y}) \otimes \varphi(\mathbf{x}) \quad (4)$$

where $\otimes$ denotes the tensor product, $\phi(\mathbf{r}|\mathbf{y})$ is the feature induced by the kernel $K_{r|y}(\mathbf{r}|\mathbf{y}, \mathbf{r}'|\mathbf{y}')$ defined over the image region and $\varphi(\mathbf{x})$ is the feature vector induced by the state/pose kernel $K_x(\mathbf{x}, \mathbf{x}')$. For the tensor product feature vector, the joint location and state kernel is chosen to have the following form [29]:

$$\begin{aligned} K(\mathbf{r}, \mathbf{z}, \mathbf{r}', \mathbf{z}') &= \Psi(\mathbf{r}, \mathbf{z})^\top \Psi(\mathbf{r}', \mathbf{z}') = \\ &= K_{r|y}(\mathbf{r}|\mathbf{y}, \mathbf{r}'|\mathbf{y}') K_x(\mathbf{x}, \mathbf{x}') \end{aligned} \quad (5)$$

Eq. (5) implies that the joint kernel is a product of components computed over image regions within the bounding box and the corresponding state, respectively. This tensor product feature is rather general and can handle many types of structured outputs, including multiclass and sequential constraints. In vision, kernels are used to compare statistics or image features, *e.g*. as inner products of histograms defined over regions. This includes for example, bag-of-feature models, regular grids like HOG, as well as spatial pyramids based on weighted combinations of histogram intersection kernels computed at multiple levels of image encoding. An attractive feature of histogram kernels is that partially overlapping image regions share underlying statistics.

On the other hand, since we work with continuous states, $\varphi(\mathbf{x})$ is a feature vector induced by the kernels defined over continuous variables. Concurrently, we wish the kernel function to be normalized to prevent the slack variable ξ from diverging to infinity for some outputs. One possible (but by no means the only) choice satisfying these desiderata is the Gaussian kernel. This is defined over continuous variables and its 2-norm $\|\varphi(\mathbf{x})\|_2 = \sqrt{K_x(\mathbf{x}, \mathbf{x})} = 1$. Thus, for most of our experiments, the state/pose kernel has the form: $K_x(\mathbf{x}, \mathbf{x}') = \exp(-\gamma_x \|\mathbf{x} - \mathbf{x}'\|^2)$.

When designing a similarity measure between two different input-output features, intuitively, we wish: 1) input-output pairs with distant inputs should be dissimilar; and 2) input-output pairs whose inputs are nearby but outputs are distant should also be dissimilar; otherwise stated *only* the input-output pairs with both similar inputs *and* similar outputs should be similar. The joint kernel we use satisfies the above conditions because it is the product of two kernels defined over inputs and outputs, respectively; hence its value is small if any one of the two kernel component values is small (dissimilar). In this respect, the joint kernel is stronger than the classical kernel (only defined over inputs), where the input-output pairs with similar inputs but dissimilar outputs have negative impact and can pull the estimate in contradictory directions. For localization and state estimation, the advantage of a joint kernel over separable ones is that given a test image, the training data with dissimilar states/poses from the test input will have reduced impact on the bounding box search and estimate.

To summarize, the combined kernel handles the output jointly, so dependencies among output variables can be accounted for (in contrast, standard regression methods predict output components independently, which can produce suboptimal models, but see also [5]). This explains why continuous structSVM achieves better performance than support vector regression (SVR) and related methods in our experiments. In addition, the model also includes search/ranking in the space of possible object locations in the image, described next.

2.2. Output Loss Function

The output loss $\Delta(\mathbf{z}, \mathbf{z}')$ should reflect how well $\mathbf{z}$ approaches the ground truth output $\mathbf{z}'$. Within our joint tensor product kernel formulation, the loss function definition should be compatible with both the image and the state/pose kernels. For the image kernel, we adapt the score used in the PASCAL visual object challenge [4]:

$$\Delta_y(\mathbf{y}, \mathbf{y}') = 1 - \frac{\mathrm{Area}(\mathbf{y} \cap \mathbf{y}')}{\mathrm{Area}(\mathbf{y} \cup \mathbf{y}')} \quad (6)$$

where the quality of object localization is based on the amount of area overlap, where $\mathrm{Area}(\mathbf{y} \cap \mathbf{y}')$ is the intersection of the two bounding boxes $\mathbf{y}$ and $\mathbf{y}'$, and $\mathrm{Area}(\mathbf{y} \cup \mathbf{y}')$ is their union.

For state/pose estimation, it is natural to consider the loss function as a square distance in the reproducing kernel Hilbert space induced by the kernel function $K_x(\mathbf{x}, \mathbf{x}')$

$$\begin{aligned} \Delta_x(\mathbf{x}, \mathbf{x}') &= \|\varphi(\mathbf{x}) - \varphi(\mathbf{x}')\|^2 = \\ &= K_x(\mathbf{x}, \mathbf{x}) + K_x(\mathbf{x}', \mathbf{x}') - 2K_x(\mathbf{x}, \mathbf{x}') \end{aligned} \quad (7)$$

This implies that if a state is far from the ground truth, it has high loss, otherwise small loss. We define the joint output

loss as the weighted sum of losses for object localization and state estimation

$$\Delta(\mathbf{z},\mathbf{z}') = \gamma\Delta_y(\mathbf{y},\mathbf{y}') + (1-\gamma)\Delta_x(\mathbf{x},\mathbf{x}') \quad (8)$$

with $0 \leq \gamma \leq 1$ balancing the two terms.

2.3. Cutting Plane Algorithm

When training the model with continuous outputs, the number of constraints is infinite, and it is infeasible to solve the optimization (2) for all the constraints. Fortunately, the maximum margin loss has a sparsity-promoting effect, with most constraints inactive in the final solution. The cutting plane algorithm creates a nested sequence of successively tighter relaxations of the original optimization problem and finds a small set of active constraints that ensures a sufficiently accurate solution—a practical training method (see our fig. 1a). The algorithm starts with an empty working set $S = \varnothing$ of constraints. At each iteration, it finds the most violated constraint for the i-th training input

$$\overline{\mathbf{z}}_i = \text{argmax}_{\mathbf{z}\in Z}\{\Delta(\mathbf{z}_i,\mathbf{z}) + \quad (9)$$
$$+ \frac{1}{N}\sum_{\overline{\mathbf{Z}}\in S}\alpha_{\overline{\mathbf{Z}}}\sum_{j=1}^{N}[K(\mathbf{r}_j,\mathbf{z}_j,\mathbf{r}_i,\mathbf{z}) - K(\overline{\mathbf{r}}_j,\overline{\mathbf{z}}_j,\mathbf{r}_i,\mathbf{z})]\}$$

If the amount of violation exceeds the current value of the slack variable ξ by more than ϵ, the potential support vector $\overline{\mathbf{Z}} = (\overline{\mathbf{z}}_1,\ldots,\overline{\mathbf{z}}_N)$ is added to the working set $S = S \cup \overline{\mathbf{Z}}$. After the working set is updated, the optimization (2) is solved in the dual with constraints $\overline{Z} \in S$

$$\min_{\alpha\geq 0}\frac{1}{2}\sum_{\overline{\mathbf{Z}}\in S}\sum_{\overline{\mathbf{Z}}'\in S}\alpha_{\overline{\mathbf{Z}}}\alpha_{\overline{\mathbf{Z}}'}H(\overline{\mathbf{Z}},\overline{\mathbf{Z}}') - \sum_{\overline{\mathbf{Z}}\in S}\Delta(\overline{\mathbf{Z}})\alpha_{\overline{\mathbf{Z}}}$$
$$s.t.\sum_{\overline{\mathbf{Z}}\in S}\alpha_{\overline{\mathbf{Z}}} = C \quad (10)$$

where

$$\Delta(\overline{\mathbf{Z}}) = \frac{1}{N}\sum_{j=1}^{N}\Delta(\mathbf{z}_j,\overline{\mathbf{z}}_j) \quad (11)$$

and

$$H(\overline{\mathbf{Z}},\overline{\mathbf{Z}}') = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}[K(\mathbf{r}_i,\mathbf{z}_i,\mathbf{r}_j,\mathbf{z}_j) - \quad (12)$$
$$- K(\mathbf{r}_i,\mathbf{z}_i,\mathbf{r}_j,\overline{\mathbf{z}}'_j) - K(\mathbf{r}_i,\overline{\mathbf{z}}_i,\mathbf{r}_j,\mathbf{z}_j) +$$
$$+ K(\mathbf{r}_i,\overline{\mathbf{z}}_i,\mathbf{r}_j,\overline{\mathbf{z}}'_j)]$$

The algorithm stops when no violation is larger than the desired precision ϵ. Notice that at the first iteration, the set of constraints S is empty, hence the second term in the optimization problem (9) vanishes—in this case finding the most violated constraint simplifies to maximizing the loss $\Delta(\mathbf{z}_i,\mathbf{z})$ with respect to the output $\mathbf{z}$. Unlike the *n*-slack formulation, the dual problem for the *1*-slack usually remains compact, as only a single constraint is added per iteration.

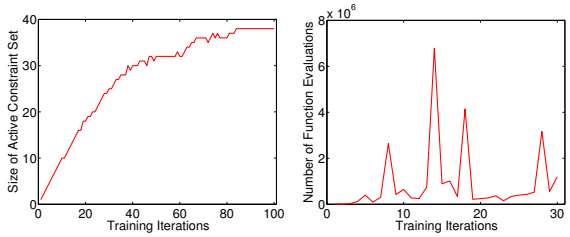

Figure 1. *(a, Left)* Size of the active constraint set (*i.e.* the number of support vectors) as function of iteration. The number of support vectors saturates at about 80 iterations, indicating convergence of the cutting plane algorithm. Notice that support vectors (points with non-zero dual variables) in the *1*-slack formulation are linear combinations of multiple examples and no longer correspond to a single training data point. *(b, Right)* Number of function evaluations for the branch-and-bound algorithm in the training stage of a joint structural SVM, based on bag-of-words SIFT (image size 640×480). Notice that the number of function evaluations is significantly higher during some of the iterations compared to the others, confirming that the hardness of search is closely linked to the structural SVM parameters, $\mathbf{w}$.

2.4. Constraint Generation

For clarity, in the sequel we rewrite the equation (9) with the bounding box and state/pose variables

$$(\overline{\mathbf{y}}_i,\overline{\mathbf{x}}_i) = \text{argmax}_{(\mathbf{y},\mathbf{x})\in(Y,X)}\{\gamma\Delta_y(\mathbf{y},\mathbf{y}') +$$
$$(1-\gamma)\Delta_x(\mathbf{x},\mathbf{x}') +$$
$$+\frac{1}{N}\sum_{(\overline{\mathbf{Y}},\overline{\mathbf{X}})\in(Y^N,X^N)}\alpha_{(\overline{\mathbf{Y}},\overline{\mathbf{X}})}\sum_{j=1}^{N}[K(\mathbf{r}_j,\mathbf{y}_j,\mathbf{x}_j,\mathbf{r}_i,\mathbf{y},\mathbf{x}) -$$
$$-K(\mathbf{r}_j,\overline{\mathbf{y}}_j,\overline{\mathbf{x}}_j,\mathbf{r}_i,\mathbf{y},\mathbf{x})]\} \quad (13)$$

In the cutting plane algorithm, the dual optimization is a constrained convex quadratic program and does not change as a function of the joint kernel. However the maximization of constraints heavily depends on its choice. Here we consider two types of optimizations: one that searches only in the space of poses, the other that searches jointly.

In the first scheme, we assume that ground truth bounding boxes are available (annotated by people or other algorithms) both in training and testing. Hence, we optimize (13) only with respect to the state/pose in the constraint generation step. This assumption is feasible in many settings, *e.g.* controlled lighting environments where good background subtraction is available. Notice that we do not limit the bounding box to a rectangular region—it can be an image region with any shape. The optimization problem (13) depends on the choice of state kernel. For instance, for the Gaussian, it is necessary to solve a nonlinear optimization problem (linear kernels induce far simpler optimizations, but are substantially less accurate). Here, we optimize (13) using a BFGS quasi-Newton method with cu-

bic polynomial search for the optimal step size. In training, we initialize using randomly chosen samples from the training set in order to diversify the support vectors. In testing, we initialize with prediction given by (unstructured) support vector regression based on models independently trained for each output dimension.

In the second scheme, we search for the bounding box and state jointly. The joint search helps because training data with similar states/poses can make a larger contribution to the location than those with dissimilar images. In particular, we maximize (13) using alternate optimization with both exact and greedy components: in one round we fix the bounding box $\mathbf{y}$ and find the pose $\mathbf{x}$ using a BFGS optimizer; in the next round, we fix the pose $\mathbf{x}$, and find the bounding box $\mathbf{y}$ using branch-and-bound or sliding window search. We use sliding windows here because although good bounds can be derived for bag-of-words models, efficiently bounding the image component of the cost is not always straightforward for features extracted on regular-grids (*e.g*. HOG or block SIFT). For testing, we initialize the localization solution (bounding box) with prediction given by a structural SVM trained for object localization. State initialization is performed as in the first scheme, using predictors independently trained for each state/pose dimension. Although branch-and-bound is guaranteed to find the global optimum, it can be rather inefficient in training for some data samples and parameter vectors $\mathbf{w}$ (these naturally change, and this modulates the search in the cutting plane algorithm). Although the empirical computational cost of branch-and-bound for object detection was found, in practice, to be $O(M^2)$ for $M \times M$ images [20], its efficiency depends on the parameter vector and can turn to $O(M^4)$ in the worst case (see also our fig. 1b).

3. Experiments

We performed several experiments at graded levels of complexity in order to test the components of the model. We illustrate the reconstruction of 3d human motion in monocular video, both in the HumanEva benchmark [25], for which 3d ground truth is available, and using images of people captured in real world environments, with complex backgrounds and uncontrolled human poses.

Features and State Representation: We consider two appearance features: block SIFT and bag-of-words SIFT, both extracted from 8×8 pixel cells and normalized by four different groups of 2×2 cells. We include both contrast sensitive and insensitive features [12], where the gradient orientations in each cell are encoded using two different quantization levels into 18 ($0^0 - 360^0$) and 9 orientation bins ($0^0 - 180^0$), respectively. This leads to a $4 \times (18+9) = 108$-dimensional feature vector. In practice, we use a 31-d analytic projection of the full 108-d feature vectors, with 27 dimensions corresponding to different orientation channels (18 contrast sensitive and 9 contrast insensitive), and 4 dimensions capturing the overall gradient energy in four blocks of 2×2 cells. For the block SIFT (used in conjunction with our sliding window-based search), the fixed 48×128 detection window is represented by 6×16 cells, to make a 2976-d feature vector per detection window. For the bag-of-words SIFT feature (used in conjunction with our branch-and-bound search), the 100,000 randomly chosen descriptors are used to generate a codebook with 1000 clusters, learned by k-means (descriptor size 1000).

Each type of descriptor (regular grid versus bag of words) offers certain advantages: block SIFT has been shown to give some of the best results for pose prediction or detection in the past, but is less amenable to constructing efficient bounding functions. In turn, bag-of-features models tend to perform less well for prediction or detection, but are easier to bound (see later discussion).

Three-dimensional human poses (output $\mathbf{x}$) are represented as 60d vectors of three-dimensional body joint positions (20 markers each with $\mathbf{r}$, $\mathbf{x}$ and $\mathbf{z}$ coordinates) using 'torsoDistal' as root. All poses are preprocessed by subtracting the root joint from all the other body joint positions.

Datasets: We use two datasets with 3d ground truth. One is HumanEva [25] which has relatively simple backgrounds, the other is a dataset constructed by us where we've automatically placed a synthetic impostor on real images. The HumanEva-I dataset [25] consists of 3 subjects performing several predefined actions, such as box, jogging, walking, *etc*. Table 1 summarizes the characteristics of the dataset used. We divided each video sequence (corresponding to each motion of each subject) into approximately two equal parts: the second half is the training set, and the first half the test set, as suggested by [25].

Action	S1	S2	S3	TrSet	TeSet	Total
Box	502	464	933	945	944	1889
Jogging	439	795	831	1033	1032	2065
Walking	1176	876	895	1474	1473	2947

Table 1. Size of dataset for block SIFT features on HumanEva-1. S1, S2 and S3 denote Subject 1, Subject 2 and Subject 3, respectively. TrSet and TeSet refer to the training and test sets. Notice that the size of the dataset is smaller than the number of image frames, as invalid mocap frames are removed.

The prediction error is the Euclidean distance between the estimated joint center and the true center averaged over all joints, per frame [25]: $\mathrm{Err}_{seq} = \frac{1}{T}\sum_{i=1}^{T} D(\mathbf{x}_i, \overline{\mathbf{x}}_i)$ where T is the length of sequence and $D(\mathbf{x}_i, \overline{\mathbf{x}}_i) = \frac{1}{M}\sum_{j=1}^{M} \|m_j(\overline{\mathbf{x}}_i) - m_j(\mathbf{x}_i)\|$, where $m_j(\mathbf{x}_i) \in \mathbb{R}^3$ extracts the three dimensional coordinates of the jth joint center, M is the number of the centers, and $\|\cdot\|$ the Euclidean distance.

The second database of quasi-real images was created by us in order to train with more challenging scenes and backgrounds. We started with images from the INRIA pedestrian dataset. In order to diversify the training data, we rendered and inserted a synthetic graphics 'impostor' for which 3d poses were known. To partially automate the process, and make sure the results have as realistic a distribution as possible in terms of relative scales, sizes, location of ground plane, *etc*., we run a person detector on images and add our impostor at a similar scale and in the vicinity of detections, properly placed according to estimates of the ground plane [15]. A few samples are shown in fig. 2. This training setting provides a certain degree of automation, but still requires visual inspection and manual cleanup. Other complementary technologies and sensors may soon become available for more realistic acquisition of 3d human motion data in the real-world, as encouragingly reported in [31].

Figure 2. Images from our dataset. The skeleton was rendered using poses from motion capture. The size and location of the object was automatically computed based on person detection results.

To give intuition on why structSVM is suitable to pose estimation, we show correlation coefficients for human walking and jogging motions in fig. 3. Entries are computed as: $\rho_{ij} = \frac{\mathrm{cov}(y^i, y^j)}{\sqrt{\mathrm{cov}(y^i, y^i)\mathrm{cov}(y^j, y^j)}}$ where $\mathrm{cov}(\cdot, \cdot)$ is the covariance between two random variables. The light, highly correlated off-diagonal components indicate non-negligible coupling between the state variables (3d joint positions).

In a first experiment, given in tables 2 and 3, we report the average joint position error of RVM (relevance vector machine), GPR (Gaussian process regression), SVR (support vector regression), HME (mixture of experts) and structural SVM models, for the simpler case of bounding boxes obtained using background subtraction. structSVM is consistently better for both activity-specific models and for generic models that were trained jointly (20% improvement on walking motions, compared to the best performing method, HME).

In the second set of experiments, we consider integrated methods for localization and pose estimation in HumanEva, without assuming knowledge of the background, hence without relying on background subtraction for person detection. We use two types of search methods for localization (bounding box search): branch-and-bound and sliding windows. For branch-and-bound, we use the bag-of-words SIFT model. For sliding windows, we use block-SIFT features extracted within a fixed detection window, and search the object over ten different image scales.

In one experiment, we first locate the human using a structural SVM trained for detecting people then estimate the 3d pose using a second, separately trained structural SVM with the given bounding box as input. In a second experiment, we learn and infer the person location and the 3d pose jointly. Our first finding is that the detection rate

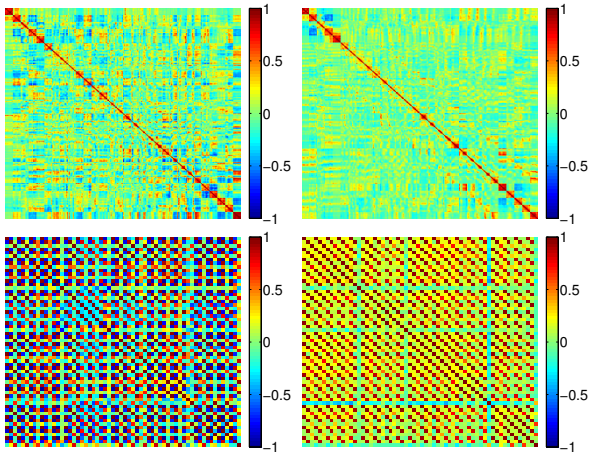

Figure 3. Matrices of correlation coefficients for block SIFT features collected on a regular grid inside the object bounding box *(top)* and 3d poses *(bottom)* (60d vectors of temporally ordered three dimensional body joint positions). *Left:* boxing. *Right:* combined boxing + jogging + walking motions. Notice non-negligible correlations among both inputs and outputs.

Motion	RVM	GPR	SVR	HME	structSVM
Box	69.2	67.8	64.2	63.5	**60.6**
Jogging	66.3	65.9	63.2	58.5	**53.9**
Walking	59.1	58.9	61.5	55.9	**45.4**

Table 2. Models trained on each motion of each subject. The descriptor is block SIFT extracted on a regular grid inside the person bounding box. 'C1' means that the image feature is extracted from images captured by the first color camera. Error, in mm, is averaged over the same motion of three subjects.

Motion	RVM	GPR	SVR	HME	structSVM
Box	71.5	71.6	70.9	67.5	**63.2**
Jogging	62.3	62.3	59.2	53.4	**47.6**
Walking	66.1	66.0	63.1	58.7	**46.3**

Table 3. Activity specific models, trained on the same motion of all subjects. The descriptor is block SIFT extracted on a regular grid inside the person's bounding box, divided into a 6×5 cell grid, with gradients in each cell quantized into 9 orientation bins ($0^0 - 180^0$) (descriptor of size 270). 'C1' means that the features are extracted from images captured by the first color camera. Error, in mm, is averaged over the same motion of three subjects.

is 100% both for the separately trained model (second experiment) and for the jointly trained model. The detection rate is computed as correct localization, where bounding boxes were considered correct if their area overlap with ground truth exceeded 50%. Hence both methods appear to have learned that many images share similar backgrounds. While this may not seem entirely remarkable, it is still non-trivial, considering how much attention background subtraction and shadow removal methods receive in almost any task where their usage is applicable—knowing the background makes a big difference in practice, see also our third finding.

Our second finding is that the jointly trained model is consistently 8% better than the separately trained model for 3d pose estimation in all the cases, very likely because of improved window localization. Our third finding is that the jointly trained model has similar (albeit slightly lower) performance w.r.t. the structSVM where the bounding box was given (*e.g*. tables 2 and 3). This is not surprising since background subtraction also exploits temporal information whereas our jointly trained model didn't (this would be easy to include, as an additional kernel linking states at successive timesteps, not just observations and states at a given timestep). However, our model works even when we have to estimate pose for a disparate set of images rather than a video (our next experiment).

We run two types of structural SVM models on the quasi-real dataset using block SIFT features, sliding window search for object localization, and gradient descent search for the remaining state components (human joint variables). In the first experiment, we initially locate the person using structural SVM for object location, then estimate the 3d pose based on the resulting bounding box. In the second experiment, we perform localization and 3d pose estimation jointly.

The error in the 3d position of the articulations for the jointly trained model is 62.3 mm on this dataset, lower than 67.6 mm obtained by the separately trained model. The detection rate of the jointly trained model is about 3% better than that of the separately trained model. Beside block SIFT, we also run two types of structural SVM models using bag of words SIFT and branch-and-bound for person localization. In this case, as well, the jointly trained model achieves better performance than the one trained separately. We also found that structural models based on bag-of-words SIFT underperformed those trained using block SIFT, perhaps because the former does not encode spatial information.

Sample localization and 3d reconstruction results obtained by our structural SVM model, trained on the quasi-real dataset and tested with complex images, are shown in fig. 4. Various types (and degree) of failure of the model are shown in fig. 5. Clearly these leave ample space for improvement in detection quality, pose prediction accuracy, and general large-scale reliability. However, we appreciate that the integrated model provides conceptual advantages and a formal unifying framework to explore alternative feature types, bounding functions, as well as inference and training procedures.

Feature	Joint Position Error		Detection Rate	
	Separate	Joint	Separate	Joint
BlockSIFT	67.6	62.3	84.8%	87.2%
HistoSIFT	81.2	75.9	78.6%	81.1%

Table 4. Comparisons of separately and jointly trained structSVM models in the quasi-real dataset (joint position error in mm).

4. Conclusions

We have presented an integrated continuous structured prediction model for joint visual object localization and state estimation. This offers an end-to-end training and inference framework that overcomes the limitations of regressive methods, as it consistently searches for both the location of the object is in the image and for its state. This is achieved by combining global inference methods like branch-and-bound with greedy (gradient descent) search in an alternation scheme. The method comes in contrast to previously proposed techniques that assume the inputs have already been selected at run-time. We illustrate the campabilities of this framework through a graded set of localization and 3d human pose reconstruction experiments in monocular images, both in the HumanEva benchmark, and in images of complex scenes captured in the real world.

Future work: We actively study alternative inference techniques based on branch-and-bound for both image and state search, as well as the design of bounding functions for image descriptors computed on regular grids.

Acknowledgements: This research was supported, in part, by awards from the European Commission (MCEXT-025481) and NSF (IIS-0535140).

References

[1] A. Agarwal and B. Triggs. Recovering 3d human pose from monocular images. *PAMI*, 2006.

[2] M. Andriluka, S. Roth, and B. Schiele. People Tracking-by-Detection and People-Detection-by-Tracking. In *CVPR*, 2008.

[3] Y. Bar-Shalom and T. Fortman. *Tracking and Data Association*. Academic Press, 1988.

[4] M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In *ECCV*, pages 2–15, 2008.

[5] L. Bo and C. Sminchisescu. Structured Output-Associative Regression. In *CVPR*, 2009.

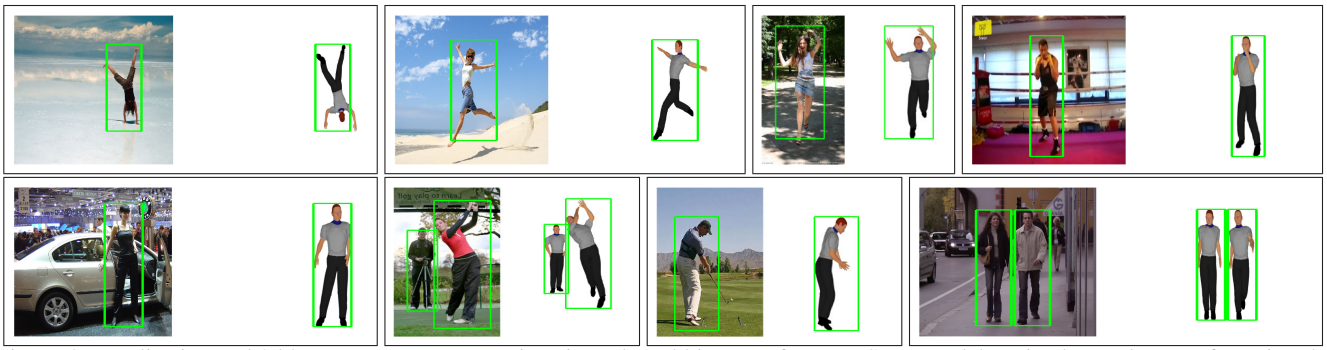

Figure 4. Localization and 3d human pose reconstructions in real world images, for structSVM models trained on a dataset of quasi-real images. The framework allows the localization and reconstruction of multiple people.

Figure 5. Failure modes of the structSVM framework. *Left:* image clutter with linear structure confuses the phase of the golf swing. *Center:* monocular ambiguities (flipped legs). *Right:* Partially incorrect reconstruction (left subject, running versus kicking a ball).

[6] M. Brubaker and D. Fleet. The Kneed Walker for Human Pose Tracking. In *CVPR*, 2008.

[7] K. Choo and D. Fleet. People Tracking Using Hybrid Monte Carlo Filtering. In *ICCV*, 2001.

[8] R. D. Cook. *Regression Graphics*. Wiley Inter-Science, 1988.

[9] C. Cortes, M. Mohri, and J. Weston. A general regression technique for learning transductions. In *ICML*, pages 153–160, 2005.

[10] J. Deutscher, A. Blake, and I. Reid. Articulated Body Motion Capture by Annealed Particle Filtering. In *CVPR*, 2000.

[11] A. Ess, B. Leibe, K. Schindler, and L. van Gool. A Mobile Vision System for Robust Multi-Person Tracking. In *CVPR*, 2008.

[12] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.

[13] T. Finley and T. Joachims. Training structural SVMs when exact inference is intractable. In *ICML*, pages 304–311, 2008.

[14] D. Heckerman, D. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *JMLR*, 1, 2000.

[15] D. Hoiem, A. Efros, and M. Hebert. Putting Objects in Perspective. In *CVPR*, 2008.

[16] M. Isard and A. Blake. CONDENSATION – Conditional Density Propagation for Visual Tracking. *IJCV*, 1998.

[17] T. Joachims, T. Finley, and C. Yu. Cutting-plane training of structural svms. *Machine Learning*, 2009.

[18] M. Kim and V. Pavlovic. Dimensionality reduction using covariance operator inverse regression. In *CVPR*, 2008.

[19] J. Lafferty, A. Mccallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

[20] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *CVPR*, pages 1–8, 2008.

[21] Y. LeCun, S. Chopra, R. Hadsell, R. M. Ranzato, and F. Huang. A Tutorial on Energy-Based Learning. In *Predicting Structured Data*. MIT Press, 2006.

[22] D. Ramanan and C. Sminchisescu. Training Deformable Models for Localization. In *CVPR*, 2006.

[23] R. Rosales and S. Sclaroff. Learning Body Pose Via Specialized Maps. In *NIPS*, 2002.

[24] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *CVPR*, 2003.

[25] L. Sigal and M. Black. Humaneva: Synchronized video and motion capture dataset for evaluation of articulated human motion. Technical report, 2006.

[26] L. Sigal and M. Black. Predicting 3d people from 2d pictures. In *AMDO*, 2006.

[27] C. Sminchisescu, A. Kanaujia, and D. Metaxas. Learning Joint Top-down and Bottom-up Processes for 3D Visual Inference. In *CVPR*, 2006.

[28] C. Sminchisescu and B. Triggs. Hyperdynamics Importance Sampling. In *ECCV*, volume 1, pages 769–783, Copenhagen, 2002.

[29] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484, 2005.

[30] R. Urtasun, D. Fleet, A. Hertzmann, and P. Fua. Priors for people tracking in small training sets. In *ICCV*, 2005.

[31] D. Vlasic, R. Adelsberger, G. Vannucci, J. Barwell, M. Gross, W. Matusik, and J. Popovic. Practical Motion Capture in Everyday Surroundings. In *SIGGRAPH*, 2007.

[32] J. Weston, B. Schölkopf, O. Bousquet, T. Mann, and W. Noble. Joint kernel maps. In *LNCS*, 2005.