Non-Rigid Object Detection with Local Interleaved Sequential Alignment (LISA)

David Hurych, *Member, IEEE,* Karel Zimmermann, *Member, IEEE,*
and Tomáš Svoboda, *Member, IEEE*

Abstract—This paper shows that the successively evaluated features used in a sliding window detection process to decide about object presence/absence also contain knowledge about object deformation. We exploit these detection features to estimate the object deformation. Estimated deformation is then immediately applied to not yet evaluated features to align them with the observed image data. In our approach, the alignment estimators are jointly learned with the detector. The joint process allows for the learning of each detection stage from less deformed training samples than in the previous stage. For the alignment estimation we propose regressors that approximate non-linear regression functions and compute the alignment parameters extremely fast.

Index Terms—non-rigid object detection, alignment, regression, exploiting features, real-time, waldboost, sliding window, sequential decision process

1 Introduction

Detection of objects with appearance altered by pose variations (including non-rigid deformations and viewpoint changes) is more difficult than the detection of objects in a single pose [1], [2]. If the detection time is constrained, exhaustive search over the space of possible poses with a single pose detector is intractable.

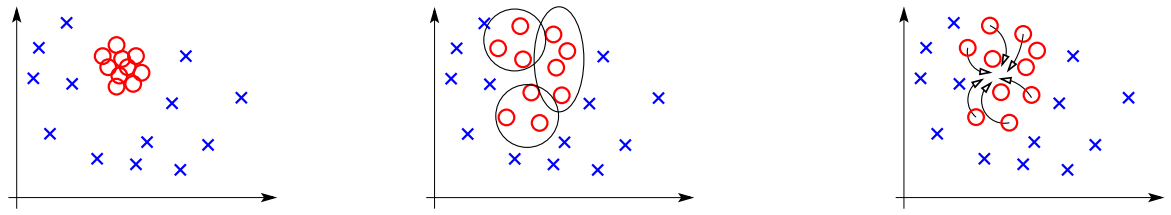
An ample amount of work has been devoted to the detection of objects deformed by pose variations. Many approaches partition the positive training samples into clusters with similar poses, see Figure 1b. Some of them [3], [4] first estimate the pose cluster and then use pose-specific classifier to decide about the object presence. Others [5], [6] estimate pose cluster simultaneously with the detection process. A fine partitioning of the pose space is desirable to achieve good detection performance. However, the finer the partitioning, the fewer training samples fall into each cluster and therefore immense training sets are often needed [5]. In contrast to these approaches, recent work [1] uses simple pose estimators which align some detection features. These pose estimators help to detect objects in an arbitrary pose without training set partitioning.

Our feature alignment remedies the partitioning of the training set. In contrast to [1] which finds simple local deformations (e.g. inplane rotations) and aligns each feature independently, we rather estimate a *global non-rigid alignment* of all the features. Importantly, the alignment is estimated solely from the features used for detection by pre-trained regressors. The alignment estimation is reduced to reading a value from a look-up table which costs negligible time. On the other hand, our approach requires annotated data for learning. Nevertheless, the use of our method does not prevent the use of [1], thus the frameworks are complementary.

In our system the features are evaluated sequentially; each one reveals a certain amount of object deformation, see Figure 2. Features are *successively aligned* to the observed deformation which makes the positive class less scattered and easier to detect, see Figure 1(c). The training set is not partitioned and the number of necessary training samples remains relatively low, even for large deformations.

We demonstrate the sequential alignment idea on a Sequential Decision Process (SDP) similar to Waldboost [7], where the successive nature of feature evaluation allows for efficient application of the estimated alignment. In the SDP a classifier cumulatively estimates a *confidence* about the object presence or absence in a given detection window. Once the confidence is sufficiently low, the window is rejected. We extend the SDP by exploiting the same features that were used for the confidence computation to estimate the alignment. The alignment is then applied on the subsequent features and the process continues with more appropriately aligned features. In consequence, both the confidence and the alignment are estimated more efficiently as it is then easier to distinguish the well aligned positive samples from the background and to estimate the alignment from a closer neighbourhood. The process continues until the rejection or acceptance is reached as in the classical SDP. Note that the confidence and alignment updates are encoded by the same feature values, therefore in comparison with the classical SDP the computational complexity of

• *All authors are with the Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Cybernetics, Center for Machine Perception.*
E-mail: hurycd1, zimmerk, svoboda@cmp.felk.cvut.cz



(a) objects in a single pose (b) partitioning into pose clusters (c) our approach–aligning features

Fig. 1: Simplified sketches of positive (red circles) and negative (blue crosses) samples in 2D feature space. (a) objects in a single pose exhibit smaller scatter than objects deformed by pose variations, (b) scattered samples are often partitioned into pose clusters with a small number of training samples, (c) our approach aligns features during the detection to compensate for object deformation consequently making positive samples less scattered.

SDP with alignment is almost preserved. We call the proposed scheme Sequential Decision Process with *Locally Interleaved Sequential Alignment* (SDP with *LISA*).

The contribution of this paper is threefold: (i) we show that features evaluated in the sliding window detection process also contain knowledge about the correct alignment of the evaluated window on the observed object deformation; (ii) we propose very efficient piecewise linear regression functions which are jointly learned with the classifier. This facilitates estimating the alignment during the detection process; (iii) we show that the estimated alignment speeds up the detection process by reducing the search space and improves the detection rates.

2 Related work

Great progress has been made in the detection of objects under varying poses and deformations [1], [8], [9], [10]. The predominant strategy is to combine a collection of classifiers, each dedicated to a single pose or deformation [2], [11], [12], [13], [14], [15]. To train multiple classifiers, either the training data need to be separated into disjoint clusters [5], [12], or the features in training samples need to be registered to lie in correspondence [1], [13], or both strategies are combined [2].

The clustering of training data imposes the need to collect large amounts of data for learning each classifier separately. Some authors try to reduce the amount of training data by sharing some features across multiple views [12], [11]. To avoid the clustering of training data, some methods [12], [13], [14], [15] align the object features in each training sample (before or during the training process) to lie in correspondence. This task usually requires a precise labelling of object features correspondence.

Other methods avoid the necessary labelling and try to align the features automatically [2], [1] before their evaluation. Usually, the feature positions and low level deformations are estimated first, e.g. by computing the dominant edge orientation in some part of the detection window and using pose-indexed features [1], [13]. The automatic feature alignment keeps the training set less scattered, and improves the detection rates but lacks interpretation. We model the alignment for a specific class of objects, and as a side product of the detection we obtain a parametrized alignment of the whole model.

Detection of deformable objects

Recently, Ali et al. [1] proposed to use pose indexed features coupled with dominant edge orientation estimation, in different scales and positions in the detection window, for feature alignment. By the feature alignment, they forgo the need to train a collection of detectors for different object poses and learn a single deformable detector. The dominant edge orientation is partitioned into 8 bins in 14 poses (1 pose in the largest scale, 4 in smaller scale and 9 in the smallest scale) and needs to be computed for every candidate position in order to estimate the features poses. For alignment estimation they need to evaluate additional $8{\cdot}14 = 112$ features apart from the detection features which is considerably more than the average number of features needed for classification in our system. We interleave object detection and alignment by regression. The detector runs on increasingly better aligned features, which consequently groups together the training samples in the feature space, as shown in Figure 1(c). Thanks to the design of regression functions and the re-use of detection features, a negligible number of additional computations are needed (reading the alignment parameters from a look-up table and moving the features) and the computational complexity grows with the dimension of the alignment space only very gently. The approach of [1] does not require the training data labelling while our method does. However, our method allows having the pose space not discretized.

In general, SDP with LISA outperforms the standard SDP's [7], [16], [17], [18], [19], [20] in both the speed and detection performance. In [2] the authors argue that the sliding window-based object detectors work best when trained on examples that come from a single coherent group with well aligned features,

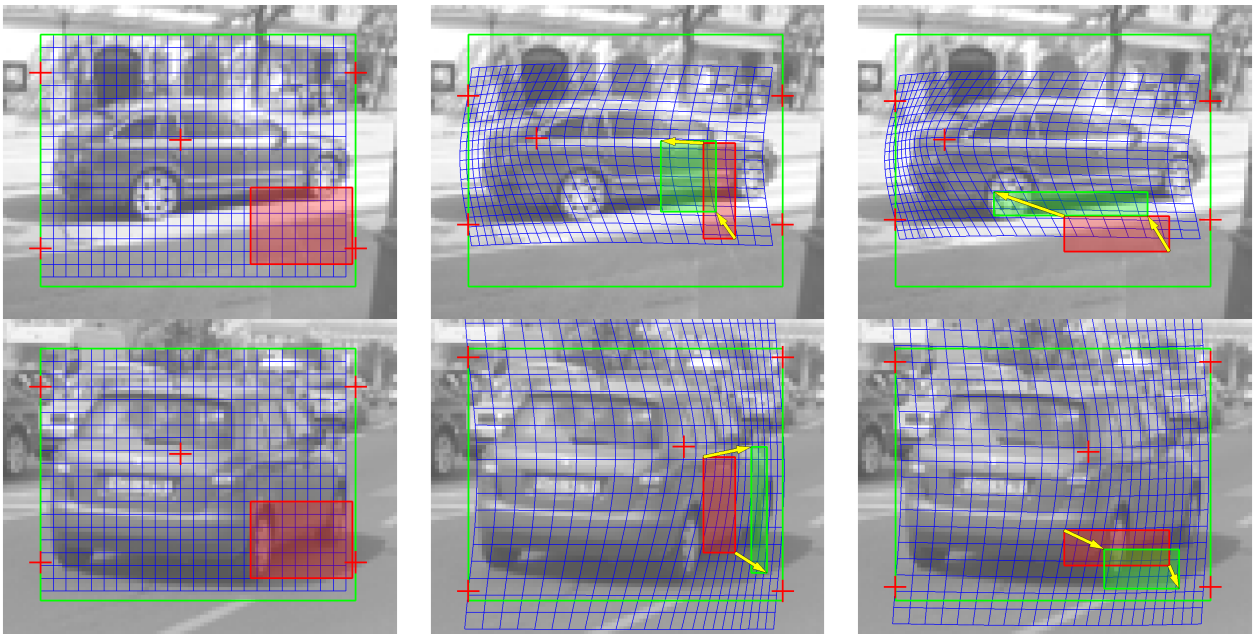


Fig. 2: **Local Interleaved Sequential Alignment (LISA):** Left: The first feature always has the same position. The deformation of the local coordinate system is outlined by the blue mesh. Features evaluated during the detection process contribute to both (i) confidence and (ii) non-rigid deformation. The second column shows how features are aligned after 30 evaluated features. A non-aligned feature position is delineated by the red rectangle; the aligned feature is green. The last column shows alignment of the last evaluated feature.

e.g. frontal faces. Our improvement in performance is caused by the ability to locally align on the displaced, rotated, and deformed object instances. The gain in speed is obtained by the search space reduction. From the point of view of 2D translation search space reduction (sparser detection grid) has already been shown for a cascade of classifiers in [21] and for SDP in [22]. Here we show that we can effectively reduce a high dimensional search space of non-rigid deformation.

Recent deformable part-based object detectors [23], [12], [13] achieve excellent detection rates, but are far from the real-time performance. The computational complexity of part-based object detectors is given governed by the detection of model parts and by the estimation of globally optimal parts configuration. Model parts are usually detected by an SVM-based [23], [12] or AdaBoost [13] classifier. The root-part of the object is detected first [23], [13], which allows reducing the search space for the remaining parts. After estimating the candidate positions of the parts, the globally optimal configuration of the model parts (or for multiple models for multiple views [12]) is found by using the dynamic programming. Our method may be compared from the computational complexity point of view with [8]. Their classifier is an SVM working with HoG features which similar to [12] for *one-view model* as well as for the *root part* of the part-based model in [23]. In [8] N denotes the number of possible locations in multiple scales and V the number of evaluated features in each subwindow. For an SVM-based detector, *VN multiplications* are needed. Since the SDP with LISA may learn to compensate 2D translation [22], the number of poses N may be significantly reduced to M and we may run the SDP with LISA on a sparser detection grid. Here we compensate non-rigid deformation of high dimensionality and $M \ll N$. Our method requires $4VM$ *additions* in the worst case (all features being evaluated without the early rejection), where 4 is the number of additions needed to transform each evaluated feature according to estimated alignment (see Section 6 for details). For performance comparison with [23], [12] and the running times of different methods see Section 8.

The Patchwork of Parts [24] builds a statistical model for detection of objects with multiple parts. The detection process loops through the image locations (positions and scales) and evaluating the classifier for every part (class) in every location. Model deformations are modelled as shifts of object parts which are than recombined using a patchwork operation. The algorithm is able to classify 100 subwindows per second which is not fast enough a real-time performance.

Search space reduction

A speeding-up of the original part based model [23] was addressed in [25]. The authors argue that the cost of detection is dominated by the cost of matching each part of the model to the image and not by the cost of computing the optimal configuration of parts. They propose to learn a multi-resolution hierarchical part based model. The parts are tested sequentially and image locations are discarded as soon as a partial

detection score falls under some threshold. The resulting algorithm achieves almost the same detection accuracy as the original algorithm and runs twice as fast.

The search space reduction for detection speed-up has been approached also in *Efficient Subwindow Search* (ESS) [26]. ESS is reducing the search space by a branch and bound algorithm. It defines multiple sets of rectangles (sets of candidate windows) in the image. After evaluation of all the features in the image the algorithm computes the upper bound (highest possible detection score) that the score function could take on any of the rectangles in each set. The authors propose an efficient scheme for going hierarchically through all the possible rectangles (scales and translations) without the need to exhaustively evaluate the detection score for all possible rectangles. Many rectangle sets are rejected as soon as the upper bound is under some acceptance threshold. The disadvantage is the need for evaluating all the features in the image first. This is well applicable for the approaches which use a bag of features or some shared low level features, usually for multiview and part-based object detection [14], [12]. After the features evaluation, the detectors need to evaluate a non-trivial score function (usually SVM-based classifiers) of all the features which fall into each particular rectangle, and here ESS brings significant speed-up [14]. A sliding window-based SDP does not need to evaluate all the features in all the positions and scales thanks to the early rejection stages. Here, ESS would actually slow down the process by the necessary evaluation of all features first. Also, in the SDP the rejection thresholds are already known in advance for each stage and no other bounds need to be computed. From the search space reduction point of view, ESS reduces the number of candidate window translations as well as scales, but does not take into account other object deformations.

The recently proposed Crosstalk Cascades [27] assume that adjacent subwindows responses at nearby locations and scales are correlated. As soon as the classification score for some location reaches some threshold, all points in the detection grid in the close neighbourhood start to be evaluated as well (excitatory cascades). When the ratio of score in the current position and in at least one position in the close neighbourhood goes under some stage specific threshold the evaluation of remaining stages in actual position stops (inhibitory cascades). Training the classifier needs perturbing each training sample by small 2D shifts to ensure the correlation of classifier answers at nearby positions. This perturbing corrupts the performance of the cascaded detector. We approach the problem from an opposite direction. We are aligning the detection window to a pose at which it fits the object better. It does not require corrupting the training samples by adding shifts to positive samples. The detection of well aligned samples is easier and needs to evaluate fewer features to reach the decision. From the point of view of search space reduction, the Crosstalk Cascades reduce the 2D sliding window translation and scale as well as [26]. Nevertheless, they still need to look at every position in the detection grid but the number of evaluated features is reduced. Our method yields the detection grid reduction because our detector is able to move and deform.

Non-rigid alignment estimation

We mention only few of the most relevant papers in this area, since our method focuses more on improvement of *object detection*, than on precise alignment estimation.

State of the art methods specialized in alignment of deformable models are now able to cope with quite large object deformations [28], [29], [30]. In [28] authors train a cascade of regressors for non-rigid face deformation estimation. Every regressor in the cascade is a *Fern*. Each Fern separates the training set into subsets (one for each leaf), where within one subset there are samples with a similar type of deformation. Therefore each Fern basically divides the space of possible deformations. This makes the alignment task easier for regressors trained on subsets in each leaf and allows coping with larger deformations. On the other hand, it is necessary to cluster the training data to learn the regressors. In comparison, our method does not need to cluster the training data for learning. We transform only the features which is faster than inversely transforming the whole image patch as in [28].

In the Boosted Appearance Model (BAM) [30] the authors propose to train a classifier which recognizes well aligned deformable model from those not well aligned. The classifier is then used in a criterial function which includes a parametrized shape of the object. The goal is to find the shape parameters that maximize the score of the learned strong classifier. The problem is solved iteratively by gradient ascent optimization. The follow-up to BAM is Boosted Ranking Model [29] (BRM). In BRM given 2 image patches a classifier is trained to recognize the better aligned image patch from the worse aligned. The shape in both BAM and BRM is a set of 2D points and the displacement of each of the points is parametrized by a 2D vector. The number of parameters is reduced by projection to a low dimensional space via PCA similarly to our approach. In comparison, we model the shape deformation by deforming the whole grid which covers the object patch. This allows us to efficiently transform the features instead of inversely transforming the whole image patch. We obtain a transformation for every pixel in the grid and not only for the control points of the shape as in [28], [29].

A Sequence of Learnable Linear Predictors for learning a fixed sequence of linear predictors (weak

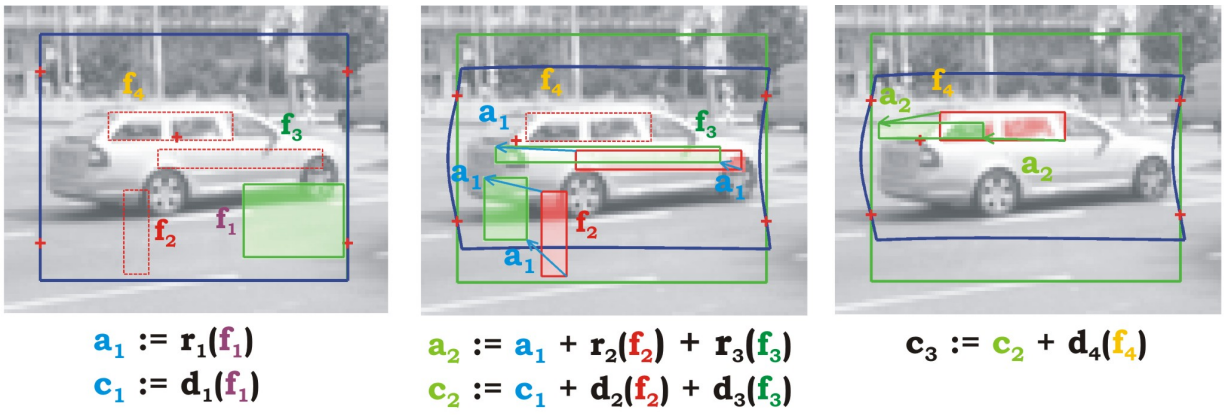


Fig. 3: **Classification:** Three steps of SDP with LISA are depicted. Left: In the initial position, only feature 1 is evaluated. From its value the first alignment $\mathbf{a}_1$ and confidence c_1 are computed. Middle: the alignment $\mathbf{a}_1$ is applied on features 2 and 3. Note that the applied alignment $\mathbf{a}_2$ is updated by contribution of two regressors, not just one. Also note that the alignment $\mathbf{a}_1$ was not applied on feature 4. Right: the last feature is moved from its initial position by the accumulated alignment $\mathbf{a}_2$.

regressors) that estimate the object alignment was proposed in [31]. Each predictor in the sequence is learned on the *estimation error* of the previous predictor. A similar idea was later proposed in [32], only they use Ferns instead of linear predictors as the weak regressors. Our work uses weak regressors proposed in [22]. Instead of using simple linear functions we propose a non-linear regression and approximate it by piecewise linear (or piecewise constant) functions which improves the alignment and keeps the fast performance of linear predictors.

3 SDP WITH LISA CLASSIFICATION

We divide the classification process into K stages. In each stage k, only one feature is evaluated. The value of this feature contributes to the confidence and the alignment. Contributions are determined by (i) a detection function $d_k : \mathbb{R} \to \mathbb{R}$, which maps the feature value to a contribution to the confidence, and (ii) a regression function $r_k : \mathbb{R} \to \mathbb{R}^m$, which assigns an m-dimensional contribution to the alignment vector $\mathbf{a}$ using the same feature value. Both the confidence and the alignment are accumulated from evaluated features. Then there is a threshold $\theta_k \in \mathbb{R}$ (estimated during the learning), which allows to reject windows with the so far accumulated confidence lower than θ_k. In each stage, the feature can potentially be aligned. This is determined by a binary value q_k, which is estimated by boosting during the training stage. If q_k is `TRUE`, this will invoke aligning of the feature, while q_k = `FALSE` means that the non-aligned feature will be used.

The alignment estimated from a single feature may be inaccurate, therefore it must be accumulated over multiple features. We keep the last *valid* alignment, denoted as $\mathbf{a}_\omega$, where index ω corresponds to the stage at which the alignment was estimated. Besides that, we also accumulate alignment updates from all evaluated features. This alignment is updated in each stage k and we denote it by $\mathbf{a}_k$. Hence, there are two alignments: (i) accumulated up to stage k denoted by $\mathbf{a}_k$ and (ii) valid, which is applied on features, denoted by $\mathbf{a}_\omega$. The stage at which the index ω is updated is determined by a binary value z_k = `TRUE`; z_k is again estimated by boosting during the training stage).

We define a feature function $f_k : (\mathcal{I} \times \mathbb{R}^m) \to \mathbb{R}$ as a mapping which assigns a feature value to a window with image data $I \in \mathcal{I}$ and m-dimensional alignment vector $\mathbf{a} \in \mathbb{R}^m$. For the sake of simplicity, we refer to the feature function as the *feature* and to image data in the sliding window as the *window*. Based on the above introduced notation, we define the strong classifier as a collection:

$$H = [f_1, q_1, d_1, \theta_1, r_1, z_1, \ldots, f_K, q_K, d_K, \theta_K]\,. \qquad (1)$$

The classification Algorithm (Figure 4) summarizes how the SDP with LISA decides about object presence or absence in a given window I with the given strong classifier H. See also Figure 3 for illustration. In the algorithm, we denote c_k as the confidence and $\mathbf{a}_k$ as the alignment, both accumulated up to stage k.

4 JOINT LEARNING OF SDP WITH LISA

The expected output from learning is the strong classifier H, Eq. (1), inputs are *training* and *validation* sets. At the beginning of each training stage, the training set with the following structure is available:

$$\begin{aligned}\mathcal{T} \;=\; \{&(I^1, \mathbf{t}^1, y^1), \;\ldots,\; (I^p, \mathbf{t}^p, y^p), \\ &(I^{p+1}, y^{p+1}), \;\ldots,\; (I^N, y^N)\},\end{aligned} \qquad (2)$$

where $I^1, \ldots, I^p$ are positive image data, $I^{p+1}, \ldots, I^N$ are negative image data, $y^1 \ldots y^N$ are labels such that $y^1 = \cdots = y^p = 1$, $y^{p+1} = \cdots = y^N = -1$ and $\mathbf{t}^1 \ldots \mathbf{t}^p$ are correct alignments of positive data. The validation

Initialize $\mathbf{a}_0 = \mathbf{0}, c_0 = 0, k = 1, \omega = 0$.
while $k \leq K$ **do**
  if $q_k = \texttt{TRUE}$ **then** *// use alignment*
    Estimate the value of feature $v = f_k(I, \mathbf{a}_\omega)$ with alignment $\mathbf{a}_\omega$.
  else
    Estimate the value of feature $v = f_k(I, \mathbf{0})$ without alignment.
  end if
  Update confidence $c_k \leftarrow c_{k-1} + d_k(v)$.
  if $c_k < \theta_k$ **then**
    reject the window and break,
  end if
  Estimate alignment $\mathbf{a}_k = \mathbf{a}_{k-1} + r_k(v)$.
  if $z_k = \texttt{TRUE}$ **then** *// alignment is valid*
    update $\omega \leftarrow k$
  end if
  $k \leftarrow k + 1$
end while
Accept the window.

Fig. 4: **Classification algorithm:** Classification of a single window by SDP with LISA

set $\mathcal{V}$ and the testing set $\mathcal{W}$[1] have the same structure.

The learning algorithm uses the following notation: $[[\Psi]]$ is a binary function equal to 1 if a statement Ψ is TRUE and 0 otherwise, and $[\Upsilon\ \Xi]$ is concatenation of Υ and Ξ. We introduce the error of the strong classifier H on validation data $\mathcal{V}$ denoted as $E(H, \mathcal{V}) = \sum_i [[H(I_i) \neq y_i]],\ \forall (I_i, y_i) \in \mathcal{V}$. For the sake of completeness we define: $E(\emptyset, \mathcal{V}) = \infty$. To simplify the notation, we also denote a weak classifier $\texttt{wc}_k$ to be the following foursome $\texttt{wc}_k = [f_k\ q_k\ d_k\ \theta_k]$.

The joint learning of SDP and LISA, see Figure 5, successively builds a strong classifier H. The current stage is denoted by the lower index k, the training samples are indexed by the upper index i. Since we allow the alignment to be accumulated over multiple stages without direct application on features, we also keep the index ω of the last valid alignment.

The algorithm (Figure 5) first constructs two weak classifiers: $\widehat{\texttt{wc}}_k$, that use features either aligned by $\mathbf{a}_{k-1}$ or not aligned at all, and $\texttt{wc}_k$, that use features either aligned by $\mathbf{a}_\omega$ or not aligned at all (lines: 4-5). Then the validation errors of $[H_k\ \texttt{wc}_k]$ (strong classifier H_k concatenated with $\texttt{wc}_k$) and $[H_k\ \widehat{\texttt{wc}}_k]$ (strong classifier H_k concatenated with $\widehat{\texttt{wc}}_k$) are compared, and the one with the lower error is selected and denoted as H_k (lines: 6-11). If the alignment a_{k-1} is used (i.e. $\widehat{\texttt{wc}}_k$ is used and $\widehat{q}_k = \texttt{TRUE}$), then ω is set to $k-1$, which makes a_{k-1} to be the valid alignment from now on. After that, we jointly re-learn regression functions $r_{\omega+1} \dots r_k$ to estimate the alignment from

1. Sets $\mathcal{T}$ and $\mathcal{V}$ are used during the learning phase and testing set $\mathcal{W}$ is used for experimental evaluation.

input: $\mathcal{T}$, $\mathcal{V}$, $\mathcal{F}$
$k = \omega = 1$, $H_0 = \emptyset$, $w^i = 1/N, i = 1 \dots N$.
while $E(H_k, \mathcal{V}) \leq E(H_{k-1}, \mathcal{V})$ **do**

//Build two weak classifiers: $\texttt{wc}_k$ *that use* $\mathbf{a}_\omega$
//and $\widehat{\texttt{wc}}_k$ *that use* $\mathbf{a}_{k-1}$.
$\texttt{wc}_k = \textbf{learn\_weak\_cls}(\mathcal{T}, \mathcal{F}, H_k, \mathbf{a}^1_{k-1} \dots \mathbf{a}^N_{k-1})$
$\widehat{\texttt{wc}}_k = \textbf{learn\_weak\_cls}(\mathcal{T}, \mathcal{F}, H_k, \mathbf{a}^1_{\omega} \dots \mathbf{a}^N_{\omega})$

//Comparison of validation errors determine, whether
//to start using $\mathbf{a}_{k-1}$ *from stage* k *or not.*
if $E([H_k\ \texttt{wc}_k],\ \mathcal{V}) > E([H_k\ \widehat{\texttt{wc}}_k],\ \mathcal{V})$ **then**
  $H_k \leftarrow [H_k\ \widehat{\texttt{wc}}_k]$
  if $\widehat{q}_k = \texttt{TRUE}$ **then** $\omega \leftarrow k - 1$ **end if**
else
  $H_k \leftarrow [H_k\ \texttt{wc}_k]$
end if

//Learn regressors estimating alignment from
//features $f_{\omega+1} \dots f_k$
$[r_{\omega+1} \dots r_k] \leftarrow \textbf{learn\_regressors}(\mathcal{T}, f_{\omega+1}, \dots, f_k)$

//Update regressors $r_{\omega+1} \dots r_k$ *in the strong classifier*
$H_k = \left[H_\omega\ [\texttt{wc}_{\omega+1}\ r_{\omega+1}\ z_{\omega+1}] \dots [\texttt{wc}_k\ r_k]\right]$

//Update weights of training samples
$w^i = \exp(-y^i H_k(I^i)),\ i = 1 \dots N$

//Collect new negative samples as FPs of H_k
$\mathcal{T} \leftarrow \textbf{update\_negative\_samples}(H_k)$
$[\mathbf{a}^1_\omega \dots \mathbf{a}^n_\omega\ \mathbf{a}^1_k \dots \mathbf{a}^N_k] \leftarrow \textbf{update\_alignment}(\mathcal{T})$
$k \leftarrow k + 1$
end while

Fig. 5: **Learning of SDP with LISA:**

features $f_{\omega+1} \dots f_k$, i.e. those features which have not yet been used for the alignment (lines 12-13). Please note, that in case where the alignment a_{k-1} is used, the re-learning reduces on the learning of the new regression function r_k, which will be used in the following stages.

Eventually, training weights are updated in line 14 and training data $\mathcal{T}$ are updated (line 15) by the new negative samples. Negative samples are collected as the false positive detections of the current strong classifier H_k. The algorithm continues until the validation error starts to increase.

The paper continues as follows: Learning of weak classifiers and alignment regressors for the group of features is described in section 5. The non-rigid deformation model is summarized in Section 6. Implementation details are explained in Section 7.

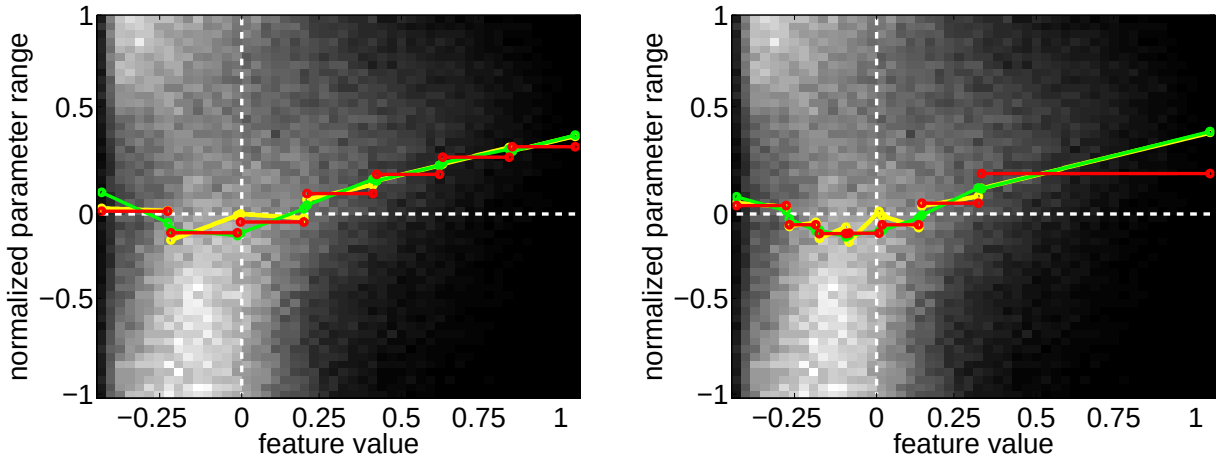


Fig. 6: **Examples of tested piecewise linear regression functions:** The density of the training data for one feature (depicted as grayscale heatmap) with fitted regression functions. The left image corresponds to *non-proportional partitioning* and the right image to the *proportional partitioning* of the feature space. Green is a piecewise affine function (i), yellow corresponds to a piecewise linear function (ii) and red is a piecewise constant function (iii).

5 Learning of Weak Classifiers and Alignment Regressors

5.1 Learning of Weak Classifiers

In our approach, the weak classifier d_k is a piecewise constant function of feature $f_k \in \mathcal{F}$ dividing feature values into U bins with sizes proportional to the training data density[2]. To simplify the notation, we define a bin assigning function $\delta_j : \mathbb{R} \to \mathbb{N}$ which assigns corresponding bin u to feature value $v = f_j\left(I^i, \mathbf{a}^i_\omega\right)$ for the i-th training sample and j-th feature. Given training samples weights w^i, the constant response κ_{ku} of d_k in bin u is computed as follows:

$$\kappa_{ku} \quad = \quad \arg\min_{\kappa} \sum_{i \in I_u} w^i (\kappa - y^i)^2 = \frac{\sum_{i \in I_u} w^i y^i}{\sum_{i \in I_u} w^i} \quad (3)$$

where $I_u = \{i \,|\, \delta(f_j(I^i, \mathbf{a}^i_\omega)) = u\}$ is the set of training samples indexes, which fell to bin u.

In the weak classifier estimation the same procedure is performed for each bin and each feature from the feature pool. Finally, we use the feature (and corresponding classifier d_k) which yields the lowest weighted error. Such approach is coincident with Gentleboost technique [33]. Rejection thresholds θ_k and $\widehat{\theta}_k$ are set in order to preserve the required maximum number of false negatives (FN) per learning stage. The FN limit is defined by the user to achieve the required running time similarly to [20].

5.2 Learning of Regressors

As already noted in section 3, the use of a regressor, learned on a single feature, may inaccurately align some positive samples and cause the lower detection rate. During the learning we do not immediately apply the estimated alignment on the feature, but we wait for the right number of features, for which jointly learned regressors yield better alignment and consequently lower the validation error of the detector.

In the learning algorithm (Figure 5, line 12), regressors $r_{\omega+1}, \ldots, r_k$ are jointly learned to compensate the residual alignment error $\Delta\mathbf{t}^i = (\mathbf{t}^i - \mathbf{a}^i_\omega)$ of preceding regressors $r_1, \ldots, r_\omega$. $\mathbf{t}^i$ is the vector of ground truth parameters of alignment. We search for regressors $r_{\omega+1}, \ldots, r_k$ which are the solution of the following problem:

$$\arg\min_{\tilde{r}_{\omega+1} \ldots \tilde{r}_k} \sum_{i=1}^{p} \left\| \Big(\sum_{j=\omega+1}^{k} \tilde{r}_j(f_j(I^i, \mathbf{a}^i_\omega)) \Big) - \Delta\mathbf{t}^i \right\|_F^2. \quad (4)$$

For simplicity we explain only one dimensional alignment estimation, i.e. with Δt^i being only a scalar for each sample instead of a vector. The higher dimensional alignment is learned for each dimension independently, therefore the following equations are valid for multiple alignment parameters estimated by each regressor. To solve the problem (4) we propose to learn a piecewise affine function by the least squares method. The feature space is divided into U bins, where each bin gives an affine response, see green lines in Figure 6.

The response of a regression function $r(v)$ is then computed as follows:

$$r(v) = \gamma_{j,\delta(v)} v + \lambda_{j,\delta(v)}, \quad (5)$$

where $\gamma_{j,\delta(v)}$ and $\lambda_{j,\delta(v)}$ are scalar coefficients. We considered (i) full affine function with $\gamma_{j,\delta(v)} \in \mathbb{R}, \lambda_{j,\delta(v)} \in \mathbb{R}$, (ii) linear function $\gamma_{j,\delta(v)} \in \mathbb{R}, \lambda_{j,\delta(v)} = 0$ and (iii) constant function with $\gamma_{j,\delta(v)} = 0, \lambda_{j,\delta(v)} \in \mathbb{R}$. Since we experimentally verified that all three functions yield

2. except the size of border bins which are $[-\infty, \texttt{min\_value}]$ and $[\texttt{max\_value}, +\infty]$

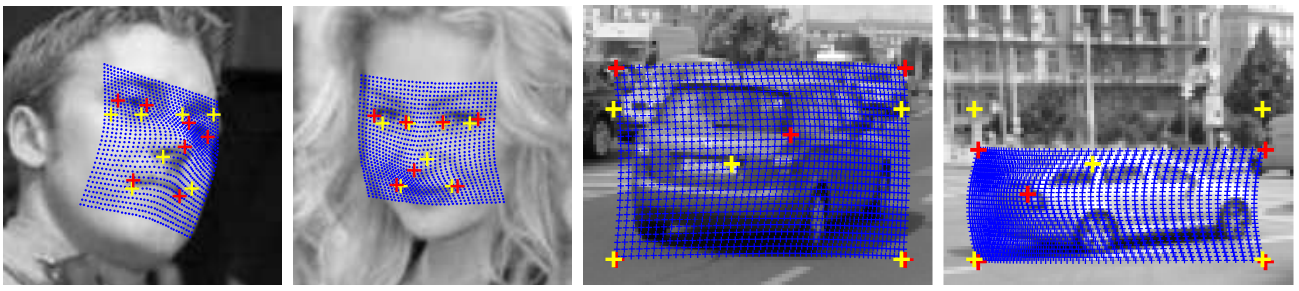


Fig. 7: Example data from the LFW and CSV datasets. The red crosses are ground truth labels. Yellow crosses depict the *mean of labelled positions for each dataset* and the blue points are the deformed grid. The deformed grid is obtained by thin plate splines non-rigid deformation using the correspondences between the yellow and red points.

similar results for a sufficient number of bins (see experimental evaluation in section 8.4), we used the piecewise constant function to speed up the alignment estimation process. Therefore problem (4) is reduced to search for coefficients $[\lambda_{\omega+1,1} \dots \lambda_{k,U}]$. We substitute Equation (5) with $\gamma_{j,\delta(v)} = 0$ into problem (4) to obtain the corresponding least squares problem:

$$\underset{\tilde{\lambda}_{\eta,u} \in \mathbb{R}}{\arg\min} \sum_{i=1}^{p} \left\| \sum_{j=\omega+1}^{k} \tilde{\lambda}_{j,\delta(f_j(I^i,\mathbf{a}^i_\omega))} - \Delta t^i \right\|^2. \quad (6)$$

To write the solution of (6) in a compact form, we further introduce a binary matrix

$$[\mathbb{A}]_{i,(ju)} = \begin{cases} 1 & \text{if } \delta(f_j(I^i, \mathbf{a}^i_\omega)) = u \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

where index i determines the row and indexes (ju)[3] determine the column. We also introduce vector $\tilde{\lambda} = [\tilde{\lambda}_{\omega+1,1} \dots \tilde{\lambda}_{k,U}]^\top$, which is concatenation of all unknown lambdas from all bins and all features. Finally, we form the vector $\Delta\mathbf{g} = [\Delta t^1 \dots \Delta t^p]^\top$ with all the residual alignments. The solution of problem (6) is then

$$\lambda = \underset{\tilde{\lambda}}{\arg\min} \left\| \mathbb{A}\tilde{\lambda} - \Delta\mathbf{g} \right\|^2 = \mathbb{A}^+ \Delta\mathbf{g}, \quad (8)$$

where $\mathbb{A}^+$ denotes Moore-Penrose pseudo-inverse [34] of matrix $\mathbb{A}$.

Two types of feature space partitionings were tested: (i) non-proportional partitioning, which divides the space into bins of equal sizes and (ii) proportional partitioning, which divides the space into bins of sizes inverse proportional to the training data density, where each bin contains the same number of training samples. See Figure 6 for example of partitioning into 7 bins with all three tested functions fit into the training data of one feature.

6 Non-rigid Deformation Model

We work with two types of alignments. The first is a simple two dimensional displacement and the second is a non-rigid deformation parametrized via PCA [35]. We define the feature as function $P : (\mathcal{I} \times \mathbb{R}^4 \times \mathbb{N}) \to \mathbb{R}$, the value of which is computed from image $I \in \mathcal{I}$ on the position specified by its left-upper corner $\alpha \in \mathbb{R}^2$ and right-bottom corner $\beta \in \mathbb{R}^2$ with type $\gamma \in \mathbb{N}$. We experimented with HoG features [27] (where γ denotes orientation of edges) and Haar features [36] (where γ stands for the feature type).

The alignment encoding the two dimensional displacement is given by two dimensional vector $\mathbf{a} = (\Delta x,\ \Delta y)^T$. Then, feature function $f(I, \mathbf{a})$ of the feature $P(\alpha, \beta, \gamma)$ aligned by the two dimensional displacement $\mathbf{a}$ is

$$f(I, \mathbf{a}) = P(I, \alpha + \mathbf{a}, \beta + \mathbf{a}, \gamma). \quad (9)$$

The non-rigid alignment deforms the position of every corner point α (resp. β) by m-dimensional vector $\mathbf{a} = [a_1 \dots a_m]^T$ as follows:

$$\alpha(\mathbf{a}) = \alpha + a_1 \mathbf{w}^1_\alpha + \dots + a_m \mathbf{w}^m_\alpha, \quad (10)$$

where $\mathbf{w}^1_\alpha \dots \mathbf{w}^m_\alpha$ are 2D Eigenvectors corresponding to deformations modelled in point α. Eigenvectors of the non-rigid deformation are obtained by PCA. Training of PCA is detailed in the next paragraph. Feature function $f(I, \mathbf{a})$ of the feature $P(\alpha, \beta, \gamma)$ aligned by the non-rigid deformation $\mathbf{a}$ is

$$f(I, \mathbf{a}) = P(I, \alpha(\mathbf{a}), \beta(\mathbf{a}), \gamma). \quad (11)$$

To train the PCA, we use the position of a few manually selected keypoints in each bounding box from the training set, see the red points in Figure 7. Given these keypoints, we compute the elastic transformation by thin plate splines transformation [37] of an orthogonal pixel grid within the bounding box for each training sample, see blue grids in Figure 7. The elastic transformation assigns a two-dimensional displacement vector to each pixel from the orthogonal grid in each bounding box. Finally, we concatenate these displacements for each training sample into one column vector and project them into the lower dimensional space by PCA.

The alignment may be applied either by deforming the features and placing them in the right position

3. (ju) denotes a linear combination of indexes j and u.

in the image or by inversely deforming the image. The latter would require the image deformation after each applied alignment update. Unfortunately, this is unthinkable for the real-time performance of the detector as we would need to transform the image (or part of it) multiple times for each candidate window. Hence we transform the features.

We need to keep the features in a rectangular shape to take advantage of the fast evaluation on integral images. Therefore the deformation of each feature is only approximated by anisotropic scaling, see Figure 2 for example. The non-rigid transformation is applied on the upper left and lower right corner of each feature. This gives us the correct positions of both corners for each feature and determines the new width and height of each feature, see Figure 2. This feature transformation costs 8 additions per feature and is very efficient. This type of feature transformation is used in the learning as well as in the detection process.

bins of feature f_k	d_k	r_k	
$(-\infty, -0.42)$	-1	0	0
$\langle -0.42, -0.27)$	0	0.05	-0.37
$\langle -0.27, -0.19)$	-0.32	-0.07	-0.39
$\langle -0.19, -0.11)$	0.91	-0.09	0.02
$\vdots$	$\vdots$	$\vdots$	$\vdots$

TABLE 1: A part of a lookup table encoding the confidence and 2D translation updates for one feature in stage k. Bin sizes and values of the first regression column correspond to the function shown in red in the right image in Figure 6.

7 Implementation Details

Both the weak classifier d_k and the weak regressor r_k in stage k are implemented as a single lookup table (see Table 1 for an example); both confidence and alignment updates are read by a single lookup. The only additional cost during the classification for the alignment is its application to the features positions. For a rigid alignment by translation, application of the alignment means only two scalar additions per evaluated stage since both corners move identically and the feature's height and width are precomputed. In non-rigid deformations, we precompute an alignment lookup table, where each alignment vector $\mathbf{a} = [a_1 \dots a_m]$ is assigned with both position corners α_i, β_i for all features f_i, see Table 2. To speed-up the m-dimensional indexing, we compute a 1-dimensional index from $a_1 \dots a_m$ by bit-shifting. As a consequence, application of the non-rigid alignment costs four additions plus one access to the lookup table per stage. The size of the lookup table is reasonable. We usually use $m = 3$ and feature corners positions are integer values, that can be encoded by one byte. Denoting: D to be the number of discrete values of a_j and F to be the total number of features,

a_1	$\dots$	a_m	f_1		$\dots$	f_k	
			α_1	β_1		α_k	β_k
0		0	(3,7)	(5, 25)		(51, 61)	(12, 18)
0	$\dots$	0.1	(4,7)	(6, 26)	$\dots$	(51, 61)	(11, 18)
$\vdots$		$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$

TABLE 2: A part of a lookup table encoding corner positions of particular features for all possible alignments.

the size of the alignment lookup table is $4FD^m$ (e.g. for $D = 100$, $m = 3$ and $F = 300$, the size is 1.2GB).

8 Experiments

The results of the experiments demonstrate the importance of (non-rigid and rigid) LISA for SDP. Section 8.1 and 8.2 describes experiments with non-rigid LISA on *Annotated Faces in the Wild dataset* (AFW [38]) and *Car Semi-profile View dataset* (CSV). Our negative data consist mostly of Google street-view images without cars and faces in total amount of 15Gpx1. Experiments with rigid LISA are detailed in [22]. Section 8.4 evaluates performance of different regression functions.

We apply our feature alignment method on SDP similar to Waldboost [7]. We refer to our method applied on SDP as SDP+LISA, reimplementation of the alignment method proposed by Ali et al. [1] applied on SDP is referred to as SDP+Ali [1]. Besides that we also evaluate SDP+LISA+Ali [1]. This method allows to combine non-aligned features with features aligned by either the Ali [1] method, LISA method, or by both methods simultaneously. The alignment method of each particular feature is determined by boosting during the training stage. In addition to that, we also show baseline given by pre-trained, publicly available models: (i) Deformable Part based Models (DPM) [15] on a CSV and AFW dataset and (ii) Zhu's and Ramanan's [12] face detector on AFW dataset. In section 8.4 we also compare the precision of our alignment to the one of Zhu [12]. Section 8.5 discusses advantages, drawbacks, and limitations of the proposed method.

Ground truth annotations contain positions of several manually annotated keypoints. AFW has 7 keypoints and CSV has only 3 keypoints (upper left, lower right for bounding box and one vertical edge point), see Figure 7. All experiments are conducted with HoG features. Detection rates are summarized in Figure 8. Section 8.3 justifies the choice of HoG over Haar features by comparing detection rates on CSV dataset.

In all experiments where sequential decision process is involved, detected windows are filtered by Non Maxima Suppression (NMS). NMS is set to suppress all detections which have mutual coverage (union of bounding boxes divided by their intersection) bigger than 0.6. Criterion for correct detection is that the

Dataset/Method	SDP	SDP+LISA	SDP+Ali [1]	SDP+LISA+Ali [1]	Zhu [12]	DPM [15]
AFW	0.129	**0.041**	0.081	**0.047**	0.069	0.115
CSV human-view	0.038	**0.001**	0.042	0.003	–	**0.001**

Fig. 8: **Experiment summary:** Comparison of FN rates for fixed number of FP per 1Mpxl equal to 10^{-2}. Results corresponds to ROC curves in Figures 10 and 12

Method	SDP+LISA	SDP+Ali [1]	Zhu [12]	DPM [15]
Running time on VGA	33ms	41ms	17.2s	10.5s

Fig. 9: **Running time:** Comparison of running time on Intel core i7 Q700, 1.7GHz. Methods SDP+LISA, SDP+Ali [1], Zhu [12] run on single core, DPM [15] uses 4 cores. We used publicly available MATLAB/MEX implementation of DPM [15] and Zhu [12], while SDP+LISA and SDP+Ali [1] is measured on our C++ implementation with image resolution known in advance and the nearest neighbour image rescaling (ROC curves correspond to the linear interpolation when rescaling images). Time reported in [1] for AdaBoost+Ali is 30ms for 120×190 images and probably only one scale. Nevertheless, we reimplemented their feature alignment method and use it in our SDP, which is significantly faster than the AdaBoost used in [1] due to early rejections.

detected bounding box and ground truth bounding box have mutual coverage bigger than 0.3.

8.1 AFW dataset results

The AFW dataset is a publicly available dataset of face images obtained by random sampling of Flicker images. We use ground truth data which specify positions of 7 manually chosen keypoints (2 for each eye, 1 for the nose and 2 for the mouth corners). Note that a considerable amount of publicly available annotations have very low accuracy of keypoint positions (errors corresponding to 15% of the face size are not an exception). Even though such annotations make it difficult to train any accurate regression function (especially in L_2-norm), we use them directly. Since our approach does not contain any decision tree, which could split frontal and profile images, we focused only on frontal images captured within the range of approximately ± 45 degrees (in-plane and out-of-plane rotations).

Figure 10 shows ROC curves of SDP, SDP+LISA, SDP+Ali [1], SDP+LISA+Ali [1] trained on the first part of the AFW dataset. We can see that LISA outperforms Ali's [1] method. However, Ali's [1] method still yields significant improvement with respect to the pure SDP. It is also worth emphasizing that the SDP+Ali [1] method only needs annotated bounding boxes, while SDP+LISA also needs annotated keypoints to learn the regressors estimating non-rigid deformation. We can also see that the SDP+LISA+Ali [1] method, which combines features aligned by Ali [1] and LISA methods, has almost the same results as the SDP+LISA method. For comparative purposes the results of publicly available pre-trained models of Zhu's and Ramanan's [12] detector and Felzenszwalb's DPM detector [15] are shown. We do not retrain their detector and use publicly available model *p146_small* and the same DPM model from [12] using 10 mixtures learned for faces and kindly provided

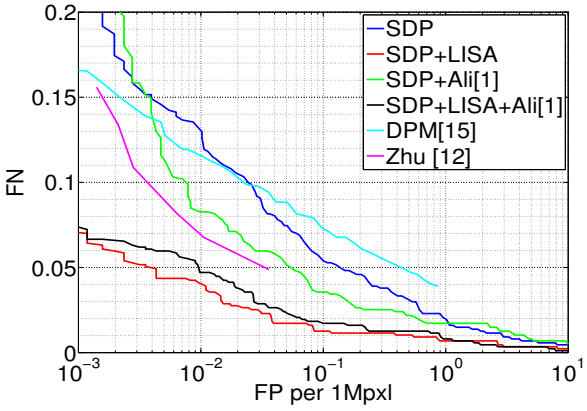


Fig. 10: **AFW-AFW ROC curves:** Comparison of different methods on the AFW dataset. SDP, SDP+LISA, SDP+Ali [1], SDP+LISA+Ali [1] trained on the first part of AFW. All methods were tested on the second part of AFW (images were captured by random sampling of Flicker images, therefore training and testing sets are independent). False positives are measured per 1 `Mpxl` of background data, false negatives per dataset.

to us by Xiangxin Zhu. Unlike our detector, Zhu's and Ramanan's detector is designed to detect high-resolution faces only (bigger than 80×80 while our detector works with 40×40 pixels). To make the comparison fair, we evaluated all methods only on faces which are bigger than 80×80.

Since AFW was captured by random sampling of Flicker images, training and testing sets are independent. Nevertheless, we also show that if we train on a subset of the BIOID dataset and a small fraction of the LFW dataset (faces already detected by Viola-Jones detector), we achieve almost the same results (see Figure 11 for the comparison of SDP and SDP+LISA trained on different datasets).

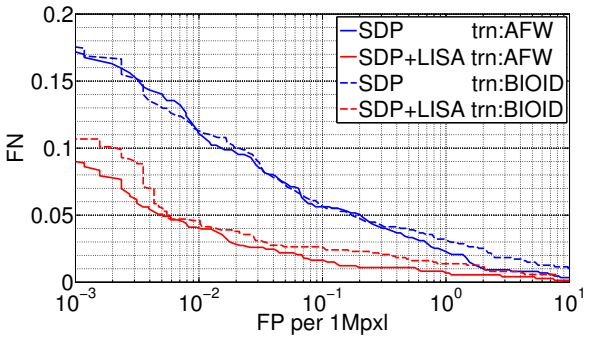


Fig. 11: **AFW-BIOID ROC curves:** Comparison of (i) SDP, SDP+LISA trained on first part of AFW (solid lines) and (ii) SDP, SDP+LISA trained on BIOID dataset (office environment images) and small fraction of LFW (dashed line). All methods were tested on the second part of AFW. False positives are measured per 1 `Mpxl` of background data, false negatives per dataset.

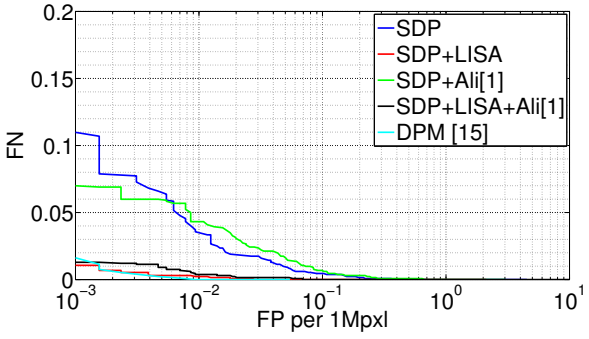


Fig. 12: **CSV ROC curves:** False positives are measured per 1 `Mpxl` of background data, false negatives per dataset.

8.2 CSV dataset results

The CSV dataset consists of 1600 images of cars taken from a semi-profile view ranging from almost pure rear view to the almost pure side view. Images are taken from a human view angle. We use 1200 images for training (training and validation sets) and 400 for testing. As the ground truth three points were marked: upper left, lower right, and a rear vertical edge point. The rear edge point corresponds to the imaginary intersection of the lower side windows line and the rear edge. The bounding box has a fixed aspect ratio. The parameters which were estimated by the regressors were selected by PCA, as described in section 6, for both modelled non-rigid deformations in the AFW and CSV datasets.

Figure 12 shows ROC curves of SDP, SDP+LISA, SDP+Ali [1], SDP+LISA+Ali [1] and publicly available DPM [15] pre-trained from VOC 2007 data. Figure 12 shows corresponding ROC curves. LISA outperformed Ali's [1] method. Actually, Ali's [1] method did not yield any significant improvement in the detection rate, because the deformations in this dataset probably could not be well modelled by the pose estimators of [1] (mainly estimating dominant edge orientation) since there is almost no in-plane rotation present in the CSV dataset. We can also see that DPM slightly outperforms SDP+LISA in human view-point images, however we still preserve real-time performance, see running time summary in Figure 9.

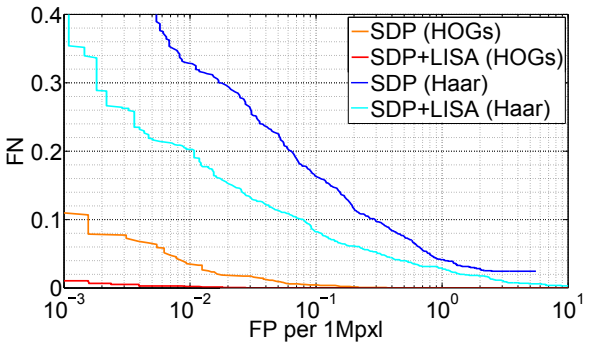


Fig. 13: **ROC curves (Haar vs HoG features):** False positives are measured per 1 `Mpxl` of background data, false negatives per dataset.

8.3 Haar vs. HoG features

We also demonstrate the influence of the choice of feature type. Figure 13 shows ROC curves of SDP and SDP+LISA methods evaluated on the CSV dataset captured from the human-view angle for (a) Haar features and (b) HoG features. While the relative improvement coming from using LISA is preserved, HoG features exhibit much better detection rates than Haars.

8.4 Regression Functions Evaluation

In this experiment we evaluate the performance of piecewise regression functions in all three variants of equation (5): (i) affine function with γ and λ, (ii) linear function with γ only and (iii) constant function with λ only (used in LISA). We learn the regression functions for different numbers of bins in combination with two types of feature space partitionings.

Here the regression functions are learned separately from the detector on their own features. In line 12 of the Learning Algorithm in Figure 5 multiple regressors are jointly learned at once. The number of jointly learned regressors is estimated automatically by observing the error on validation data. Five regressors are jointly learned on average. Therefore for performance evaluation, we also jointly learn 5 regressors.

As a criterion for selection of the best performing function and feature space partitioning we use the *mean regression error* (MRE) of the alignment parameters (selected by PCA) estimation with respect to the parameters computed from the deformed grids computed from ground truth annotations, depicted in

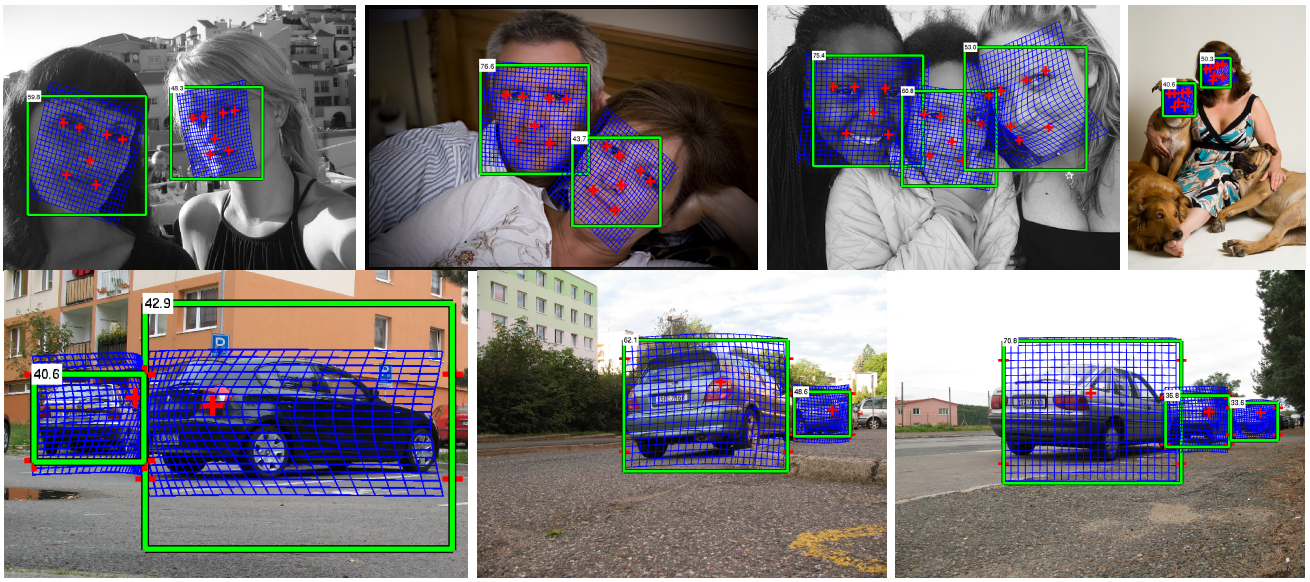


Fig. 14: **Detections:** with the SDP+LISA detector. Upper row shows some faces from AFW testing set. The last image in the upper row shows FP detection. The bottom row shows semi-profile cars from robot view angle on CSV testing images.

Figure 7. Each features' regressor estimates all $m = 3$ alignment parameters. In this section we denote r_j^k a part of the $j-$th regressor, which estimates single alignment parameter k. The MRE of the positive samples in the testing set is then computed as follows

$$\text{MRE} = \sum_{i=1}^{p} \left(\sum_{k=1}^{m} \left(\sum_{j=1}^{5} r_j^k(v_j^i) - \Delta t^{ki} \right)^2 /m \right) /p, \quad (12)$$

where p is the number of positive image samples, v_j^i is the $j-$th feature value of sample i and t^{ki} is the samples' $k-$th ground truth parameter.

The resulting MREs on testing data for all variants of tested regression functions dependent on the number of bins are depicted in Figure 15. The MREs are normalized by the initial MRE of the testing set when no alignment is applied. The results in Figure 15 correspond to non-rigid alignment estimation on the testing part of the LFW dataset. The results for other datasets are similar.

The important observation here is that MREs of the third function (piecewise constant - dark and light red bars in Figure 15) decrease very quickly with the growing number of bins. At approximately 15 bins it reaches the testing error of the first two types of functions with the slope parameter γ. Also the proportional partitioning variants (bars in light tones) perform better than the non-proportional ones (bars in dark tones). In our algorithm we use the *piecewise constant function* with the *proportional partitioning* (light red bars). It achieves a good alignment precision and is extremely fast to evaluate.

In the last experiment we evaluate the precision of estimated alignment and we compare our results to

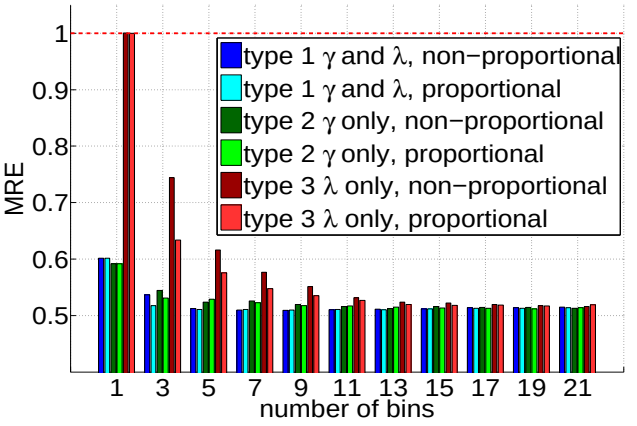


Fig. 15: **Mean Regression Error (Eq. 12) as a function of the number of bins for LFW alignment learned on 5 features.** Please note that piecewise constant function (dark and light red bars) quickly reaches the testing error of the first two functions, which use the slope parameter γ. Also note that the proportional partitioning of the feature space yields better results than the non-proportional one. MREs are normalized, with 1 being the testing set error when no alignment is estimated.

[12] on facial features. Alignment precision is evaluated on a testing part of the AFW dataset. Our model was trained on the training part of the AFW dataset. The same model was used to generate red curve in Figure 10. The publicly available model *p146_small* from [12] works with faces larger than 80×80. That is why we made a selection of images with faces appearing in larger resolution. We took the positive

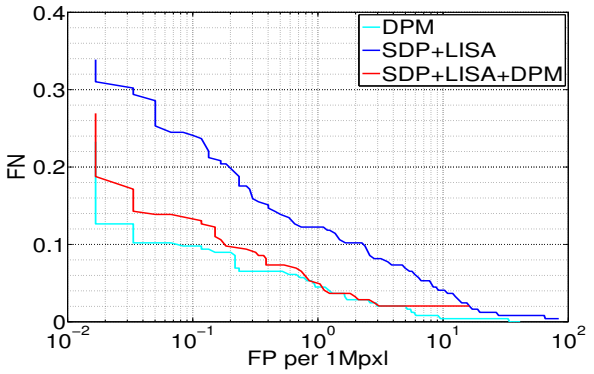


Fig. 16: **SDP+LISA+DPM pipeline:** Comparison of SDP+LISA, SDP+LISA+DPM and DPM on a hard face dataset with wide range of poses, illuminations and occlusions.

detections of [12], which corresponded to one of 7 frontal face models (out of 13). The remaining are side-view models, which do not contain all the facial features necessary for comparison with our model. From this selection we made an intersection of true positive detections of both our method and the one of [12] in order to evaluate the alignment on the exact same images. A total of 338 images from the testing set were selected for this experiment. The computed errors are Euclidean distances of estimated facial features positions from ground truth facial features positions relative to face size (to compensate for different face scales). 7 facial features were used: 4 eye corners, 1 nose tip and 2 mouth corners. The resulting mean error of [12] is 0.0513, median 0.0441 with variance 0.0012. I.e. for a face of size 100×100 pixels, there is a mean error of 5.13 pixels for each facial feature. The resulting mean error of our method is 0.0472, median 0.0410 with variance 0.0009. For the same face size, we achieve a lower mean error of 4.72 pixels for each facial feature.

8.5 Discussion

The proposed local interleaved sequential alignment improves the sequential decision process. The main competitors are the SDP itself and local pose estimators proposed by Ali [1], which may be combined with the SDP as well.

The SDP is favorable for its high detection speed, see running time comparison in Figure 9. On the other hand, it is known that the greedy learning suffers from lower generalization when compared to SVM based approaches like [12], [15]. Notice that in the previous experiments we used publicly available models of [12], [15] to show the baseline. When the DPM with 4 components [15] and SDP+LISA are trained on the same dataset and tested on a harder face dataset with wide range of poses, illuminations and occlusions, then the detection rate of DPM is indeed better, see Figure 16. To achieve both high speed of SDP+LISA and high detection rate of DPM, we propose a combined SDP+LISA+DPM pipeline. The SDP+LISA step reduces the number of possible sub-windows and the strong but slow DPM runs only on the remaining small fraction of all sub-windows. In this experiment, the SDP+LISA leaves only 15 sub-windows per 1MPxl image in average for additional DPM evaluation, while only 2% of true positives are rejected. Remaining 15 sub-windows are finally evaluated by the DPM in a negligible time.

LISA is based on regression functions, which sequentially compensate deformation of the object in the evaluated sub-window. The advantage of the regression functions is that the computational complexity grows only linearly with the dimensionality of the pose space. However, accurate regression is usually possible only for a limited range of local deformations.

Main drawbacks of the proposed method are: (i) inherently limited generalization of SDP methods, (ii) limited range of deformations and (iii) keypoint annotations needed for learning. The main advantages are: (i) high detection speed, (ii) better detection rate than other SDP methods, (iii) global object deformation is estimated as a side-product of the detection process.

9 Conclusion

We have proposed an efficient approach for aligning detection features with observed non-rigid object deformation in a real-time. The idea was shown on sequential decision process (SDP), where pre-trained features are successively evaluated in a detection window. The successive feature evaluation allows for efficient alignment estimation by pre-learned regressors during the detection process. The estimated alignment is directly applied to not yet evaluated features which significantly improves the detection rates.

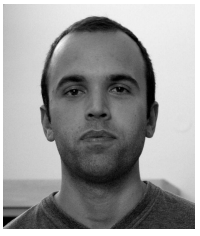
Acknowledgments

The first and second authors were supported by the Czech Science Foundation P103/10/1585 and Projects P103/11/P700 respectively. The third author was supported by EC project FP7-ICT-247870 NIFTi. Any opinions expressed in this paper do not necessarily reflect the views of the European Community. The Community is not liable for any use that may be made of the information contained herein.

References

[1] K. Ali, F. Fleuret, D. Hasler, and P. Fua, "A real-time deformable detector," *TPAMI*, vol. 34, no. 2, pp. 225–239, 2012.
[2] B. Babenko, P. Dollár, Z. Tu, and S. Belongie, "Simultaneous learning and alignment: Multi-instance and multi-pose learning," in *ECCV*, 2008.
[3] P. Viola and M. Jones, "Fast multi-view face detection," in *MERL*, 2003.
[4] J. Zhang, S. Zhou, L. McMillan, and D. Comaniciu, "Joint real-time object detection and pose estimation using probabilistic boosting network," in *CVPR*, 2007.

[5] C. Huang, H. Ai, Y. Li, and S. Lao, "Vector boosting for rotation invariant multi-view face detection," in *ICCV*, 2005, pp. 446–453.
[6] S. Li and Z. Zhang, "Flatboost learning and statistical face detection," *TPAMI*, vol. 26, no. 9, pp. 1112–1123, 2004.
[7] J. Šochman and J. Matas, "Waldboost - learning for time constrained sequential detection," in *CVPR*, 2005, pp. 150–157.
[8] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005, pp. 1–8.
[9] A. Mohan, C. Papageorgiou, and T. Poggio, "Example-based object detection in images by components," *TPAMI*, vol. 23, no. 4, pp. 349–361, 2001.
[10] P. Viola, M. Jones, and D. Snow, "Detection pedestrians using patterns of motion and appearance," in *ICCV*, 2003, pp. 734–741.
[11] A. Torralba, K. Murphy, and W. Freeman, "Sharing visual features for multiclass and multiview object detection," *TPAMI*, vol. 29, no. 5, pp. 854–869, 2007.
[12] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *CVPR*, 2012.
[13] F. Fleuret and D. Geman, "Stationary features and cat detection," *Journal of Machine Learning Research*, vol. 9, pp. 2549–2578, 2008.
[14] I. Kokkinos, "Rapid deformable object detection using dual-tree branch-and-bound," in *Advances in Neural Information Processing Systems*, 2011, pp. 2681–2689.
[15] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *TPAMI*, vol. 32, no. 9, pp. 1627–1645, 2010.
[16] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Machine Learning*, 1996, pp. 148–156.
[17] R. Schapire, "The boosting approach to machine learning an overview," in *MSRI Workshop on Nonlinear Estimation and Classification*, 2001.
[18] D. L. Shrestha and D. P. Solomatine, "Experiments with adaboost.rt, an improved boosting scheme for regression," *Neural Computation*, vol. 18, pp. 1678–1710, 2006.
[19] P. Bühlmann and T. Hothorn, "Boosting algorithms: Regularization, prediction and model fitting," *Statistical Science*, vol. 22, no. 4, pp. 477–505, 2007.
[20] L. Bourdev and J. Brandt, "Robust object detection via soft cascade," in *CVPR*, vol. 2, 2005, pp. 236–243.
[21] K. Zimmermann, D. Hurych, and T. Svoboda, "Improving cascade of classifiers by sliding window alignment in between," in *ICARA*, 2011, pp. 196–201.
[22] ——, "Exploiting features - locally interleaved sequential alignment for object detection," in *ACCV*, vol. 1. Springer LNCS 7724, 2012, pp. 437–450.
[23] P. Felzenszwalb, R. Girshick, and D. McAllester, "Cascade object detection with deformable part models," in *CVPR*, 2010, pp. 2241–2248.
[24] Y. Amit and A. Trouvé, "Pop: Patchwork of parts models for object recognition," *IJCV*, vol. 75, no. 2, pp. 677–692, 2007.
[25] M. Pedersoli, A. Vedaldi, and J. Gonzàlez, "A coarse-to-fine approach for fast deformable object detection," in *CVPR*. IEEE Computer Society, 2011, pp. 1353–1360.
[26] C. H. Lampert, M. B. Blaschko, and T. Hoffmann, "Efficient subwindow search: A branch and bound framework for object localization," *TPAMI*, vol. 31, no. 12, pp. 2129–2142, 2009.
[27] P. Dollár, R. Appel, and W. Kienzle, "Crosstalk cascades for frame-rate pedestrian detection," in *ECCV*, 2012, pp. 645–659.
[28] X. Cao, Y. Wei, F. Wen, and J. Sun, "Face alignment by explicit shape regression," in *CVPR*. IEEE Computer Society, 2012, pp. 2887–2894.
[29] H. Wu, X. Liu, and G. Doretto, "Face alignment via boosted ranking model," in *CVPR*. IEEE Computer Society, 2008, pp. 1–8.
[30] X. Liu, "Generic face alignment using boosted appearance model," in *CVPR*. IEEE Computer Society, 2007, pp. 1–8.
[31] K. Zimmermann, J. Matas, and T. Svoboda, "Tracking by an optimal sequence of linear predictors," *TPAMI*, vol. 31, no. 4, pp. 677–692, 2009.
[32] P. Dollar, P. Welinder, and P. Perona, "Cascaded pose regression," in *CVPR*, 2010, pp. 1078 –1085.
[33] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *Annals of Statistics*, vol. 28, p. 2000, 1998.
[34] R. Penrose, "A generalized inverse for matrices," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 51, no. 3, pp. 406–413, 1955.
[35] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, vol. 2, no. 6, pp. 559–572, 1901.
[36] P. Viola and M. Jones, "Robust real-time face detection," in *ICCV*, vol. 2, 2001, pp. 747–757.
[37] F. L. Bookstein, "Principal warps: Thin-plate splines and the decomposition of deformations," *TPAMI*, vol. 2, no. 6, pp. 567–585, 1989.
[38] M. Koestinger, P. Wohlhart, P. M. Roth, and H. Bischof, "Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization," in *First IEEE International Workshop on Benchmarking Facial Image Analysis Technologies*, 2011.



David Hurych received his MSc degree in computer science from the Báňská Technical University of Ostrava, Czech Republic, in 2007. Now he is a PhD student at Czech Technical University in Prague where he also works as a researcher. In 2008, during his PhD studies, he spent three months on an internship at the NII in Tokyo, Japan. He has published papers on learnable tracking methods, unsupervised incremental learning, fast object tracking and detection.

Karel Zimmermann received his PhD degree in cybernetics from the Czech Technical University in Prague, Czech Republic, in 2008. He worked as postdoctoral researcher with the Katholieke Universiteit Leuven (2008-2009). Since 2009 he has been a postdoctoral researcher at the Czech Technical University. He serves as a reviewer for major journals such as PAMI and conferences such as CVPR and ICCV. He received the best reviewer award at CVPR 2011 and the CSKI prize for the best PhD work in 2008. His current research interests include learnable methods for tracking, detection and robotics.

Tomáš Svoboda received his PhD degree in artificial intelligence and biocybernetics at the Czech Technical University in Prague in 2000. He spent three years as post-doc with the Computer Vision Group at the ETH, Zürich, Switzerland. He is currently deputy head of the Department of Cybernetics, FEE CTU in Prague. He has published papers on multicamera systems, omnidirectional cameras, image based retrieval, and learnable tracking methods. He is a regular member of PC of most important computer vision conferences and also serves as reviewer for major journals like IEEE PAMI and IJCV.