

---

# Part based object detection with a flexible context constraint

---

**Supporting object detection with easy context**

Diploma thesis by Karl Robert Biehl from Berlin-Wilmersdorf

June 2013



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Fachbereich Informatik  
Graphisch-Interaktive Systeme

Part based object detection with a flexible context constraint  
Supporting object detection with easy context

Submitted diploma thesis by Karl Robert Biehl from Berlin-Wilmersdorf

1. Reviewer: Prof. Stefan Roth, PhD
2. Reviewer: M. Sc. Thorsten Franzel

Date of submission: June 8, 2013

---

# Erklärung zur Diplomarbeit

## Declaration of academic honesty

Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I hereby declare that this diploma thesis has been created by myself. Information derived from work of others has been acknowledged in the text and bibliography. This diploma thesis has not been submitted to other universities or institutes in any form.

Darmstadt, June 8, 2013

---

(R. Biehl)

---

---

## Abstract

---

### **Part based object detection with a flexible context constraint**

*Supporting object detection with easy context*

by Karl Robert Biehl

This work describes an object detection system which integrates flexible spatial context constraints to improve detection performance. It allows spatial and scale deformation of the object relative to its context. The contextual model extends an existing deformable parts model and is trained on partially labeled data using a latent SVM. The approach can be applied to any object detection problem where the object class always exists in one typical image context, but the context can appear independently. A new scoring method is used to model the asymmetric relationship between object and context. Furthermore, the system enables the use of contextual non-maximum suppression, a context sensitive way to discard redundant detections. Trained on our combined dataset of dresses and persons, the system achieves a significant improvement in detection performance when compared with basic deformable parts models.

**Keywords.** spatial constraints, context constraints, clothing detection, object detection, deformable models

---



---

## Contents

---

<b>List of Tables</b>	<b>i</b>
<b>List of Figures</b>	<b>ii</b>
<b>List of Algorithms</b>	<b>iii</b>
<b>List of Nomenclature</b>	<b>iv</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Related work</b>	<b>6</b>
2.1 Clothing detection . . . . .	6
2.2 Spatial context in object detection . . . . .	6
2.3 Context rescoring in deformable parts models . . . . .	8
<b>3 Contextual model</b>	<b>9</b>
3.1 Deformable parts models . . . . .	9
3.2 Contextual model . . . . .	9
<b>4 Matching</b>	<b>13</b>
4.1 Original model . . . . .	13
4.2 Contextual model . . . . .	14
4.2.1 Defining a contextual scoring function . . . . .	14
4.2.2 Contextual model scoring . . . . .	16
<b>5 3D distance transform on score pyramid</b>	<b>19</b>
<b>6 Histogram of oriented gradients</b>	<b>21</b>
<b>7 Latent SVM</b>	<b>23</b>
7.1 Latent SVM with standard hinge loss . . . . .	23
7.2 Latent SVM with contextual loss . . . . .	25
7.2.1 Semi-convexity . . . . .	26
7.2.2 Semi-convexity through context regularization . . . . .	28
<b>8 Training contextual models</b>	<b>33</b>
8.1 Deformable parts model initialization . . . . .	33
8.2 Training algorithm . . . . .	33
8.3 Contextual model initialization . . . . .	34
8.4 Contextual model training . . . . .	35
<b>9 Implementation</b>	<b>37</b>
9.1 Framework version 4 . . . . .	37
9.2 Extended framework . . . . .	38
9.3 Feature extraction and backtracking . . . . .	38
9.4 Outlook: framework version 5 . . . . .	39
<b>10 Empirical results</b>	<b>41</b>
10.1 Training and Testing Data . . . . .	41
10.2 Object and context model training . . . . .	42
10.3 Contextual model initialization and training . . . . .	44
10.4 Performance evaluation metrics . . . . .	46
10.5 Post processing . . . . .	47
10.6 Performance results . . . . .	47
10.7 Qualitative evaluation of results . . . . .	48
10.7.1 Development of deformation costs . . . . .	49

---

10.7.2 Development object and context models from initial states . . . . .	50
10.7.3 Development of the context model's threshold . . . . .	53
10.8 Quantitative evaluation of results . . . . .	53
10.8.1 Contextual non-maximum suppression . . . . .	54
10.8.2 Performance of models using RegB objective function . . . . .	54
<b>11 Conclusion</b>	<b>55</b>
<b>12 Future work</b>	<b>56</b>
12.1 Detection performance . . . . .	56
12.1.1 Allowing for different context anchor positions . . . . .	56
12.1.2 Scale sensitive mixture model initialization . . . . .	56
12.1.3 Training optimization . . . . .	56
12.1.4 Increase bounding box size . . . . .	57
12.1.5 Integrating related work on grammar models . . . . .	57
12.2 Potential speed improvements . . . . .	57
12.2.1 Speed improvements for detection . . . . .	57
12.2.2 Speed improvements for training . . . . .	58
12.3 Improve and extend training set . . . . .	59
12.4 Add multilateral spatial constraints . . . . .	59
12.5 Practical use of the model with flexible context constraint . . . . .	60
<b>13 Bibliography</b>	<b>v</b>



---

**List of Tables**

---

1	Trained configurations and results . . . . .	48
2	Deformation costs results (contextual model 2.3) . . . . .	50
3	Deformation costs results (contextual model 2.4) . . . . .	50
4	Positive Loss development in training . . . . .	57

---

## List of Figures

---

1	Previous work that defines contextual model structure . . . . .	3
2	Overall model structure of $M_{contextual}$ in the case of one object and one context model. . . . .	11
3	Contextual matching process at a single scale . . . . .	18
4	3D distance transform on score pyramid . . . . .	19
5	Centered scale deformation . . . . .	20
6	Visualization of a HOG descriptor. . . . .	21
7	Plots of hinge loss(2D) and contextual loss (3D). . . . .	27
8	Annotation and segmentation tool . . . . .	41
9	Examples from the “dress” training set . . . . .	42
10	Fully trained two component dress deformable part mixture model. . . . .	43
11	Fully trained one component person deformable part mixture model. . . . .	44
12	Detection error tradeoff . . . . .	48
13	Detection error tradeoff with CNMS . . . . .	49
14	Dress model before and after training of the contextual model . . . . .	51
15	Person model before and after training of the contextual model . . . . .	52
16	Mockup of system to automatically detect and segment all clothing items a person is wearing. . . . .	60
17	High confidence contextual model setections . . . . .	61
18	Contextual model detections with CNMS . . . . .	61
19	Interesting contextual model detections . . . . .	62
20	False positive detections due to wrong dress size . . . . .	62

---

## List of Algorithms

---

1	Applying 3D distance transform to score pyramid . . . . .	20
2	Stochastic gradient descent with feature cache . . . . .	25
3	LSVM data-mining algorithm . . . . .	25
4	Training procedure . . . . .	34
5	Scale backtracking on score pyramid . . . . .	39

---

## Nomenclature

---

AP	Average Precision, page 46
CNMS	Contextual Non-Maximum Suppression, page 47
Context Model	Context models form the restricting part of a contextual model. A context model and its class is only used to restrict viable areas in an image where the object should be detected, page 9
Contextual Model	A contextual model consists of an object model and one or more context models, page 9
Contextual object hypothesis	The overall hypothesis of an object hypothesis taking into account other context hypotheses, page 14
DET	detection error tradeoff, page 46
FPPI	False positives per image, page 46
HOG	histogram of oriented gradients, page 2
LSVM	Latent Support Vector Machine, page 4
MR	Miss rate, page 46
Object Model	The object model is the core part of a contextual model. The parent contextual model's object class is equal to the object model's class, page 9
PCA	Principal Component Analysis, page 9
PR	precision vs. recall, page 46
SVM	Support Vector Machine, page 28

---

## 1 Introduction

---

Clothing detection in images is an area where current object detection methods fail to achieve good performance. However, detection of object classes like faces, pedestrians and cars has been targeted by many researchers. A typical benchmark for object detectors is the PASCAL Visual Object Challenge dataset. It includes several object classes which vary strongly in difficulty. While top of the line object detectors trained on bikes, cars, horses and people usually have good detection performance, other object classes like chairs, dogs, cows and plants tend to have lower results.<sup>1</sup> While not a part of this dataset, clothing object classes seem to be in this latter difficult category.<sup>2</sup>

Interestingly, pedestrian and people detectors fare relatively well.<sup>3</sup> In many real life situations clothes are mostly present when they are being worn by people. Thus, in most photos clothes should appear in the wearer's direct proximity. Furthermore, different types of clothes have different locations on the wearer's body, e.g. most clothes like trousers, shirts, dresses are centered horizontally on the body, whereas hats, shirts, belts, trousers, shoes all have distinct vertical positions on the wearer's body. To make use of these spatial relationships between wearer and clothing objects this work describes an object detector, that combines two deformable parts models by Felzenszwalb et al. [2010a], one representing the object, the other representing the context, where the context model enforces a flexible context constraint on the object model.

The idea in object detection to infer knowledge between instances of multiple object classes has been examined on several occasions already. Torralba et al. for instance introduced Boosted Random Fields (BRFs), which leverage boosting to learn the graph structure and locality potentials of a conditional random field (CRF).<sup>4</sup> They are training their model to "first detect easy (and large) objects, since these reduce the error of all classes the fastest. The easy-to-detect objects can then pass information to the harder ones"<sup>5</sup>. For example they infer the position of cars by detecting the road first, or detecting a computer screen, then the keyboard and ultimately the mouse.<sup>6</sup>

Assuming a perfect people detector and that clothing is only present when being worn, one could implement a model which detects clothes given the presence of a person. The resulting classifier would prevent false positive detections in image areas where no person exists. According to our assumptions if we know there is no person, there cannot be any clothes, therefore reducing the size of the clothing classification problem. Instead of having to look for each type of clothing in the whole image the possible locations are at best tightly constrained. This should lead to a higher precision compared to a classifier without this contextual constraint.

In this work we try to leverage these spatial constraints between clothes and the wearer – or in general between objects and their context. This should still be possible with an imperfect but sufficiently good people detector as is available today.

Our goal is defined by leveraging knowledge from the following assumptions:

### Contextual model assumptions:

There is an object class and a context class with the following relationship:

- Object instances can only exist in context instances.
- Context instances can exist with or without an object instance.

This leaves the question: why is detection of clothing so difficult? There are different possibilities:

**Taxonomy:** It is difficult to categorize different types of clothing items. One could differentiate between *top* and *bottom*, between *dress*, *shirt*, etc., or between *mini*, *maxi* or *evening dress*.

A practical taxonomy would be as coarse as possible without allowing too much inner class variability. Training data has to be obtained, so each category increases costs when using a supervised learning method. Increased inner class variability on the other hand makes classification for object detectors harder.

**Appearance:** Clothing items come in a variety of shapes, material and textures. First, clothes have no rigid shape, they partly gain their shape by the person wearing it. In this work the shape of clothes is defined by the shape the item portrays in a worn state. Most items of clothing cannot be correctly classified even by humans when just being dropped on the floor. The item's material influences reflection. Denim mostly has a matte appearance. Shining leather types on the other hand can exhibit large contrasts dependent on the light source and thus introduce edges which could impair many gradient feature based detectors. The same can be true for textures and prints. They often contain higher contrasts than the item relative to its background.

---

<sup>1</sup> See results of competition "comp3" in Everingham et al. [2012].

<sup>2</sup> The detection performance of deformable parts models by Felzenszwalb et al. [2010a] trained on our dresses dataset suggests this.

<sup>3</sup> See results for category *people* in Everingham et al. [2012] and results for person classifier trained on INRIA Person data set in Girshick et al. [2012a].

<sup>4</sup> Torralba et al. [2004, p. 2].

<sup>5</sup> Torralba et al. [2004, p. 8].

<sup>6</sup> See Torralba et al. [2004, p. 8].

---

**Occlusion:** Clothing items are usually worn in combination. Especially accessories like scarves or bags often occlude other items. Wearing a jacket on top of a dress might occlude the dresses' arm parts left and right, while still leaving the center and bottom visible.

This work does not focus on finding ways to solve these problems of clothing detection directly, instead this work focusses on using domain specific knowledge that could significantly prune candidate clothing locations in an image, and therefore simplify the object detection problem. This could reduce false positive detections as well as enable the object detector to directly focus on the smaller problem, responding to more specific details, which would have less meaning or could have been deceptive in a global scope.

This work extends the deformable parts object detector by Felzenszwalb et al. with knowledge about the object's spatial position relative to its context. An obvious choice as the context for clothing items is the person wearing them. As noted above, using the relative spatial position one can constrain object detections to certain areas near the context. Context is provided to the object detector in the form of a priori knowledge from a simpler and more reliable pedestrian detector. In the following, the term object will be used to describe the class that should be detected, and context will be used to describe the class that is used to gain a priori knowledge.

Ultimately this approach could lead to a model where one context detector is responsible for classifying rough obvious features and one object detector can focus solely on fine details. In practice this could mean that the context model detects the person, and different clothing detectors primarily need to discern between themselves instead of themselves and an arbitrarily large set<sup>7</sup> of background objects.

## Approach

The decision to use deformable parts models by Felzenszwalb et al. as the framework for this thesis is based on its

*“strong low-level features based on histograms of oriented gradients (HOG)[,]  
efficient matching algorithms for deformable parts models (pictorial structures) [and]  
discriminative learning with latent variables (latent SVM).”<sup>8</sup>*

Additionally, Felzenszwalb et al. have an emphasis on building flexible grammar based models. Grammar based models are relatively flexible and modular constructs of terminals (filters) and non-terminals (e.g. rules).<sup>9</sup> Their model discerns two types of rules, deformation rules, which allow for spatial uncertainty, and structural rules, which are used to aggregate scores from multiple deformation rules. Such a grammar based model offers a high grade of modularity and extensibility. The base grammar can be extended with additional rules to model relationships between object and context.

The deformable parts models are grouped in mixture models as components. Each component of the mixture model contains a root filter – determining the bounding box of the object – and several smaller parts which have a higher resolution than the root filter. Each part is spatially connected to the root filter by a standard offset in  $x$  and  $y$  direction. Possible deformations are realized through “spring-like connections”<sup>10</sup>. Part deformations are penalized by a weighted cost function growing quadratically. Examples of trained deformation costs can be seen in Figure 1 visualized as ellipses. The deformable parts model by Felzenszwalb et al. is partly based on the Dalal-Triggs model, which achieved good object detection performance with HOG features.<sup>11</sup> As Felzenszwalb et al. extended the Dalal-Triggs model with deformable parts, this work extends the deformable parts model with a flexible context constraint.

## Flexible context constraint

A context is “flexible” in two ways:

1. Flexibility through a spring-like deformation model. Similar to parts in the deformable parts based models the context can deform (move) from its mean location relative to the object. This is being moderated by deformation costs, which determine how far the context can move from its common location.
2. The context constraint is also flexible in the way, that its influence on the object detector can be mediated by the training algorithm. If a context does not help the object detector much, its impact into the overall score will be diminished, if the context is a significant aid in finding the object, the impact on the overall score will be increased.

Based on this grammar based model a meta model is being described which extends the model architecture by another level. The meta model contains multiple deformable parts models (one object class model, one or more context class models). The resulting model is then used to classify the instances of the object class which appear in conjunction with

---

<sup>7</sup> Rather than saying “infinitely” large set, since the amount of background objects is determined by the training data.

<sup>8</sup> Felzenszwalb et al. [2011]

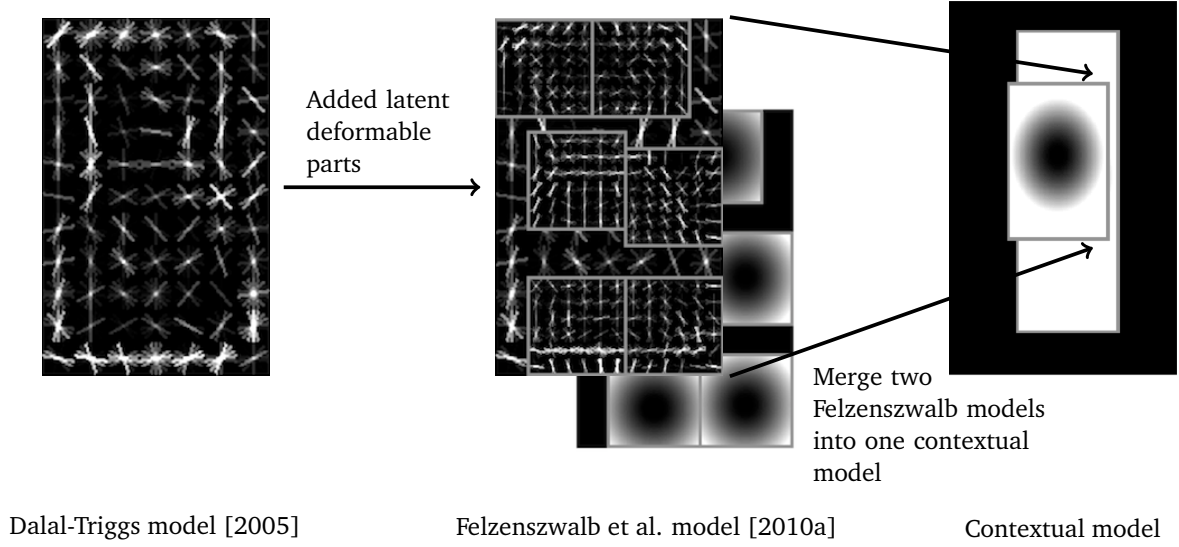
<sup>9</sup> See Felzenszwalb and McAllester [2010, p 1].

<sup>10</sup> Felzenszwalb et al. [2010a, p. 1].

<sup>11</sup> See Felzenszwalb et al. [2010a, p. 2], Dalal and Triggs [2005].



instances of the context class. From now on meta models will be referred to as *contextual models* and the deformable parts models by Felzenszwalb et al. as *base models*.



**Figure 1:** How previous works define the contextual model structure. Development of the model by Dalal and Triggs [2005] with extensions by Felzenszwalb et al. [2008, 2010a] which is used in the model with a flexible context constraint proposed in this thesis.

The spatial relationship between object and each context is analogous to the way the relationship between root filter and parts inside the deformable parts models is modeled. The deformable parts model is a recursive system of objects containing parts, which are themselves objects.<sup>12</sup> Now we have a contextual object model, which contains object and context models, which in turn contain parts. Therefore, context model deformation can be compared with part deformation, however, it is extended by scale deformation. Scale deformation should make the contextual model robust enough to, for example, detect both smaller and larger bags being worn by a person. This is one of the bigger changes to the model, affecting both grammar and implementation.

### Contextual Score

After defining the model architecture by extending the grammar a new scoring method is being developed. Two possible scoring functions for merging object and context scores are being discussed.

One could either multiply both scores, which is complicated with SVM scores<sup>13</sup>, or add them, just as the part scores are being summed up in the deformable part-based models<sup>14</sup>. Multiplying two scores between 0 and 1 only produces a high result if both scores are high, which assures that both object and context have been detected. If one score is low, the other cannot easily compensate. This aligns with our assumption that the object requires a context to exist in. By adding both scores one could compensate the other one completely if it is just high enough. E.g. that could mean that a very high context score could counterbalance a low object score and thus lead to a false positive. Instead a hybrid approach is being chosen. A scoring function which adds the thresholded context score to the object score, making sure that negative (below the threshold) context score penalizes the object score but a high context score does not impact the result. So in the contextual model an example is scored by combining the object's score and the context's score in the following function which will further be referred to as "conditional-sum":

$$\text{score}_{\text{contextual}}(\text{objectscore}, \text{contextscore}) = \text{objectscore} + \min(t, \text{contextscore}), \quad (1)$$

where  $t$  is the threshold at which the context score is being capped. A standard value is  $t = 0$ , where positive context scores have no impact, but negative context scores reduce the overall score. This conditional-sum function reflects the assumption that the object can only exist in its context, but the context can exist on its own. Choosing  $t$  offers flexibility and discretion to the SVM as to how much impact the context score is allowed to have.

<sup>12</sup> See Felzenszwalb and McAllester.

<sup>13</sup> Since we are working with SVM scores one would have to map them to a  $[0,1]$  interval first using sigmoids.

<sup>14</sup> See Felzenszwalb et al. [2010a, p. 7].

---

For root and part filter descriptors, HOG is used. These descriptors have been introduced by Dalal and Triggs and achieved superior results for human detectors when compared with other descriptor methods. We think that due to our primary goal of creating a model to improve detection of clothes worn by people HOG descriptors should be a very good fit. Also Felzenszwalb et al. show that good results can be achieved using HOG on many different object classes.<sup>15</sup>

## Training

Felzenszwalb et al. use what they call a latent SVM (or LSVM) to train their model with partially labeled data. Their LSVM approach allows latent variables like part configurations (e.g. their position) to be trained automatically.<sup>16</sup> For contextual models a new objective function defining the optimization problem has to be defined based on the new scoring method. An optimization problem has to be semi-convex so it can be trained using a latent SVM. Semi-convex means that the optimization problem is convex on its negative examples, but not on its positives. Felzenszwalb et al. show that confining the amount of possible part configurations to one per training algorithm iteration makes the optimization problem convex.<sup>17</sup>

When developing the objective function to train the contextual model it became clear that just using the same partially labeled data as Felzenszwalb et al., consisting of just bounding boxes for the object class, the optimization problem would not be convex. Context scores would grow until all positive examples would no longer be penalized by the context.<sup>18</sup> Thus, in an extreme case every detection window on an image could be classified as a positive context.

To prevent this behaviour the context has to be regularized. This requires additional class labels to be added to the training data for the context. Both object and context appearances have to be annotated in the training data, their respective class labels are  $y$  and  $\hat{y}$ . The training data is divided into two sets: positive object examples  $P$  and background images (negative examples)  $N$ . Positive and negative context examples are spread through both sets. This is due to the asymmetric relationship between object and context. Recall that objects always appear in context, but context can appear alone. This means that also context appearances in  $N$  have to be annotated.

Two ways to regularize the context are being described. The first way is to handle all different cases of positive and negative object and context class labels differently in the loss function. The other solution is the addition of another term to the objective function, which in parallel trains the context classifier. Both are discussed and validated in section 10 and only the first one leads to satisfactory results. Also the semi-convexity property of the new optimization problem – a requirement for the latent SVM – is being shown.

We further describe the initialization which is an important aspect of training and has a potentially large impact on the final model. The two-step LSVM training algorithm is susceptible to local minima.<sup>19</sup> Both the object and context part-based models are being initialized and trained beforehand according to the work by Felzenszwalb et al. and merged to a contextual mixture model. The initial spatial relationships between the models are derived directly from the training data.

The LSVM training algorithm uses data-mining of “hard” negative examples, which violate the SVM loss function the most. This mechanism is of the utmost importance for training the contextual model. Since the assumption behind the contextual model is to find objects in their context, one wants to discard as many false examples through classifying the context alone, and thus softly restrict the object classifier to the area defined by the context. If the context classifier is doing a good job, most hard examples will fall into the area which is not penalized by the context. In this case the training algorithm can focus on training the object detector to classify examples, which the context detector did not penalize.

The contextual model has then been implemented on top of the deformable parts model framework version 4 provided by Felzenszwalb et al.. Rules for modeling the new scoring function and scale deformation have been added to the framework. Also the new objective functions for the contextual model have been added to the training algorithm implementation. Details like scale offsets of values other than a doubling or halving the size<sup>20</sup> and the fact that now multiple class labels are present in the training data lead to significant changes to the codebase.

To implement scale deformation as an addition to deformation in  $x$  and  $y$  dimension on a feature pyramid, we created a process to convert the feature pyramid to and from a grid and apply distance transform in three dimensions. Distance transform can only be applied across scale levels if all levels have the same dimensions. This process introduces rounding error, making backtracking the source location for feature extraction more difficult.

---

<sup>15</sup> See Felzenszwalb et al. [2010a, p. 19].

<sup>16</sup> See Felzenszwalb et al. [2010a, p. 2].

<sup>17</sup> See Felzenszwalb et al. [2010a, p. 8].

<sup>18</sup> This problem is being described in detail in section 7.

<sup>19</sup> See Felzenszwalb et al. [2010a, p. 11].

<sup>20</sup> In the original framework it is only possible to have an offset of a full scale octave. “The scale sampling in a feature pyramid is determined by a parameter  $\lambda$  defining the number of levels in an octave. That is,  $\lambda$  is the number of levels we need to go down in the pyramid to get to a feature map computed at twice the resolution of another one.” Felzenszwalb et al. [2010a, p.5].

---

## Results

Using this extended framework multiple experiments with different configurations of the contextual model have been executed. Due to time constraints the experiments were confined to one dataset for the “dress” object class with the “person” context class. The training dataset holds 1974 positive examples and 1614 negative images. The test dataset contains 1201 images with 846 positive examples.

In some of the experiments all context model parameters except the bias have been fixed. This was done to keep the context model independent from the object model, so that it could be reused with other object classes sharing the same context. The initial person context model is trained using the INRIA Person dataset.<sup>21</sup>

Based on the object class “dress” this work examines different ways of configuring and training the proposed part based model with flexible context constraint.

This work shows that the deformation parameters which model spatial relationship reach significant weights. This tells us that they help classification of our training data and have been chosen by the training algorithm to penalize negative examples.

Using fixed context model parameters and only training the context deformation and object model parameters, detection performance of the contextual model is comparable with that of the original deformable parts model. The INRIA Person deformable parts model used as the context model to initialize the overall contextual model, had a low performance, when tested on the context bounding boxes in our training data. This shows us that the contextual model is very robust against bad context models and can still leverage low performance classifiers to increase precision in low recall situations.

When training all parameters, the development of the object and context model show clear signs of adapting to their new task. The object model assumes features, that it did not respond to before being constrained by a context. Additionally, the context model also assumes rough features of the object. The resulting contextual model outperforms the original deformable parts models on our training set.

Furthermore, this work describes a secondary use of the context constraint – the application of non-maximum suppression to the context detection windows. With this method all but the best scoring detections sharing the same context detection window, or overlap it by at least 50%, are getting discarded. This is based on the domain knowledge, that one person can only wear one dress. This step decreases false positive detections significantly.

---

<sup>21</sup> See Dalal [2005], trained by Felzenszwalb et al. [2011]

---

## 2 Related work

---

As noted in the above the contextual model is based on the deformable parts models by Felzenszwalb et al. [2010a] which we want to build upon to leverage contextual spatial constraints for objects. Using a part-based approach seems important because of the deformable nature of clothes. There are other approaches which can be used to automatically learn parts of objects like Schnitzspan et al. [2010] which has a much more flexible understanding of parts. Schnitzspan et al. allow for “flexible [part] shapes and sizes”<sup>22</sup> and can learn arbitrary part topologies instead of being restricted to a star-shaped part constellation.<sup>23</sup> The decision to go with the former has been based primarily on their approach to use extendable grammar based models which gives reason to expect that an extension of their existing framework is feasible. The main goal is to improve detection rate for clothing being worn by people. Therefore, this section is being divided into work related to fashion and clothing detection and the use of context in object detection.

---

### 2.1 Clothing detection

---

Fashion and clothing in computer vision has received a lot of research. In the fashion industry research has mostly targeted retrieval through similarity matching, e.g. via shape, texture, patterns, which is used in fashion search engines like *empora.com* or *like.com* (now Google Shopping). These approaches are very practical and mostly work with product photos with very simple backgrounds.<sup>24</sup> There are examples of clothing segmentation in more complex images though, like Hu et al. [2008]. Some even use clothing segmentation to recognize people.<sup>25</sup>

The work by Bertelli et al. [2011] focusses on object segmentation. They also work with the object class dresses, and use a hybrid system of object detector and color models. Their example images contain relatively simple backgrounds, mostly solid or smooth gradients in the background, and some people standing around, sourced from professional product photography. Their innovation lies primarily in the combination of top-down and bottom-up segmentation approaches. Therefore, they use simple, or out-of-the box object detectors – e.g. also the deformable parts models by Felzenszwalb et al. [2010a]. Therefore, this can be seen as an extension to object detectors.<sup>26</sup>

The detection and classification of clothes in real world images has received far less attention. There is an approach by Liu et al. using real-world photos of people wearing clothes with complex backgrounds to retrieve images from a product database through parts alignment.<sup>27</sup> In their work parts are detected by a part-based human detector with 20 upper-body parts and 10 lower-body parts. Unlike in our work they use a large set of features including HOG, Local Binary Patterns, color histogram and skin descriptor features.<sup>28</sup> Their input images are very similar to the ones used in our training set – street style photos with upright people and complex backgrounds. Since their approach works through part alignment they rely on a distinctive human shape and images of products with people wearing them. In that way they are also leveraging the human context. This approach does not, however, target an explicit object detection.

---

### 2.2 Spatial context in object detection

---

Using the spatial configuration of objects to improve the detection performance has already been the subject of much research. There are different approaches on how to model relationship between objects and their context. Jahangiri et al. [2010] propose a “conditional random field model for labelling parts of building scenes”<sup>29</sup>. Others propose post-processing methods which can be applied to any object detection model. Rabinovich et al. [2007] propose a conditional random field (CRF) post-processing framework, which maximizes object label agreement according to contextual relevance. Their method can either be trained using labeled training data, or by querying the search engine of Google to determine interconnections between categories.<sup>30</sup> Their model only incorporates coarse spatial layout information.<sup>31</sup> Another contextual post-processing method is part of the work about deformable part based models by Felzenszwalb et al. [2010a] and will be described at the end of this section.

A different approach is to learn spatial context between “stuff” and “things”. Stuff represents types of objects of “amorphous spatial extent”, which could be more easily defined by regions than by detection windows, which is a good method

---

<sup>22</sup> Schnitzspan et al. [2010, p. 2]

<sup>23</sup> See Schnitzspan et al. [2010, p. 2]

<sup>24</sup> See for example for image similarity Aysal and Heesch [2009], for object segmentation in (fashion) product photos Jahangiri and Heesch [2009].

<sup>25</sup> See Gallagher and Chen [2008]

<sup>26</sup> See Bertelli et al. [2011, p. 2f].

<sup>27</sup> See Liu et al. [2012, p. 3330]

<sup>28</sup> See Liu et al. [2012, p. 3334]

<sup>29</sup> Jahangiri et al. [2010, p. 6]

<sup>30</sup> See Rabinovich et al. [2007, p. 4f].

<sup>31</sup> See Rabinovich et al. [2007, p. 3].

---

define the location and extent of things.<sup>32</sup> Their method “automatically groups regions based on both their appearance and the relationships to the detections in the image”<sup>33</sup> without the need of an explicit training set, and helps improve detection performance over other detectors.<sup>34</sup>

In this work we use a predetermined relationship of inferring a hard object class “dress” from an easier one “person”. This is similar to the approach by Torralba et al., who want to improve scene understanding by inferring knowledge gradually from easier objects to harder objects – which in some cases un-identifiable on their own, even for humans. In one example they successfully classify small blobs of pixels as computer mice through inference from keyboard and computer screen detections.<sup>35</sup> They note that CRFs have problems “capturing important long distance dependencies between whole regions and across classes”<sup>36</sup>. Instead Torralba et al. propose Boosted Random Fields (BRFs), where the graph structure is learned “by using boosting to select from a dictionary of connectivity templates (derived from labeled segmentations)”<sup>37</sup>, and also learning local evidence potentials.<sup>38</sup> The resulting dense graph can pool information from large regions of the image, instead of just having connections between pixels.<sup>39</sup>

Another approach which rather targets multilateral spatial constraints has been proposed by Desai et al. [2011]. In their work Desai et al. introduce “inhibitory intra-class constraints (NMS) and inhibitory inter-class constraints (Mutual Exclusion) in a single unified model along with contextual cuing and spatial co-occurrence”<sup>40</sup>.

The idea to include spatial cueing into a model as a way to enhance detection results is a core aspect of the model proposed in this work. There are differences between our flexible context model and their multi-class model in how the spatial cueing aspect is being modeled.

While Desai et al. [2011] focus on multi-class detection and all their unknown bilateral relations<sup>41</sup> we focus on a “known” context and only model the relations between object classes and one context class. For  $N$  classes Desai et al. [2011] have  $N^2$  relations, while we only have  $N$  relations. While making our model less complex it is also less flexible in the way that the model can learn which objects in a scene have any relevance to the object one wants to detect. So with our flexible context model one could encounter the situation that the context class which was chosen by hand is completely irrelevant to the objects. Therefore, this work relies heavily on domain knowledge in the sense that to be able to choose a context class one must be sure that the object in question only appears in the chosen graphical context. Accordingly we focus on the domain of worn clothing, where a person provides the context and the different types of clothing are the object classes.

To model the spatial relationships between objects they define a fixed set of canonical relations like “above, below, overlapping, next-to, near, and far”<sup>42</sup>. For each object class pair weights for these relations are being trained. That means that the spatial relations by Desai et al. [2011] are modeled in a relatively coarse manner. Our model follows a different approach. Instead of using a histogram and sampling the different relations into bins we model the spatial relations between model and context analogously to how spatial relations and deformation is designed in the deformable parts model by Felzenszwalb et al. [2010a].

An advantage of using offsets and distance transform based deformation is that on the one hand spatial relations can be very precise and on the other hand are only limited by the model’s resolution. In this case the deformable parts model uses histogram of gradient features with a bin size of 8 meaning that the model’s resolution would be 8 times less than the image resolution. This is still a lot finer-grained than the canonical relation representation by Desai et al. They also use a binary vector to store whether or not a relation is being satisfied.<sup>43</sup> The deformable parts model on the other hand uses quadratic deformation costs to penalize unnatural part locations.<sup>44</sup> Common to their and our approach is the promise that modelling precise spatial relationships between object classes improves classification performance. One could view object classes in our model as optional parts of the context class, thus adding another hierarchy level to the existing model structure.

E.g. dresses, trousers, hats, shoes are being worn by people and become parts of them. While the flexible context model assumes that objects can only appear in their context – a dress can only appear when being worn by a person – it is

---

<sup>32</sup> See Heitz and Koller [2008, p. 1]

<sup>33</sup> Heitz and Koller [2008, p. 1]

<sup>34</sup> See Heitz and Koller [2008, p. 13]

<sup>35</sup> See Torralba et al. [2004, p. 9].

<sup>36</sup> Torralba et al. [2004, p. 3].

<sup>37</sup> Torralba et al. [2004, p. 3].

<sup>38</sup> See Torralba et al. [2004, p. 3].

<sup>39</sup> See Torralba et al. [2004, p. 2].

<sup>40</sup> See the tabular visualization in Desai et al. [2011, Fig. 6]

<sup>41</sup> Desai et al. [2011, p. 2]

<sup>42</sup> Desai et al. [2011, p. 3]

<sup>43</sup> See Desai et al. [2011, p. 3]

<sup>44</sup> See Felzenszwalb et al. [2010a, p. 6].

---

different the other way around, the context can appear with or without any associated object. A person can be naked, wearing other clothes like T-shirts, but not mandatorily a dress, etc. Hence. objects in our case must be seen as optional parts of their context. The non existence of the context can disprove the objects existence but the context's existence does not prove the objects existence.

This comes at a cost that an object can only have one mean location relative to their context. In the multi-class model by Desai et al. allows for multiple spatial constraint relations, e.g. to have higher weights on the canonical relation above and below the object at the same time, and lower weights on the relation next-to. While this makes sense to model relations between arbitrary classes it might not be needed in all cases. In the domain of clothing most item classes have a fixed place relative to the wearer's body. However, there are exceptions, loosely worn accessories like bags tend to be worn in the hip region, but they can be located either right, left or in front of the person holding it.

While there are similarities, Desai et al. [2011] focusses on a different problem than this work. That being said one could imagine applying their canonical relations on top of this flexible context model. In the domain of clothing this would make a lot of sense. There are rather obvious relations between some clothing types. E.g. trousers are rarely worn together with dresses. Also a person usually only wears one of most clothing items. So rather than being redundant the flexible context model could be extended with the work by Desai et al..

---

### 2.3 Context rescoring in deformable parts models

---

Felzenszwalb et al. [2010a] also propose a method that uses contextual information to improve results. In their case it is a post-processing step, that rescores detections. However, the approach is very different. They use global contextual information provided by “a set of detections obtained by using [...] different models [...] for different object categories”<sup>45</sup>. A detection is then rescored by a classifier which receives the input vector  $g = (\sigma(s), x_1, y_1, x_2, y_2, c(I))$ , where  $\sigma(x)$  is a renormalizing function,  $x_1, y_1, x_2$  and  $y_2$  define the detection's bounding box (which is normalized by the size of the image) and  $c(I)$  is the image context vector containing the best normalized scores for each different model  $\sigma(s_k)$ , where  $s_k$  is the highest score of the  $k$ -th model in the image  $I$ . The rescoring classifier itself is trained on images with bounding boxes of multiple object classes.<sup>46</sup> Such an approach would possibly be a useful addition to the model when multiple clothing categories are detected on the same image.

---

<sup>45</sup> Felzenszwalb et al. [2010a, p. 15].

<sup>46</sup> See Felzenszwalb et al. [2010a, p. 15].



---

### 3 Contextual model

---

In the following we describe the contextual model. Since it is based on the deformable parts models by Felzenszwalb et al. [2010a], we start by recapping its architecture and then describe how the contextual model extends the existing model.

---

#### 3.1 Deformable parts models

---

In a deformable parts model, an object is represented as a set of filters, one root filter  $F_0$  covering the whole object, and  $n$  part filters  $F_i (i = 1 \dots n)$  of higher resolution covering smaller object parts. The complete model can be specified as an  $(n + 2)$  tuple.

$$M = (F_0, P_1, P_2, \dots, P_n, b), \quad (2)$$

where

$$P_i = (F_i, v_i, d_i) \quad (3)$$

$$v_i \in \mathbb{R}^2 \text{ (offset relative to root filter)} \quad (4)$$

$$d_i \in \mathbb{R}^4 \text{ (cost of deviating from offset } v_i) \quad (5)$$

$$b \in \mathbb{R} \text{ (a normalization to allow for mixture models)} \quad (6)$$

Each part is defined by a 3-tuple, the part's filter  $F_i$  and parameters to model their spatial relation to the root filter.

Parameter  $v_i \in \mathbb{R}^2$  can be thought of as the spatial prior. It is the mean location relative to the object's bounding box and therefore the root filter  $F_0$ . The model also allows for deformable parts. Felzenszwalb et al. are using a springlike model. A part can move from its mean position but it is being penalized. There is a deformation cost vector  $d_i$  for each part  $i$ . The parameter  $b$  represents the SVM bias and normalizes the model w.r.t. other models in a mixture model.

The filters are using histogram of oriented gradients descriptors, where Principal Component Analysis (PCA) has been applied<sup>47</sup>, which will be described in section 6.

Multiple models of like  $M$  can be grouped into a single mixture model to be able to detect multiple classes. As noted before  $b$  is being used to normalize each model's score to make it comparable. Usually object classes that are perceived by humans as an 'atomic' class are being divided into multiple object classes by orientation (e.g. left and right facing) and aspect ratio (often similar to viewpoint).<sup>48</sup> But of course any group of classes can be merged into a mixture model. This is part of the model initialization which will be described in section 8.

---

#### 3.2 Contextual model

---

Now we describe how we extended the deformable parts models to introduce the flexible context constraint. To improve detection of objects which typically reside in context we introduce additional spatial constraints by nesting multiple models. Our extended model spans a hierarchy of deformable parts models. Here we introduce four terms to describe the extended model and its sub-models:

- The term *contextual model* is being used to describe the overall model. This model consists of two parts, the object model and the context model.
- The *object model* is a deformable parts model like described in Felzenszwalb et al. [2010a]. The object class is the class one wants to detect with the contextual model.
- The *context model* is also a deformable parts model like the object model. The context class is the class of the context one assumes the object class to be in. In the end the contextual model is using the context model to penalize (preferably false) object detections outside of the detected context locations.

A contextual model can be constructed by combining one object model  $M_0$  and  $m \in \mathbb{N}$  context models  $C_j$  and adding information about their spatial relation.

The contextual model is defined by

$$M_{\text{contextual}} = (M_0, C_1, \dots, C_m) \quad (7)$$

---

<sup>47</sup> See Felzenszwalb et al. [2010a, p. 4,13].

<sup>48</sup> See Felzenszwalb et al. [2010a, p. 12].

where

$$C_j = (M_j, V_j, S_j, D_j) \quad (8)$$

$$M_j = (F_{j,0}, P_{j,1}, P_{j,2}, \dots, P_{j,n_j}, b_j) \quad (9)$$

where

$$n_j \in \mathbb{N} \text{ is the amount of filters in } M_j \quad (10)$$

$$V_j \in \mathbb{R}^2 \text{ (offset relative to context filter)} \quad (11)$$

$$S_j \in \mathbb{R}^1 \text{ (scale relative to context filter)} \quad (12)$$

$$D_j \in \mathbb{R}^6 \text{ (cost of deviating from offset } V_j \text{ and scale } S_j) \quad (13)$$

where  $M_j$  are instances of  $M$  as defined in (2).

It is obvious that the contextual model structure bears a big resemblance to the deformable parts model structure. Basically an additional part hierarchy has been added to the system, however, there are significant differences.

The spatial prior  $V_j$  is now accompanied by a scale prior  $S_j$ , since the object and context models can be trained in different resolutions and sizes.  $S_j$  is being derived from the mean scale difference between object and context model and represents this resolution change. Due to the fact that features are generated from a gaussian pyramid, a model's resolution defines its scale. With  $S_j$  it is possible to map the response of the context models to the object model.  $S_j$  represents the number of octaves, therefore the scale difference is  $2^{S_j}$ .<sup>49</sup>

Interestingly this already happens in the model by Felzenszwalb et al. between root filter and part filters. If the gaussian pyramid, which was used to generate the features, has  $\lambda$  steps per scale octave (a 2x scale change) then part filters are positioned  $\lambda$  steps below the root filter. That means the filter's resolution doubles while still covering the same area on the root filter's level. Adhering to the variable naming convention of capital letters for  $M_{contextual}$ 's spatial parameters and lowercase letters for  $M$ 's spatial parameters one could say that the counterpart to  $S_j$  also exists in  $M$  as a fixed parameter  $s_i = 1$ .<sup>50</sup> In more recent work Girshick et al. extended their model to allow for a range of other integer values for  $s_i$ .<sup>51</sup>

In our case  $S_j$  is a real number. As a feature pyramid only has  $\lambda$  levels per scale octave this value has to be rounded at runtime to the closest level difference. There are also implications for the training algorithm implementation, since scale jumps with  $S_i \in \mathbb{R}$  instead of a confined number of integer values introduces rounding errors, which will be discussed in section 9.3.

The most prominent case for  $M_{contextual}$  is a contextual model which contains one context model  $M_0$  and one object model  $M_1$ . An example of such a model is depicted in Figure 2 — The boxes representing the context and object model contain root and part filters. Their spatial relation is being described by the scale, offset and deformation parameters illustrated in the circle.

## Structure

The overall model  $M_{contextual}$  is very similar to the structure of its sub-models. The sub-models in the overall model act similar to the parts in the sub-models. Like parts in the deformable parts model by Felzenszwalb et al. the sub-model locations can be deformed and their detection scores are being aggregated. There are three core differences in score aggregation, deformation and offsets.

### Score aggregation:

While the part scores in Felzenszwalb et al. [2010a] are being summed up, the score of context and object scores in  $M_{contextual}$  works differently. Summing would mean that the presence of a positive context score would increase the overall detection's score. This contradicts our assumption that a context only enables the presence of an object. The context can exist without the presence of an object, the object can only exist together with the context. That is why the context score is being thresholded at a certain point. See 4.2.1 Option 1: Conditional-sum.

### Object Deformation:

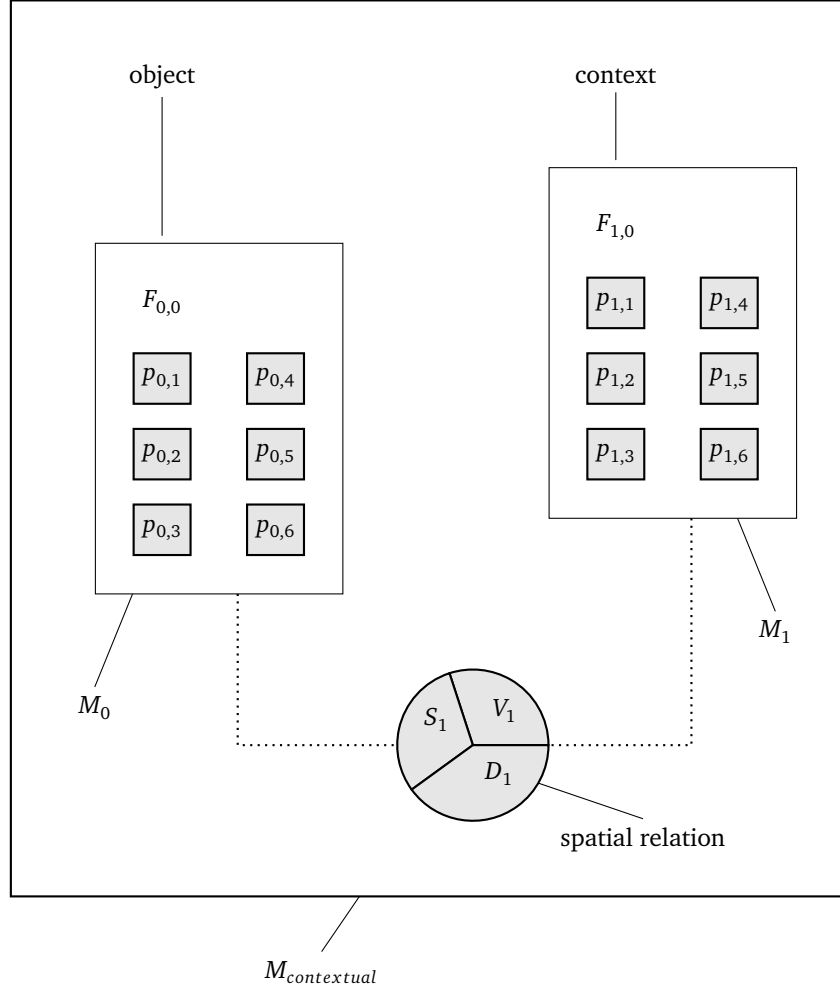
Additionally to the deformation in the  $x$  and  $y$  dimensions this model allows a change of scale. This is important because the object and context models represent their own objects which might vary in relative size. A practical example would

<sup>49</sup> Some examples: The scale change for  $S_j = 1$  is 2 (double the original size), for  $S_j = 0.5$  it is  $\sqrt{2} \approx 1.4142$  (about 1.4142 the original size).

<sup>50</sup> See Felzenszwalb et al. [2010a, p. 5].

<sup>51</sup> See framework version 5 change notes by Girshick et al. [2012a]





**Figure 2:** Overall model structure of  $M_{contextual}$  in the case of one object and one context model.

be the size of the bag (object) relative to the person (context) holding it. One could imagine that there are bags of different sizes. Moreover, the object and context model could be trained in different scales and resolutions. This results in different score locations on the scale pyramid. To reflect such scale differences, the part offsets have to contain a scale prior (see  $S_1$ ).

#### Scale Offset:

Felzenszwalb et al. use the part filter’s anchor points (top left corner) as the locations to compute relative offsets on. These anchors are easy to use, since the filter response locations represent the part locations, however, relative anchor positions are not scale invariant! In the former case scaling parts away from their mean size would increase anchor offsets and lead to additional deformation costs in  $x$  and  $y$  directions. Instead the offsets between context and object model are based on their center locations.

Specifically the offset is defined by the object detection’s anchor and the context detection’s center, since only the context detection is being deformed in scale.

#### 3D Context Deformation

Felzenszwalb et al. are using the method by Felzenszwalb and Huttenlocher [2004] to calculate the distance transform which is needed to model part deformations.

*A distance transform of a binary image specifies the distance from each pixel to the nearest non-zero pixel.*<sup>52</sup>

<sup>52</sup> Felzenszwalb and Huttenlocher [2004, p. 1]

---

The generalized distance transform method is not limited to 1D or 2D (image) applications but can be easily applied to additional dimensions. The algorithm can be computed in linear time. That means it runs along each dimension in linear time. Adding another dimension to the problem increases the problem size by the additional dimension's extent.

Our model adds another hierarchy to the model – the context. The context can be deformed in both the  $x$  and  $y$  dimension – just like object parts – and also in scale. To allow for deformation on the relative scale between the object and the context we need to add another dimension to the distance transform.<sup>53</sup> Applying the distance transform to the pyramid's scale dimension is not trivial. The score pyramid contains the score responses of parts, or whole deformable parts models. In the first case part filters are applied to a feature pyramid.<sup>54</sup> The result is a score pyramid.

To be able to apply the distance transform in the scale dimension across each level of the scale-space pyramid, the pyramid has to be transformed into a 3-dimensional matrix. That means each of the pyramid levels have to be scaled up to the smallest common size. The distance transform can subsequently be applied in the  $z$ -dimension (scale dimension) of the resulting 3-dimensional grid before converting it back to a score pyramid. The distance transform algorithm is then being applied to the  $x$  and  $y$  axes on each scale level.

---

<sup>53</sup> See Felzenszwalb and Huttenlocher [2004]

<sup>54</sup> See Felzenszwalb et al. [2010a, p. 5].

## 4 Matching

### 4.1 Original model

Given an object model  $M$ , one can use it to score an object hypothesis  $z$ , specified in terms of the positions of the various filters  $p_i$  in the feature pyramid which is generated from an input image  $I$ . In particular  $p_i = (x_i, d_i, l_i)$  specifies the location of the  $i$ -th part in the feature pyramid  $H$  and  $z = (p_0, \dots, p_n)$  defines the object hypothesis.

The score is being calculated by summing all filter responses (filter term), subtracting all part's deformation costs and adding mixture model bias  $b$ . The filter responses are the dot product between filter vector  $F_i$  and the feature vector  $\phi(H, p_i)$  which holds the features of part  $p_i$ 's location in feature pyramid  $H$ . The deformation costs are being calculated using the deformation costs  $d_i$  and the deformation features  $\phi_d(dx, dy) = (dx, dy, dx^2, dy^2)$ . The part displacements  $dx$  and  $dy$  are relative to the first (root) filter. The displacement of the  $i$ -th part is defined as  $(dx_i, dy_i) = (x_i, y_i) - (2(x_0, y_0) + v_i)$ .<sup>55</sup>

According to Felzenszwalb et al. [2010a, p. 6] the score of a configuration specified by  $z$  is defined by the following equation (14).

$$\text{score}(z) = \text{score}(p_0, \dots, p_n) = \left( \underbrace{\sum_{i=0}^n F_i \phi(H, p_i)}_{\text{filter term}} - \underbrace{\sum_{i=1}^n d_i \phi_d(dx_i, dy_i)}_{\text{deformation term}} \right) + b \quad (14)$$

The overall score of a root location is then calculated as the score of the best possible placement of the parts,

$$\text{score}(p_0) = \max_{p_1, \dots, p_n} \text{score}(p_1, \dots, p_n). \quad (15)$$

Felzenszwalb et al. "use dynamic programming and generalized distance transforms (min-convolutions) [...] to compute the best locations for the parts as a function of the root location"<sup>56</sup>. Given computed filter responses, their method requires  $O(nk)$  time, with  $n$  parts and  $k$  locations in the feature pyramid. First, the generalized distance transform algorithm is being applied to filter response  $R_{i,l}$  of each part, where  $R_{i,l}$  is a cross-correlation between part filter  $F_i$  and level  $l$  of the feature pyramid. The algorithm can be computed in linear time per dimension.<sup>57</sup> This transformation is being defined mathematically as

$$D_{i,l}(x, y) = \max_{dx, dy} (R_{i,l}(x + dx, y + dy) - d_i \cdot (dx, dy)), \quad (16)$$

where the  $D_{i,l}(x, y)$  represents the maximum score contribution (penalized by the deformation cost) of part  $i$  to the root location  $(x, y)$  at level  $l$ .<sup>58</sup>

Then the overall score is being calculated as the sum of the root filter responses  $R_{0,l}$  on each level  $l$  and the transformed part filter responses  $D_{i,l}$  of all parts. Before summing, the transformed part filter responses are being shifted by the offset  $v_i$  and moved down in the score pyramid by  $\lambda$  levels, specifically

$$\text{score}(x_0, y_0, l_0) = R_{0,l_0}(x_0, y_0) + \sum_{i=1}^n D_{i,l_0-\lambda}(2(x_0, y_0) + v_i) + b. \quad (17)$$

Using the generalized distance transform algorithm the transformation  $D_{i,l}(x, y)$  can also be inverted<sup>60</sup>, so the corresponding part locations can be found starting from a high scoring root location. (This is important for feature extraction from examples, which is needed to train the model using stochastic gradient descent.)

### Mixture models

Mixture models are being used to combine multiple classifiers in a single model. Felzenszwalb et al. define a mixture model with  $m$  components as an  $m$ -tuple,  $M = (M_0, \dots, M_m)$ . The object hypothesis in a mixture model specifies the component  $c$  part locations,  $z = (c, p_0, \dots, p_{n_c})$ , where  $n_c$  is the number of parts in model  $M_c$ . The overall score in a mixture model at a particular location  $(x, y, l)$  is the maximum of component models  $M_c$  at that location.

<sup>55</sup> See Felzenszwalb et al. [2010a, p. 6].

<sup>56</sup> citet[p. 6]felzenszwalb2010object.

<sup>57</sup> See Felzenszwalb et al. [2010a, p. 6], Felzenszwalb and Huttenlocher [2004, p. 9]

<sup>58</sup> See Felzenszwalb et al. [2010a, p. 6].

<sup>60</sup> See Felzenszwalb et al. [2010a, p. 6], Felzenszwalb and Huttenlocher [2004, p. 7]

---

## 4.2 Contextual model

---

A scoring function had to be developed that suits the assumptions of the contextual model. Recall that the object class is supposed to only appear in its context, but the context class can exist with or without the object. Thus, the goal was to find a scoring function, which

- **does not** infer the object’s existence by the likely existence of the context
  - but **does** infer the object’s non-existence by the likely non-existence of the context.
- 

### 4.2.1 Defining a contextual scoring function

---

Using a sum approach as in the basic scoring function (14) obviously does not fit this goal, which would cause the object and context scores to be plainly added, leading to unwanted behavior. For instance, this could mean that a very high context score could more than compensate a low object score, possibly leading to false positive detections.

In this section two options to construct a scoring method for the contextual model are being discussed. The first one adheres to the assumptions and allows the object score to compensate a bad context score, however, the context score can not improve a bad object score. This is being solved by thresholding the context score before summing. We call this approach ‘conditional-sum’. This is also the method that has been chosen for further development of the contextual model. The second approach is a product approach where object and context SVM scores – with a theoretically unlimited range – that have been mapped to probability estimates in the interval  $[0, 1]$  – e.g. using sigmoid fitting. A sigmoid is an “S”-shaped function which can map scores from  $\mathbb{R}$  to  $[0, 1]$ .<sup>61</sup> Generating an overall score by multiplying two scores  $s_1, s_2 \in [0, 1]$  ensures that both object and context must have a sufficiently high score to generate a positive detection. The detailed definitions of both scoring methods and the reasons for picking the first approach are described in this section.

Not unlike a mixture model a contextual model also contains multiple deformable parts models. To score a contextual model  $M_{\text{contextual}}$  we have to aggregate the scores of object and context models. As defined above a contextual model  $M_{\text{contextual}}$  has an object model  $M_0$  and  $m - 1$  context models  $C_j, j \in \{1, \dots, m - 1\}$ . We define the *contextual object hypothesis* as a pair  $Z = (z_0, \dots, z_m)$  consisting of both the context and object hypotheses. All models  $M_j$  contained in  $M_{\text{contextual}}$  are referred to as sub-models,  $C_j, 0 < j \leq m$  represent the context models. To account for the indices of multiple sub-hypotheses we define  $z_j = \{p_{j,0}, p_{j,1}, \dots, p_{j,n_j}\}j$ , where  $j$  is the  $j$ -th sub-model hypothesis.

#### Option 1: Conditional-sum

As defined in (14) we score a sub-model hypothesis  $z_j$  with  $\text{score}(z_j) = \text{score}(p_{j,0}, \dots, p_{j,i})$ . As a simple example we now look at a contextual model without deformation costs. Recall that the scores of part hypotheses are independent from each other and thus can be aggregated after calculating the scores for each part. This is also true for the sub-models in a contextual model. The difference lies in the aggregation method. Instead of summing up the part scores we limit the context hypothesis scores to a certain threshold to reflect the asymmetric inference between object and context. Specifically we score  $Z$  with

$$\text{score}_{\text{contextual}}(Z) = \text{score}_{\text{contextual}}(z_0, \dots, z_j) = \text{score}(z_0) + \sum_{j=1}^m \text{score}_{\text{context}}(z_j), \quad (18)$$

where

$$\text{score}_{\text{context}}(z) = \begin{cases} \text{score}(z) & \text{score}(z) < 0 \\ 0 & \text{otherwise} \end{cases} \quad (19)$$

is the contribution of a context hypothesis to the score of  $Z$ . On the one hand, capping the context score at a certain threshold controls and limits the inference from the context hypothesis  $z_j$  to a contextual object hypothesis  $Z$ . On the other hand, the object model hypothesis  $z_0$  has full impact on the overall score.

This scoring function is designed to penalize the overall object score when a negative context score is present, while a positive context score leaves the overall object score unchanged. The threshold in this scoring function to enable or disable context influence on the overall score is 0. While this threshold is fixed at 0 in (19), the threshold  $b_j$  of context model  $C_j$  can be indirectly influenced by the model parameters of  $C_j$ .

---

<sup>61</sup> There are different methods to generate probability estimates from classifier scores. One example is the work by Zadrozny and Elkan [2002], which use sigmoid fitting and score binning to estimate probabilities.

The context score is being formed out of a dot product between the filter vector  $\beta$  and the context feature vector  $\Phi(x, z)$  for an example  $x$  and a hypothesis  $z$ . A threshold  $b$  which  $\text{score}_{\text{contextual}}$  should be capped can now be chosen by using homogeneous coordinates. The threshold is defined by the following equation:

$$\text{let } v = \hat{\Phi}(x, z) \quad (20)$$

$$\beta \cdot v = (\beta_1, \dots, \beta_n) \cdot \begin{pmatrix} v_1 \\ \dots \\ v_n \end{pmatrix} = b \quad (21)$$

$$\iff (\beta_1, \dots, \beta_n, b) \cdot \begin{pmatrix} v_1 \\ \dots \\ v_n \\ 1 \end{pmatrix} = 0 \quad (22)$$

$$(23)$$

By extending  $\beta$ , the threshold  $b$  now is a part of the model parameters, and, in spite of being fixed in the scoring function, can be influenced indirectly. Actually  $b$  is already present in the deformable parts models as the SVM bias parameter, which makes additional changes unnecessary. Therefore,  $b$  is equivalent to the bias defined in (2).

### Option 2: Product of probability estimates

**(Not currently pursued):** The original score function is being changed to make sure that a good context match is not directly leading to an object detection. Since we only want an object match if context and object has a good detection score we sum each filter response for the context and object model separately and then multiply their estimated probabilities. An example to estimate probabilities from classifier scores would be through sigmoid fitting. Sigmoid functions  $\text{sig}_i$  can translate SVM scores into a  $[-1, 1]$  interval. Therefore, one could define the estimated probabilities of  $\text{score}(Z_i)$  being positive with

$$p_{Z_i} = \frac{(1 + \text{sig}_i(\text{score}(Z_i)))}{2} \in [0, 1]. \quad (24)$$

The score could then be calculated by multiplying the probabilities  $p_{Z_0}$  and  $p_{Z_1}$ . An example  $X$  would then be scored by

$$P(X) = p_{Z_0} \cdot p_{Z_1} \quad (25)$$

$$(26)$$

The resulting (estimated) probability  $p_Z$  represents the contextual model's classifier score, which is thresholded to get a binary detection result.

### Comparing both matching methods

When comparing both methods, several reasons come to mind why it might be better to choose the conditional-sum approach.

- Conditional sum is similar to the standard hinge loss (hinge loss is defined as  $\max(0, 1 - \beta \cdot \hat{\Phi}(x_i, z(x_i)))$ ) instead of the conditional-sum term  $\max(0, \beta \cdot \hat{\Phi}(x_i, z(x_i)))$ . This could be more easily applied to the loss function of an SVM classifier.
- With the conditional-sum approach the dependency on a high context score is not as strong as in the product approach, allowing the object score to compensate penalty by a mediocre to low context score. This could lead to a more robust model when the context classifier misclassifies positives as false negatives.

For the second approach, when multiplying scores between 0 and 1, a single score can hardly compensate if the other score is close to zero. We might not want to enforce the fact that the context classifier is not allowed to be wrong. It should be for the discretion of the training algorithm if it allows instances where a high object score compensates a low context score to still achieve a positive detection.

- It is a rather complex problem to generate probability estimates for classifier scores, especially in a multi-class setting.<sup>62</sup> Going this path would have introduced a lot more complexity into the contextual model.

Based on the reasons above, the conditional-sum approach is being used to score the contextual model going forward.

<sup>62</sup> See work by Platt et al. [1999], Zadrozny and Elkan [2002], Milgram et al. [2005].

#### 4.2.2 Contextual model scoring

Matching in a contextual model is largely similar to matching in a deformable part model as described above. This section will transfer the matching method by Felzenszwalb et al. [2010a] to contextual models including the new conditional-sum score aggregation as well as scale deformation.

As defined above in section 4.2.1, the contextual object hypothesis  $Z$  contains multiple sub-model hypotheses  $z_j$ . We now extend (18) for contextual models with deformation costs,  $D_j \neq (0, 0, 0, 0, 0, 0)$ . To calculate displacement and deformation costs of a context hypothesis  $z_j$ , we require its location. The location  $\text{loc}(z_j)$  is defined by its root filter location,

$$\text{loc}(z_j) = p_{j,0} = (x_{j,0}, y_{j,0}, l_{j,0}). \quad (27)$$

Contrary to the parts in the model by Felzenszwalb et al. [2010a], the context can also move within the scale space. There the anchor for the  $i$ -th part in  $M_j$  are defined by the mean distance from the top-left corner of the first part  $p_{j,0}$  (root) to the top-left corner of part  $p_{j,i}$ . In our model the anchor for the  $j$ -th context model is defined as the mean distance between the center of the object  $\text{center}(z_0)$  and the top-left corner of the context  $\text{loc}(z_j)$ , where

$$\text{center}(z_j) = \text{loc}(z_j) + \frac{(w_j, h_j, 0)}{2} \in \mathbb{Z}^3 \quad (28)$$

and  $(w_j, h_j)$  represents the width and height of root filter  $F_{j,0}$ .

Since the object model  $M_0$  is just a deformable parts model its object hypothesis  $z_0$  is being scored by (14). Each context model  $C_j$  consist of a deformable parts model  $M_j$  appended by 3-dimensional deformation parameters  $D_i$  and offset and scale priors  $V_i$  and  $S_i$ . These act analogous to the deformation parameters and offsets for parts  $d_i$  and  $v_i$  in the deformable parts model. Thus, the score for the object hypothesis  $z_0$  is defined by

$$\text{score}_{\text{def}}(z_j) = \underbrace{\text{score}_{\text{context}}(z_j)}_{\text{context score}} - \underbrace{D_j \phi_D(dx_{j,0}, dy_{j,0}, ds_{j,0})}_{\text{context deformation costs}} \quad (29)$$

where

$$ds_j = \frac{l_j}{\lambda} - \left( \frac{l_0}{\lambda} + S_j \right) \quad (30)$$

$$(dx_j, dy_j) = (x_j, y_j) - (2^{ds_j} \cdot \text{center}(z_0) + V_i) \quad (31)$$

is the displacement of the  $j$ -th context relative to its anchor position in a score pyramid  $H$  with  $\lambda$  levels per octave<sup>63</sup> and

$$\phi_D(dx, dy, ds) = (dx, dy, ds, dx^2, dy^2, ds^2) \quad (32)$$

are context deformation features. Recall that  $\text{score}_{\text{context}}(z_j)$  in (29) is the thresholding function (19). The context deformation term  $D_j \phi_D(dx_{j,0}, dy_{j,0}, ds_{j,0})$  is subtracted after the context score has been thresholded because we want to penalize the object for not being close enough to the context, not the other way around.

The resulting score for a contextual object hypothesis  $Z$  is defined by

$$\text{score}_{\text{contextual}}(Z) = \text{score}(z_0) + \sum_{j=1}^m \text{score}_{\text{def}}(z_j). \quad (33)$$

<sup>63</sup> The scale offset parameter  $S_i$  and scale displacement  $ds_i$  as well as the deformation costs in  $\phi_D$  need to be invariant to the scale resolution of the score pyramid  $H$ . This is why it makes sense to describe scale changes as the number of scale octaves. The level change can then be determined by rounding the scale change according to the pyramid scale resolution. E.g. a scale displacement  $ds = 0.25$  in a pyramid with  $\lambda = 5$  levels per octave would result in a level displacement  $dl = \lfloor ds \cdot \lambda + 0.5 \rfloor = 1$

## Matching

Previously, we have defined how contextual models are being scored, including score aggregation and 3-dimensional deformation. To detect objects with the contextual model, we need to calculate the overall score of an object location in a contextual model. It is defined as the score of the best possible placement of its sub-models. In particular

$$\text{score}(z_0) = \max_{z_0, \dots, z_1} \text{score}(z_0, \dots, z_1) \quad (34)$$

Let  $R'_{j,l}(x, y) = \text{score}(x, y, l)$  be an array which stores the response of model  $M_j$  in  $M_{\text{contextual}}$  in level  $l$  of the score pyramid, corresponding to (17), and

$$R'^{<0}_{j,l} = \text{score}_{\text{context}}(x, y, l) = \min(0, R'_{j,l}(x, y)) \quad (35)$$

an array which stores the thresholded responses of context models  $M_j$ ,  $0 < j \leq m$  in level  $l$  of the score pyramid according to (19).

Then we transform the thresholded response to allow for deformation in  $x$ ,  $y$  and  $l$  with

$$D'_{j,l}(x, y) = \max_{dx, dy, ds} \left( R'^{<0}_{j,l}(x + dx, y + dy, l + \lambda ds) - D_j \cdot (dx, dy, ds) \right). \quad (36)$$

The array  $D'_{j,l}(x, y)$  stores the maximum score contribution of context model  $j$  to the object location  $(x, y)$  at level  $l$ .

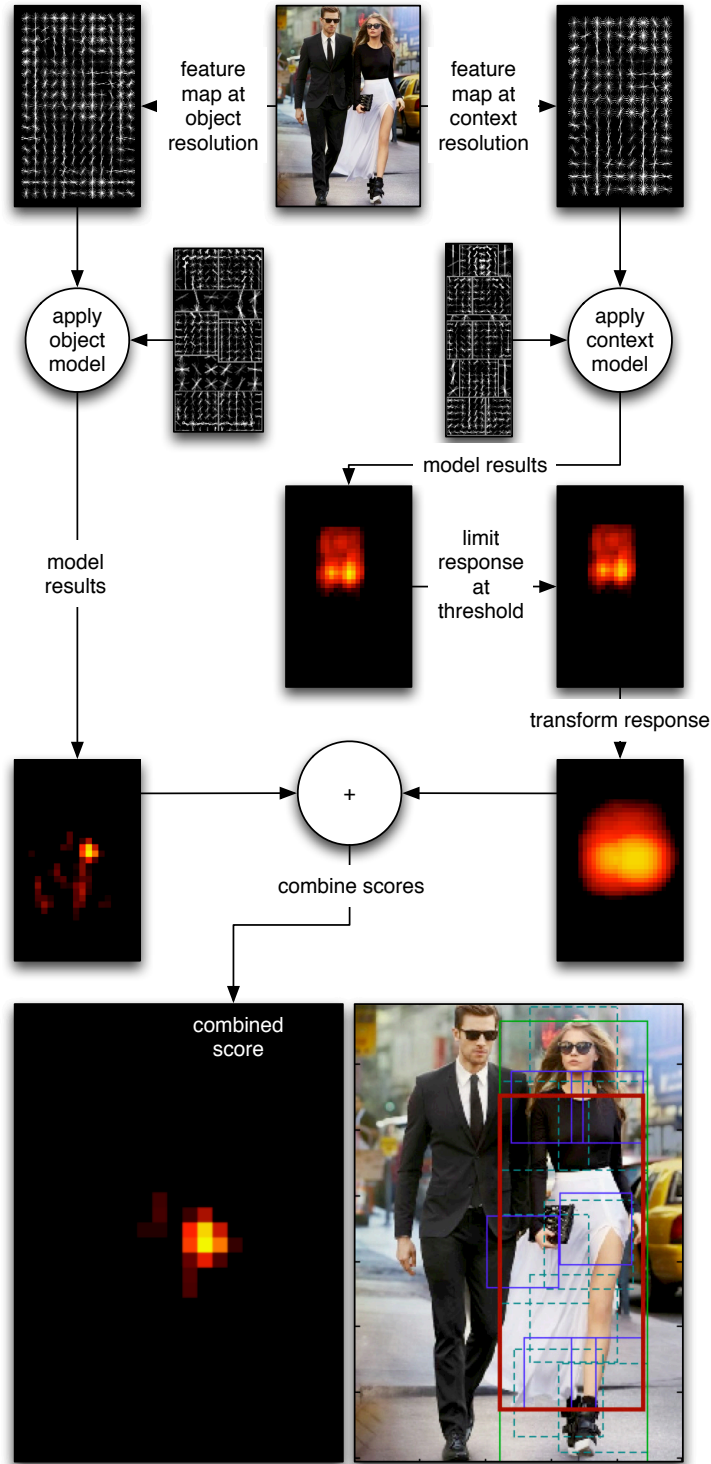
Finally, the overall score can be calculated as the sum of the root filter responses  $R'_{0,l}$  on each level  $l$  and the transformed responses  $D'_{j,l}$  of all context models. Before summing the transformed part filter responses, they are being shifted by the spatial offset  $V_j$  and scale offset  $S_j$

$$\text{score}_{\text{contextual}}(x_0, y_0, l_0) = R'_{0,l_0}(x_0, y_0) + \sum_{i=1}^n D'_{j,l_0 - \lambda S_j} \left( 2(x_0, y_0) + V_j \right). \quad (37)$$

(37) is very similar to (17), but has no normalization parameter  $b$ . The whole detection process on the level of the contextual model is depicted in Figure 3.

The transformation defined in array  $D_{i,l}(x, y)$  for the deformable parts model can also be backtracked. This means, that, given a root location, the optimal displacements for a part can be determined.<sup>64</sup> As noted before, this is required for feature extraction and model training. The transformation defined by  $D_{i,l}(x, y)$  does only allow for spatial uncertainty in the  $x$  and  $y$  dimensions and thus affects only the  $l$ -th level of the score pyramid. The transformation defined by  $D'_{j,l}(x, y)$  does also allow for scale uncertainty. Unfortunately the generalized distance transform algorithm is not defined for the scale dimension in a pyramid. Instead it can only work on a grid. The next section explains how we solve this problem. But ultimately it means that backtracking through scale space is not possible using the generalized distance transform algorithm. This is why a brute force backtracking method needs to be used, which is being described in section 9.3.

<sup>64</sup> See section 4.1 and Felzenszwalb et al. [2010a, p. 6].



**Figure 3:** Contextual matching process at a single scale. The figure is analogous to the matching process in Felzenszwalb et al. [2010a, p. 7] but transferred to the new hierarchical level of a contextual model. The object and context models usually (but do not have to) require feature maps from different levels in the feature map. The context model's response is being thresholded as defined in (33). Next, the thresholded response is transformed according to the deformation costs and combined with the object model's unaltered response. One can see that the scores far away from the context are suppressed. Note that the context transformation process also spreads the score across pyramid levels. The score maps in this figure are normalized, therefore the thresholded context model response becomes "redder" in some areas because score variance is reduced. HOG filter images created with the framework by Felzenszwalb et al. [2011].



## 5 3D distance transform on score pyramid

Felzenszwalb et al. use the generalized distance transform algorithm to efficiently calculate the cost of part deformations.<sup>65</sup>

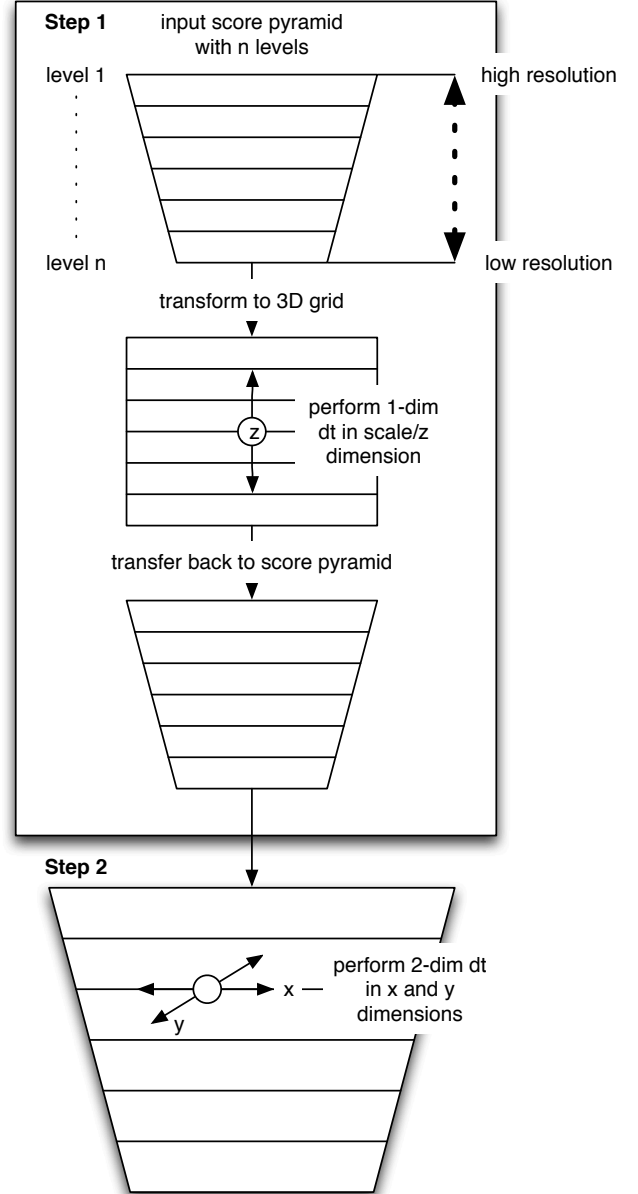
To allow for scale deformation of the context relative to the object we need to add another dimension to the distance transform. Generalized distance transform can be applied to a grid with an arbitrary dimension, running in “ $O(dk)$ , where  $d$  is the dimension of the grid and  $k$  is the overall number of grid locations”<sup>66</sup>. The problem with this is not applying the distance transform itself, but preparing the data to be able to do so. The score maps for different scale levels are part of a score pyramid where each level  $l$  has a different resolution depending on the scale  $s = 2^{l/\lambda}$ , where  $\lambda$  is the amount of levels per scale octave.

In our case each pyramid is also surrounded by a fixed amount of padding  $p \in \mathbb{N}^2$ , which is needed to apply filters and receive responses in the fringe regions of the feature and score pyramids. Therefore,  $p$  is determined by the maximum filter size used in a model. Consequently the feature maps on each level cannot just be up-sampled to the same resolution. Since the padding is a fixed amount, it makes up a larger percentage of the feature map in lower levels of the pyramid than in higher levels. Therefore, virtual padding is needed to translate coordinates from one scale level to another.

Let  $C_i, C_j \in \mathbb{R}^2$  be coordinates on levels  $i, j \in [1, \dots, n]$  of the pyramid with  $n \in \mathbb{N}$  levels and  $k \in \mathbb{N}$  intervals per octave. Also let  $P \in \mathbb{R}^2$  be the padding applied to each pyramid level. Downsampling a matrix by an octave would reduce its resolution by half.  $C_i$  and  $C_j$  represent the same location in an image, just on different scale levels on the pyramid.

$$\text{Then } C_i = 2^{\left(\frac{j-i}{k}\right)} C_j + 2^{\left(\frac{j-i}{k}-1\right)} P. \quad (38)$$

To perform distance transform on the scale axis we first need to convert the pyramid to a 3-dimensional grid. We do this by translating scale levels to the first scale level with (38). The grid’s size will be determined by the size of the top level plus two times the virtual padding  $2^{\left(\frac{j-i}{k}-1\right)} P$  we get from translating level  $i = n$  to level  $j = 1$ . Therefore, virtual padding can be relatively large. With 40 levels and an interval of 10 for example the virtual padding is getting almost 14 times as large as the padding itself. Therefore, the grid can occupy a significant amount of memory.<sup>67</sup>



**Figure 4:** Process to apply 3D distance transform on score pyramid according to Algorithm 1. In step 1 the distance transform algorithm is being applied to the scale dimension only. After the score matrix has been transformed back to a score pyramid in step 2 the distance transform algorithm is being applied to  $x$  and  $y$  dimensions.

<sup>65</sup> See Felzenszwalb et al. [2010a, p. 6].

<sup>66</sup> Felzenszwalb and Huttenlocher [2004]

<sup>67</sup> This is especially a problem for distance transform running on GPUs, since memory is much more expensive in graphics cards. This can be overcome through partitioning which adds complexity to the program.

---

**Algorithm 1** Applying 3D distance transform to score pyramid

---

```
function DT3D( $P, d, a$ )  
   $P \leftarrow \text{shiftPyramidLevels}(P, a)$   
   $M \leftarrow \text{gridFromPyramid}(P)$   
   $M \leftarrow \text{dtScale}(M)$   
   $P \leftarrow \text{pyramidFromGrid}(M)$   
   $P \leftarrow \text{shiftPyramidLevels}(P, -a)$   
   $M \leftarrow \text{dt2D}(P)$   
end function
```

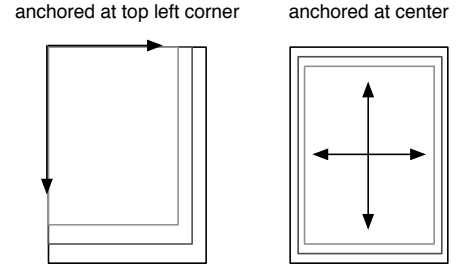
---

In Algorithm 1 the process of applying the 3D distance transform is described in pseudo code. It starts by shifting the pyramid’s feature maps by the anchor offset  $a = (w, h)/2 \in \mathbb{N}^2$ , where  $w$  and  $h$  are the width and height of the model’s detection window. So  $\text{shiftPyramidLevels}(P, a)$  shifts all score locations in the pyramid to the center of the detection window. Then the pyramid is transformed to a grid to perform the distance transform.

In line 4 the call “dtScale” applies the distance transform 1-dimensionally to the 3rd/scale dimension. Then the grid is transformed back to a score pyramid, and the previous shift is inverted with  $\text{shiftPyramidLevels}(P, -a)$ . Both  $\text{shiftPyramidLevels}$  calls before and after scale deformation make it a centered operation, meaning that the detection windows corresponding to a location on the feature map expand or shrink from the center (see Figure 5).

In line 7 the 2D distance transform algorithm is being applied to the  $x$  and  $y$  dimensions on each of the pyramid’s levels, analogous to Felzenszwalb et al. [2010a] .

When downsampling with  $\text{pyramidFromGrid}$ , it is important to make sure that local maxima do not get thrown out. To prevent that, the feature maps in each grid level are being scaled up sparsely to a size which is a  $2^n$  multiple of the target size, with  $n \in \mathbb{N}$ . “Sparsely” in this context means that the entries in the source feature map get transferred uniquely to the larger matrix, without replicating the values to fill up space. Then the feature map is downsampled to the size of the pyramid’s target level via a max-pooling algorithm.



**Figure 5:** Scale deformation anchored at top left corner of detection window (left) and scale deformation anchored at center of detection window (right).

---

## 6 Histogram of oriented gradients

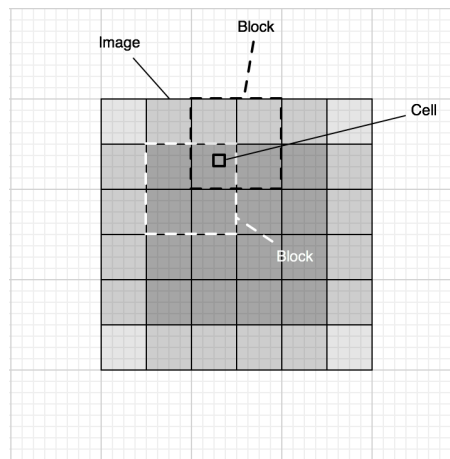
---

In their work Dalal and Triggs show that HOG descriptors perform very well in object detection, especially for human detection.<sup>68</sup> Since Felzenszwalb et al. base their work on the Dalal-Triggs human detection model they also use HOG as their image descriptors. Theoretically the HOG descriptor could be easily substituted for other descriptor methods. However, based on the findings by Dalal and Triggs [2005] and Felzenszwalb et al. [2010a] HOG is the best descriptor for most object detection problems.

While Dalal and Triggs showed that their HOG-based model would outperform previously existing feature sets for human detection Felzenszwalb et al. could achieve cutting-edge detection performance on many more object categories.<sup>69</sup>

The method to generate HOG descriptors contains four steps. At first the input image is normalized regarding gamma and color. Then gradients are computed by convoluting gradient filter templates with the preprocessed input image. The image is divided into cells, which are described as “small spatial regions”<sup>70</sup>. Finally, the edge orientations from the computed gradients in each cell are combined to form a histogram representing the gradients of a certain spatial region of the input image.<sup>71</sup> The number of buckets in the histogram can be chosen depending on requirements. For most situations 18 buckets is enough, leading to an orientation resolution of  $10^\circ$ .

To generate a descriptor, which is invariant with respect to changes in illumination and contrast, a local normalization must take place. This is being done by further grouping multiple adjacent cells in so called blocks.



**Figure 6:** Visualization of a HOG descriptor. The image is divided into small cells and overlapping blocks. Each cell can influence multiple blocks.

As depicted in Figure 6 these blocks are overlapping. Each cell can be part of multiple blocks. Now contrast normalization is being performed on each block. These normalized blocks are the actual histogram of oriented gradients descriptors. The whole dense grid of overlapping blocks extracted from an input image then forms the feature vector that can be used to train a classifier.<sup>72</sup>

This process leads to the following key features of HOG. Due to localized cells and overlapping blocks the descriptor is invariant to “local geometric and photometric transformations”<sup>73</sup>. The degree of invariance can be controlled by changing the cell and block size. Larger cell sizes for example lead to higher geometric invariance, while losing spatial detail.

The main configuration parameters for HOG are the number of orientation bins in the histogram which determines the resolution in which gradients are described, the cell size and block size. If for example 18 orientation bins are used the whole range of gradients –  $0^\circ$  to  $180^\circ$  since the direction of the gradient is ignored – is divided into bins of  $10^\circ$  resolution. The cell size determines how large the image area is to construct the histogram. The block size determines how many cells are being combined to a block and thus determining the contrast normalization area. For the human detector Dalal and Triggs came to the conclusion that the configuration of 9 bins, cell size of 6 by 6 pixels and a block size of 3 by 3 cells leads to the best results. Interestingly, the optimal cell size of 6 to 8 pixel for their human detector is reflected by

---

<sup>68</sup> See Dalal and Triggs [2005, p. 1].

<sup>69</sup> See Dalal and Triggs [2005, p. 4], Felzenszwalb et al. [2010a, p. 16].

<sup>70</sup> Dalal and Triggs [2005, p. 2].

<sup>71</sup> See Dalal and Triggs [2005, p. 2].

<sup>72</sup> See Dalal and Triggs [2005, p. 2].

<sup>73</sup> Dalal and Triggs [2005, p. 2].

---

the human limb sizes in their training data.<sup>74</sup> This could mean that the optimal configuration of the HOG descriptor is dependent on the object class and should be considered when optimizing models.

Another advantage of HOG is that “translations and rotations make little difference if they are much smaller [... than] the local spatial or orientation bin size”<sup>75</sup>. Still it is important that the objects remain in the same rough orientation. For human detection this means that Dalal and Triggs use only images of persons that are roughly upright.<sup>76</sup>

Especially for the target domain of this work – fashion – their previous results show a great prospect. In most cases clothing follows the wearer’s body shape. In this case it should be possible to leverage the advantages the HOG descriptor showed in the work by Dalal and Triggs due to similarity of body shapes. Other instances of clothing which are either stand-alone, like bags, or superimpose the body shape, like some types of dresses, usually show a clear contrast relative to the wearer and background.

Also local contrast normalization can be really important for clothes. A person can wear multiple items with different colors, patterns and materials, showing varying levels of contrast. The same object could have strong contrast w.r.t the background for example, but a low contrast w.r.t an adjacent item of clothing. In such occasion normalizing contrasts could make shapes much more distinguishable.

Patterns and prints, however, could become a challenge for a HOG classifier. Stripes, checkers or printed images on items of clothing could confuse the classifier.

Felzenszwalb et al. propose using PCA to reduce the size of the HOG-descriptor. They achieve a reduction from 36 dimensions to just 11 without reducing the model’s performance by selecting the most relevant eigenvalues. Since these are costly to compute, they define descriptors with 13 dimensions, based on the knowledge gained by analyzing the eigenvalues, which are easier to compute. All descriptors have the same detection performance.<sup>77</sup>

---

<sup>74</sup> See Dalal and Triggs [2005, p. 5].

<sup>75</sup> Dalal and Triggs [2005, p. 2].

<sup>76</sup> See Dalal and Triggs [2005, p. 2].

<sup>77</sup> See Felzenszwalb et al. [2010a, p. 4,13].

---

## 7 Latent SVM

---

In this section the Latent SVM used in Felzenszwalb et al. [2010a] is being recapitulated, so later modifications w.r.t the objective function can be examined.

---

### 7.1 Latent SVM with standard hinge loss

---

Felzenszwalb et al. propose a classifier that scores an example  $x$  with

$$f_{\beta}(x) = \max_{z \in Z(x)} \beta * \Phi(x, z). \quad (39)$$

$\beta$  is a vector holding all model parameters. The possible configurations example  $x$  can have, also known as latent values, are defined by  $Z(x)$  –  $z$  is one latent value configuration. Latent values contain for example both part configuration and the component label. The binary label  $y \in \{-1, 1\}$  for the example  $x$  results from the thresholding  $g_{\beta}(x)$ .<sup>78</sup>

Then  $\beta$  is being trained using the binary labeled examples  $D = (\langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle, \dots, \langle x_n, y_n \rangle)$  minimizing the objective function

$$L_D(\beta) = \underbrace{\frac{1}{2} \|\beta\|^2}_{\text{Regularization term}} + C \sum_i \underbrace{\max(0, 1 - y_i f_{\beta}(x_i))}_{\text{Standard hinge loss}}. \quad (40)$$

The constant  $C$  determines the relative weight of the regularization term  $\frac{1}{2} \|\beta\|^2$  w.r.t. the standard hinge loss term. To be able to train an SVM classifier with latent variables one has to show that the optimization problem is convex.

#### Semi-convexity

“A latent SVM leads to a non-convex optimization problem ...”<sup>79</sup>. Felzenszwalb et al. show that by specifying the latent values for each positive example the objective function of their model becomes convex. This is what they call a semi-convex optimization problem. Linear SVMs are a special case of latent SVMs “if there is a single possible latent value for each example”<sup>80</sup>. This means if  $|Z(x_i)| = 1$  then  $f_{\beta}$  is linear in  $\beta$ .

With a linear SVM we know that the hinge loss  $\max(0, 1 - y_i f_{\beta}(x_i))$  is convex for  $y_i = -1$  since the maximum of multiple convex functions is convex – (39) is defined as the maximum of convex functions.<sup>81</sup> Hence, the objective function is convex in  $\beta$  for all negative examples.

The moment you consider positive examples ( $y_i = 1$ ) then  $\max(0, 1 - y_i f_{\beta}(x_i))$  is no longer convex, as it is the maximum of a convex and a concave function  $1 - y_i f_{\beta}(x_i) \Rightarrow 1 - f_{\beta}(x_i)$  with  $y_i = 1$ . This is why we assign each positive example a single latent value ( $|Z(x_i)| = 1$ ) which renders  $f_{\beta}(x_i)$  linear and the hinge loss convex for positive examples.<sup>82</sup> This shows that (40) is convex.

#### Optimization

When the latent values are being limited to one per example with  $|Z(x_i)| = 1$  let  $Z_p$  define a single latent value for each positive example in  $D$ . Then the optimization algorithm needs to optimize both the latent values  $Z_p$  as well as the model parameters in  $\beta$ . Felzenszwalb et al. use a coordinate descent approach, meaning that the algorithm alternates between optimizing one of the two variables  $Z_p$  and  $\beta$ .

They define an auxiliary objective function

$$L_D(\beta, Z_p) = L_{D(Z_p)}(\beta), \quad (41)$$

where compared to (40) the training set  $D$  has been replaced by  $D(Z_p)$ , a training set where latent values have been restricted to  $Z_p$ . For each positive  $Z_p$  specifies the latent value  $z_i$ , the only possible latent value for the example  $x_i$ , therefore setting  $Z(x_i) = \{z_i\}$ .

---

<sup>78</sup> See Felzenszwalb et al. [2010a, p. 8].

<sup>79</sup> Felzenszwalb et al. [2010a, p. 8].

<sup>80</sup> Felzenszwalb et al. [2010a, p. 8].

<sup>81</sup> See Felzenszwalb et al. [2010a, p. 8].

<sup>82</sup> See Felzenszwalb et al. [2010a, p. 8].

The new auxiliary objective function bounds the LSVM objective with

$$L_D(\beta) = \min_{Z_p} L_{D(Z_p)}(\beta) \quad (42)$$

and  $L_D(\beta) \leq L_{D(Z_p)}(\beta)$  by fixing (39) to the currently best latent configuration.<sup>83</sup>

The coordinate descent algorithm alternates between the following two steps:

1. Relabel positive examples. In this step  $L_D(\beta, Z_p)$  is being optimized over  $Z_p$  “by selecting the highest scoring latent value for each positive example”<sup>84</sup>. In practice this can be described as a data-mining step, where for each positive example  $P_i$  and bounding box  $(I, B)$ , the highest scoring detection that has sufficient overlap with  $B$ , is being stored in  $z_i$ . In short,  $z_i = \operatorname{argmax}_{z \in Z(x_i)} \beta \cdot \Phi(x_i, z)$ .<sup>85</sup>
2. Optimize beta. “Optimize  $L_D(\beta, Z_p)$  over beta by solving convex optimization problem defined by  $L_{D(Z_p)}(\beta)$ .”<sup>86</sup>

Felzenszwalb et al. note that by searching exponentially-large space of latent values for positive and negative examples one gets a “relatively strong local optimum”<sup>87</sup> after the algorithm has converged. Step one searches the space of latent values for positive examples, while step 2 does that implicitly for negatives.<sup>88</sup>

It may be important to initialize  $\beta$  carefully to train a good model. A bad  $\beta$  especially impacts the relabeling of positive examples in step 1, since it is important there to select sensible latent values. The initialization is being described in section 8.<sup>89</sup>

### Stochastic gradient descent

The convex optimization problem defined by  $L_{D(Z_p)}(\beta)$  is being solved via stochastic gradient descent.

Stochastic gradient descent algorithm approximates  $\nabla L_D$ , the sub-gradient of the LSVM objective function, by taking a step in the negative direction of a subset of examples.

Using the sub-gradient of the LSVM objective function (40)

$$\nabla L_D(\beta) = \beta + C \sum_i h(\beta, x_i, y_i) \quad (43)$$

$$h(\beta, x_i, y_i) = \begin{cases} 0 & \text{if } y_i f_\beta(x_i) \geq 1 \\ -y_i \Phi(x_i, z_i(\beta)) & \text{otherwise} \end{cases} \quad (44)$$

one can approximate the term  $\sum_{i=1}^n h(\beta, x_i, y_i)$  with  $nh(\beta, x_i, y_i)$ . It has been shown that by repeating this step many times and slowly reducing the step size the global minimum can be reached almost surely.<sup>90</sup>

Felzenszwalb et al. improve this method by using a feature vector cache  $F$  for  $D(Z_p)$  instead of a cache of examples  $x$ . “This makes it possible to avoid doing inference over all of  $Z(x)$  in the inner loop of an optimization algorithm such as gradient descent.”<sup>91</sup>

The feature vector cache consists of  $(i, v) \in F$  where  $i$  is the index of an example  $x_i$  and the  $v = \Phi(x_i, z)$  the feature vector for  $z \in Z(x_i)$ .  $F$  can contain multiple feature vectors of the same *negative* example  $x_i$  with different latent values  $z$ .<sup>92</sup> All  $i$  indexed by  $F$  are contained in  $I(F)$ . Now instead of optimizing  $L_D(\beta)$  they optimize  $L_F \beta$  – a slightly modified objective function which only considers the feature vectors indexed by  $F$ :

$$L_F(\beta) = \frac{1}{2} \|\beta\|^2 + C \sum_{i \in I(F)} \max(0, 1 - y_i (\max_{(i, v) \in F} \beta \cdot v)). \quad (45)$$

Similar to (43) we can derive the sub-gradient  $\nabla L_F(\beta) = \beta + C \sum_i h(\beta, x_i, y_i)$  with

$$h(\beta, x_i, y_i) = \begin{cases} 0 & \text{if } y_i (\max_{(i, v) \in F} \beta \cdot v) \geq 1 \\ -y_i \Phi(x_i, z_i(\beta)) & \text{otherwise} \end{cases} \quad (46)$$

Then  $\beta$  can be updated using the following algorithm<sup>93</sup>:

<sup>83</sup> Felzenszwalb et al. [2010a, p. 8].

<sup>84</sup> Felzenszwalb et al. [2010a, p. 8].

<sup>85</sup> See Felzenszwalb et al. [2010a, p. 8].

<sup>86</sup> Felzenszwalb et al. [2010a, p. 8].

<sup>87</sup> Felzenszwalb et al. [2010a, p. 8].

<sup>88</sup> See Felzenszwalb et al. [2010a, p. 8].

<sup>89</sup> See Felzenszwalb et al. [2010a, p. 8].

<sup>90</sup> See Bottou [1998], Kiwiel [2001].

<sup>91</sup> Felzenszwalb et al. [2010a, p. 10].

<sup>92</sup> This is not the case for positive examples, since they have a single latent value  $z$  defined by  $Z_p$ .

<sup>93</sup> See Felzenszwalb et al. [2010a, p. 10].

---

**Algorithm 2** Stochastic gradient descent with feature cache

---

- 1) Let  $\alpha_t$  be the learning rate for iteration  $t$
  - 2) Let  $i \in I(F)$  be a random example  $i$  indexed by  $F$
  - 3) Let  $v_i = \operatorname{argmax}_{v \in V(i)} \beta \cdot v$
  - 4) If  $y_i(\beta \cdot v_i) \geq 1$  set  $\beta = \beta - \alpha_t \beta$
  - 5) Else set  $\beta = \beta - \alpha_t(\beta - C n y_i v_i)$
- 

The learning rate  $\alpha_t$  defines the step size in each iteration. A larger learning rate is not always favorable, since one could step far further than and diverge from the minimum. Therefore, the learning rate should decrease each iteration. According to Shalev-Shwartz et al. [2007] setting  $\alpha_t = \frac{1}{t}$  works well for linear SVMs. In their implementation of the gradient descent algorithm, however, Felzenszwalb et al. use a learning rate of  $\alpha_t = \frac{1}{\min(\frac{n}{2}, t+10000)}$  for a maximum of  $n$  iterations, basically starting with a learning rate which is already well below one and keeping it at  $\frac{2}{n}$  for larger  $t$ .

**Data-mining hard examples**

The size of the feature cache  $I(F)$  basically determines the runtime of the gradient descent algorithm. While it is very efficient in the sense that each gradient descent iteration can be computed very fast, it is still important to select examples that get into the  $I(F)$  wisely, so that the algorithm converges faster. This is especially important since negative examples are so numerous. Obtaining a training set of negatives, images not containing the object class, is very inexpensive compared with the process of finding and annotating positive examples. One image can already have  $10^5$  detection windows to consider. A typical training set with 2000 negative images would lead to 200 million feature vectors to consider while training. Therefore, only "hard negatives", and of course positives, should be used to form  $I(F)$ .<sup>94</sup>

Hard negatives are negative examples which violate the SVM margin the most. In a Latent SVM the hard and easy feature vectors of a training set  $D$  are defined as  $H(\beta, D)$  and  $E(\beta, D)$  respectively:

$$H(\beta, D) = \{i, \Phi(x_i, z_i) | z_i = \operatorname{argmax}_{z \in Z(x_i)} \beta \cdot \Phi(x_i, z) \text{ and } y_i \beta \cdot \phi(x_i, z_i) < 1\} \quad (47)$$

$$E(\beta, D) = \{(i, v) \in F | y_i \beta \cdot v > 1\} \quad (48)$$

Felzenszwalb et al. propose an iterative algorithm which alternates between training the model and mining hard examples.

---

**Algorithm 3** LSVM data-mining algorithm

---

- 1) Let  $\beta_t := \beta^*(F_t)$  (train the model)  $t$
  - 2) If  $H(\beta, D(Z_p)) \subseteq F_t$  stop and return  $\beta_t$
  - 3) Let  $F'_t := F_t \setminus X$  for any  $X$  such that  $X \subseteq E(\beta_t, F_t)$  (shrink the cache)
  - 4) Let  $F_{t+1} := F'_t \cup X$  for any  $X$  such that  $X \cap H(\beta_t, D(Z_p)) \setminus F_t \neq \emptyset$
- 

The LSVM Data-mining algorithm (Algorithm 3) starts by training the model using the initial cache of feature vectors  $F_1$ .  $\beta^*(D)$  is defined as  $\operatorname{argmin}_{\beta} L_F(\beta)$ . The algorithm stops as soon as all hard negatives in  $D(Z_p)$  are included in  $F_t$  ( $H(\beta, D(Z_p)) \subseteq F_t$ ). Since this implies that all examples in  $F_t \setminus X$  have zero loss on  $\beta$ , Felzenszwalb et al. conclude that this must mean that one has found  $\beta = \beta^*(D(Z_p))$ .<sup>95</sup> They also show that the algorithm is going to terminate after a finite number of iterations because  $L_{F_t}(\beta^*(F_t))$  grows with each iteration while being bounded by  $L_{D(Z_p)}(\beta^*(D(Z_p)))$ .<sup>96</sup> In step three the cache is being shrunk, that means all examples  $(i, v)$  which are elements of  $E(\beta, F_t)$  are being discarded from the cache. Step four is the actual data mining step, where the shrunk cache  $F'_t$  is being replenished with newly found hard negatives in  $F(Z_p)$ .

---

**7.2 Latent SVM with contextual loss**

---

In the following section we describe how contextual model is being trained with a Latent SVM analogous to the LSVM Data-mining algorithm described above.

---

<sup>94</sup> See Felzenszwalb et al. [2010a, p. 9].<sup>95</sup> See Felzenszwalb et al. [2010a, p. 9f].<sup>96</sup> See Felzenszwalb et al. [2010a, p. 9f].



### 7.2.1 Semi-convexity

To be able to apply the LSVM Data-mining algorithm to our contextual model, we have to show that its optimization problem is semi-convex as well.

The changes to the model structure in this work are limited to appending existing elements of the model structure to form a metamodel, basically adding another hierarchy level. Two deformable parts models in accordance with Felzenszwalb et al. are being grouped and used as components of a new model, linked by deformation rules. Thus, this resulting metamodel can be expected to have the same properties.

It still has to be shown that the objective function also is convex.

With our new scoring function  $\text{score}_{\text{context}}$  the function, which scores an example  $x$  changes from

$$f_{\beta}(x_i) = \beta \cdot \Phi(x_i, z_i(\beta)) \quad (49)$$

in the original deformable parts model to

$$g_{\beta}(x_i) = \beta \cdot \tilde{\Phi}(x_i, z_i(\beta)) + \min(0, \beta \cdot \hat{\Phi}(x_i, z_i(\beta))) \quad (50)$$

$$= \beta \cdot \tilde{\Phi}(x_i, z_i(\beta)) - \max(0, -\beta \cdot \hat{\Phi}(x_i, z_i(\beta))) \quad (51)$$

in the contextual model.

To be able to discern between the object and context part of the model parameters, the feature vectors  $\tilde{\Phi}(x_i, z_i(\beta))$  and  $\hat{\Phi}(x_i, z_i(\beta))$  are being introduced in the previous function. These vectors are sparse and contain the values  $\neq 0$  of  $\Phi(x_i, z_i(\beta))$  of the respective sub-model.  $\tilde{\Phi}(x_i, z_i(\beta))$  represents the object sub-model, while  $\hat{\Phi}(x_i, z_i(\beta))$  represents the context sub-model so that

$$\Phi(x_i, z_i(\beta)) = \tilde{\Phi}(x_i, z_i(\beta)) + \hat{\Phi}(x_i, z_i(\beta)). \quad (52)$$

Also there are flavours of  $f_{\beta}(x_i)$  which only score either the context or object in  $x_i$ :

$$\tilde{f}_{\beta}(x_i) = \beta \cdot \tilde{\Phi}(x_i, z_i(\beta)) \quad (53)$$

and

$$\hat{f}_{\beta}(x_i) = \beta \cdot \hat{\Phi}(x_i, z_i(\beta)). \quad (54)$$

The SVM loss function corresponding to  $g_{\beta}(x_i)$  consists of two loss functions

$$L(s) = \max(0, 1 - s) \quad (55)$$

for object classification, and

$$L_{\text{context}}(s) = -\min(0, s) \quad (56)$$

for the context classification, which are being nested to achieve the wanted dependency between object and context – see (57). The output of  $L_{\text{context}}$  is being used as a modifier to the input for  $L$ . The variable  $s$  in our case would be filter response scores by either the object or the context model.

The resulting *contextual loss* function with the context score  $s_0$  and object score  $s_1$  is

$$L_{\text{contextual}}(s_0, s_1) = L(s_1 + L_{\text{context}}(s_0)). \quad (57)$$

In Figure 7 one can see that the value of  $L_{\text{contextual}}(s_0, s_1)$  is being influenced for  $s_0 < 0$ . That means the loss increases if  $s_0$  is too low. For  $s_0 \geq 0$  the graph of  $L_{\text{contextual}}$  is equal to the standard SVM hinge loss  $\max(0, 1 - y_i \tilde{f}_{\beta}(x_i))$ . In other words, a context score  $s_0$  below zero shifts the standard hinge loss left. A way to make this clearer is to alter the SVM hinge loss notation from  $\max(0, 1 - y_i \tilde{f}_{\beta}(x_i))$  in a standard SVM to

$$\max(0, a - y_i \tilde{f}_{\beta}(x_i)) \text{ where } a = 1 - \max(0, \hat{f}_{\beta}(x_i)). \quad (58)$$

The original soft margin optimization problem with the slack variable  $\xi_i$  and a linear loss function is defined by

$$\min_{\beta, \xi_i, b} \left( \frac{1}{2} \|\beta\|^2 + C \sum_i \xi_i \right) \quad \text{s.t. } y_i(\beta x_i - b) \geq 1 - \xi_i, \quad \xi_i \geq 0. \quad (59)$$



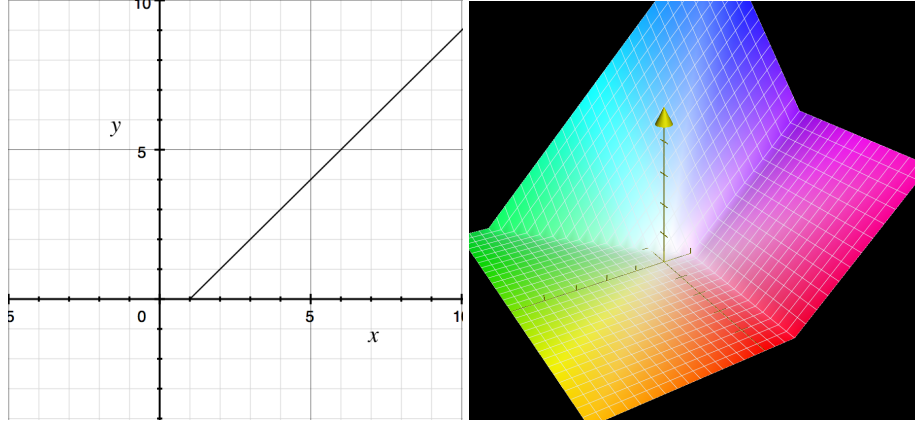


Figure 7: 2D plot of the loss functions  $L$ (left) and 3D plot of  $L_{\text{contextual}}(s_0, s_1)$ (right)

Using the standard loss function and the Latent SVM according to Felzenszwalb et al. [2010a, p. 8], the original objective function is defined as (40):  $L_D(\beta) = \frac{1}{2}\|\beta\|^2 + C \sum_i \max(0, 1 - y_i f_\beta(x_i)) = \frac{1}{2}\|\beta\|^2 + C \sum_i L(y_i f_\beta(x_i))$ .

Now the contextual model's objective function can be constructed. Felzenszwalb et al. "train  $\beta$  from labeled examples  $D = (\langle x_1, y_1 \rangle, \dots, \langle x_n, y_n \rangle)$ , where  $y_i \in \{-1, 1\}$ , by minimizing the objective function" <sup>97</sup> (40). An objective function can be formed which reflects the overall model's scoring function (50):

$$L_{D_{\text{contextual}}}(\beta) = \frac{1}{2}\|\beta\|^2 + C \sum_i y_i g_\beta(x) \quad (60)$$

$$= \frac{1}{2}\|\beta\|^2 + C \sum_i L(y_i \tilde{f}_\beta(x_i) - y_i L_{\text{context}}(\hat{f}_\beta(x_i))) \quad (61)$$

$$= \frac{1}{2}\|\beta\|^2 + C \sum_i \max(0, 1 - y_i \tilde{f}_\beta(x_i) + y_i \max(0, -\hat{f}_\beta(x_i))). \quad (62)$$

The loss function's subgradients are as follows:

$$\nabla L_{D_{\text{contextual}}}(\beta) = \beta + C \sum_i h(\beta, x_i, y_i) \quad (63)$$

$$h(\beta, x_i, y_i) = \begin{cases} 0 & \text{if } y_i \tilde{f}_\beta(x_i) - y_i \max(0, -\hat{f}_\beta(x_i)) \geq 1 \\ -y_i \tilde{\Phi}(x_i, z_i(\beta)) & \text{if } y_i \tilde{f}_\beta(x_i) < 1 \wedge \hat{f}_\beta(x_i) \geq 0 \\ -y_i \tilde{\Phi}(x_i, z_i(\beta)) - y_i \hat{\Phi}(x_i, z_i(\beta)) & \text{otherwise} \end{cases} \quad (64)$$

Now, having defined the optimization problem by  $L_{D_{\text{contextual}}}(\beta)$ , we have to show that it is in fact semi-convex according to Felzenszwalb et al. [2010a]. Recall that the hinge loss in a linear SVM is convex for  $y_i = -1$ . We now have to show that this is also true for our contextual loss. It is obvious that there is a problem with the convexity w.r.t.  $\hat{f}_\beta(x_i)$ :

$$\max(0, 1 - y_i \tilde{f}_\beta(x_i) + y_i \max(0, -\hat{f}_\beta(x_i))) \quad (65)$$

$$\text{Let } y_i = -1 \quad (66)$$

$$\implies \max(0, 1 + \tilde{f}_\beta(x_i) - \max(0, -\hat{f}_\beta(x_i))) = \max(0, 1 + \tilde{f}_\beta(x_i) + \min(0, \hat{f}_\beta(x_i))) \quad (67)$$

We can see that  $\min(0, \hat{f}_\beta(x_i))$  is concave, hence, the loss is not convex for  $y_i = -1$ . This also gets apparent when looking at the sub-gradient of the objective function.

That the objective function in this state is not semi-convex can also be explained intuitively. Now, if the contextual model would be trained with equation (62), one would encounter the problem that the context model would only be affected

<sup>97</sup> Felzenszwalb et al. [2010a, p. 8].

by the positive examples in a way that the context scores would increase until  $\hat{f}_\beta(x_i) \geq 0$  for all positive examples. Negative examples loose impact as soon as their context score  $\hat{f}_\beta(x_i) \geq 0$ . One could see that this optimization problem is not semi-convex. Given a context model which scores every example with a score larger than zero, there would be no pressure by the objective function to train it anymore.

With each relabeling step new positive examples can be detected with context scores below zero. In the following gradient descent step, the context model parameters increase again, so that more positive and negative examples have a greater or equal than zero context score. This happens until all positive examples have a context score of  $> 0$ . Thus, the optimal model parameters could never get reached. To break this cycle, we have to introduce additional constraints to the objective function.

## 7.2.2 Semi-convexity through context regularization

The solution is to make use of an additional class label  $\hat{y}_i$  for the context classifier  $\hat{f}_\beta(x)$ . That requires that additional training data has to be acquired. The context label  $\hat{y}_i$  is being introduced. It is independent from the object class label  $y_i$  and therefore context bounding boxes have to be annotated in negative example images. Hence, the training dataset  $D$  has to be extended with context labels  $\hat{y}_i$ . The resulting dataset is called  $E$ . With these extended labeled examples  $E = (\langle x_1, y_1, \hat{y}_1 \rangle, \dots, \langle x_n, y_n, \hat{y}_n \rangle)$  where  $y_i, \hat{y}_i \in \{-1, 1\}$ ,

We explore two ways to avoid context model parameters from drifting to unreasonably large values. In the first approach the contextual hinge loss from (62) is modified directly by adding negative pressure to context model parameters, in the second approach an additional weighted standard SVM hinge loss is being added to the objective function (62).

### 1. Integrated context regularization with additional context class label

For the first approach we now consider the four resulting cases of combining the possible values of the object class  $y_i$  and context class  $\hat{y}_i$ .

When applying the soft margin optimization problem

$$\frac{1}{2} \|\beta\|^2 + C \sum_i \xi_i \quad (68)$$

to this model, with the assumption that a positive object example always means that there is also a context nearby, we only need to distinguish between the following three cases:

$$\begin{aligned} \text{s.t. } \quad & \forall : y_i = +1 \Rightarrow \hat{y}_i = +1 \quad \tilde{f}_\beta(x_i) - \max(0, -\hat{f}_\beta(x_i)) \geq 1 - \xi_i \\ & \forall : y_i = -1 \wedge \hat{y}_i = +1 \quad -\tilde{f}_\beta(x_i) \geq 1 - \xi_i \\ & \forall : y_i = -1 \wedge \hat{y}_i = -1 \quad -\tilde{f}_\beta(x_i) - \hat{f}_\beta(x_i) \geq 1 - \xi_i \end{aligned} \quad (69)$$

When comparing (62) with (69) the cases with  $y_i = -1$  are handled differently. The max function, which previously enforced the context score threshold, is gone. This change is based on the following assumption, which is very similar to the way standard linear SVMs are being trained. Usually the class label  $y_i$  is being used to force the score of positives and negatives into opposite directions when optimizing by giving them inverted loss functions. In our case a second class label  $\hat{y}_i$  has to be used to define 'different' loss functions for each combination – which are later of course combined into one loss function, with parameters  $y_i$  and  $\hat{y}_i$ .

**Correctness Assumption:** The threshold which is enforced by max is only relevant for  $\hat{f}_\beta(x_i) > 0$ . Now we want the final trained model to have  $\hat{f}_\beta(x_i) < 0 \forall x_i, i \in \mathbb{N}$  where  $\hat{y}_i = -1$ . This would mean that in the final model all examples with  $\hat{y}_i = -1$  would have negative context scores, hence, thresholding with max would have no effect. This means that the loss function is deliberately altered, depending on the object and context label configuration. Now each label configuration has an own loss term. Because these changes do not affect the final model – assumed the context appearances training data can be classified correctly in the end ( $\hat{y}_i$ ). While in reality there might certainly be some outliers in the training set that are impossible to classify correctly, if the training data has a certain quality the model should classify most of the context training data correctly. In the training context, given the model can reach good results on the training data, this might be a feasible approach to regularize the context parameter while training.

**Correctness assumption:** assume that  $\hat{y}_i = \text{sign}(\hat{f}_\beta(x_i)) \forall \langle x_i, y_i, \hat{y}_i \rangle \in E$  after the contextual model has been trained.

Based on knowledge we gain from  $\hat{y}_i$ , limiting  $\hat{f}_\beta(x_i)$  using max is being omitted in cases of  $\hat{y}_i = -1$ . This basically means negative context examples have a different loss function than positive context examples. This is analogous to the standard SVM hinge function, where the example's class label determines the loss term.

That way the training algorithm “knows” that it can decrease a negative examples’ loss (and score) by reducing the context’s score. This would not be the case if the context score would still be thresholded in the objective function because then the gradient w.r.t. context model parameters would equal zero. Ultimately, the training algorithm would have no indicator how to decrease the loss of negative examples using the context.

1. Case one ( $y_i = +1 \Rightarrow \hat{y}_i = +1$ ) covers the positive object examples. This case is equivalent to the global scoring function  $f_\beta(x_i)$ . The other two cases differentiate between available and unavailable context.
2. For the second case the knowledge about a positive context ( $\hat{y}_i = +1$ ) tells us that in this instance a successful classification has to be achieved through object model.
3. In the third case  $y_i = -1 \wedge \hat{y}_i = -1$  a classification should be made as a joint effort by the object classifier and context classifier, this assumption means that in this case  $\hat{f}_\beta$  ought to be negative. This assumption is reinforced by the context class label, since a negative context label in the training data should mean that this example would get a negative score/sign in the trained model. This is why we are removing the threshold previously enforced by the max function.

This results in the following loss function.

$$y_i \tilde{f}_\beta(x_i) - \left( \frac{y_i + 1}{2} \right) \max(0, -\hat{f}_\beta(x_i)) - \left( \frac{-y_i + 1 - \hat{y}_i + 1}{2} \right) \hat{f}_\beta(x_i) \geq 1 - \xi_i \quad (70)$$

$$\Rightarrow L_E(\beta) = \frac{1}{2} \|\beta\|^2 + C \sum_i \max \left( 0, 1 - y_i \tilde{f}_\beta(x_i) + \left( \frac{y_i + 1}{2} \right) \max(0, -\hat{f}_\beta(x_i)) + \left( \frac{-y_i + 1 - \hat{y}_i + 1}{2} \right) \hat{f}_\beta(x_i) \right) \quad (71)$$

Now we show that this loss function forms a semi-convex optimization problem for all combinations of  $y_i$  and  $\hat{y}_i$ . For argumentation purposes we first show semi-convexity for case three and one, and lastly for case two.

- ( $y_i = -1 \wedge \hat{y}_i = -1$ ) If both object and context class are negative, then the loss for these examples is defined as  $\max(0, 1 + \tilde{f}_\beta(x_i) + \hat{f}_\beta(x_i))$ . Equations (53, 54) and (52) show that for negative examples contextual loss and linear SVM hinge loss with scoring function (39) are equal, that is

$$\max(0, 1 + \tilde{f}_\beta(x_i) + \hat{f}_\beta(x_i)) = \max(0, 1 + f_\beta(x_i)), \quad (72)$$

which proves that is convex for negative examples.

- ( $y_i = +1 \Rightarrow \hat{y}_i = +1$ ) In an LSVM the optimization problem (71) is not convex for positive examples, but it becomes convex when we limit latent values per example  $|Z(x_i)| = 1$ . This works completely analogous to the way semi-convexity has been shown for the linear SVM hinge loss.
- ( $y_i = -1 \wedge \hat{y}_i = +1$ ) This case is rather simple. With  $\hat{y}_i = 1$  the loss function (71) for an example becomes equal to the linear SVM hinge loss:

$$\max \left( 0, 1 - y_i \tilde{f}_\beta(x_i) + \left( \frac{y_i + 1}{2} \right) \max(0, -\hat{f}_\beta(x_i)) + \left( \frac{-y_i + 1 - \hat{y}_i + 1}{2} \right) \hat{f}_\beta(x_i) \right) \text{ with } y_i = -1 \wedge \hat{y}_i = +1 \quad (73)$$

$$\Rightarrow \max(0, 1 - y_i \tilde{f}_\beta(x_i) + (0 \cdot \max(0, -\hat{f}_\beta(x_i)) + (0 \cdot 0) \hat{f}_\beta(x_i)) \quad (74)$$

$$\Rightarrow \max(0, 1 - y_i \tilde{f}_\beta(x_i)) \quad (75)$$

Since we know that the standard hinge loss is convex for negative examples this case is also semi-convex.

After showing that the optimization problem defined by  $L_E(\beta)$  is indeed semi-convex we, show that  $L_E(\beta)$  can be derived directly from the non-semi-convex optimization problem defined by (62). One can show that equation (71) is in fact equivalent to the contextual loss term of (62) when substituting the fixed class label  $\hat{y}_i$  with  $\text{sign}(\hat{f}_\beta(x_i))$ , where

$$\text{sign}(a) = \begin{cases} -1 & a < 0 \\ 1 & a > 0 \end{cases}, a \in \mathbb{R}. \quad (76)$$

This also means it reflects the scoring function (50).

$$y_i \tilde{f}_\beta(x_i) - \left( \frac{y_i + 1}{2} \right) \max(0, -\hat{f}_\beta(x_i)) - \left( \frac{-y_i + 1 - \hat{y}_i + 1}{2} \right) \hat{f}_\beta(x_i) \quad (77)$$

$$\iff y_i \tilde{f}_\beta(x_i) - \begin{cases} \max(0, -\hat{f}_\beta(x_i)) & \text{if } y_i = 1 \\ \left( \frac{-\hat{y}_i + 1}{2} \right) \hat{f}_\beta(x_i) & \text{if } y_i = -1 \end{cases} \quad (78)$$

$$\iff y_i \tilde{f}_\beta(x_i) - \begin{cases} \max(0, -\hat{f}_\beta(x_i)) & \text{if } y_i = 1 \\ 0 & \text{if } \hat{y}_i = 1 \\ \hat{f}_\beta(x_i) & \text{if } \hat{y}_i = -1 \end{cases} \quad \text{if } y_i = -1 \quad (79)$$

$$\text{let } \hat{y}_i := \text{sign}(\hat{f}_\beta(x_i)) \quad (80)$$

$$\implies y_i \tilde{f}_\beta(x_i) - \begin{cases} \max(0, -\hat{f}_\beta(x_i)) & \text{if } y_i = 1 \\ 0 & \text{if } \text{sign}(\hat{f}_\beta(x_i)) = 1 \\ \hat{f}_\beta(x_i) & \text{if } \text{sign}(\hat{f}_\beta(x_i)) = -1 \end{cases} \quad \text{if } y_i = -1 \quad (81)$$

$$\iff y_i \tilde{f}_\beta(x_i) - \begin{cases} \max(0, -\hat{f}_\beta(x_i)) & \text{if } y_i = 1 \\ \min(0, \hat{f}_\beta(x_i)) & \text{if } y_i = -1 \end{cases} \quad (82)$$

$$\iff y_i \tilde{f}_\beta(x_i) - \begin{cases} \max(0, -\hat{f}_\beta(x_i)) & \text{if } y_i = 1 \\ -\max(0, -\hat{f}_\beta(x_i)) & \text{if } y_i = -1 \end{cases} \quad (83)$$

$$\iff y_i \tilde{f}_\beta(x_i) - y_i \max(0, -\hat{f}_\beta(x_i)) \quad (84)$$

$$\iff y_i (\tilde{f}_\beta(x_i) - \max(0, -\hat{f}_\beta(x_i))) \quad (85)$$

$$\iff y_i g_\beta(x_i) \quad (86)$$

As one can see by substituting  $\hat{y}_i$  with  $\text{sign}(\hat{f}_\beta(x_i))$  with (80) the result is equivalent to the loss term in (62). When defining  $\hat{y}_i := \text{sign}(\hat{f}_\beta(x_i))$  we let the classifier  $\hat{f}_\beta(x_i)$  determine the class label.

Conversely we could say that by adding the information about context class label  $\hat{y}_i$  to the first proposed loss function (62) and thus enforcing the context class through labeled training data we end up with (69), again showing the requirement of the correctness assumption if  $\hat{y}_i = \text{sign}(\hat{f}_\beta(x_i))$ .

As noted in Felzenszwalb et al. [2010a] the standard latent SVM optimization problem is semi-convex, therefore relying on a fixed set of positive examples to become convex. If all cases in (69) would be handled with the term

$$y_i \tilde{f}_\beta(x_i) - y_i \max(0, -\hat{f}_\beta(x_i)) \geq 1 - \xi_i \quad (87)$$

the optimization problem would not be convex, even with defining  $|Z(x_i)| = 1$ , since  $\hat{y}_i$  would not be used to regularize the context parameters. Recall that an SVM optimization algorithm wants to maximize the margin between the Hyperplane defined by the SVM and the positive and negatives examples. In a convex optimization problem the algorithm should be able to modify all parameters. However, as noted above, the sub-gradients of (87) can become zero w.r.t. the context parameters for negative examples  $y_i$ , thus being stuck at certain suboptimal values.

If the context score is always greater zero, the sub-gradients are always zero for context parameters in (64). However, defining the optimization problem by (69) solves this problem.

With (71) we can define the objective function:

$$L_E(\beta) = \frac{1}{2} \|\beta\|^2 + C \sum_i \max(0, 1 - y_i \tilde{f}_\beta(x_i) + \left( \frac{y_i + 1}{2} \right) \max(0, -\hat{f}_\beta(x_i)) + \left( \frac{-y_i + 1 - \hat{y}_i + 1}{2} \right) \hat{f}_\beta(x_i)) \quad (88)$$

The resulting subgradients are as follows:

$$\nabla L_D(\beta) = \beta + C \sum_i h(\beta, x_i, y_i, \hat{y}_i) \quad (89)$$

$$h(\beta, x_i, y_i, \hat{y}_i) = \begin{cases} 0 & \text{if } y_i \tilde{f}_\beta(x_i) - \left( \frac{y_i + 1}{2} \right) \max(0, -\hat{f}_\beta(x_i)) - \left( \frac{-y_i + 1 - \hat{y}_i + 1}{2} \right) \hat{f}_\beta(x_i) \geq 1 \\ -y_i \tilde{\Phi}(x_i, z_i(\beta)) & \text{if } y_i \tilde{f}_\beta(x_i) < 1 \wedge ((\hat{f}_\beta(x_i) \geq 0 \wedge y_i = +1) \vee (y_i = -1 \wedge \hat{y}_i = +1)) \\ -y_i \tilde{\Phi}(x_i, z_i(\beta)) - \hat{y}_i \hat{\Phi}(x_i, z_i(\beta)) & \text{otherwise} \end{cases} \quad (90)$$

The label  $\hat{y}_i$  is equivalent to  $y_i$  in row three of (90) due to first two cases. Row three can only be reached when  $y_i = \hat{y}_i$ . That means we can simplify the subgradient with (52) to:

$$h(\beta, x_i, y_i, \hat{y}_i) = \begin{cases} 0 & \text{if } y_i \tilde{f}_\beta(x_i) - \left(\frac{y_i+1}{2}\right) \max(0, -\hat{f}_\beta(x_i)) - \left(\frac{-y_i+1}{2} \frac{-\hat{y}_i+1}{2}\right) \hat{f}_\beta(x_i) \geq 1 \\ -y_i \tilde{\Phi}(x_i, z_i(\beta)) & \text{if } y_i \tilde{f}_\beta(x_i) < 1 \wedge ((\hat{f}_\beta(x_i) \geq 0 \wedge y_i = +1) \vee (y_i = -1 \wedge \hat{y}_i = +1)) \\ -y_i \Phi(x_i, z_i(\beta)) & \text{otherwise} \end{cases} \quad (91)$$

This also confirms that the context parameters (or rather just its bias parameter, since we do not retrain the context model) is never being modified alone, only in conjunction with the object parameters. Only the object parameters are being trained on their own if the context's score is large enough. This reflects the conditional scoring function (33) where the object score  $\text{score}_{def}(Z_1)$  is a mandatory term and  $\text{score}(Z_0)$  only added if it is below the threshold.

Models that have been trained using this approach are marked with the keyword “RegA” in the result section.

## 2. Independent context regularization with hinge loss

The other approach to regularize context parameters introduces another term to the objective function (62). The additional standard SVM hinge loss is supposed to keep the context parameters so that context training data is still being correctly classified.

$$L_E(\beta) = \frac{1}{2} \|\beta\|^2 + C \sum_i L_{\text{contextual}}(\hat{f}_\beta(x_i), y_i \tilde{f}_\beta(x_i)) + D \sum_j L(\hat{y}_j \hat{f}_\beta(x'_j)) \quad (92)$$

$$= \frac{1}{2} \|\beta\|^2 + C \underbrace{\sum_i \max(0, 1 - y_i \tilde{f}_\beta(x_i) + y_i \max(0, -\hat{f}_\beta(x_i)))}_{\text{Term 1: contextual loss (57)}} + D \underbrace{\sum_j \max(0, 1 - \hat{y}_j \hat{f}_\beta(x'_j))}_{\text{Term 2: standard hinge loss (55)}}. \quad (93)$$

As shown before with (62) also (93) is not semi-convex. Though (93) is semi-convex in  $\tilde{f}_\beta(x)$ . This means that just the parameters in  $\beta$  where  $\tilde{\Phi}(x_i, z_i(\beta))$  is non-zero will step in the wrong direction (of course this later also influences the other ones). To prevent that, another standard hinge loss term is being introduced – see Term 2 in (93), which we know is semi-convex. This term only trains the context parameters. Unlike in the previous approach the context examples  $x'_j$  that are used in this term can also come from a completely different training set.

This creates a competitive relationship between the contextual loss term and the added standard hinge loss. As discussed before in 7.2.2 term 1 tends to larger  $\tilde{f}_\beta(x_i)$  to ensure there is no context penalty on positive examples. Term 2 on the other hand now tends to keep the parameters reasonable to be able to classify the context class correctly. The variable  $D$  is being used to change the relative weight between all terms and the impact of the new context regularization term.

Models that have been trained using this approach are marked with the keyword “RegB” in the result section.

## Comparison of context regularization methods

The two context regularization methods are very different. The first approach works analogous to a standard hinge loss where  $y_i$  is already known, by also fixing the context class  $\hat{y}_i$  while training. This solves the problem of the previous non-convex objective function.

Both approaches require additional labels for the training data. Without  $\hat{y}_i$  the context parameters in  $\beta$  cannot be prevented from deviating to a local minimum. Still, the training data requirements are still different. While the integrated approach with (88) requires both object and context labels (bounding boxes) to be labeled in the same training data, (93) can work with an integrated trainingset as well as with two different trainingsets for the object and context classes. This can be a huge disadvantage for the first approach. Most of the time training sets of objects only contain bounding boxes of a single class. Even if there are multiple class annotations in a single data set, there is still a big chance that these classes together do not have a relationship as stated in our assumptions on page 1.

In these cases it could be more feasible to train a contextual model through the independent context regularization approach. Training sets for both model and context could be selected independently. Existing datasets could be recombined to test out different object/context combinations. In our case of clothing objects and person context models we could fall back on an existing extensive amount of annotated human datasets. Only the clothing dataset would have to be generated from scratch. Of course one would have to choose only images for the clothing dataset where also the context is available. It does not have to be annotated though, thus, saving time.

---

Other than that, the independent context regularization approach has many drawbacks. There is another constant  $D$  which together with  $C$  defines the relative weights of the three loss terms.  $D$  is not independent from  $C$  through the term  $\frac{1}{2}||\beta||^2$ , which, together with a typical model training time of several hours, makes it very time-consuming to find suitable values. Also it is hard to show that the optimization problem becomes semi-convex by adding the additional hinge loss for the context model.

The integrated approach on the other hand is a) proven to be semi-convex and b) does not introduce additional weights between object and context model – the impact of the context model in the contextual model is instead trained implicitly. For this reason we decide primarily use the first approach, objective function  $L_E(\beta)$  as defined in (88).

---

## 8 Training contextual models

---

A contextual mixture model is being initialized by combining fully trained deformable part mixture models, in our case one object class mixture model and one context class mixture model. In this section model initialization and training for deformable parts models, according to Felzenszwalb et al., will be recapitulated.

---

### 8.1 Deformable parts model initialization

---

The models are being trained by the LSVM coordinate descent algorithm. It “is susceptible to local minima and thus sensitive to initialization”<sup>98</sup>.

The deformable part mixture models are being initialized in multiple phases. We have positive training data  $P$  containing image, bounding box pairs  $(I, B)$ . Negative training data  $N$  just contains images.

1. In the first phase the root filters  $F_i$ <sup>99</sup> for  $m$  components are being initialized. Bounding boxes from the training data are being sorted by their aspect ratio and divided into  $m$  equally sized groups. “Aspect ratio is used as a simple indicator of extreme intraclass variation.”<sup>100</sup> Now for each component group root filters  $F_i, i \in [1, \dots, m]$  are being trained. Their dimensions are being defined by the mean aspect ratio of the current group and the area at the 80th percentile. “This ensures that for most pairs  $(I, B) \in P_i$  we can place  $F_i$  in the feature pyramid of  $I$  so it significantly overlaps with  $B$ .”<sup>101</sup>  $F_i$  is being trained with a standard SVM. Positive examples are extracted directly from the image region under each bounding box  $B$  in  $P_i$  and warped to obtain a feature map with the same dimensions as the root filter. Negative examples in the filter size are extracted randomly from the negative data set  $N$ .
2. In phase two the components, respectively the root filters, are being merged into a single mixture model. Now the mixture model is being trained using the full training datasets  $P$  and  $N$  and the LSVM algorithm. The latent variables in this phase are the root location and the component label. As Felzenszwalb et al. point out, the LSVM training algorithm can be interpreted as an alternating clustering method. In the relabeling step each example is assigned a mixture label (equivalent to the cluster label) while in the gradient descent the root filters representing the cluster “means” are being estimated.<sup>102</sup>
3. In phase three the model is being extended by adding parts for each component. Felzenszwalb et al. are proposing the heuristic of choosing six parts per component and placing them greedily in high energy regions of the root filter.<sup>103</sup> The energy of the region where the part has been placed is set to zero and the next best high energy region is being chosen for the next part, until all parts have been chosen. The part weights are initialized by doubling the resolution of the root filter and extracting the weight in the part’s area.<sup>104</sup> Felzenszwalb et al. [2010a] allow for a small set of different rectangular part shapes and require the parts to be either centered vertically or have a symmetric counterpart. In their newer code release from 2011 Felzenszwalb et al. only use quadratic part shapes of a single size.<sup>105</sup> Additionally, they introduce mirrored components to the mixture model which makes it possible to place the parts everywhere in the root filter without creating symmetric counterparts. Each part does not have its symmetric counterpart inside the same component model, but rather in another completely symmetric component model.<sup>106</sup> Deformation model parameters for all parts are being initialized to  $d_i = (dy, dx, dy^2, dx^2) = (0, 0, 0.1, 0.1)$  which “pushes part locations to be fairly close to their anchor position”<sup>107</sup>.

---

### 8.2 Training algorithm

---

Felzenszwalb et al. define the procedure *Train* by combining the LSVM training algorithm in section 7.1 and the data-mining of hard examples algorithm 3. To train a model for an object class  $c$ , the procedure expects positive examples  $P$ , negative images  $N$  and the initial model parameters  $\beta$ .  $P$  contains pairs of images and bounding boxes  $P = \{(I_1, B_1), \dots, (I_n, B_n)\}$  defining object locations,  $N$  is a set of background images – images without objects of class  $c$ .  $\beta$  are the parameters of a mixture model  $M$  that has been initialized according to the method described above.<sup>108</sup>

---

<sup>98</sup> Felzenszwalb et al. [2010a, p. 11].

<sup>99</sup> Each root filter  $F_i$  in a mixture model represents  $F_0$  in single model, see equation(2).

<sup>100</sup> Felzenszwalb et al. [2010a, p. 11].

<sup>101</sup> Felzenszwalb et al. [2010a, p. 12].

<sup>102</sup> See Felzenszwalb et al. [2010a, p. 12].

<sup>103</sup> See Felzenszwalb et al. [2010a, p. 12].

<sup>104</sup> See Felzenszwalb et al. [2010a, p. 12].

<sup>105</sup> Possibly to speed up the feature extraction. With parts of the same size only one feature map per window location has to be extracted from the feature pyramid. This could increase performance significantly.

<sup>106</sup> See Felzenszwalb et al. [2011]

<sup>107</sup> Felzenszwalb et al. [2010a, p. 12].

<sup>108</sup> See Felzenszwalb et al. [2010a, p. 11].



As described above, the procedure (see algorithm 4) alternates between optimizing the latent values “by selecting the highest scoring latent value for each positive example”<sup>109</sup>, and optimizing  $\beta$ . Both steps are being executed multiple times resulting in nested **for** loops, the outer *relabel*-loop and the inner *datamine*-loop. The variable *num-relabel* determines how often the outer loop is being executed, where all positive examples are being relabeled, hence, optimizing the latent values  $Z_p$ . The variable *num-datamine* determines how often hard negatives are being mined and the stochastic gradient descent algorithm is being carried out. The variable *memory-limit* depends on the runtime environment and on the size of each feature vector  $(i, v) \in F_n$ .

---

**Algorithm 4** Procedure Train<sup>110</sup>

---

```

function TRAIN( $P, N, \beta$ )
   $F_n \leftarrow \emptyset$ 
  for relabel = 1  $\rightarrow$  num-relabel do
     $F_p \leftarrow \emptyset$ 
    for  $i = 1 \rightarrow n$  do
      Add detect-best( $\beta, I_i, B_i$ ) to  $F_p$ 
    end for
    for datamine = 1  $\rightarrow$  num-datamine do
      for  $j = 1 \rightarrow m$  do
        if  $|F_n| < \text{memory-limit}$  then
          Add detect-all( $\beta, J_j, -(1 + \delta)$ ) to  $F_n$ 
        end if
         $\beta \leftarrow \text{gradient-descent}(F_p \cup F_n)$ 
        Remove  $(i, v)$  with  $\beta \cdot v < -(1 + \delta)$  from  $F_n$ 
      end for
    end for
  end for
end function

```

---

In general the algorithm could also stop if **while** loops instead of fixed-length **for** loops would be used. This is not being reflected in the depicted Algorithm 4. The first loop has an exit condition, that stops as soon as the relabeling step improves the loss of all positive examples by less than 1%. This exit condition is implemented in the framework by Felzenszwalb et al. [2011]. The data-mining algorithm has an explicit exit condition, which stops the loop when all hard negatives of the training data  $N$  are in the feature cache  $F_n$ .<sup>111</sup>

Felzenszwalb et al. explained, that the algorithm runs a fixed number of iterations for practical reasons. While not further describing these practical reasons, one could imagine that it is reasonable to prevent an algorithm from running an indefinite amount of time.

When training models using the PASCAL training data, they execute the procedure *Train* with *num-relabel* = 8 and *num-datamine* = 10, while using a smaller subset  $N'$  of their negative images  $N$ . Then they run an additional data-mining loop – see Algorithm 4 – with *num-datamine* = 5 for the whole set of negative images  $N$ .<sup>112</sup> This two-step approach might make sense to learn latent values fast using  $N'$ , and then use the whole set  $N$  to optimize  $\beta$  on its own. These details might be highly dependent on the training set.

---

### 8.3 Contextual model initialization

---

To construct a contextual mixture model we first train two deformable part mixture models according to the algorithm described above. The object class mixture model is being trained on our object training dataset. The context class mixture model on the other hand is preferably being trained on a disconnected training dataset to assure that object specific features have less impact.

After both mixture models have been trained, they can be merged to form an initial contextual mixture model. Since in each mixture model each component has a right and left facing version, two ways to merge the mixture models into a contextual model come into mind. We refer to these initialization options as *lr* and  $l \times r$ . The *lr* option groups each left facing component of the object model with each left facing component of the context model, the same is being done with the right facing components. The  $l \times r$  option on the other hand groups each component of the object model with

---

<sup>109</sup> Felzenszwalb et al. [2010a, p. 8].

<sup>110</sup> Felzenszwalb et al. [2010a, p. 11, Procedure Train].

<sup>111</sup> See Felzenszwalb et al. [2010a, p. 9f].

<sup>112</sup> See code in framework version 4, Felzenszwalb et al. [2011].



each of the context model's components. While the latter is more thorough and considers each possible combination of object and context appearance, the  $lr$  approach is simpler and faster to train. The  $lr$ -type contextual models contain half the amount of parameter than  $l \times r$ -type contextual models.

For example we have a deformable parts model  $A$  with 4 components (2 component sizes, each left and right) and a deformable parts model  $B$  with 2 components (1 component size, which also exists in left and right versions). Be  $A$  the object model and  $B$  the context model. Let  $A_i$  be the  $i$ -th component of  $A$  and  $B_i$  the  $i$ -th component of  $B$  respectively. When merging both models to a contextual mixture model using the  $lr$  approach one would get a 4-component mixture model with the component permutations  $[(A_1, B_1), (A_2, B_2), (A_3, B_1), (A_4, B_2)]$ . By using the  $l \times r$  approach one would get an 8-component model with the component permutations  $[(A_1, B_1), (A_1, B_2), (A_2, B_1), (A_2, B_2), (A_3, B_1), (A_3, B_2), (A_4, B_1), (A_4, B_2)]$ .

For example let's look at the number of components  $m$  of a contextual mixture model  $M_{contextual}$  which is being generated from the mixture model  $M_0$  with  $m_0$  components and context mixture model  $M_1$  with  $m_1$  components. Let every second component of  $M_0$  and  $M_1$  be a mirrored version of the previous component. Thus, each component exists in a left and in a right facing version, meaning we have an even amount of components in each sub-model,  $m_0 \% 2 = m_1 \% 2 = 0$ .

Using  $lr$  merging  $M_{contextual}$  would have  $m_{lr} = 2 \cdot (\frac{m_0}{2} \cdot \frac{m_1}{2})$  components. In other words, first the sub-models' left facing components are being grouped in all permutations and then all left facing components. In general  $m_{lr} = 2 \prod_{c=1}^n \frac{m_c}{2}$ , where  $n$  is the number of sub-models in  $M_{contextual}$ . Using  $l \times r$  merging  $M_{contextual}$  would have  $m_{l \times r} = m_0 \cdot m_1$  components. Hence, a model using  $l \times r$ -merging has  $2^n$  times more permutations, resulting in a much higher complexity and a proportional growth of the number of model parameters in  $\beta$ ,

$$m_{l \times r} = 2^{n-1} \cdot m_{lr}. \quad (94)$$

Each component pair  $(A_i, B_i)$  is then merged to a contextual model. All resulting contextual models together form one contextual mixture model. To merge a component pair we need to initialize the offset, scale and deformation cost parameters  $V_i$ ,  $S_i$  and  $D_i$  respectively as defined in (7), since these parameters define their spatial relation.  $V_i$  and  $S_i$  are calculated as the mean location and scale offsets of the object and context bounding boxes in the training dataset. It is worth noting that, while annotated data for their relative configuration is available, the object location w.r.t. its context is still considered to be a latent variable and is being trained. The reason for this is that the initial mean location and scale offsets can change when examples get assigned different component labels in the relabeling step of the training algorithm.

The deformation costs for each  $(A_i, B_i)$  are being initialized at  $D_i = (dy, dx, ds, dy^2, dx^2, ds^2) = (0, 0, 0, 0.01, 0.01, 0.02)$ , leaving the deformation costs low and pushing the object only slightly to its context. Each parameter in the model has customizable learning and regularization rate parameters. For the object deformation costs the same rates are chosen as the part deformation parameters of the original deformable parts model by Felzenszwalb et al.. These parameters could have effects on model performance, but, since the results of changing learning and regularization rates can only be observed after training the whole model, experiments were not feasible due to time constraints.

---

## 8.4 Contextual model training

---

If one wants to keep the context model independent from the object in a contextual model the learning and regularization rates for its parameters – except the bias parameter  $b$  – can be set to zero. Keeping the context model independent enables the reuse of the same model in multiple contextual models. We also do this for some of the contextual models we show in section 10. This adheres to the vision to reuse and share context models across multiple object categories. If a large dataset of different objects is available for training one could also leave the context model parameters trainable. More objects trained at the same time would decrease the dependency between a single object class and the context. In this case we are limited to keep the context model parameters static to simulate this independency because we only have one training dataset for the object category “dress”.

After the initialization the contextual model is being trained analogous to the deformable parts-based model, using either the objective function (71) or (93) defined in section 7.2.2.

The deformable part model's training algorithm is the same as in the work by Felzenszwalb et al.. There are several parts that need to be changed.

### Learning the threshold and model degeneration

Augmenting a deformable parts model with a context should lead to a model which performs equally or better than the original. Therefore, the influence of the contextual constraints should be weighed by the training algorithm. This makes the model robust to context models which do not significantly correlate with the object. In such a case the threshold

parameter  $b$  could be raised by the training algorithm to a level where all detection windows result in a positive context score. This would remove any context constraint and impact of deformation costs.

Imagine a contextual model  $M_{contextual} = (M_0, C_1)$  with their object hypothesis  $z_0$  and context hypothesis  $z_1$ . Now consider the  $C_1$ 's bias  $b_1$ . There must be a value for  $b_1$ , where

$$\lim_{b_1 \rightarrow \infty} \text{score}(z_1) \geq 0 \text{ with } z_1 = (p_{1,0}, \dots, p_{1,n_1}) \forall p_{1,i} \in \mathbb{N}^3, \quad (95)$$

where  $n_1$  is the number of parts of context model  $C_1$ . With (19) this implies that the impact of any context hypothesis  $z_1$  on the score of the overall  $Z$  is completely mitigated.

$$\Rightarrow \lim_{b_1 \rightarrow \infty} \text{score}_{contextual}(Z) = \text{score}_{def}(Z_1) + \min(0, \lim_{b_1 \rightarrow \infty} \text{score}(Z_0)) \quad (96)$$

$$= \text{score}(z_0) + 0 \quad (97)$$

$$= \text{score}(p_{0,0}, p_{0,1}, p_{0,2}, \dots, p_{0,n_0}) \quad (98)$$

$$= \text{score}(p_0, \dots, p_{n_0}). \quad (99)$$

In that case the contextual model  $M_{contextual}$  degenerates to a model, which is equivalent to the input object model  $M_0$  ( $\lim_{b_1 \rightarrow \infty} M_{contextual} = M_0$ ).

The opposite is also possible. If the bias  $b_0$  became low enough, the context score would never have to be thresholded. In that case the threshold becomes irrelevant and the contextual model just becomes the sum of its sub-models, ( $\lim_{b_1 \rightarrow -\infty} M_{contextual} = M_0 + M_1$ ).

This shows that the context model's bias parameter moderates the context's impact.

---

## 9 Implementation

---

### 9.1 Framework version 4

---

While this work is based on deformable parts models by Felzenszwalb et al. [2010a] the implementation is also based on their framework *voc-release4*.<sup>113</sup> This framework is programmed in Matlab as well as in C for modules demanding faster performance. The reason why Version 5 of the framework is not used here, is that it has not been published at the time of implementation of the contextual model described here. A further look at the newer framework and how its new features could be leveraged has been appended to the end of this section.

#### Model blocks

The model structure itself resides in Matlab. The model parameters in  $\beta$  are encoded as a weight vector and divided and stored in *blocks*. Blocks represent related weights. Each element of deformable parts model  $M$ <sup>114</sup>,  $F_i$ ,  $v_i$ ,  $d_i$  or  $b$  has their own block. For instance each part's weights are being stored in their own block. Each block has an associated *blocklabel*. Having blocks instead of a single weight vector allows for sparse feature vectors, which can speed up dot product calculation for mixture models needed in the gradient descent step.<sup>115</sup> This also enables the possibility to identify different types of blocks, generating multiple dot products and handling them differently in the gradient descent.<sup>116</sup> This is a core requirement to be able to implement the contextual model without introducing deep changes to the model structure.

#### Grammar

Additionally, the model contains *symbols* and *rules*. Felzenszwalb et al. [2011] constructed their model so that it can be described by an acyclic grammar out of symbols. Each symbol has a label and a type. Their types can either be *terminal* or *non-terminal*. Each terminal symbol represents a filter (like a root or a part filter), while each non-terminal symbol represents a rule. Rules represent the grammar's productions.<sup>117</sup> A rule holds information about the left hand side (lhs, its own symbol label) and the right hand side (rhs) of the production. The *rhs* can consist of one or more symbol labels.

In their framework Felzenszwalb et al. use two different rules:

- **Structural rules** merge all symbols on the *rhs* by adding their score pyramids and applying an offset to the resulting score pyramid which is stored in the *lhs*-symbol. This offset represents the bias  $b$  of model  $M$ . Therefore, the offset has an associated block to store its weight.
- **Deformation rules** apply distance transform to the *rhs*-symbol. The rule contains the deformation cost parameters which like the structural rule's offset have an associated block. When this rule is being applied, the modified score pyramid is stored and together with lookup tables generated as a by-product of the distance transform algorithm. These lookup tables allow backtracking of the deformation in  $x$  and  $y$  direction which is needed to look up the original part locations before deformation. With these part locations the features can be extracted and stored in feature vectors used in the gradient descent.

#### Symmetry

The mixture models' symmetry allows for a parameter reduction by half.<sup>118</sup> In Felzenszwalb et al. [2010a] model components could only be self-symmetric. This means that parts in the model had to be positioned in a symmetric manner. In this case parameter reduction could be achieved within a single model. The framework *voc-release4* Felzenszwalb et al. [2011] introduced asymmetric models, in the sense that now left and right orientations of objects could be modelled. To make it possible to model non-symmetric objects each model has a mirrored counterpart, e.g. a car from the left and a car from the right side. The parameters of this mirrored counterpart do not need to be redundantly stored. Instead a "flip"-flag is being added to the mirror filters and rules, which lets the framework know that these are just mirrored copies of the original filters and rules.

---

<sup>113</sup> See download at Felzenszwalb et al. [2011]

<sup>114</sup> See equation (2).

<sup>115</sup> See Felzenszwalb et al. [2011]

<sup>116</sup> Gradient descent with the subgradients in equation XYZ requires the knowledge of the individual object and context scores and need to modify object and context parameters individually as well. This is possible due to the division into blocks.

<sup>117</sup> See "documentation.txt" in Felzenszwalb et al. [2011]

<sup>118</sup> See Felzenszwalb et al. [2010a, p. 11].

---

## 9.2 Extended framework

---

The grammar based model is a great opportunity for extensions. At least on the structural level this is easily done. The offset and deformation linking the object and context model as depicted in Figure 2 can be described by two rules.

- **Extended structural rules:** We need to extend the structural rules to be able to offset parts (or sub-models) by any arbitrary scale difference. While there is actually no need to change the data encoded in the rule, we need to change the parts of the program where structural rules are being processed. In the original framework only octave scale changes are supported by the implementation, while any scale change is allowed by the grammar. This means that the extended structural rule implementation can be used as a replacement of the old ones. The grammar does not need to change, but the implementation does. The final program does not need to make a distinction between extended structural rules and original structural rules.
- This is not the case for **extended deformation rules:** These rules cannot be used as a replacement for original deformation rules, since they introduce new grammar through new data and functionality. The extended deformation rule applies 3-dimensional distance transform to the whole score pyramid as described in section 5. As such the rule holds a weight vector with the length of six instead of four to include linear and quadratic scale deformation costs. Like in the original deformation rules,  $x/y$  coordinate lookup tables are being stored in the rule for backtracking. There is no scale-space lookup table though. Backtracking w.r.t. the scale level is being done using a simple brute force approach for reasons explained in the following section 9.3.

Before applying the 3-dimensional distance transform, the extended deformation rule also applies the threshold cap to the context scores according to the contextual scoring function (33).

---

## 9.3 Feature extraction and backtracking

---

Implementing the 3-dimensional distance transform has been one of the major changes enable the framework of Felzenszwalb et al. to support contextual models. In the earlier sections we described how generalized distance transform can be applied to score pyramids. In the following chapter we explain how the distance transform backtracking works through the scale levels of score pyramids.

To train the model positive and negative examples must be extracted from the detections. This is done by starting at the detection window and tracing back offsets and deformation to the part filter locations. The process to backtrack and extract an example contains three distinctive steps:

1. Backtracking offsets.
2. Backtracking deformation.
3. Extracting root and part filter features.

To backtrack offsets (e.g. used in spatial positioning of parts and sub models) the coordinates are being translated by the negative offset and scaled according to the scale level offset. All levels in the feature pyramid are padded by a fixed amount. This means that in addition to scaling, the virtual padding between two levels has to be subtracted from the resulting coordinate. This virtual padding can introduce rounding errors into the resulting coordinates.

$$p_{virtual}(\delta s) = p \cdot (2^{\delta s} - 1), p \in \mathbb{N} \quad (100)$$

In the case that all scale changes are either  $\frac{1}{2}$  or 2, which means that the scale change is always a whole octave<sup>119</sup>, the introduction of rounding error can be prevented by choosing an even valued padding.

$$p := 20 \quad (101)$$

$$p_{virtual}(1) = p \cdot (2^1 - 1) = 20 \quad (102)$$

$$p_{virtual}(-1) = p \cdot (2^{-1} - 1) = -10 \quad (103)$$

This is the fact for Felzenszwalb et al. “[In their] models the part filters capture features at twice the spatial resolution relative to the features captured by the root filters”<sup>120</sup>. While in our model the part filters are equivalent to the ones by Felzenszwalb et al., the spatial resolution of the features captured by the context model relative to the features captured by the object model is not fixed. The difference in resolution depends on the actual resolution of the trained object and context models, as well as their relative sizes in the real world or rather in the training data. Thus,  $\delta s$  may assume any value in  $\mathbb{R}$ .

---

<sup>119</sup> A scale change by an octave halves or doubles the frequency/resolution of the resulting feature matrix.

<sup>120</sup> See Felzenszwalb et al. [2010a, p. 2].

Therefore, virtual padding  $p_{virtual}$  can have non-integer values, also leading to non-integer coordinates. Since integer coordinates are needed to extract values in the matrix and perform further backtracking, these coordinates have to be rounded. This leads to several differences in the backtracking implementation between our code and the code supplied by Felzenszwalb et al.<sup>121</sup>. This will be explained below.

Additionally to the offsets, the newly introduced scale deformation of the context leads to an additional potential rounding error. While backtracking the deformation in  $x$  and  $y$  dimension is easily done with a coordinate lookup, the same is not possible for scale deformation. The lookup table can be generated as a byproduct of the actual distance transform. Scale deformation is applied to a grid matrix. Then the matrix is transformed back to the feature pyramid, thus, rendering the previously built lookup table useless.

In our implementation the scale dimension lookup table is completely discarded and instead a scan for the expected score takes place on the score pyramid. This is being done under the assumption that there is only one location on the score pyramid where deformed score minus scale deformation cost equals original score. This assumption should be true for the majority of examples. In reality the probability of this happening is very low as the score is calculated with dot products of double precision feature and filter vectors with dimensions which are usually larger than 1000.<sup>122</sup>

With  $n$  levels the algorithm starts at location  $S$  on level  $l_{start}$  with the score  $s$  to scan the score pyramid  $P$  for the expected score  $e$  and finally returns its source location  $A$ . The process can be written in pseudo code as shown in Algorithm 5.

---

**Algorithm 5** Backtracking function to scan score pyramid for expected score.

---

```

function SCANPYRAMIDFORSCORE( $P, s, l_{start}$ )
  for  $l \leftarrow 1, n$  do
     $e \leftarrow s - \text{deformationCost}(l_{start} - l)$ 
     $(w, h) \leftarrow \text{sizeForLevelInPyramid}(l, P)$ 
    for  $x \leftarrow 1, w$  do
      for  $y \leftarrow 1, h$  do
         $s_{source} \leftarrow P(x, y, l)$ 
        if  $e = s_{source}$  then
           $A \leftarrow (x, y, l)$ 
        end if
      end for
    end for
  end for
end function

```

---

In practice the implementation of this brute-force algorithm only scans a small area of each pyramid level for the expected score  $e$ . The area depends on the difference of the start level  $l_{start}$  and potential source level  $l$ . Is  $\delta l = l_{start} - l$  smaller or equal to one the area in which we scan for  $e$  is small. In this case the area should span the largest possible rounding error that can be expected from the scale deformation operation.<sup>123</sup> In the case that  $\delta l$  is larger than 1 we have to consider the error that may result from upsampling. The larger  $\delta l$ , the larger the error may be:

$$\text{error} = \left[ - \left\lceil 2^{\delta s} - 1 \right\rceil, \left\lceil 2^{\delta s} - 1 \right\rceil \right] \quad (104)$$

While the backtracking approach by Felzenszwalb et al. is much simpler, this brute force approach is still fast enough for training when several hundreds hard negatives can be detected per negative example image. When the model is nearing its optimum most images lead to less detections the scan does not affect detection speeds significantly. If one is only interested in the object's bounding box and not the part's the whole backtracking process can be ignored.

---

#### 9.4 Outlook: framework version 5

---

Girshick et al. released a new version 5 of the framework in late 2012. It improves the implementation in multiple ways.<sup>124</sup>

---

<sup>121</sup> See Felzenszwalb et al. [2011]

<sup>122</sup> Assuming a model of root filter and 8 parts, a root feature resolution of 40x40 and part feature resolutions of 20x20 and histogram of gradients for each feature with 9 buckets the feature and filter vectors would have a dimension of 43200.

<sup>123</sup> See transformation from score pyramid to 3d matrix to score pyramid in 5 3D distance transform on score pyramid

<sup>124</sup> See Girshick et al. [2012a] and Girshick et al. [2012b] for change log.

- 
- It now contains generalized code for training LSVM and weak labeled structured SVMs (WL-SSVM). The latter is used for training occlusion sensitive models in Girshick et al. [2011], which could help improving the contextual model later on.
  - It includes a cascade classifier to improve detection speed by Felzenszwalb et al. [2010b]
  - Improved optimizer, which speeds up convergence. The implementation also does not use hard disk to cache features anymore.
  - The model now includes a scale prior, which calibrates detection scores at different scales.
  - Improved modularity. For example objective functions can now be changed without editing many locations in the implementation code.

How these improvements could be used in the current contextual model is being described in section 12, Future work.

---

## 10 Empirical results

---

Before we will present the empirical results in this section, we describe the methodology of how the models have been trained exactly, what training and test data has been used and how that data has been generated. First we look at the training and testing data set.

---

### 10.1 Training and Testing Data

---

As described in the first section, one of the goals of the contextual model should be to improve the detection of clothing items which are being worn by a person. Therefore, our training and testing data focusses on clothing (objects) and the person wearing them (context). Unfortunately for comparison purposes there are no datasets available for our target domain – clothing. Thus, comparison with other object detection methods will be limited and training and testing data had to be generated from scratch.

Data annotation has been done through microwork platforms like Crowdfunder and Amazon Mechanical Turk. Images have been taken from the Fashionfreax.net image database. For the annotation steps we have used our own HTML5 canvas annotation tool, which supports segmentation with vertices as well as simple bounding boxes (see Figure 8).

The data generation process consists of 5 successive steps:

1. Preselecting images: In the first step images from the Fashionfreax.net image database which have been tagged with certain clothing categories have been selected. The clothing categories for this step have been:
  - bags
  - boots
  - dresses
  - high heels
  - jackets
  - skirts
  - shirts, blouses
  - shoes
  - trousers
  - T-shirts

These alphabetically ordered classes have been chosen because these categories make up most of the items tagged on outfit photos by users on Fashionfreax.net. Positives as well as negatives will be taken from the Fashionfreax.net database of outfit photos.

2. Filtering positives for context appearance: Since our assumption for contextual models is that the object class we are looking for exists in a certain context, we have to make sure that our training data reflects that. In this case we want images containing people wearing clothes, however, not clothes by themselves (e.g. product photos). So this step consists of filtering out images where either no person is visible or where they are not completely visible from head to toe. Also images where the person is not standing are being filtered. *The reason why also partly visible people are being filtered is that the context model to detect these people should not be too complex in these experiments. Instead a simple standing person model is being used. In general it would be possible to build a contextual model with a more complex person model, e.g. one that can also match upper body, or even portrait photos.*
3. Validating and Completing Tags: This step has been crowdsourced. Here, workers go through each of the images and select which of the aforementioned categories are present in the photo. This step is necessary because it cannot be guaranteed that the images on Fashionfreax.net have been tagged exhaustively. Quality control in this step has been supported by the use of pre-tagged examples. All user's had to tag these examples randomly throughout their task to determine if their results are trustworthy.<sup>125</sup>



Figure 8: Annotation and segmentation tool

---

<sup>125</sup> See Crowdfunder [2013]



4. Annotating Clothes: In this step all clothes which have been selected in the first step are being segmented. Crowdflower workers segment one item of clothing at a time by adding vertices until the item is fully segmented (See Figure 8). Vertices based (almost pixel-wise) segmentation has been used to make sure that the worker includes all of the item and the output is more precise. Additionally, this can be used at a later stage as training data for clothing segmentation. We then automatically fit a bounding box around the segments so the whole item is inside the bounding box.

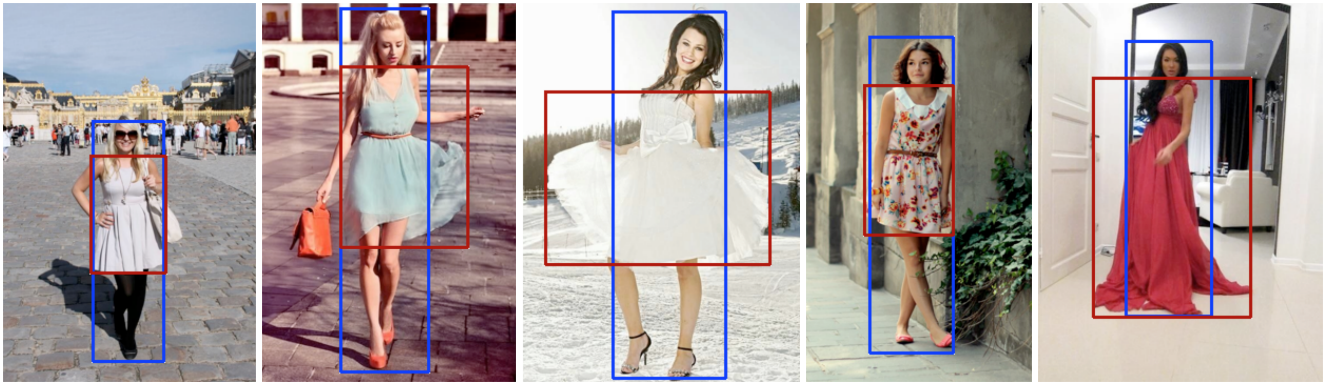
5. Annotating Persons: This step consists of segmenting each person with a simple bounding box. In this case we also have to annotate persons in the negative training set. While this reduces the advantages of hard negatives mining when training an LSVM, this is mandatory for our approach. Here the relationship between context and object is key. The object in a contextual model can only be present if the context is also present. Thus, for positive examples this is relatively simple. In the positive training data each annotated object needs an annotated context – object class label  $y_i$  and context class label  $\hat{y}_i$  are equal – see first case in equation (69).

However, negative examples are not as simple. We have to distinguish between two types of negative examples. While negative examples never feature the object, they might or might not contain the context. Hence, the two cases of class labels  $y_i = -1, \hat{y}_i = 1$  and  $y_i = -1, \hat{y}_i = -1$  exist. Negative examples only mean that they are negative examples for the object training, not necessarily for the context. Thus, bounding boxes for context appearances in all negative examples are needed. This can also be seen when looking at equation (91), the LSVM objective function's sub-gradients contain context and object class labels, which do not have to be equal. Unlike the positive examples negative examples can have zero to many context annotations.

Both annotation steps have been done by a small number of instructed workers. Annotations have been manually validated at the beginning of the task, to ensure the workers were reliable. The order in which the images have been annotated was random.

The final set we used for training and evaluating models in this work contains the following data:

- 2820 positive examples of annotated dresses and their associated persons. These are being split up in a 70/30 ratio. Thus, there are **1974 positive training examples** and **846 positive testing examples**.
- **1614 negative images** with annotated persons.<sup>126</sup>
- The resulting testing set contains **1201** images with 846 positive examples.



**Figure 9:** Several examples of bounding boxes from the “dress” training set. All persons wearing a dress are upright and come with many variations in pose, appearance and backgrounds. The dresses vary in appearance, size, type and shapes impacted by the wearer’s pose.

## 10.2 Object and context model training

Before creating a contextual model the object and context class have to be defined. In this evaluation the object class is *dress* and the context class the person wearing it.

<sup>126</sup> The reason that in this case it is not “1614 negative examples” but “1614 negative images” is that each image contains many possible negative examples. The Latent SVM algorithm is taking advantage of this fact by data mining hard negative examples, examples whose score is larger than -1, the margin defined by the loss function in equation (55). In the case of positive examples, the training set contains only one positive example per image. Thus, for positive examples the number is the same.



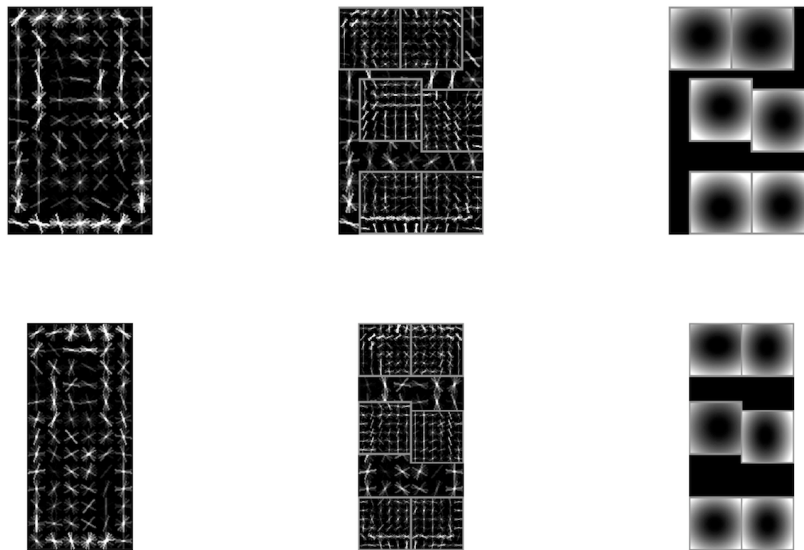
## Object classifier

The object classifier consists of standard deformable parts models. The deformable parts models are trained with the whole training set (1974 positive examples, 1614 negative images). The positive training data is divided into two sets to train a mixture model of two components. The final mixture model then contains 2 unique models and 2 mirrored models. The models are trained with root filters and six parts each.<sup>127</sup>

The resulting dress classifier is being used as both a benchmark for the resulting contextual dress classifier as well as the initialization for the object mixture model used in the contextual mixture model. However, using a fully trained object mixture model to initialize the context mixture model is not mandatory, since the object models are being fully trained again, when training the contextual models. In our case this was a convenient implementation choice, since the object mixture model structure is readily available for integration into the contextual model structure.<sup>128</sup>

The final mixture model is visualized in Figure 10. The dress outline is visually very distinctive in both components. Per definition they differ in aspect ratio<sup>129</sup>, but interestingly significant differences can be seen in HOG weights in the center parts. While the first component exhibits a distinctive vertical line (which could represent a belt or color contrast of the dress itself), the second component does not show a similar feature. One could imagine that both components are trained to classify very different kinds of dresses. While the first component conveys an hour glass shape often seen with casual dresses, the second component seems to have a much more pronounced part below the hips, and a less pronounced upper body part. This is typical for floor-length evening gowns/dresses.

Additionally, there are differences between the deformation parameters of the components parts. While the parts of the first component have relatively similar deformation costs in both the  $x$  and  $y$  direction, the top-left and right-center parts of the second component have very different deformation costs in both dimensions. This results in oval-looking part deformation visualizations in Figure 10.



**Figure 10:** Fully trained two component dress deformable part mixture model.<sup>130</sup>

## Context classifier

The context classifier is also a deformable parts mixture model like the object classifier. However, since its purpose is interchangeability – so it can be reused in conjunction with different object classes which share the same context – a pre-trained person model is being used. It has been trained on the INRIA Person Dataset. This dataset contains

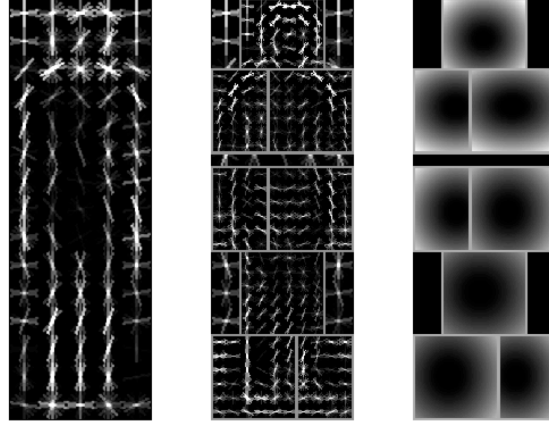
<sup>127</sup> See model initialization and training in Felzenszwalb et al. [2010a, p. 11].

<sup>128</sup> If initializing with a fully trained mixture model versus an untrained mixture model increases or decreases the time to reach the global optimum could be an interesting topic for discussion.

<sup>129</sup> See model initialization Felzenszwalb et al. [2010a, p. 11].

<sup>130</sup> Model visualized with framework by Felzenszwalb et al. [2011]. Redundant mirrored models have been hidden.

upright persons.<sup>131</sup> It achieved an average precision of 0.882 “using the PASCAL evaluation methodology in the complete test dataset, including images without people”<sup>132</sup>. This shows that this is an up-to-date person detector, which should perform well detecting upright people wearing clothes (dresses). However, the pre-trained context model only achieves an average precision of 0.581 on our training set which suggests that it might not be a perfect fit for our data. We will still use it to see how robust the contextual model approach is w.r.t. imperfect context models.



**Figure 11:** Fully trained one component person deformable part mixture model.<sup>133</sup>

The reasoning behind choosing a context classifier trained on a third party dataset is that it should be as independent as possible. Training a person classifier on the data we generated would mean that it is being trained on an outstanding amount of persons wearing dresses. Hence, at least parts of the person classifier would be especially fit to respond to features in dresses. This is redundant and does not align with the context classifiers envisaged purpose. Still, there might be the risk that an independently trained context classifier does not reflect the context in your training data, possibly leading to subpar results.

There might be a reason to do a tradeoff between context classifiers that are independent enough from the object classifier and (at least partly) trained on the object class dataset. In this work the approach to use an independently trained context classifier has been chosen to show its effect in less than perfect situations. In production one would train the context classifier on the context annotated in the training data. If there are multiple object classes having the same context class the context class should be trained on all of them. Adding more and more object classes and training on their annotated context examples should lead to a sufficiently independent context classifier. At the end most of the inter class similarity should be modeled by the context classifier, while the inter class dissimilarity should be modeled by the object classifiers themselves. Together they could form a contextual mixture model covering a multitude of classes. This aspect will be discussed more thoroughly in section 12.

### 10.3 Contextual model initialization and training

Contextual models are being initialized using the object and context model described above. Then the spatial prior parameters  $V_i$  and  $S_i$  have to be initialized. These are being calculated as the mean spatial (including scale) offsets found in the training data. There priors are being calculated separately for each component pair which form new components in the resulting contextual mixture model. The scale offsets are needed to compare scores of models which have been trained in different resolutions. This is apparent when looking at both the dress model in Figure 10 and the person model in Figure 11. The root filters of the dress model are 11, respectively 13 blocks in height, the person model’s root filter is 15 blocks in height. Also the dress root filters are 7, respectively 6 blocks in width, while the person root filter is only 5 blocks in width. Together with knowledge about the real (mean) sizes of these objects a conversion scale can be calculated. In this case the person model is being offset by  $V_{c,j} = -0.5$  scale octaves w.r.t the first dress component  $c = 1$ , and  $V_{c,j} = -0.4$  scale octaves w.r.t. the second dress component  $c = 2$ , which means that the person model’s components are applied to a lower resolution version of the input image than the dress model’s.

<sup>131</sup> See Dalal [2005].

<sup>132</sup> Felzenszwalb et al. [2011]

<sup>133</sup> Model visualized with framework by Felzenszwalb et al. [2011]. Redundant mirrored models have been hidden.

The deformation costs for the context models have been initialized at  $D_i = (dy, dx, ds, dy^2, dx^2, ds^2) = (0, 0, 0, 0.01, 0.01, 0.02)$ . This allows the object to move freely relative to its context. Part deformation costs in the deformable parts models on the other hand are being initialized much higher, allowing less movement initially.<sup>134</sup> Low initial deformation costs have been chosen because as part of the evaluation the development of deformation parameters should be examined. By keeping the initial value low we can be relatively sure that if it increases in value the assumed spatial relations also exist in the training data, and the training algorithm is working correctly.

The objective functions used for training, all contain the parameter  $C$  to control regularization term weights. The value has been set to  $C = 0.002$  as it has been used in the framework version 4 by Felzenszwalb et al. [2011] for training models on training data from the PASCAL VOC challenge, and has been shown to work well for different object classes. Due to the significant increase of model parameters in contextual models it might be useful to experiment with this value. However, for experiments in this work, the value has been kept fixed.

We trained multiple contextual models in different configurations with four main parameters to compare their impact on performance. These include the choice of *training dataset*, *objective function*, merging method ( $lr$  vs.  $l \times r$ ) and context model training (initial parameters vs. trained parameters).

### Training sets – ALL, MEDIUM, FAST

There are three training set sizes which have been used to train models:

- ALL: 1974 positives, 1614 negative images
- MEDIUM: 700 positive examples, 1614 negative images
- FAST: 500 positives examples, 474 negative images

The choice of the training set depends largely on practical reasons. FAST and MEDIUM are used to reduce the training time. While models across different training sets are not comparable, some models have been primarily trained for comparison of different combinations of configuration parameters, in which case using training sets, which reduce the contextual model's training time is reasonable.

### Objective functions – RegA, RegB

In section 7.2.2 two different objective functions have been defined, referred to as **RegA** and **RegB**. In a nutshell RegA defines a semi-convex optimization problem, while requiring object and context annotations on the same training set. RegB allows spreading object and context annotations over two standalone training sets, but has the disadvantage of having increased complexity through the additional term weight parameter  $D$ . Also the resulting optimization problem is not semi-convex as defined by Felzenszwalb et al. [2010a]. Instead an auxiliary term with weight  $D$  is being used to regularize the contextual loss concave term. Most models trained in this work for evaluation use RegA as the objective function due to reasons described above and in section 7.2.2. However, some contextual models have been trained using RegB to examine its practicability.

### Merging methods – $lr$ , $l \times r$

As described before, there are two ways to merge object and context mixture models to form a contextual mixture model,  $lr$  and  $l \times r$ . (94) shows that models initialized with  $l \times r$  are much more complex than if they are initialized using  $lr$ . In our case of merging dress mixture model  $M_0$  with 4 components and the person mixture model  $M_1$  with 2 components  $lr$  leads to a contextual mixture model  $M_{lr}$  with 4 components,  $l \times r$  to a contextual mixture model  $M_{l \times r}$  with 8 components.  $M_{lr}$  is comparable to the original object model in terms of amount of components. It also leads to a less complex model, increasing training and classification speed compared to  $M_{l \times r}$ . With  $M_{l \times r}$  one would end up with double the amount of components in the original object model. This has implications on comparability. Training a dress deformable parts model using 8 components instead of 4 could already lead to a higher detection performance.<sup>135</sup> Therefore, any performance improvement of  $M_{l \times r}$  relative to  $M_0$  could be ascribed to the increased number of components, not necessarily to use of additional contextual information. A solution to that would be the reuse of model components and their parameters across different components in the contextual mixture model, so that the real amount of object components does not increase relative to the original object model. This option has not been implemented yet.

<sup>134</sup> See Felzenszwalb et al. [2010a, p. 12].

<sup>135</sup> Increasing the amount of components in a mixture model usually increases detection performance, see performance results in Felzenszwalb et al. [2010a, p. 16]. However, one has to consider the amount of positive training data per component. If the ratio between training set size per component gets to small lower detection performance could be expected.

## Context model parameter training – TRAIN, FIX

As described in 8.4 one has the option of training the context model parameters or keeping them fixed when training the contextual model. Fixed context model parameters prevent them from assuming object specific features and, therefore, stay independent from the object model parameters. Keeping the context model parameters flexible on the other hand could lead to a better fit to the training data and better model performance. We investigate both options in our performance results. Contextual models where the context parameters are being kept fixed are tagged with the keyword **FIX** and if not with **TRAIN** in results Table 1.

In all cases the context model’s bias parameter as well as the context deformation costs are being trained, since they are mandatory to train a contextual model. The bias, as noted in section 8.4, is used to moderate the impact of the context to the overall score and, therefore, is of importance for a robust contextual model.

*Fitting Training Data:* For contextual models, where context parameters stay fixed, the positive training examples are being pruned to guarantee a certain fit between training data and the context model. Depending on the context model positive examples with a context score below a certain threshold  $t$  are being discarded –  $t = (-1 + t')/2$ , where  $t'$  is the minimal score of any example in the context model’s own training data. With a perfect context model and typical SVM margins of 1 the threshold should become  $t = 0$ . In our case the context  $t \approx -0.8$ , which means that only very low scoring examples are being discarded.

The different configurations that have been used to train contextual models are noted in Table 1.

### 10.4 Performance evaluation metrics

To quantify the performance of different models we use plots of detection error tradeoff (DET) curves. These are commonly used in place of precision vs. recall (PR) curves because they “allow small probabilities to be distinguished more easily”<sup>136</sup>. In DET curves one plots the miss rate (MR) versus false positives per image (FPPI). This tradeoff displays the same information as so-called receiver operating characteristics (ROC), e.g. PR curves, but better results are smaller instead of larger. Some researchers also use MR v.s. false positives per window (FPPW) instead of FPPI. FPPW sometimes leads to problems when comparing different detectors because the amount of windows in the dataset can vary depending on the descriptor and its configuration. In this case we decided to use FPPI, since it renders the results much more comprehensible.

$$MR = 1 - HR = \frac{\sum FN}{\sum (TP + FN)} \quad (105)$$

$$FPPI = \frac{\sum FP}{|I|}, \quad (106)$$

where FN are false negative detections, TP are true positive detections and FP are false positive detections.  $|I|$  is the number of images in the test set.

Additionally, the Average Precision (AP) metric is used as a rough model quality indicator in Table 1. It is defined as the mean precision in the whole precision v.s. recall graph. Precision is defined as Usually AP is approximated by taking the mean value of a fixed amount of points in the graph. The PASCAL VOC Challenge manual proposes that AP should be calculated as the mean precision at 11 different recall values 0.0 to 1.0 in steps of size 0.1.<sup>137</sup> This approximation is defined as

$$P_{avg} = \frac{1}{11} \sum_r p_{interp}(r), \text{ where } r \in \{0, 0.1, 0.2, \dots, 1.0\} \quad (107)$$

and  $p_{interp}(r)$  is the maximum precision available where the recall is larger than  $r$ .<sup>138</sup> However, it is important to note that a finer grained approximation may increase “the impact of the wiggles in the precision/recall curve, caused by small variations in the ranking of examples”<sup>139</sup>.

AP can give an idea of how well a model does balance precision and recall across the precision vs. recall graph, however, it does not indicate how the detection error tradeoff curve looks, which is why DET curve graphs are needed to assess model performance in detail.

Detections are being evaluated with a minimal overlap of 0.5. The overlap  $a_o$  is defined as

$$a_o = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})} \quad (108)$$

<sup>136</sup> Dalal and Triggs [2005, p. 2].

<sup>137</sup> See Everingham et al. [2010, p. 313].

<sup>138</sup> See interpolated average precision definition in Salton and McGill [1986].

<sup>139</sup> Everingham et al. [2010, p. 313].

where  $B_p$  is the bounding box of the detection and  $B_{gt}$  is the bounding box of the ground truth.<sup>140</sup> Multiple positive detections on the same positive example will be counted as false positives.

---

## 10.5 Post processing

---

Non-maximum suppression (NMS) is being used to greedily select high scoring object detections and remove others that overlap the same area analogous to Felzenszwalb et al. on all models.<sup>141</sup> For contextual models non-maximum suppression is also used on the context detection windows. Since we assume that a person can only wear a single item of a class, we can limit redundant object detections in the same context. In the tests executed in this work the context detection windows are greedily selected by the overall object detection score. All models have been evaluated on the test data using NMS, and the contextual models also using NMS based on the context detection windows – contextual non-maximum suppression (CNMS). Both results are displayed in Table 1. Using CNMS improves the detection performance significantly for important areas in the DET graph.




---

## 10.6 Performance results

---

Table 1 gives an overview over the different models and configurations that have been trained. The first model that has been trained 1.1 is a standard deformable parts model trained with framework version 4 by Felzenszwalb et al. [2011] using 4 components (2 aspect ratio sizes, left and right orientation). Its DET curve and average precision ( $AP = 0.701$ ) acts as the base line with which the contextual models are being compared with.

In the current implementation detecting dresses in an image takes about 5 (10) times or longer for the trained contextual models in  $lr$  ( $l \times r$ ) configuration when compared to the original deformable parts model. Additionally to having the complexity and amount of parameters of two part based models the speed decrease is mostly attributable to running the distance transform algorithm in 3 dimensions, with the overhead of transforming score pyramids to grids and vice versa. Possible improvements in that matter are being discussed in the section 12, Future work.

The AP of Contextual model 2.4  using CNMS has been marked as the “winner”, since 2.4 has the same amount of components as 1.1 , making for a better comparison. Contextual model 2.5 , which has been trained using  $l \times r$ , has more components – therefore some of the performance increase could be due to more model components.







The following sections 10.7 and 10.8 evaluate and discuss the the trained models in a qualitative and quantitative manner.

---

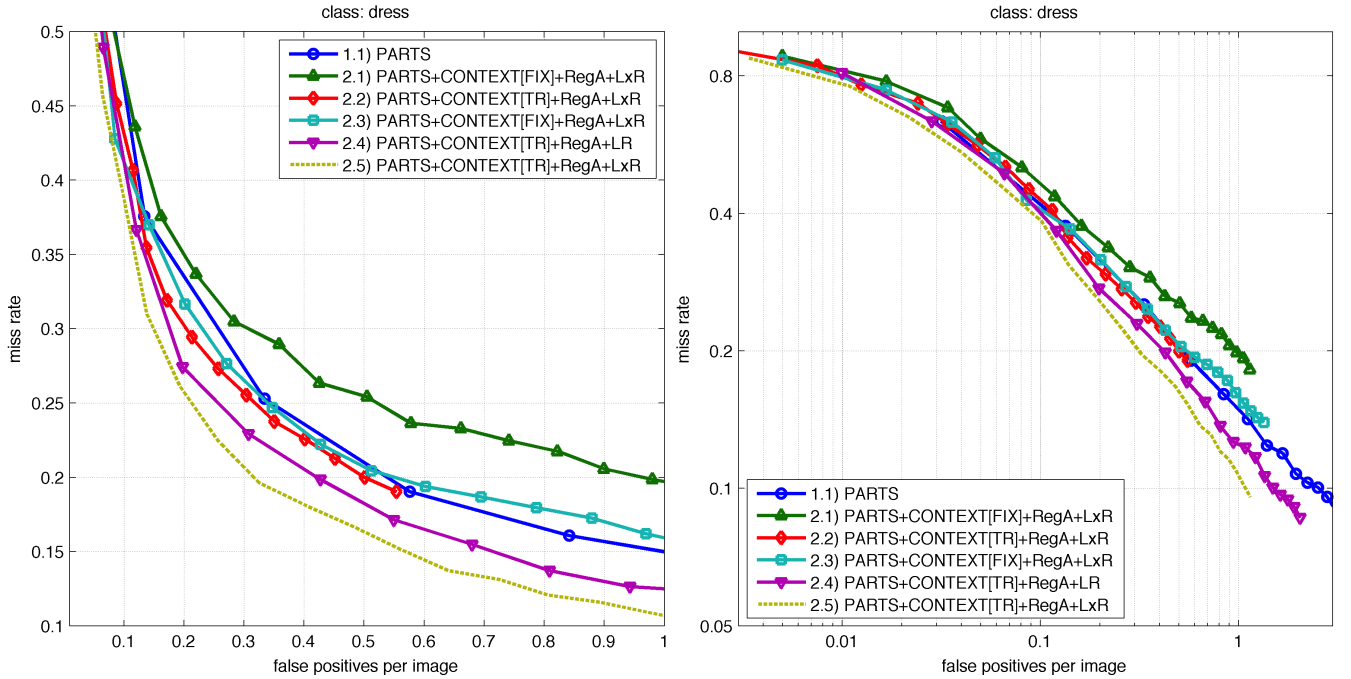
<sup>140</sup> See Everingham et al. [2010, p. 314].

<sup>141</sup> See Felzenszwalb et al. [2010a, p. 14f].

<sup>142</sup> Tested on context annotations in positive training set.

#	Model Type	Comment	Plot	Params	Merging	Object	Context	AP (NMS)	AP (CNMS)
1	Base Model								
1.1	Parts	ALL			$lr$	Dress	n/a	0.701	
1.2	Parts	INRIA Person <sup>142</sup>			$lr$	Person	n/a	0.581	
2	RegA								
2.1	Contextual	FAST			$l \times r$	Dress	FIX	0.651	0.634
2.2	Contextual	MEDIUM			$l \times r$	Dress	TRAIN	0.685	0.646
2.3	Contextual	ALL			$l \times r$	Dress	FIX	0.682	0.691
2.4	Contextual	ALL			$lr$	Dress	TRAIN	0.719	<b>0.721</b>
2.5	Contextual	ALL			$l \times r$	Dress	TRAIN	0.750	0.724
3	RegB								
3.1	Contextual	MEDIUM		$D := C$	$l \times r$	Dress	FIX	0.596	0.611
3.2	Contextual	MEDIUM		$D := \frac{1}{2}C$	$l \times r$	Dress	FIX	0.614	0.631
3.3	Contextual	MEDIUM		$D := 2C$	$l \times r$	Dress	FIX	0.633	0.641

**Table 1:** Configurations and results for trained contextual models. The column Model Type defines if the model is a deformable parts model ('Parts') as defined by Felzenszwalb et al. [2010a] or a contextual model as defined in this work ('Contextual'). The other keywords used in this table are described in section 10.3.

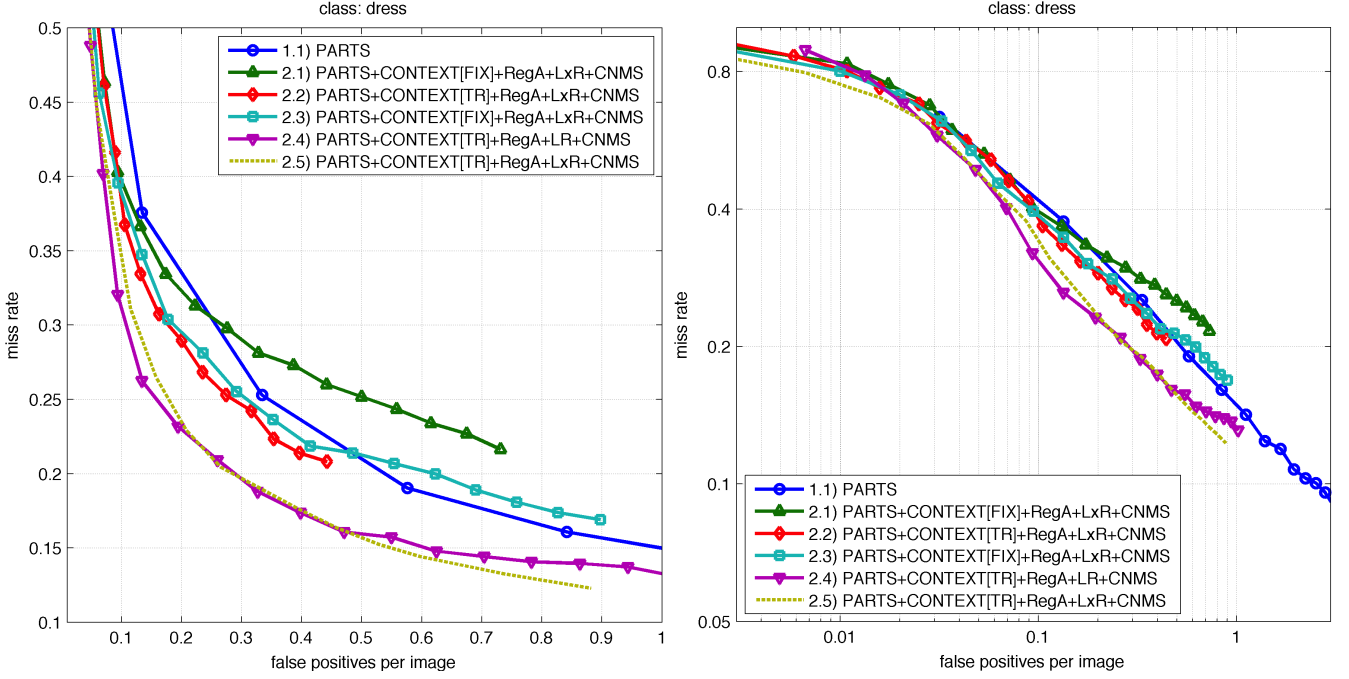


**Figure 12:** Detection error tradeoff plots for all models trained with RegA in Table 1, post-processed using standard NMS. (Linear plot on the left, log-log plot on the right)

## 10.7 Qualitative evaluation of results

First, the trained models are being evaluated qualitatively. Here the focus is to determine if the approach of a model with a flexible context constraint works and if expected spatial and scale relationships exist between object and context – in our case dress and person. In particular, the following two aspects are being evaluated: the development of context





**Figure 13:** Detection error tradeoff plots for all models trained with RegA in Table 1 which are post-processed using CNMS. (Linear plot on the left, log-log plot on the right). Due to obvious reasons 1.1 —●— is post-processed using standard NMS.

deformation costs and the development of the object and context sub-models after training starting from their initial states.

#### 10.7.1 Development of deformation costs

The deformation costs  $D_j$  for the  $j$ -th context model are the core metric in which the spatial and scale relationship between object and context is encoded. Initial values before training have been set to very low values to allow for relatively free movement. In practice there is also a lower bound for  $D_j$  which is used to keep deformation costs from becoming too “flat”.<sup>143</sup> We now look at the deformation costs of different contextual mixture models that have been trained.

*Expectations:* In the real world dresses (as most clothes) are centered horizontally on the wearer’s body. Furthermore, dresses start closely below the wearer’s head. This can also be observed in our training data ( see Figure 9). There are only small deviation of the dress relative to the person in the horizontal axis. Also the bounding box of the dress starts below the persons’s chin, or for some dresses above the décolleté. However, the lengths of the dresses may vary a lot. The contextual model anchors at the center of the body, which means that deformation in  $y$ -dimension could be needed to correct the changed  $y$  position of the scaled object. Consequently, one would expect high horizontal and lower vertical deformation costs.


Contextual model 2.3 with configuration [ALL, RegA,  $l \times r$ , FIX], shows that the deformation costs have been learned by the training algorithm to improve classification and are in consequence non-trivial. Table 10.7.1 shows the deformation costs for each of its 4 left facing components (the other 4 right facing components are equal in value, except mirrored in  $x$ ). The data clearly shows that the deformation costs are significantly higher after training the model than the values used for initialization. This makes sure that the model is training correctly.

Recall that the deformation costs for an example is computed by  $D_j \cdot \phi(dx, dy, ds)$  where  $\phi_D(dx, dy, ds) = (dx, dy, ds, dx^2, dy^2, ds^2)$ .


For deformation costs  $D_{j,4}$  in  $x$ -dimension all components are relatively similar across all components. This aligns with the expectation that all dresses, regardless of type or size, have the same horizontal position. However, the actual value of  $D_{j,4}$  is relatively low. In no way does it really enforce a centered position on the body.

<sup>143</sup> See Felzenszwalb et al. [2010a, p. 11].

$D_j$ Component	1 $dx$	2 $dy$	3 $ds$	4 $dx^2$	5 $dy^2$	6 $ds^2$
initial values	0.0	0.0	0.0	0.01	0.01	0.02
1	0.0140	-0.0042	-0.0258	0.0424	0.0611	0.0859
3	0.0041	-0.0260	-0.0504	0.0487	0.0320	0.0267
5	-0.0135	0.0112	-0.0083	0.0398	0.0569	0.0578
7	-0.0141	0.0737	0.0279	0.0320	0.0316	0.0533

**Table 2:** Deformation costs for each component in model 2.3  from Table 1. The first three data columns contain the actual deformation costs. The last 3 columns are learned corrections for spatial and scale priors.

$D_j$ Component	1 $dx$	2 $dy$	3 $ds$	4 $dx^2$	5 $dy^2$	6 $ds^2$
initial values	0.0	0.0	0.0	0.01	0.01	0.02
1	0.0284	-0.0498	-0.0903	0.0324	0.0527	0.0419
3	-0.0208	0.0945	-0.0050	0.0275	0.0574	0.0543

**Table 3:** Deformation costs for each component in model 2.4  from Table 1. The first three data columns contain the actual deformation costs. The last 3 columns are learned corrections for spatial and scale priors.

The deformation costs  $D_{j,5}$  in  $y$ -dimension show much more variation across the components, which is also true for the costs  $D_{j,6}$  in scale dimension. It seems like components with lower vertical deformation costs also have lower scale deformation costs. Interestingly, the linear correlation between  $D_{j,5}$  and  $D_{j,6}$  is 0.81.

The values in  $D_j$  for the first two indices is learned correction of the spatial prior, and  $D_{j,3}$  is the learned correction of the scale prior. Component 3 for example has the strongest correction of the scale prior, and also the lowest scale deformation costs  $D_{j,6}$ .

The trained weights of the deformation costs do not point to the expected strong horizontal relationship of object and context. There could be multiple reasons for that in initialization and training of the model.

Foremost, it could be possible that each component had to cater to a large variety of dress location and scales. This can be partly controlled by the component clustering for the mixture model generation: components should not only be discerned by aspect ratio but also by the size relative to their context. Deformation costs in experiment 2.3 could suggest that. Component 1, 5 and 7 have relatively high scale deformation costs, while component 3 has a much lower scale deformation cost. This could mean that component 3 has to handle a larger interval in the scale-space and, thus, has increased spatial uncertainty. This change in component clustering is also described in section 12, Future work.

Another source for large spatial uncertainty could be the method for applying scale distance transform to the score pyramid described in section 5. It introduces  $x$  and  $y$  error when downsampling transformed grid levels to convert them back to pyramid levels.

Another possibility is that the performance of the context model determines how stable spatial relationships are in the resulting contextual model and, thus, influencing the magnitude of the deformation costs. In reality the fact that, while training, the context and object bounding boxes are defined by the training data, contradicts this theory. Instead the spatial relationship should be greatly impacted by the annotations themselves and less by the performance of the context model. Also the model of experiment 2.4 where the context model parameters have been trained, suggesting a better context model fit, does not yield the expected results. There the difference between  $x$  and  $y$  deformation costs is even higher, see Table 10.7.1, contradicting the expected results described above even more.<sup>144</sup>

## 10.7.2 Development object and context models from initial states

Context deformation costs  $D_j$  model the spatial and scale constraints between object and context. Recall that these constraints should redistribute responsibility between object and context models. Therefore, one would expect that the object model parameters develop further from initial values. In the next paragraphs we will compare 'before' (initial) and

<sup>144</sup> To further investigate the development of deformation costs one could define a verification experiment with controlled and automatically generated training data. E.g. a training data of two simple shapes (triangles, the objects, centered in rectangles, the context) where triangles can appear outside and inside of the rectangle, rectangles could appear without triangles. Triangles which are not centered in the rectangles would not count as the object class. It should be possible to automatically generate images and annotations and train a contextual model. In that environment contextual models could be trained with training sets of different variations in deformation.



'after' (trained) states of the dress and person models. The contextual model 2.4 has been chosen for this comparison, because due to  $lr$  merging it has the same amount of components as the original dress model. Specifically let  $M_{contextual} = (M_0, C_1)$ , where  $M_1 = (M_1, V_1, S_1, D_1)$ , the initialized contextual model, and  $M_0, M_1$  initial dress and person models respectively. Then  $M'_{contextual} = (M'_0, C'_1)$  is the contextual model in its trained state, and  $M'_0, M'_1$  are the trained versions of  $M_0$  and  $M_1$ . Next, the differences between  $M_0$  and  $M'_0$  as well as  $M_1$  and  $M'_1$  are being examined.

### Dress model development

The initial dress object model has four components, in two left/right orientation groups by aspect ratio. This model's development is depicted in Figure 14. The root and part filter are HOG filters where each block has been visualized as a set of oriented bars. Thicker, more visible bars represent higher weights at that certain orientation.

Starting with the root filters there are not many differences between initial components and the trained versions. The overall visible shape of the HOG visualization stays the same, even after training. Maybe the most prominent differences lie in the inner blocks of components' root filters, the area which represents the skirt-shaped part of a dress. There the initial root filters have relatively low weights, and the visualization appears darker. The trained components on the other hand show higher weights in the blocks in this area. The second component usually matches larger dresses, which reach further down to the floor.

If one would lay both root filters on top of each other, they would have to be aligned at the top border. Then the blocks in which the weights have increased also lie roughly at the same location in both root filters. Since the data set for training the dress on its own and for training the contextual model have been the same, these weights could be ascribed to the effect of a contextual constraint.

As described in the introduction the goal of adding a contextual constraint was the reduction of the actual object detector's problem size, freeing up scope to assimilate features which are significant for the object class in context but could lead to false reasonings outside of the context. Therefore, these changes in block weights after training the contextual model could be signs of such an adjustment and indicate that the contextual model assumptions might be true.

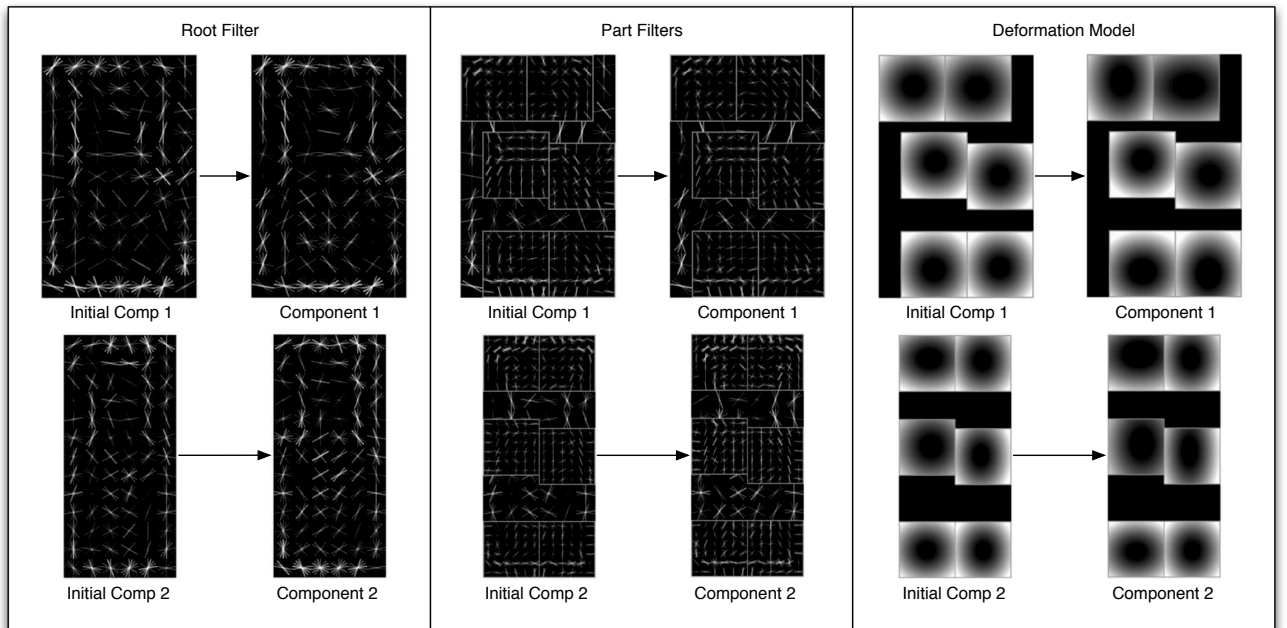
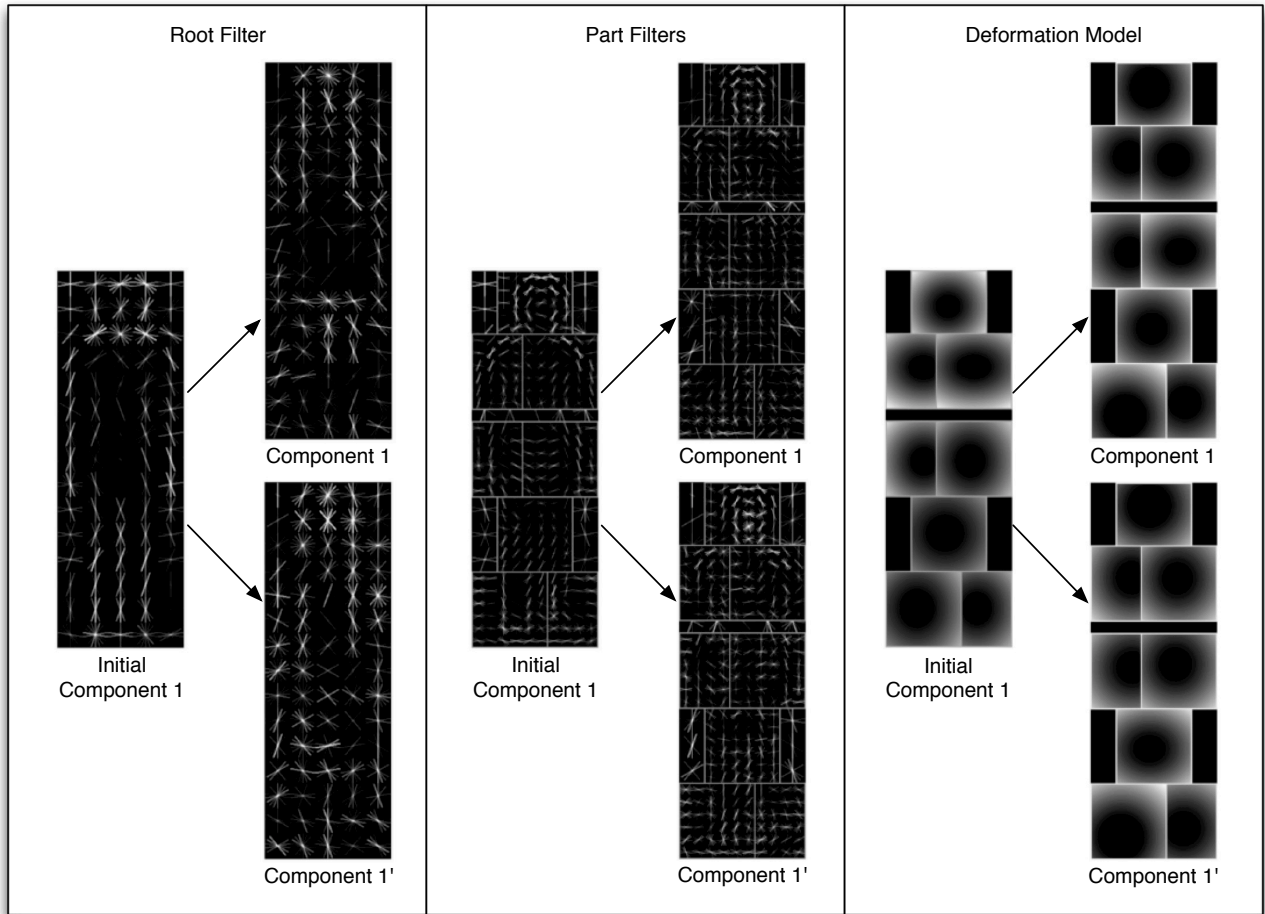


Figure 14: Dress model before and after training of the contextual model in experiment 2.4.

### Person context model development

Comparing the weights of the initial and trained context model show considerable differences. The root and part filters seem to have assumed a lot of object specific information. The HOG visualization in Figure 15 shows that both person components now contain features that are redolent of dress features. The person model deviates partly to a "person wearing a dress"-model, with visible bottom lines of the dress. The bottom line is visible further up in component 1 than in component 2. This reflects the shape of the dress model components.



**Figure 15:** Person model before and after training of the contextual model in experiment 2.4. Due to duplication of context components when initializing the contextual model, component 1 of the person model has been trained into two different components. The left column shows the development of the root filter, the middle one of the part filters, and the right column of the deformation model. The resulting part filters show an assimilation of domain specific features from the dress training data.

This was to be expected because the training set only contained the class dress, hence, any person not wearing a dress is irrelevant for the classification task of the context. In another situation where multiple clothing items share the same person model as their context this 'feature assimilation' would not happen in this extent.

There are also other differences in the HOG filters of the trained person model, where the original model had strong weights for vertical edges (thick vertical lines in visualization), the trained person context model seems to have a much higher variation in gradient orientations (many star or cross shaped histogram visualizations). Also the chest area of the original model had relatively low weights, which can be seen as the dark areas in the visualization. After training most of these low weight, dark areas seem to have vanished. Especially the visualizations of the root filters become noisy after training. The previously distinctively visible shape of a person becomes hardly visible. One of the few parts, which reflect the expected shape of a body part, is the head. Both components still present a distinctive head-like shape. Still there are significant weight differences. While the initial person model had stronger weights in the outer areas of the head, these weights are less pronounced and other weights in the inner blocks of the part become more distinct.

Looking at the parts deformation costs, strong changes can be observed, too. The deformation costs of most parts decrease after training allowing for higher spatial uncertainty. This is visible in Figure 15 as the visualized distributions gain a higher variance after training and appear darker and further spread out. Additionally, the spatial priors change as well. Especially the top part (head) and one of the bottom parts (foot) are getting moved further out after training, enabling detections of taller persons.

Many of these differences between the initial person model and the resulting person model could be attributed to fitting the person model to the new training data, since the original model was trained on the INRIA Person training set. Apart from that the person model also gained a different purpose when being part of a contextual model. It did not have to explicitly classify persons anymore. Its purpose in a contextual model is to penalize non-person detection windows, and stay neutral if there is a person. The classification task itself becomes less important than the task to provide information for the object model.

### 10.7.3 Development of the context model's threshold

Now that we examined the development of the context deformation costs and the sub-models' filters and part deformation costs, the third aspect of the contextual model, the score threshold enforced by  $score_{context}$  defined in (19) is being analyzed. Unfortunately there is no exact value that could be read. The threshold itself is always zero. Therefore, to lower the threshold the bias increases, etc. In fact the threshold is an indirect value and depends on the whole context model's parameters. We can only examine the threshold relative to the scores that are being achieved by the context models  $C_j$  inside the trained contextual models.

By definition a threshold is low, if it is often reached, and a threshold is high, if it is hardly reached. Therefore, a context model  $M_j, j > 0$  with a low learned threshold would often have a score  $\geq 0$ , a context model with a high learned threshold would rarely reach a score  $\geq 0$ . Recall that also the object model has a bias parameter. In the case of a low scoring context model due to a high threshold, the object model bias could compensate.





Let  $b'$  be an auxiliary threshold parameter, with




$$score_{context}(z) = \begin{cases} score(z) & score(z) < b'_t \\ b'_t & otherwise \end{cases}, \quad (109)$$




where  $t \in 0, 1$  determines if the model has been trained (1) or not (0). Now we set  $b'_0 := 0$ . This is an arbitrary choice, but the simplest. Now we try to determine if  $b'_1 < b'_0$  or  $b'_1 > b'_0$  for the trained contextual models. This can be approximately determined by analyzing the scores on all detection windows in the training data of the initial context model, and the context model after training the contextual model. If the score distribution would have increased after training, this would indicate that  $b'_1 < b'_0$ , and vice versa.

In detail, we inspect the resulting contextual model of experiment 2.4. When analyzing the training data the average scores of the context model increases from -4.81 to -1.91 after training, also the rate of detection windows which receive a score  $> 0$  increases from 0.0029% to 0.0046%. This gives reason to assume that  $b'_1 < b'_0$ , so the contextual model reduced the impact of the context model on the overall score compared with its initial state.

## 10.8 Quantitative evaluation of results

**Fixed context parameters (FIX):** First we analyze the results of the contextual models with fixed context parameters. For the objective function RegA these are experiments 2.1, trained on FAST, and 2.3, trained on ALL. As one would expect 2.1 , which was using the smaller training set, has a higher miss rate curve in the DET graph than 2.3 . Especially compared to the original dress model 1.1 , 2.1  cannot compete. The DET curve of 2.3 on the other hand does reach almost the same performance as the original dress model. Only in low miss/high FPPI regions suffered a little bit (see log-log graph in Figure 12). This is a relatively uninteresting region, though, because there the miss rate/false positive tradeoff is not acceptable any more for most applications.

As depicted in Table 1 the initial context model which was trained using the INRIA Person training set does only have an average precision of 0.581 on the context annotations of the training data. This shows that contextual models are relatively robust against context models, which do not reflect the training data. In that way the context model was possibly not able to contribute a lot of meaningful information to contextual model 2.3  to surpass the original dress model 1.1 . There are several FPPI intervals, where 2.3  outperforms the baseline by a little margin, but also others where 1.1 prevails.

**Trained context parameters (TRAIN):** Next, we look at contextual models where the context model parameters were not fixed. Starting with 2.2 , we can see that using only the MEDIUM training set a competitive performance to the original model 1.1 can be reached in the interval below 0.6 FPPI. Finally, model 2.4  which has been trained using the ALL training set is able to significantly outperform 1.1 . This result should validate the contextual model approach, especially since 2.4. has been initialized using the  $lr$  merging approach, so it has the same amount of components as the original dress model 1.1.

There is still small FPPI interval where 1.1 outperforms 2.4 – in the area where  $MR > 0.65$  and  $FPPI < 0.04$  (not visible in the DET graph). This is very interesting because the intuitive expectation when defining the contextual model was,


---

that it will increase precision and, thus, especially have advantages in that region. In reality the contextual model 2.4 achieves the best results relative to 1.1 for FPPI > 0.15.


---

### 10.8.1 Contextual non-maximum suppression

---

Using CNMS to post-process detections helps increasing the detection performance of some of the contextual models. Model 2.4  has an average precision of 0.719 before, and 0.721 after CNMS, however, the AP does not indicate the real changes. When comparing Figure 12 with Figure 13, one can see that the contextual model curves achieve a better detection error tradeoff (they achieve lower false positives per image while the miss rate stays the same). The downside is that applying CNMS increases the minimum miss rate that can be reached with the model, also decreasing recall and thus affecting AP adversely. This is why the AP for most contextual models decreases when applying CNMS.

One could compare the CNMS method with simple methods that only allow one detection per image. This would be a bad practice, since it relies on knowledge about the testing data, only containing one object in the image. In our case we adopt this idea, by only allowing one detection per context. This knowledge is not inferred from the testing data, but from the training data, since a person can either wear no dress or exactly one dress. Therefore, this is a viable method in all images, independent of how many people with or without dresses are present.

The improved MR/FPPI tradeoffs are also noticeable for contextual models that have been trained with fixed context parameters (see DET graph of 2.3 ). This shows that even independent, imperfect context detectors provide useful information to increase an object detector's performance in the most relevant regions of the DET graph.

---

### 10.8.2 Performance of models using RegB objective function

---

Three models have been trained using the RegB objective function as defined in (93) – 3.1, 3.2 and 3.3. All these models have been trained using the MEDIUM training set,  $l \times r$  mixture model generation and fixed context model parameters (FIX). The only parameter that has been varied was the relative term weight  $D$ , which controls the impact of the additional context loss term in (93), while their values have been set relative to the contextual loss term weight  $C$ . Tested values are  $D \in \{C, \frac{1}{2}C, 2C\}$ .

The best performance of these three options can be reached using  $D := 2C$  (AP = 0.633). The results suggest that the best performance can be achieved by increasing  $D$  even more. A more thorough grid search is needed to get more significant results. The current data implies that models trained with RegB have a subpar performance to models trained with RegA. Even with a smaller training set, RegA model 2.1 has a higher AP than RegB model 3.1. Further exploration of different values for  $C$  and  $D$  are needed to confirm or reject this result.

---

## 11 Conclusion

---

In this work we proposed a method to model and train an object detector with flexible context constraints based on the deformable parts models by Felzenszwalb et al. [2010a]. It uses the same star-shaped architecture to model constraints between an object class and other context (object) classes. It is also possible for different object classes to share the same context class each having individual constraints.

For modeling the unique asymmetric relationship of object and model we proposed a new way to score combined examples of object and context which mediates the impact of context score on the overall score.

The trained contextual models suggest that

- expected spatial and scale relations between object and context do exist in the training data, as the deformation costs increased to significantly higher values after training than the initial values, showing that the contextual model uses them to classify objects.
- the object deformable parts model inside the contextual model can assimilate more object variations, adapting to the contextual environment by changing its filter weights and deformation costs. This is observable by comparing the filter weights and deformation costs of the initial object model with the object model after training the contextual model.

The contextual models achieved a significantly higher detection performance on the dresses dataset in most areas of the detection error tradeoff graph, when compared with deformable parts models. In addition contextual non-maximum suppression increased the detection performance even more and showed that the contextual information can be used in two different ways. First, the context penalizes bad object locations in the object detection step, then multiple object detections of the same class referring to the same context detection window as their context location can be suppressed.

Also this work identifies several methods which could be used to subsequently improve contextual models. The following section describes some potential solutions for the current contextual model's shortcomings and an outlook on how the model could be used in a production environment later on.

---

## 12 Future work

---

The result evaluation shed some light on the contextual model's performance and properties. As described above, there are some areas the contextual model can be improved in, to possibly achieve increased detection performance and speed.

---

### 12.1 Detection performance

---

This section will describe several methods that could be used to improve the model's detection performance, starting with a focus on improving the modeling of real-world spatial relationships.

---

#### 12.1.1 Allowing for different context anchor positions

---

Currently the object's spatial and scale prior is anchored to the center of the context. A centered anchor has been introduced together with scale deformation, so that the bounding box of the scale deformed model expands or shrinks in all directions equally. As for dresses, the training data suggests that while being centered vertically on the person's body most dresses start at the shoulders and have different lengths. Therefore, it could make sense to change the anchor to a top-center position. Then, deforming dresses in scale would make them wider and longer, but their bounding box's top edge would be anchored to the shoulders. This would reduce interaction between spatial and scale transformation, because the scale deformation does not ignore spatial constraints. This new anchor could be determined directly from the training data, and could improve the contextual model's ability to model real world spatial and scale constraints. This could have implications on the development of deformation costs, while training, and could, in the end, increase detection performance of the resulting model.

---

#### 12.1.2 Scale sensitive mixture model initialization

---

Another related method to improve the modeling of spatial relationships lies in the mixture model initialization. To initialize a mixture model of deformable parts models with  $m$  components, bounding boxes from the training data are being sorted by their aspect ratio and divided into  $m$  equally sized groups. As Felzenszwalb et al. state "aspect ratio is used as a simple indicator of extreme intraclass variation."<sup>145</sup> The current implementation of contextual models is directly being initialized using a deformable parts mixture model. Therefore, the same approach is being inherited.

In reality for contextual models there is another important indicator for extreme intraclass variation, the relative scale, which is currently being ignored. Figure 9 shows examples of dresses in different sizes relative to the wearer. The two rightmost examples show dresses which on their own exhibit very similar aspect ratios and, according to a plain aspect ratio clustering approach, would be most likely assigned to the same component. When initializing the contextual model the object and context windows in that component get a fixed spatial and scale prior, which obviously cannot model both of these examples very well at the same time.

The effects of this can be seen in the deformation costs, as stated in section 10.7.1, where some components have significantly reduced scale deformation costs to make up for this problem. As noted in the previous section, a higher scale uncertainty also leads to increased spatial uncertainty. Scaling is not invariant w.r.t  $x$  and  $y$  location if the anchor of that operation does not equal the top left corner's location of the detection window. Therefore, it would make sense to cluster bounding boxes by aspect ratio AND their relative scale to their context bounding box.

It is expected that the resulting contextual model can then increase deformation costs significantly due to reduced scale and spatial uncertainty. This could result in a contextual model with a higher ability to model the contextual relationships, thus, possibly achieving better detection performance.

---

#### 12.1.3 Training optimization

---

There are many parameters in the training algorithm and objective function. It would be interesting to see if better settings could be found.

The objective function contains a term weight  $C$ , which controls the relative weight between the regularization and loss term. The current value has been adopted from the implementation by Felzenszwalb et al. [2011]. They argue that it has worked well with many object categories. It could be that for contextual models other weights achieve better results. Additionally, the training algorithm itself has many more parameters. First, there are learning and regularization rates, which can be chosen independently for each parameter in the model. These are also dependent on  $C$ , which makes it hard to optimize them. Still it would be interesting to see how changing the learning rate and regularization rate of newly added contextual 3D deformation parameter affects model performance.

---

<sup>145</sup> Felzenszwalb et al. [2010a, p. 11].



Iteration	Positive Loss Before	After	Ratio
1	n/a	12.0012	n/a
2	4.068	3.4582	0.8501
3	5.228	4.9283	0.9427
4	4.5951	4.4585	0.9706
5	4.6083	4.5247	0.9819
6	4.5847	4.5269	0.9874
7	4.5433	4.4976	0.9899
8	4.1822	4.0937	0.9788
End			

**Table 4:** Positive loss development while training contextual model 2.4. Even at the last iteration it still improved by over 2%, thus, it might make sense to increase the amount of training algorithm iterations.

In the current implementation, the training algorithm runs at a fixed number of iterations, in particular it runs the relabel step 8 times (optimizing latent variables) and for each relabel step there are 10 iterations of data-mining hard examples and gradient descent step. Table 4 shows the development of the positive loss while training contextual model 2.4. It shows that at the last iteration the positive loss still decreased significantly by over 2%. Standard deformable parts models with less parameters and latent variables tend to converge faster. This is why it may be useful to change the number of optimization iterations and see if it improves the model.

Also currently only 600 negative images are used in the main training process. After the 8 relabeling steps have been finished all 1614 negative images are used in 5 final data-mining and gradient-descent iterations. This is also due to adopting the existing framework implementation and configuration by Felzenszwalb et al. [2011] for practical purposes. Due to the fact that contextual models are more complex it might be useful to integrate more negative images in the main training process. The downside of this would be that the amount of time needed for training would increase significantly.

#### 12.1.4 Increase bounding box size

The bounding boxes in the dresses dataset envelop the objects tightly. This originates from using annotated segmentations and fitting a bounding box tightly around it. It might have a positive effect to the object detector’s performance if it would be trained on bounding boxes which also encompass some of the surrounding area of the object. It would also be important to see if the performance advantage of contextual models still persists, if the deformable parts object model itself has some of the context information through an increased bounding box.

#### 12.1.5 Integrating related work on grammar models

There is a significant amount of related work to grammar models and deformable parts models by the same, or closely connected authors. Some of these works focus on speed improvements, like the cascade detection approach to prune areas of the image to reduce the work done.<sup>146</sup> These will be described below.

There are also approaches which improve the grammar model’s performance in certain situations like object occlusion. They introduce a model of occlusion for partially visible objects.<sup>147</sup> This is especially useful for improving the person context model inside the contextual model. With the special focus on clothing detection, the shape of the person is often occluded or concealed by clothing. Especially dresses, bags, hair and hats can alter the visible shape significantly. Therefore, introducing occlusion reasoning, one could improve the context detector in the model and therefore achieve more accurate context constraints, improving the whole contextual model.

### 12.2 Potential speed improvements

After describing some of the potential performance improvements this section will focus on improving the detection speed.

#### 12.2.1 Speed improvements for detection

Currently the implementation has very high memory requirements, mainly because of the costly grid conversion needed for applying the distance transform algorithm in the scale dimension.

<sup>146</sup> See Felzenszwalb et al. [2010b, p. 4].

<sup>147</sup> See Girshick et al. [2011].

---

### 3D distance transform

In the current implementation, the 3D distance transform is the main reason the detection speeds of contextual models are much slower than of the deformable parts models. Currently the pyramid  $\leftrightarrow$  grid conversion is done in Matlab, which could be improved by moving this part to compilable code. Generating a grid from a pyramid is very space consuming. Depending on the model's feature and score padding the overhead in scaling the smallest pyramid level to the size of the largest pyramid level is enormous. Let score pyramid  $H$  have 50 levels,  $l_H = 50$  with  $\lambda = 10$  levels per octave (a typical pyramid size), and each score level is padded by  $p = (p_x, p_y) = (20, 10)$  (the original padding of the evaluated contextual models). Let  $s_l$  be the size of level  $l$  in pyramid  $H$ , with  $s_{l_H} = (1, 1) + 2p = (41, 21)$  being the size of the feature map at the smallest level. Converting the feature map at level  $l_H$  to the size of  $s_1$  it has to be scaled by the factor of  $2^{(l_H-1)/\lambda} = 2^{4.9}$ , resulting in a grid where each of the feature maps is of size  $S = s_{l_H} \cdot 2^{4.9} \approx (1224, 627)$ . Now the scale distance transform has to be applied to all locations, with 8 byte double precision score values, going through over 292MB of data. This operation has to be done for each component in the contextual model.

Reducing memory requirements also has the possibility to increase performance. Usually detection is being run in parallel threads. With many parallel threads there is a chance that memory runs low and the whole process gets slowed tremendously when virtual memory is getting used. For that case it might be useful to move to a sparse data representation, because large areas of the 3D grid is occupied by empty areas of normal and virtual padding. On the execution side the distance transform could also be stopped as soon as it enters these empty padding areas.

Generalized distance transform is a highly parallelizable operation, so it might be feasible to improve the execution speed by implementing pyramid  $\leftrightarrow$  grid conversion as well as the distance transform algorithm on a graphics processing unit (GPU). With a grid of the size mentioned above the scale distance transform could be split into  $1224 \cdot 627 = 767448$  parallel threads, opening up a lot of potential for performance increase through parallel computing.<sup>148</sup>

### Integrating related work on grammar models

Other speed improvements could be integrated into the model which have been proposed by Felzenszwalb et al. [2010a,b]. Since these speed improvements have been developed as extensions of the deformable parts and grammar models, these could be used to increase the relative detection speed of the contextual model approach.

In Felzenszwalb et al. [2010a] PCA is being used to reduce the filter sizes and therefore increase the detection speeds. They show that the using PCA one can reduce feature dimension from 36 to 11 without decreasing detection performance. Though, "some of the gain is lost because [they] need to perform a relatively costly projection step when computing feature pyramids."<sup>149</sup>

Felzenszwalb et al. [2010b] introduce a cascade detection method into the grammar model, that prunes locations in the input images by using intermediate scores after evaluating each part filter.<sup>150</sup> Using this approach the amount of work that has to be done when detecting objects in an image is reduced significantly, which leads to a 22 fold increase in detection speed on average, without sacrificing performance.<sup>151</sup>

These methods are available in framework version 5 and, thus, could be used in an updated contextual model implementation fairly easily.

---

#### 12.2.2 Speed improvements for training

---

The speed in which a model is being trained is based on three aspects: the training set size, the detection speed and the gradient descent implementation speed. The training set size should usually not be reduced for pure training speed reasons. Also we described some methods to increase detection speeds above.

Training a model like 2.4 which has 4 contextual components currently takes about 48 hours using a high end PC with 8 cores. Speed and memory usage improvements that have been described above should speed this up significantly.

Additionally, the new framework version 5 introduces an implementation of the optimization algorithm which circumvents writing the feature vector cache to the hard disk, and directly trains the model using in memory data. Also the optimization algorithm is supposed to converge faster.<sup>152</sup>

However, in the end the training algorithm mostly consists of relabeling positives and data-mining hard-negatives and running gradient descent in between, which is why detection speeds should be the focus of improvement.

---

<sup>148</sup> Experimental implementations for parallel execution of distance transform on the GPU have been already developed for the purposes of this work, but not used due to Matlab integration issues.

<sup>149</sup> Felzenszwalb et al. [2010a, p. 13].

<sup>150</sup> See Felzenszwalb et al. [2010b, p. 4].

<sup>151</sup> See Felzenszwalb et al. [2010b, p. 7].

<sup>152</sup> See Girshick et al. [2012b].



---

For the future it is planned to extend the training set to share context with multiple object classes. In that case it is important to increase training speeds to make training of large multi-class mixture models feasible.

---

### 12.3 Improve and extend training set

---

Generating a training set to train contextual models is very expensive. Not only do positive examples have to have two bounding boxes – for both object and context – also negative images need to be annotated with context examples, diminishing some of the advantages of the LSVM training algorithm, where data-mining can make efficient use of massive sets of negative examples.

Recall that the contextual model depends on a training set, which has context bounding boxes in the negative images training set  $N$ . This can lead to problems with low quality training sets. For example, if one uses an out-of-the-box person model as the context model (which we do) the problem that many of these context bounding boxes do not fit the context model can arise. In these cases the training algorithm gets a wrong label  $\hat{y}$ . The context label  $\hat{y}$  is determined positive if the context detection window overlaps the context bounding box by 70%, else it is determined to be negative.

The potential problem is that resulting contextual model could get overconfident of people appearances if negative object labels do not correlate enough with positive context labels. This could result in a high number of false positives in test sets where people appear without wearing dresses.

This could be an issue with the trained contextual models shown in Table 1. The current contextual models work well: the contextual model 2.4 outperforms the original model 1.1 in low FPPI intervals of the DET graph, however, implementing methods to prevent overconfidence w.r.t. context appearances from the training data, could further improve the detection performance!

There are two possible approaches: First, the required overlap of context bounding boxes to set the context label  $\hat{y}$  to 1 could be decreased. For example to a minimum overlap of 50%, which is commonly used for testing. This would result in more context instances being detected in negative images, decreasing the overall confidence of the context. Second, it could make sense to increase the amount of negative images with context appearances. Currently the number of context appearances in the negative examples is less than in the positive examples (numerically, since there are 1974 positive training example (with context) and 1614 negative images containing about one context bounding box each). This could reduce the correlation between positive context and positive object, therefore moving the contextual further to the original assumption – where non-existing context inhibits object detections, but existing context does not promote object detections.

The next steps will focus on building a larger dataset, containing the 10 object categories listed above. When training multiple clothing classes one could take advantage of reusing some of the positive examples of other classes, since clothes often appear in combination. Also images with positive examples of a certain class could be used as negative images for other classes if they do not appear in the same image, reusing the required context bounding box. Therefore, the cost to add more and more clothing classes should decrease. This requires structured knowledge about what class is, and is not in each image.

Having more object classes also opens the possibility to analyze and make use of multilateral constraints between items of clothing.

---

### 12.4 Add multilateral spatial constraints

---

The current contextual model has a star-based architecture, which only allows to model spatial relations from context to a single object. Especially in the domain of fashion and street style photography it would be interesting to evaluate multilateral spatial constraints between objects. In addition to the relationship between clothes and wearer, which has been addressed in this work, different types of clothes could also have relationships. Many types of relationships between different clothing items come to mind. For instance spatial relations, e.g. t-shirts are above trousers, some items could be mutually exclusive, like a suit and a dress. Moreover, some categories could have mutual positive correlations, like high heels and dresses, or gender specific clothes in general, others could have negative correlations, like trousers and skirts.

Having a local contextual scope in the form of a person could help leveraging these relationships, especially simple relative occurrence correlations. Integrating multilateral spatial constraints on the other hand is hard using the current model. Many aspects of the model, like part deformations, are based on the fact, that the model is star-shaped and there is no interaction between the parts and their scores. As noted in section 2 Related Work Desai et al. [2011] developed a method which could be used to model spatial relations between items of clothing. As noted before the spatial model in their approach is much more coarse than the flexible deformation model described in this work, which makes it more useful in an environment of multilateral relations.

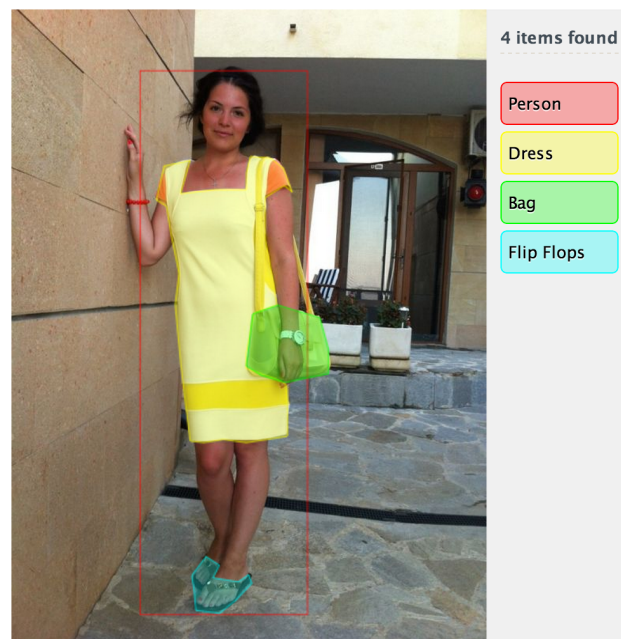
In the future the contextual model will be trained with many more clothing categories, which makes it possible to analyze potential multilateral relationships. The feasibility of making use of these relationships in the current model will vary

considerably, but the goal is to integrate some of them to build a clothing detection framework with a high enough detection performance for use in consumer facing products.

### 12.5 Practical use of the model with flexible context constraint

The goal is to build a system which accepts street style photos (one or a few persons in the foreground, complex backgrounds) and classifies all clothes which are being worn by the visible persons. Having created an object detector with flexible context constraint as described in this work is the first step. Later a large mixture model with many different fashion related object classes sharing the same context should be trained. There will be an substantial amount of work to generate the needed training sets. As noted above, another aspect is the incorporation of methods to exploit other contextual relationship between classes of clothing.

Having detected the object class and location, the next step is to analyze it's properties. Specific shape, colors, texture, patterns and material are especially relevant to find similar products. To analyze these products the specific area of the piece of clothing has to be determined by segmenting it from the background.



**Figure 16:** Mockup of system to automatically detect and segment all clothing items a person is wearing.

Bertelli et al. [2011] offer an approach, which useskerneled structural SVM to perform object segmentation. They define multiplicative kernel consisting of an object similarity kernel and a mask similarity kernel, accommodating a top-down approach using information from object detection in the object similarity kernel and bottom-up cues from color and shape information in the mask similarity kernel.<sup>153</sup> The object similarity kernel can make use of a descriptor generated from the coordinates of an object detector. In fact, they use the deformable parts model detector by Felzenszwalb et al. [2010a] as an example.<sup>154</sup> This makes their approach a sensible extension to the contextual model. The mask similarity kernel in turn contains a shape kernel, local color model kernel and a global color model kernel.<sup>155</sup> Similar to the data-mining of hard negatives in Felzenszwalb et al. [2010a], their algorithm also focusses on learning from hard examples, meaning bad segmentations and, thus, training a model that “knows what good segmentations are”<sup>156</sup>.

The final system could be integrated into a fashion related image database, which then would be automatically searched for items of clothing. Based on these detections, further object analysis as described above could be conducted. Then similar products could be linked to the image and displayed users as related information when viewing images from the database, like shown in Figure 16.

<sup>153</sup> See Bertelli et al. [2011, p. 1f].

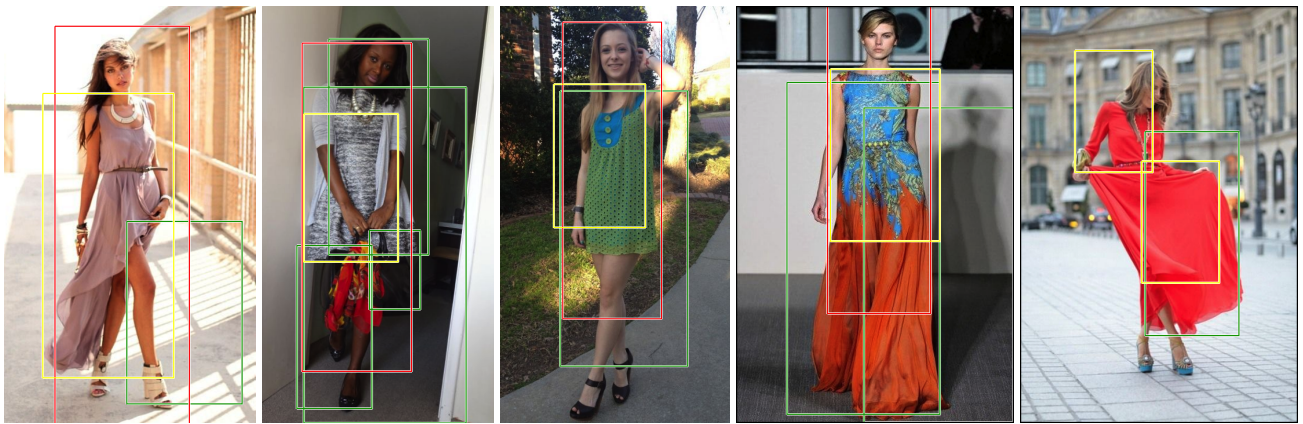
<sup>154</sup> See Bertelli et al. [2011, p. 5].

<sup>155</sup> See Bertelli et al. [2011, p. 3f].

<sup>156</sup> Bertelli et al. [2011, p. 1].

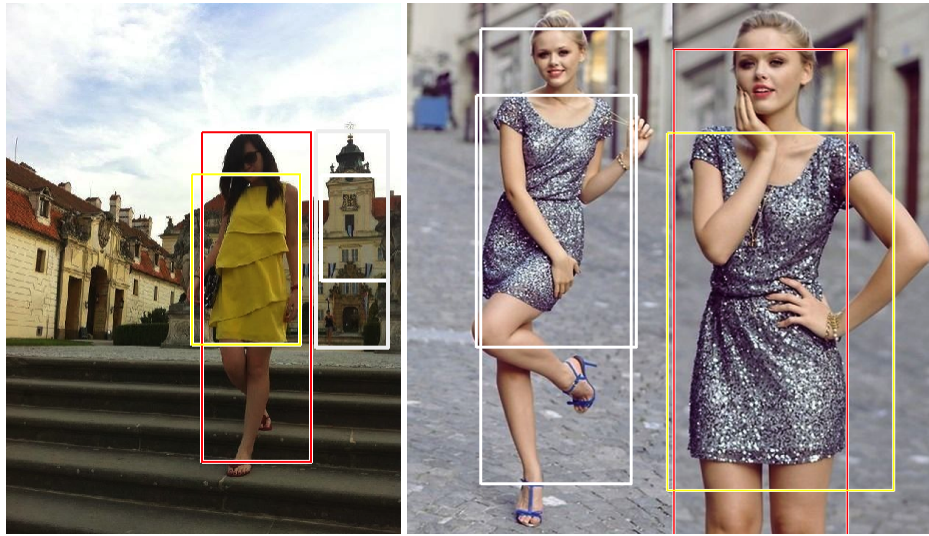


**Figure 17:** Detections using contextual model 2.4. True positives, where original 1.1 model was not confident enough. The threshold for each model has been set to the lowest score where FPPI > 0.3



**Figure 18:** Detections using contextual model 2.4. Yellow boxes show boxes after applying CNMS using red box. Green boxes are being discarded. Left: CNMS helps discarding false positives. Second from right: CNMS discards a true positive detection. Far right: false positive detections. Not sorted or filtered by score.





**Figure 19:** Interesting detections. Left: confident true positive detection (yellow/red) and low confidence false positive at clock tower (white). The clock tower somehow resembles the shape of a person. Right: confident true positive detection (yellow/red) and low confident true positive (white). The pose of the person reduces context score.



**Figure 20:** Detections Contextual Model 2.4 (green). Examples the dress was detected by the contextual model but the dress size did not match. The rightmost photo also contains a false positive from original model 1.1 (red). (FPPI > 0.3)

---

## 13 Bibliography

---

- Tuncer C Aysal and Daniel C Heesch. Multi-resolution and multi-bit representation for image similarity search. In *Multimedia, 2009. ISM'09. 11th IEEE International Symposium on*, pages 290–297. IEEE, 2009.
- Luca Bertelli, Tianli Yu, Diem Vu, and Burak Gokturk. Kernelized structural svm learning for supervised object segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2153–2160. IEEE, 2011.
- Léon Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17:9, 1998.
- Crowdfower. Gold. Retrieved May 30, 2013, from <https://crowdfower.com/docs/gold>, 2013.
- Navneet Dalal. Inria person dataset. Retrieved April 23, 2013, from <http://pascal.inrialpes.fr/data/human/>, 2005.
- Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.
- Chaitanya Desai, Deva Ramanan, and Charless C Fowlkes. Discriminative models for multi-class object layout. *International journal of computer vision*, 95(1):1–12, 2011.
- M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.
- Mark Everingham, Luc van Gool, Chris Williams, John Winn, and Andrew Zisserman. Voc2012 results. Retrieved June 6, 2013, from <http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2012/results/index.html>, 2012.
- P F Felzenszwalb, R. B. Girshick, and D. McAllester. Discriminatively trained deformable part models, release 4. Retrieved April 17, 2013, from <http://people.cs.uchicago.edu/~pff/latent-release4/>, 2011.
- Pedro Felzenszwalb and David McAllester. Object detection grammars. *University of Chicago, Computer Science TR-2010-02, Tech. Rep*, 2010.
- Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.
- Pedro Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010a.
- Pedro F. Felzenszwalb and Daniel P Huttenlocher. Distance transforms of sampled functions. Technical report, Cornell Computing and Information Science, 2004.
- Pedro F Felzenszwalb, Ross B Girshick, and David McAllester. Cascade object detection with deformable part models. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 2241–2248. IEEE, 2010b.
- Andrew C Gallagher and Tsuhan Chen. Clothing cosegmentation for recognizing people. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.
- R. B. Girshick, P F Felzenszwalb, and D. McAllester. Discriminatively trained deformable part models, release 5. Retrieved April 17, 2013, from <http://people.cs.uchicago.edu/~rbg/latent-release5/>, 2012a.
- R. B. Girshick, P F Felzenszwalb, and D. McAllester. Changelog voc-release5. Retrieved April 17, 2013, from <http://www.cs.berkeley.edu/~rbg/latent/changelog.txt>, 2012b.
- Ross Girshick, Pedro Felzenszwalb, and David McAllester. Object detection with grammar models. *IEEE TPAMI*, 33:12, 2011.
- Jeremy Heitz and Daphne Koller. Learning spatial context: Using stuff to find things. In *Computer Vision–ECCV 2008*, pages 30–43. Springer, 2008.
- Zhilan Hu, Hong Yan, and Xinggang Lin. Clothing segmentation using foreground and background estimation based on the constrained delaunay triangulation. *Pattern Recognition*, 41(5):1581–1592, 2008.
- Mohammad Jahangiri and Daniel Heesch. Modified grabcut for unsupervised object segmentation. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 2389–2392. IEEE, 2009.

- 
- Mohammad Jahangiri, Daniel Heesch, and Maria Petrou. Crf based region classification using spatial prototypes. In *Digital Image Computing: Techniques and Applications (DICTA), 2010 International Conference on*, pages 510–515. IEEE, 2010.
- Krzysztof C Kiwiel. Convergence and efficiency of subgradient methods for quasiconvex minimization. *Mathematical programming*, 90(1):1–25, 2001.
- Si Liu, Zheng Song, Guangcan Liu, Changsheng Xu, Hanqing Lu, and Shuicheng Yan. Street-to-shop: Cross-scenario clothing retrieval via parts alignment and auxiliary set. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3330–3337. IEEE, 2012.
- Alvin Martin, George Doddington, Terri Kamm, Mark Ordowski, and Mark Przybocki. The det curve in assessment of detection task performance. Technical report, DTIC Document, 1997.
- Jonathan Milgram, Mohamed Cheriet, and Robert Sabourin. Estimating accurate multi-class probabilities with support vector machines. In *Neural Networks, 2005. IJCNN’05. Proceedings. 2005 IEEE International Joint Conference on*, volume 3, pages 1906–1911. IEEE, 2005.
- S. Miller, M. Fritz, T. Darrell, and P. Abbeel. Parametrized shape models for clothing. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4861–4868. IEEE, 2011.
- John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.
- Andrew Rabinovich, Andrea Vedaldi, Carolina Galleguillos, Eric Wiewiora, and Serge Belongie. Objects in context. In *ICCV, Rio de Janeiro*, 2007.
- Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986. ISBN 0070544840.
- Paul Schnitzspan, Stefan Roth, and Bernt Schiele. Automatic discovery of meaningful object parts with latent crfs. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), San Francisco, 06/2010* 2010. Technische Universität Darmstadt.
- Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th international conference on Machine learning*, pages 807–814. ACM, 2007.
- Antonio Torralba, Kevin P Murphy, and William T. Freeman. Contextual models for object detection using boosted random fields. In *In NIPS*, 2004.
- Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699. ACM, 2002.