Detecting partially occluded objects in images

Samarth Manoj Brahmbhatt

A THESIS

in

ROBOTICS

Presented to the Faculties of the University of Pennsylvania in Partial

Fulfillment of the Requirements for the Degree of Master of Science in Engineering

2014

Dr. Kostas Daniilidis

Supervisor of Thesis

Dr. C. J. Taylor

Graduate Group Chairperson

Dedication

To the Flow

Acknowledgements

First and foremost I would like to thank my advisor Professor Kostas Daniilidis for his help and valuable insights throughout the thesis.

I would like to also thank Menglong Zhu for his friendly advice at various stages, and Thomas Koletschka, Qin He, Qiong Wang, Larry Abraham and Dinesh Thakur for good times at the GRASP lab, whether it be day or night.

Thanks are also due to my flat mates Priyank Chaudhary and Chaitanya Bhargava, who are ignorant but jestingly outspoken critics of my all work, and Tarun Bhattacharya, and Anoushka Shankar and Daft Punk, whose music kept me going through all those dataset annotations.

Lastly, I thank my family for supporting me throughout the Masters' program and giving me hope in the few times when I felt let down.

Abstract

Object detection is one of the oldest problems in computer vision which have eluded a 'grand unified theory' till date. Occlusion of objects is a major concern for many object detection algorithms, and indeed all algorithms that output a simple bounding box. The first part of this thesis explores existing research in *segmentation-aware object detection*, which depends on pixel level binary object/not-object segmentation of the contents of the bounding box for detection. This incorporation of pixel-level labelling provides an intuitive method of dealing with occlusion. The labelling is achieved by performing maximum a-posteriori inference on a grid binary Markov Random Field (MRF) defined over the bounding box. The second part of this thesis describes an effective approach for defining pairwise terms in the MRF that encourages visual consistency and a strategy using this framework for detecting pairs of objects - localizing them and labelling the contents of their bounding boxes such that the 'agreement' between the two objects is maximized.

Contents

List of Figures

List of Tables

Introduction

Object detection is one of the basic problems in computer vision and has been extensively researched. Given a test image, every object detection algorithm outputs one or more detection hypotheses, and associates a 'score' with each hypothesis. One can then threshold this score and get detections in the image. The most popular form of representing a detection hypothesis is a bounding box, which is a rectangular region enclosing the area in the image in which the detector thinks an object is present. While a rectangular is the simplest way to indicate a region of the image, it is not optimal for many applications. If the object itself is not rectangular, or if the object is partially occluded, not all the pixels in the bounding box belong to the object. Many applications such as pose estimation, scene understanding and augmented reality require a pixel- or small block-level segmentation of the object. Broadly, there are two types of object detection algorithms prevalent today.

Keypoint-based detectors

Keypoint-based algorithms try to search for distinctive points in the test image see if they 'look' like some stored keypoints (extracted from training images) in some feature space. If a lot of keypoint matches are found in a region of the test image, that region gets a higher score. Typically, the keypoints are extracted by maximizing some measure of distinctiveness and repeatability. Features are extracted from a support region around the keypoint to capture local appearance information. By normalizing the feature descriptor extraction process with respect to scale and in-plane rotation the keypoint descriptors gain invariance to changes in size and in-plane rotation of the object. For example, the popular SIFT features [17] define keypoints as extrema in the Difference of Gaussian filter applied to the image pyramid, and construct the feature descriptor by stacking histograms of gradient orientations in a circular region around the keypoint whose size proportional to its scale in the pyramid. SURF [3] also uses a gradient orientation histogram based approach but uses integral images and box filter approximation of Gaussan filters to speed up keypoint extraction. ORB

[20] attains rotation invariance by modifying BRIEF [6], which uses a binary features extracted by comparing point intensity values around the keypoint. The use of binary feature descriptors speeds up descriptor matching considerably. The advantage of keypoint based object detectors is that the search space is greatly reduced by the keypoint extraction step and that partial occlusion can be handled because the object is not represented holistically. On the other hand, to get good repeatable keypoints, both the keypoint extraction and description steps tend to be quite involved. Simpler descriptors tend to produce many false-positive matches at test time, leading to large high-dimensional matching problems. But the biggest drawback of keypoint based approaches is that since they are only interested in distinctive points, the object of interest needs to be well-textured. Additionally, since only sparse areas of the object are stored in the model, it is not possible to get a pixel level binary segmentation of the bounding box unless keypoint descriptors are extracted in a dense grid, which essentially reduces to a sliding window approach discussed next.

Sliding Window detectors

Sliding window based algorithms work densely on the whole image (and often also at all levels of the image pyramid). They basically convert the image to some feature space and learn weights for a rectangular filter such that the convolution of the filter with the feature-image will produce high scores at regions where the filter covers the object. The filter weights are often learned with the help of a Support Vector Machine (SVM) with a linear kernel. Recently the Histogram of Oriented Gradients (HOG) features [7] have become quite popular. HOG divides the image into small cells (usually 8x8 pixels) and constructs a histogram of gradient orientation from every pixel in the cell, weighted by the gradient magnitude at the cell. Histograms of blocks of cells are then L_1 normalized to gain contrast invariance. A PCA-reduced and slightly modified version of HOG is used as the main feature in the highly successful Deformable Part Model (DPM) object detection system [8]. The advantage of sliding window based methods is that detection is as simple as convolving a filter with the feature-image and picking the best after a non-maximum suppression step. However, the convolution needs to be done over the whole image pyramid. In addition, algorithms using simple features like HOG cannot be rotation invariant, a problem that has to be handled by learning different filters for different object poses and picking the maximum response. Occlusion decreases the filter response. Hence if a tight threshold is used on the score to increase precision, occlusion cases will not be handled, causing a sharp drop in recall on a challenging dataset. This seeming drawback can be used to our advantage, though.

If one models not only the object but also occlusion and at every HOG cell picks the one with the higher response, occlusion will not decrease the filter response (provided the occlusion-modelling filter is general enough). This strategy will prevent occlusion cases from being discarded when a tight threshold is used on the score to decide detections. This is the main idea behind segmentation aware object detection [10].

Part-based detectors

Another class of object detectors are worth mentioning here - part-based detectors. These have been made popular recently by a series of papers [8, 9, 11] and good performance in the Pascal VOC challenges. These algorithms also learn filters for dense features, but do not represent objects as a whole. Instead they divide the object into parts and allow the parts to be displaced about the center by paying a deformation cost. The detection is then the location at which the sum of the part appearance scores (convolution of part filter with image feature) and deformation cost (a negative quantity) is maximized. This introduces flexibility and performs better than algorithms that represent the object as a single filter.

In this thesis I will be working with a sliding window detector that represents the the object holistically. Ideas for extending the algorithm to a part-based framework are discussed in section 4. The basic idea is to train two HOG filters - one that models the appearance of the object, and one that models occlusion, and then choose the label with the higher response at every cell. The second filter does not model any object specifically, it just serves the purpose of giving a higher response at occluding areas than the object filter. In this sense, the second filter will be better (more general) if the dataset has a lot of varied cases of occlusion. In practice due to limited training images, errors in ground truth labelling and the high dimensionality of features involved, the filter weights will not be optimal. Hence naively picking the label with the highest response at every HOG cell does not produce visually consistent results. One often needs to make a grid MRF by making a node at every HOG cell and connecting pairs nodes with edges. The unary potentials at each node are the responses of the two filters, and pairwise potentials are edge weights. Typically a labelling scheme gets a ‘reward’ equal to the edge weight if an only if two nodes connected by an edge have the same 0/1 label. Maximum a-posteriori labelling is achieved by picking the labelling that maximizes the sum of all unary and pairwise potentials. A combinatorial search through all labellings would require time exponential in the number if nodes, so we use network-flow theory to get a quick (and in many cases, optimal) solution.

Related Work

Combining detection and segmentation

This thesis builds upon the research and framework of [10], in which the authors use a structured output SVM to train the HOG filter weights and then emply Quadratic Pseudo-Boolean Optimization (QPBO) to infer a MAP labelling on the MRF. They define pairwise terms over nodes that belong horizontal and vertical edges inside the bounding box, thus encouraging label consistency along horizontal and vertical edges. [13] also uses detection to aid segmentation but the object detector is fixed and does not take advantage of segmentation. [22] uses visibility labelling for pixels inside the bounding box. However, the visibility is not inferred from appearance, and depends just on the position of the bounding box in the image. Pixels in the bounding box being truncated by the image boundary are labelled as invisible.

MRF optimization

MAP inference in MRFs is a very active topic of research. All the following algorithms are ways to pose the MAP inference problem as a graph-cut problem. For MRFs with just pairwise terms, algorithms such as α-expansion and $\alpha - \beta$ swap algorithms [4] can be used to infer both binary and multi-class labels. One α-expansion move allows all nodes to either keep their current label or take on the label α. An iteration of the α-expansion move involves such moves for all labels in the label set in some order. According to [4] if all pairwise potentials define a metric, the energy of the labelling computed using this algorithm is guaranteed to be within a known approximation factor of the optimum energy. One move of the $\alpha - \beta$ swap algorithm allows all nodes whose current label is either α or β to to take on the label either α or β. An iteration of the algorithm involves such moves for all pairs α and β in the label set in some order. According to [4] if all pairwise potentials define a semi-metric, the energy of the labelling computed using this algorithm is guaranteed to be within a known approximation factor of the optimum energy.

QPBO [16] can be used for exact MAP inference in binary MRFs containing pairwise terms when all the pairwise potentials are submodular [1]. However, it does not perform exact inference when some of the pairwise terms are not submodular in the sense that it gives null labels for some nodes. In this case, the persistency property of the algorithm states that there exists an optimum labelling such that its labellings, for all the nodes for which QPBO does produce a non-null label, are the same.

Recently [15] has noted better results by using higher-order potentials as opposed to just pairwise potentials. Intuitively, higher order terms consist of cliques of nodes (such as nodes belonging to the same segment in the output of an image segmentation algorithm) and the algorithm encourages all nodes in the same clique to have the same label. This is achieved by converting certain classes of higher order potentials to submodular functions, which are then optimized as usual using graph-cuts.

It should be mentioned that since we are considering pairwise rewards for two nodes to have the same label and filter responses as unary potentials, it is intuitively easier to think about the optimum labelling as one that *maximizes* the total energy. In addition, the structural SVM formulation for learning the filter weights introduced later also defines a detection as one that maximizes the dot product of feature and weight vectors. However, all the graph-cut algorithms *minimize* the total energy of the MRF. The conversion from maximization to minimization is described in Section 2.2.1.

Learning MRF weights

In our case learning MRF weights involves learning HOG filter weights (for unary potentials) and the pairwise weight (same for all pairs). A lot of the recent MRF segmentation papers that use MRF optimization use weights for features selected by cross validation [15]. [19] uses iterated estimation of GMM parameters from image data and user interaction to learn the MRF parameters. However, both these options are intractable in our case since the stacking of 31-dimensional HOG features for all cells in the bounding box makes a high dimensional space. [12] uses structural SVMs to learn parameters for unary and higher order potentials for binary segmentation, while [18] extends the framework to multi-class segmentation.

[1] $\psi(0,0) + \psi(1,1) \leq \psi(0,1) + \psi(1,0)$

Chapter 1

Feature representation

Given an image $\mathbf{x}$ we want to predict a bounding box enclosing the object of interest and infer the pixels inside the box that belong to the object. The image is represented in terms of its HOG features. The 31-dimensional variant of HOG described in [8] is used, with cell size of 8x8 pixels. A bounding box is represented as $\mathbf{p} = (p_x, p_y)$, indicating the position of the top left corner of the bounding box. The width and height of the box are fixed as w and h respectively. Since HOG features are used to represent an image, the unit of w and h is hog cells. To achieve scale invariance one can extract features and search over the entire test image pyramid instead of just the image itself. The vector $\mathbf{v}$ consists of $w * h$ binary variables, each indicating whether the corresponding HOG cell is labelled as belonging to the object or not. Hence the labelling of an image is of the form $\mathbf{y} = (p_x, p_y, \mathbf{v})$.

A joint feature vector is constructed by stacking five components described below.

The first component corresponds to the HOG features, two groups stacked together. The HOG features of the cells labelled as not-object are zeroed out in the first group, and vice versa for the second group.

$$\begin{bmatrix} (\mathbf{v} \otimes \mathbf{1}_N) \odot \phi(\mathbf{x}, \mathbf{y}) \\ (\mathbf{1}_{wh} - \mathbf{v} \otimes \mathbf{1}_N) \odot \phi(\mathbf{x}, \mathbf{y}) \end{bmatrix} \tag{1.1}$$

where $\odot$ denotes component-wise multiplication between two vectors and $\otimes$ is the Kronecker product. $\phi(\mathbf{x}, \mathbf{y})$ denotes the vector formed by stacking the HOG features of all cells of image $\mathbf{x}$ inside bounding box $\mathbf{p}$.

The second component is used to learn a prior for a cell to not belong to the object.

$$[\mathbf{1}_{wh} - \mathbf{v}] \tag{1.2}$$

The third component is used to learn a cost for truncation by the image boundaries.

$$\mathbf{c}(\mathbf{p}) \tag{1.3}$$

which denotes the number of cells in bounding box **p** lying outside the image boundaries.

The fourth component encourages smoothness of the visibility labelling and is used to learn the weight for pairwise terms. First, we oversegment the image using the gPb - UCM algorithm [1]. To maximize the fraction of pairs of nodes having the same labelling, expected over all image segments, we include

$$\frac{1}{|C|}\sum_{c \in C} \frac{1}{\binom{|c|}{2}} \sum_{(i,j) \in c} \mathbf{1}(v_i = v_j) \tag{1.4}$$

where C is the set of all segments.

The last component is used to learn a bias for the model, and is used while training a joint model for multiple views of the same object or while training models for two objects that need to be compatible with each other (see Section 3.1).

$$1 \tag{1.5}$$

Chapter 2

Structured Training and Inference

2.1 Training feature weights with a Structured SVM

Our goal is to learn a function $f: X \rightarrow Y$ that predicts a bounding box and labelling $\mathbf{y} \in Y$ given an image $\mathbf{x} \in X$. This is done using the structured learning framework [14] as:

$$f(\mathbf{x}) = \underset{\mathbf{y} \in Y}{\operatorname{argmax}}\, \mathbf{w}^T \Psi(\mathbf{x}, \mathbf{y}) \tag{2.1}$$

where $\mathbf{w} \in \mathbb{R}^d$ is the vector of learned model parameters (weights) and $\Psi(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^d$ is the joint feature vector constructed as described in Chapter 1.

The model parameters are learned in the max-margin framework with the 1-slack formulation [14]:

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

s.t.

$$\forall (\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots, \hat{\mathbf{y}}_N) \in Y^N : \frac{1}{N} \mathbf{w}^T \sum_{i=1}^{N} (\Psi(\mathbf{x}_i, \mathbf{y}_i) - \Psi(\mathbf{x}_i, \hat{\mathbf{y}}_i)) \geq \frac{1}{N} \sum_{i=1}^{N} \Delta(\mathbf{y}_i, \hat{\mathbf{y}}_i) - \xi \tag{2.2}$$

Intuitively this formulation requires that the score $\Psi(\mathbf{x}_i, \mathbf{y}_i)$ of any ground truth labelled image $\mathbf{x}_i$ must be greater than the score $\Psi(\mathbf{x}_i, \hat{\mathbf{y}}_i)$ of any other labelling $\hat{\mathbf{y}}_i$ by the distance between the two labellings $\Delta(\mathbf{y}_i, \hat{\mathbf{y}}_i)$ minus the slack variable ξ, where $\mathbf{w}$ and ξ are minimized. The above formulation has $\mid Y \mid^N$ (exponential) constraints, one for each possible combination of labels $(\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots, \hat{\mathbf{y}}_N) \in Y^N$. Because training by satisfying such a huge number of constraints is intractable, the cutting plane training algorithm deals with this problem by maintaining a working set W of most violated constraints and choosing combinations $(\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots, \hat{\mathbf{y}}_N)$ only from W. The procedure for computing the most violated constraint for a given labelled training example is called 'loss augmented inference' and is very similar to the actual inference procedure. It is described in the next section.

The distance between the two labellings, also known as the loss function, depends just on the portion of overlap between the boxes:

$$\Delta(\mathbf{y}_i, \hat{\mathbf{y}}_i) = \begin{cases} 0 & \text{if } \mathbf{y}_i = \hat{\mathbf{y}}_i = \varnothing \\ 1 & \text{if } \mathbf{y}_i = \varnothing, \hat{\mathbf{y}}_i \neq \varnothing \text{ or } \mathbf{y}_i \neq \varnothing, \hat{\mathbf{y}}_i = \varnothing \\ 1 - \frac{\text{area}(\mathbf{y_i} \bigcap \hat{\mathbf{y}}_i)}{\text{area}(\mathbf{y_i} \bigcup \hat{\mathbf{y}}_i)} & \text{otherwise} \end{cases} \tag{2.3}$$

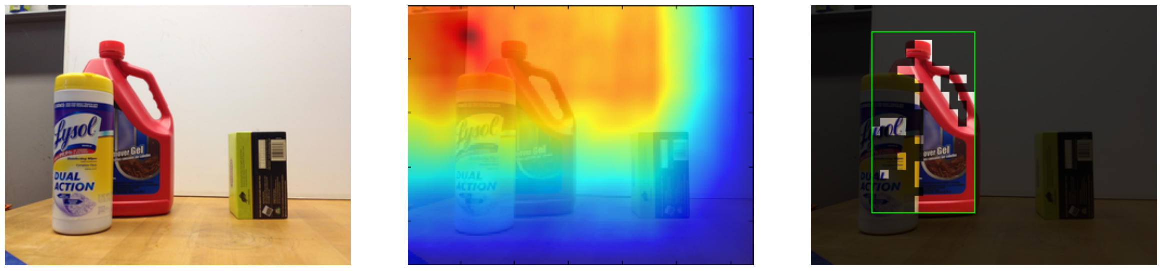
We use Andrea Vedaldi's MEX implementation [21] of Joachims' Structural SVM [14] to learn the model.

2.2 Inference

Inference for an image $\mathbf{x}$ is performed as:

$$\mathbf{y}^* = \underset{\mathbf{y} \in Y}{\operatorname{argmax}} \, \mathbf{w}^T \Psi(\mathbf{x}, \mathbf{y}) \tag{2.4}$$

Figure 2.1 shows the result of inference in terms of a heatmap of scores $\mathbf{w}^T \Psi(\mathbf{x}, \mathbf{y}$ and the subsequent output. Note that redder regions represent higher scores in the heatmap and the heatmap corresponds to location of the top-left corner of the bounding box.



(a) Image (b) Score heatmap (c) Detection

Figure 2.1: Detection process

For an image $\mathbf{x}_i$ with ground truth labelling $\mathbf{y}_i$ loss augmented inference gives the most violated constraint $\hat{\mathbf{y}}$ as:

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in Y}{\operatorname{argmax}} \, \mathbf{w}^T \Psi(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y}) \tag{2.5}$$

Here $\mathbf{y} \in Y = P \times V$, where P is the space of all possible locations of the top left corner of the bounding box and V is the space of all possible $0/1$ labels for the $w * h$ HOG cells in the bounding box. Searching through this exponential space for the most violated constraint is impractical. Hence in this section we discuss an inference strategy where the bounding box locations are enumerated and given a location $\mathbf{p}$, $\mathbf{v}$ is inferred by graph cut. In Section 2.3 we discuss a strategy for pruning the P space by using Deformable Part Model detections.

Given a bounding box location $\mathbf{p}$, the visibility variables $\mathbf{v}$ are inferred using the following formulation:

$$\mathbf{v}^* = \operatorname*{argmax}_{\mathbf{v}} \Bigg(\sum_{i=1}^{w*h} \left(\mathbf{w}_{h,i}^T \phi_i(\mathbf{x}, \mathbf{p})\right) \cdot v_i + \sum_{i=1}^{w*h} \left(\mathbf{w}_{\hat{h},i}^T \phi_i(\mathbf{x}, \mathbf{p})\right) \cdot (1 - v_i) + \sum_{i=1}^{w*h} w_{p,i} \cdot (1 - v_i) + \frac{1}{|C|} \sum_{c \in C} \frac{1}{\binom{|c|}{2}} \sum_{(i,j) \in c} w_e \cdot \mathbf{1}(v_i = v_j) \Bigg) \tag{2.6}$$

The first two terms are unary potentials, third term corresponds to the prior for a cell to not belong to the object and the last term is made of pairwise potentials. $\mathbf{w}_{h,i}$ are the weights for cell i when it is visible (first group of first component in the feature vector), $\mathbf{w}_{\hat{h},i}$ are the weights for cell i when it is not visible (second group of first component in the feature vector) and $\phi_i(\mathbf{x}, \mathbf{p})$ are the HOG features for cell i. Refer (1.1). $w_{p,i}$ is the prior for cell i learnt from the second component of the feature vector (refer (1.2)). w_e is the weight of pairwise terms (a positive number because of the maximizing formulation) learnt from the fourth component of the feature vector (refer (1.4)). Problem (2.6) can be solved using QPBO [16], but before that it has to be formulated as a minimization problem.

2.2.1 Inferring Visibility variables by QPBO

Suppose that

$$Q = \sum_{i=1}^{w*h} \mathbf{w}_{h,i}^T \phi_i(\mathbf{x}, \mathbf{p}) + \mathbf{w}_{\hat{h},i}^T \phi_i(\mathbf{x}, \mathbf{p}) \tag{2.7}$$

Then it follows that the first two terms of (2.6) can be rewritten as

$$\sum_{i=1}^{w*h} \left(\mathbf{w}_{h,i}^T \phi_i(\mathbf{x},\mathbf{p})\right) \cdot v_i + \sum_{i=1}^{w*h} \left(\mathbf{w}_{\hat{h},i}^T \phi_i(\mathbf{x},\mathbf{p})\right) \cdot (1-v_i) = Q - \left(\sum_{i=1}^{w*h} \left(\mathbf{w}_{\hat{h},i}^T \phi_i(\mathbf{x},\mathbf{p})\right) \cdot v_i + \sum_{i=1}^{w*h} \left(\mathbf{w}_{h,i}^T \phi_i(\mathbf{x},\mathbf{p})\right) \cdot (1-v_i) \right) \quad (2.8)$$

And hence (2.6) can be rewritten as a minimization problem as follows:

$$\mathbf{v}^* = \operatorname*{argmin}_{\mathbf{v}} \left(\sum_{i=1}^{w*h} \left(\mathbf{w}_{\hat{h},i}^T \phi_i(\mathbf{x},\mathbf{p})\right) \cdot v_i + \sum_{i=1}^{w*h} \left(\mathbf{w}_{h,i}^T \phi_i(\mathbf{x},\mathbf{p})\right) \cdot (1-v_i) + \sum_{i=1}^{w*h} -w_{p,i} \cdot (1-v_i) + \frac{1}{|C|} \sum_{c \in C} \frac{1}{\binom{|c|}{2}} \sum_{(i,j) \in c} -w_e \cdot \mathbf{1}(v_i = v_j) \right) \quad (2.9)$$

The procedure is summed up in Algorithm 1. We use Shai Bagon's MEX wrappers [2] to the QPBO library by Kolmogorov [5] to run the QPBO optimization in Matlab.

Algorithm 1 Seg-Detect: Detect object by searching over whole image

```
Require: HOG-image $\mathbf{x}$, Model weights $\mathbf{w}$, feature function $f(\mathbf{x},\mathbf{p},\mathbf{v})$ implementing Chapter 1
  $Y \leftarrow \{(i,j) : (i,j) \in \mathbf{x}\}$
  $\mathbf{p}^* \leftarrow (0,0)$
  $\mathbf{v}^* \leftarrow \mathbf{0}$
  $maxscore \leftarrow 0$
  for all $\mathbf{p}$ in $Y$ do
    $\mathbf{v} = \operatorname*{argmax}_{\mathbf{v}} \left( \sum_{i=1}^{w*h} \left(\mathbf{w}_{h,i}^T \phi_i(\mathbf{x},\mathbf{p})\right) \cdot v_i + \sum_{i=1}^{w*h} \left(\mathbf{w}_{\hat{h},i}^T \phi_i(\mathbf{x},\mathbf{p})\right) \cdot (1-v_i) + \sum_{i=1}^{w*h} w_{p,i} \cdot (1-v_i) + \frac{1}{|C|} \sum_{c \in C} \frac{1}{\binom{|c|}{2}} \sum_{(i,j) \in c} w_e \cdot \mathbf{1}(v_i = v_j) \right)$
    $score \leftarrow \mathbf{w}^T f(\mathbf{x},\mathbf{p},\mathbf{v})$
    if $score > maxscore$ then
      $\mathbf{p}^* \leftarrow \mathbf{p}$
      $\mathbf{v}^* \leftarrow \mathbf{v}$
    end if
  end for
  return $\mathbf{y}^* = (\mathbf{p}^*, \mathbf{v}^*)$
```

2.3 Using DPM detections to prune the bounding box search space

Deformable Part Models [8] is a highly successful algorithm for detecting objects (refer to the section on 'Part-based detectors' in Introduction for more information). However, occlusion reduces the score of the part being occluded and hence the score of the entire detection. So occluded objects are often discarded if the score threshold is set high, and if the scene is highly textured the score of an occluded detection will often be less than that of a false positive. But due to the part-based approach occluded detections almost always show up in the top few detections and hence they can be used to prune the space P. In practice we take the first 3 detections made by a DPM model trained using images of the fully visible object and restrict $\mathbf{y}$ in (2.4) and (2.5) to the union of these 3 bounding boxes. The search is still required because DPM boxes are usually not tight around the object and the segmentation-aware detection has a box of fixed size. This makes normal and loss-augmented inference significantly faster. The procedure is summed up in Algorithm 2.

Algorithm 2 DPM-Seg-Detect: Detect object by searching over DPM hypotheses

```
Require: HOG-image $\mathbf{x}$, Model weights $\mathbf{w}$, feature function $f(\mathbf{x},\mathbf{p},\mathbf{v})$ implementing Chapter 1,
    DPM detection bounding boxes $D_1, D_2, D_3$
  $Y \leftarrow \{(i,j) : (i,j), (i+w, j+h) \in D_1 \bigcup D_2 \bigcup D_3\}$
  $\mathbf{p}^* \leftarrow (0,0)$
  $\mathbf{v}^* \leftarrow \mathbf{0}$
  $maxscore \leftarrow 0$
  for all $\mathbf{p}$ in $Y$ do
    $$\mathbf{v} = \operatorname*{argmax}_{\mathbf{v}} \left( \sum_{i=1}^{w*h} \left(\mathbf{w}_{h,i}^T \phi_i(\mathbf{x},\mathbf{p})\right) \cdot v_i + \sum_{i=1}^{w*h} \left(\mathbf{w}_{\hat{h},i}^T \phi_i(\mathbf{x},\mathbf{p})\right) \cdot (1-v_i) + \sum_{i=1}^{w*h} w_{p,i} \cdot (1-v_i) + \frac{1}{|C|} \sum_{c \in C} \frac{1}{\binom{|c|}{2}} \sum_{(i,j) \in c} w_e \cdot \mathbf{1}(v_i = v_j) \right)$$
    $score \leftarrow \mathbf{w}^T f(\mathbf{x},\mathbf{p},\mathbf{v})$
    if $score > maxscore$ then
      $\mathbf{p}^* \leftarrow \mathbf{p}$
      $\mathbf{v}^* \leftarrow \mathbf{v}$
    end if
  end for
  $Dset = \{D : \mathbf{p}^* \in D, D \in \{D_1, D_2, D_3\}\}$
  return $\mathbf{y}^* = (\mathbf{p}^*, \mathbf{v}^*), D^* = \operatorname{argmin}_{D \in Dset} area(D)$
```

This algorithm also has better performance (in terms of precision-recall, refer Section 4.2) than DPM alone on our datasets, because in some occluded cases the true detection is second or third ranked in terms of DPM score, but since we search over all three top boxes we find the correct location. Figure 2.2 shows two such cases. The top left corners of the bounding boxes show their rank and score.

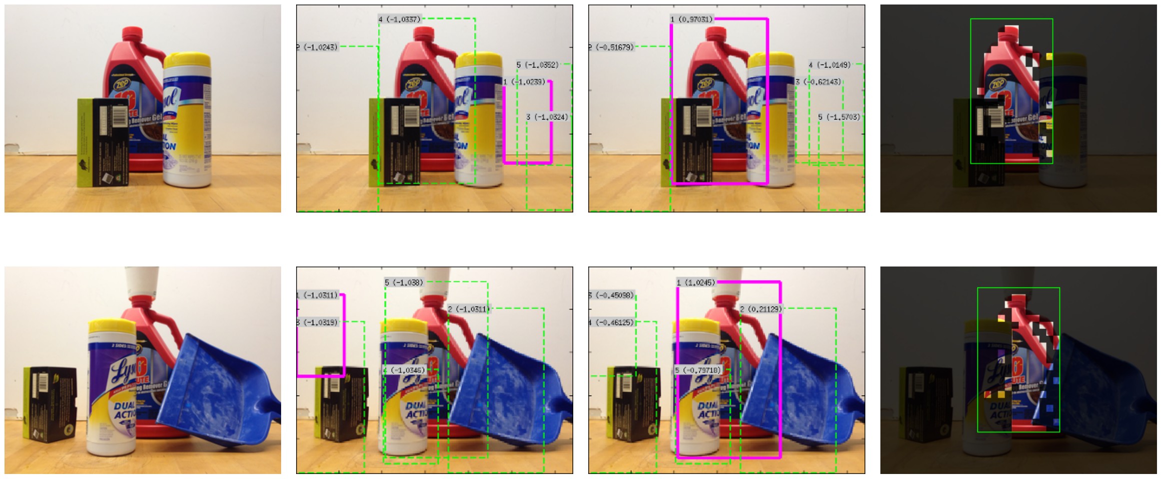

(a) Image (b) DPM detections (c) Rescored DPM (d) Final Detection

Figure 2.2: Detecting Pairs of Objects

2.4 Extending the algorithm

The algorithms described above have two important shortcomings: they can detect objects of a single size and pose. However, both these shortcomings have known solutions. To enable detection of objects independent of scale the test image is converted in to an image pyramid, features are extracted over the whole pyramid and the search for most violated constraint as well as for score maximize detection also proceeds over the whole pyramid. To enable the algorithm to detect objects in multiple poses one needs to train a joint model by stacking feature vectors for all components into one long feature vector. For mode details refer Section 3.1, which discusses joint models for pairs of objects.

Chapter 3

Detecting Pairs of Objects

In this section we discuss an approach for detecting ‘configurations of objects’ i.e. detecting not only an object of interest but also all other objects occluding it. As a first step we have developed an algorithm for detecting the object of interest and the one other object occluding it.

3.1 Training joint models

As explaned in Section 3.2, detecting pairs of objects involves using the filter weights of one model while inferring visibility for another object. Weights in individual models trained as described in Section 2.1 are not compatible with weights in models for other objects. Hence, the two models must be trained jointly. The dataset must be marked with model ID along with bounding box top-left location, and it should have separate visibilities for both the objects. While training, the feature vectors must be stacked together, with a zero vector if a particular object is not present in the image. Here the fifth component of the feature vector, the constant 1, comes in handy. The SVM learns a weight for this constant feature (bias) for both models that makes them interoperable. The loss function, as described in (2.3) now returns the loss of a particular label with the ground truth label of the same class having the highest overlap.

3.2 Inference

First we run a DPM detector for both objects individually and establish a few hypotheses. Next we re-score these hypotheses by running the segmentation aware object detector as described in Algorithm 2. This algorithm returns the highest scoring bounding box location $\mathbf{p}^*$, its labelling $\mathbf{v}^*$

and the DPM box $\mathbf{D}^*$ that $\mathbf{p}^*$ belongs to. If the $\mathbf{D}^*$ for the two objects do not intersect nothing more needs to be done; their respective $\mathbf{p}^*$ and $\mathbf{v}^*$ are returned. Else, we run a second pass, searching simultaneously over the two intersecting DPM boxes. This time we run an α-expansion algorithm in the area of intersection of the two HOG filters for all pairs of filter areas within their respective DPM boxes. The first iteration of the α-expansion chooses between yes/no object-1 and the second iteration chooses between the previous label and object-2. In the rest of the area a simple binary yes/no choice is made for the corresponding object. The procedure is summed up in Algorithm 3. Note that $p_1(i)$ means the linear index position in $\mathbf{p}_1$ of the position that is represented by i in $\mathbf{p}_2$.

Algorithm 3 Pair-Detect: Detect object by searching over DPM hypotheses

```
Require: HOG-image $\mathbf{x}$, Model weights $\mathbf{w}_1$ and $\mathbf{w}_2$, feature function $f(\mathbf{x}, \mathbf{p}, \mathbf{v})$ implementing Chapter 1
  $(\mathbf{p}_1^*, \mathbf{v}_1^*, D_1^*) \leftarrow$ DPM-seg-detect$(\mathbf{x}, \mathbf{w}_1, f)$
  $(\mathbf{p}_2^*, \mathbf{v}_2^*, D_2^*) \leftarrow$ DPM-seg-detect$(\mathbf{x}, \mathbf{w}_2, f)$
  if $D_1^* \bigcap D_2^* = \emptyset$ then
    return $\mathbf{y}^* = (\mathbf{p_1}^*, \mathbf{v_1}^*, \mathbf{p_2}^*, \mathbf{v_2}^*)$
  else
    $Y_1 \leftarrow \{(i,j) : (i,j), (i + w_1, j + h_1) \in D_1^*\}$
    $Y_2 \leftarrow \{(i,j) : (i,j), (i + w_2, j + h_2) \in D_2^*\}$
    $\mathbf{p}_1^* \leftarrow (0,0), \mathbf{p}_2^* \leftarrow (0,0)$
    $\mathbf{v}_1^* \leftarrow \mathbf{0}, \mathbf{v}_2^* \leftarrow \mathbf{0}$
    $maxscore \leftarrow 0$
    for all $(\mathbf{p}_1, \mathbf{p}_2) \in Y_1 \times Y_2$ do
      $$\mathbf{v}_1 = \operatorname*{argmax}_{\mathbf{v}} \Bigg( \sum_{i=1}^{w_1 * h_1} \big(\mathbf{w}_{1,h,i}^T \phi_i(\mathbf{x}, \mathbf{p}_1)\big) \cdot v_{1,i} + \sum_{i=1}^{w_1 * h_1} \Big(\mathbf{w}_{1,\hat{h},i}^T \phi_i(\mathbf{x}, \mathbf{p}_1)\Big) \cdot (1 - v_{1,i}) + \sum_{i=1}^{w_1 * h_1} w_{1,p,i} \cdot (1 - v_{1,i}) + \frac{1}{|C|} \sum_{c \in C} \frac{1}{\binom{|c|}{2}} \sum_{(i,j) \in c} w_{1,e} \cdot \mathbf{1}(v_{1,i} = v_{1,j}) \Bigg)$$
      $score \leftarrow \mathbf{w_1}^T f(\mathbf{x}, \mathbf{p_1}, \mathbf{v_1})$
      $$\mathbf{v}_2 = \operatorname*{argmax}_{\mathbf{v}} \Bigg( \sum_{i=1}^{w_2 * h_2} \big(\mathbf{w}_{2,h,i}^T \phi_i(\mathbf{x}, \mathbf{p}_2)\big) \cdot v_{2,i} + \sum_{i=1}^{w_2 * h_2} \Big[ \Big(\mathbf{w}_{1,h,p_1(i)}^T \phi_i(\mathbf{x}, \mathbf{p}_2)\Big) \cdot v_{1,p_1(i)} + \Big(\mathbf{w}_{1,\hat{h},p_1(i)}^T \phi_i(\mathbf{x}, \mathbf{p}_2)\Big) \cdot (1 - v_{1,p_1(i)}) \Big] \cdot (1 - v_{2,i}) \cdot \mathbf{1}(1 \leq p_1(i) \leq w_1 * h_1) + \sum_{i=1}^{w_2 * h_2} \Big(\mathbf{w}_{2,\hat{h},i}^T \phi_i(\mathbf{x}, \mathbf{p}_2)\Big) \cdot (1 - v_{2,i}) \cdot \mathbf{1}(1 < p_1(i) > w_1 * h_1) + \sum_{i=1}^{w_2 * h_2} w_{1,p,i} \cdot (1 - v_{2,i}) + \frac{1}{|C|} \sum_{c \in C} \frac{1}{\binom{|c|}{2}} \sum_{(i,j) \in c} w_{2,e} \cdot \mathbf{1}(v_{2,i} = v_{2,j}) \Bigg)$$
      $score \leftarrow score + \mathbf{w_2}^T f(\mathbf{x}, \mathbf{p_2}, \mathbf{v_2})$
      if $score > maxscore$ then
        $\mathbf{p_1}^* \leftarrow \mathbf{p_1}, \mathbf{p_2}^* \leftarrow \mathbf{p_2}$
        $\mathbf{v_1}^* \leftarrow \mathbf{v_1}, \mathbf{v_2}^* \leftarrow \mathbf{v_2}$
      end if
    end for
  end if
  return $\mathbf{y}^* = (\mathbf{p_1}^*, \mathbf{v_1}^*, \mathbf{p_2}^*, \mathbf{v_2}^*)$
```

Chapter 4

Results

4.1 Datasets

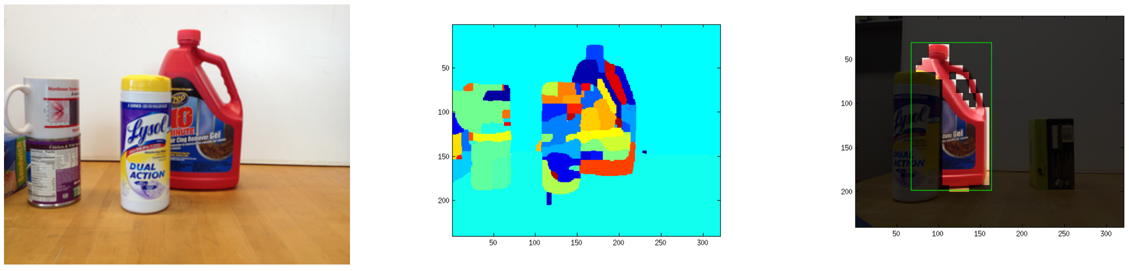
Evaluation has been performed on the following datasets:

Dataset	**Description**
Lysol	43 color images, 320x240, main object: Lysol bottle
Cleaner	13 color images, 320x240, main object: Cleaner bottles
Cleaner-2	62 color images, 320x240, main object: Cleaner bottle
Ziplock	40 color images, 320x240, main object: Ziplock box
Double	49 color images, 320x240, main objects: Lysol bottle and Cleaner bottle

Table 4.1: Datasets used for evaluation

The main object(s) in all datasets were occluded with various household objects.

The datasets were labelled with location of the top-left corner of the bounding box and pixel-level visibility labels inside the bounding box. In case of Double dataset, images containing two objects together were labelled with both. Figure 4.1 shows a sample ground truth labelling, taken from the Cleaner-2 dataset and the gPb-UCM segmentation that is used to define the pairwise terms.



(a) Image (b) Segmentation (c) Labelling

Figure 4.1: Ground truth labelling

4.2 Performance Evaluation

We perform precision-recall and average precision evaluation on all the datasets (Pair-Detect on Double dataset and DPM-Seg-Detect on other datasets). For comparison, we also run the same evaluations using a DPM model trained using about 5 images of the fully visible object. Table 4.2 shows the AP performance of both algorithms.

Dataset	**DPM-Seg-Detect AP**	**DPM AP**
Lysol	0.901	**1**
Cleaner	**1**	0.706
Cleaner-2	**0.908**	0.785
Ziplock	**1**	**1**

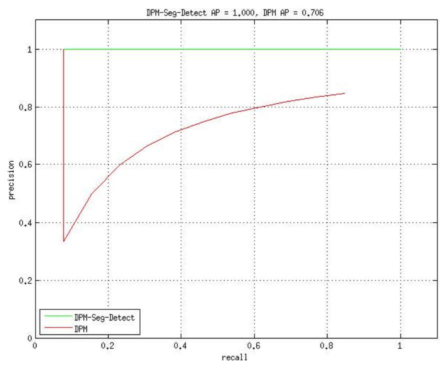
(a) DPM-Seg-Detect vs. DPM

Dataset	**Pair-Detect AP**	**DPM AP**
Double (object: Cleaner)	**0.772**	0.524
Double (object: Lysol)	**1**	**1**

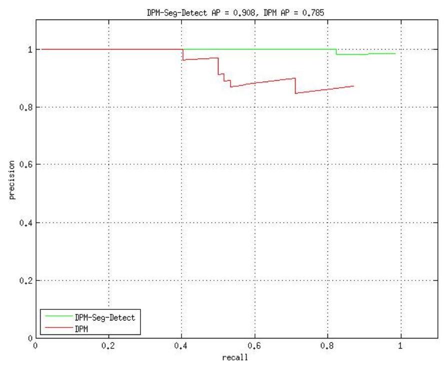
(b) Pair-Detect vs. DPM

Table 4.2: Average Precision performance

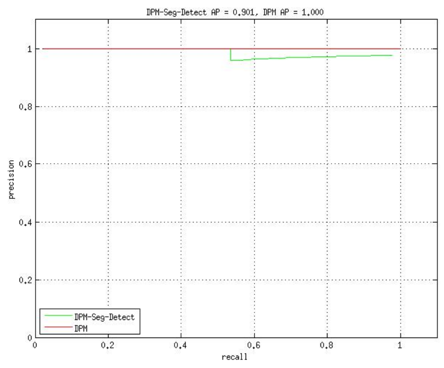
Figure 4.2 shows the various Precision-Recall curves.



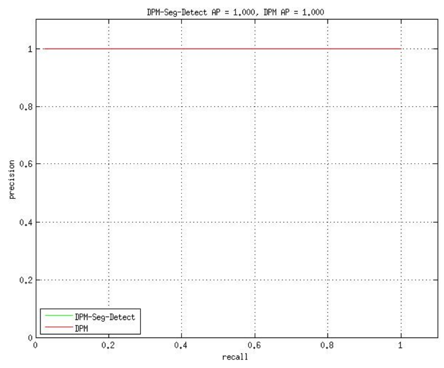
(a) Cleaner dataset P-R



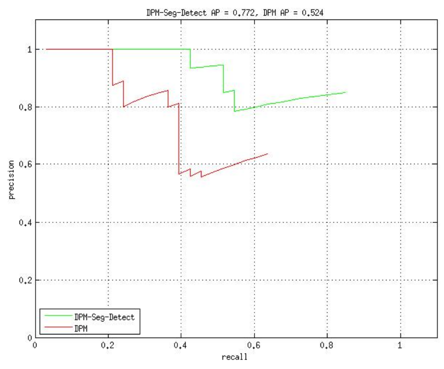
(b) Cleaner-2 dataset P-R



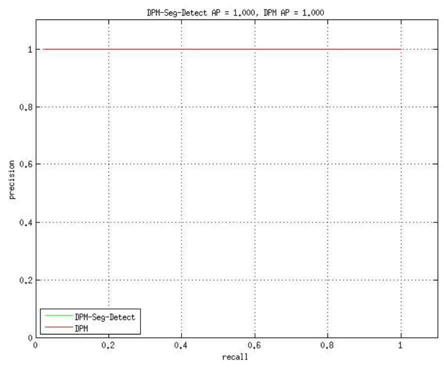
(c) Lysol dataset P-R



(d) Ziplock dataset P-R



(e) Double (object: Cleaner) dataset P-R

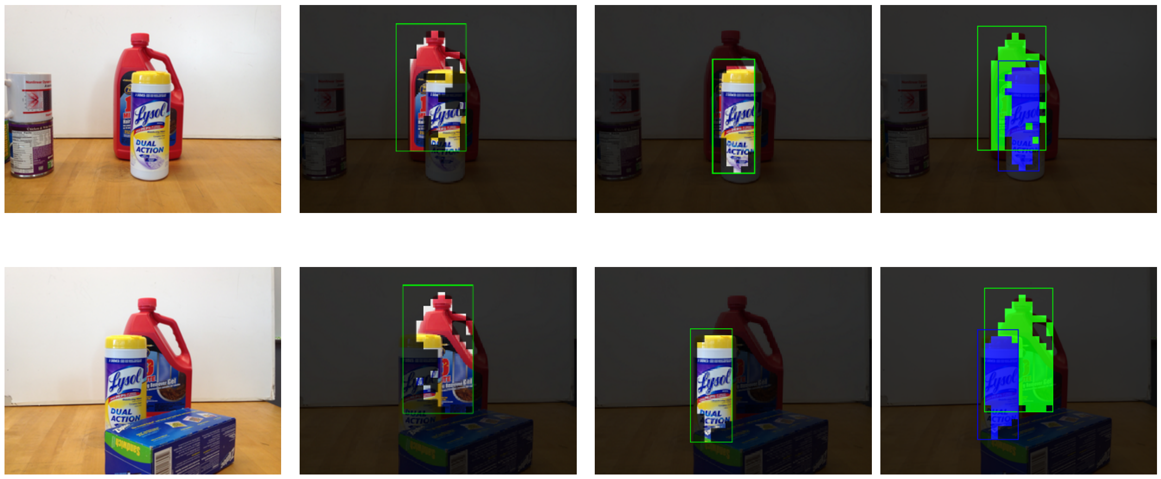


(f) Double (object: Lysol) dataset P-R

Figure 4.2: Precision-Recall curves

4.3 Pairs of Objects

In this section we show two examples of the effectiveness of using the Pair-Detect Algorithm 3 involving α-expansion in predicting a consistent visibility labelling pattern for both objects. The images are taken from the Double dataset. We also show the detections and visibility labelling predicted by the simple DPM-Seg-Detect Algorithm 2 to compare against in Figure 4.3.



(a) Image (b) DPM-Seg-Detect(1) (c) DPM-Seg-Detect(2) (d) Pair-Detect

Figure 4.3: Detecting Pairs of Objects

Chapter 5

Future Work

The object detection system described in this thesis can be made more generalized by making it scale-invariant and able to detect objects at different discretized poses, as mentioned in Section 2.4.

One drawback of the current approach is the use of rigid filters, which are not very good at handling small changes in perspective and scale. Recent research, as described in the introduction section, has shown that a part-based approach is much more robust in this aspect. So an important extension for this work would be to make it part-based. This can be done by modifying the part filters of the DPM model to be Seg-Detect (Algorithm 1) implementations.

To introduce more robustness, pairs of part filters that overlap with each other can also be Pair-Detect (Algorithm 3) implementations. This would not only force clean boundaries between the parts, but more importantly allow a bit more of deformation in lieu of better agreement between parts regarding visibility of their respective pixels.

Since DPM has a Latent SVM training approach, careful thought must be given to the training stage of such a hybrid algorithm.

Bibliography

[1] Pablo Arbelaez et al. “Contour Detection and Hierarchical Image Segmentation”. In: (2010).

[2] S. Bagon and M. Galun. “Large Scale Correlation Clustering Optimization”. In: *arXiv* (2011).

[3] Herbert Bay et al. “Speeded-Up Robust Features (SURF)”. In: *Comput. Vis. Image Underst.* (2008).

[4] Y. Boykov, O. Veksler, and R. Zabih. “Fast approximate energy minimization via graph cuts”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2001).

[5] V. Lempitsky C. Rother V. Kolmogorov and M. Szummer. “Optimizing binary MRFs via extended roof duality”. In: *Computer Vision and Pattern Recognition* (2007).

[6] Michael Calonder et al. “BRIEF: Binary Robust Independent Elementary Features”. In: *Proceedings of the 11th European Conference on Computer Vision: Part IV* (2010).

[7] Navneet Dalal and Bill Triggs. “Histograms of Oriented Gradients for Human Detection”. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2005).

[8] P. F. Felzenszwalb et al. “Object Detection with Discriminatively Trained Part Based Models”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2010).

[9] P.F. Felzenszwalb, R.B. Girshick, and D. McAllester. “Cascade object detection with deformable part models”. In: *2010 IEEE Conference on Computer Vision and Pattern Recognition* (2010).

[10] Tianshi Gao, B. Packer, and D. Koller. “A segmentation-aware object detection model with occlusion handling”. In: *IEEE Conference on Computer Vision and Pattern Recognition* (2011).

[11] Ross B Girshick, Pedro F Felzenszwalb, and David McAllester. “Object Detection with Grammar Models”. In: (2011).

[12] Stephen Gould. “Max-margin Learning for Lower Linear Envelope Potentials in Binary Markov Random Fields”. In: 2011.

[13] Stephen Gould, Tianshi Gao, and Daphne Koller. “Region-based Segmentation and Object Detection”. In: *Advances in Neural Information Processing Systems* (2009).

[14] Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. “Cutting-plane Training of Structural SVMs”. In: *Machine Learning* (2009).

[15] Pushmeet Kohli, Philip HS Torr, et al. “Robust higher order potentials for enforcing label consistency”. In: *International Journal of Computer Vision* (2009).

[16] Vladimir Kolmogorov and Carsten Rother. “Minimizing nonsubmodular functions with graph cuts-a review”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* (2007).

[17] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *Int. J. Comput. Vision* (2004).

[18] Kyoungup Park and Stephen Gould. “On Learning Higher-Order Consistency Potentials for Multi-class Pixel Labeling”. In: *European Conference on Computer Vision* (2012).

[19] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. “”GrabCut”: Interactive Foreground Extraction Using Iterated Graph Cuts”. In: *ACM Trans. Graph.* (2004).

[20] E. Rublee et al. “ORB: An efficient alternative to SIFT or SURF”. In: *IEEE International Conference on Computer Vision (ICCV)* (2011).

[21] A. Vedaldi. *A MATLAB wrapper of* SVM$^{\text{struct}}$. `http://www.vlfeat.org/~vedaldi/code/svm-struct-matlab.html`. 2011.

[22] Andrea Vedaldi and Andrew Zisserman. “Structured output regression for detection with partial occulsion”. In: *Advances in Neural Information Processing Systems* (2009).