Coursework on detection of objects in scenes using machine learning

Yaroslav Ganin
MSU, Department of Mathematics and Mechanics
dr.death.tm@gmail.com

Abstract

In this paper, an algorithm for detection and classification of objects on the image is presented. It is based on supervised machine learning technique (SVM).

An input image is separated into segments that are considered to be objects. So the problem reduces to the classification of those segments. Each of them is represented as a feature vector which components has been chosen in a way to be well discriminative.

Vectors that correspond to the test data are fed to the input of the SVM classifier that was previously trained using pre-labeled vectors.

1. Introduction

The main goal of my work is studying of known methods in the field of computer vision to develop a solution for a particular problem. The problem is as follows. A set of photographs is given. For each photograph we have hand-marked regions (polygons, to be precise) which correspond to particular object in the scene. We also have an unmarked image. We should detect objects on it and make a classification of them. Let's consider an example. Suppose we have an input photo of a street which contains people on the sidewalk, cars on the road, trees, several buildings. We are trying to develop a special mechanism that will produce an image with the following structure: an input photo is divided into connected pieces; every piece should contain only one object (*e.g.* a car, a pedestrian, a tree *etc.*); moreover pieces that contain the same type of object (*e.g.* all pieces with cars) are floodfilled with the same color. In other words the algorithm should build up the semantic map of the input photograph.

Modern methods used for the solution of such type of problems are described in [10], [11], [15], [19], [20].

My approach is to use a supervised *machine learning*. According to encyclopedia the definition of the "machine learning" notion is as follows. "Machine learning is a scientific discipline that is concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data, such as from sensor data or databases. A major focus of machine learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data; the difficulty lies in the fact that the set of all possible behaviors given all possible inputs is too complex to describe generally in programming languages, so that in effect programs must automatically describe programs". [21]

Supervised learning is a machine learning technique for deducing a function from training data. The task of the supervised learner is to predict the value of the function for any valid input object after having seen a number of training examples. To achieve this, the learner has to generalize from the presented data to unseen situations in a "reasonable" way.

A particular method that was used in my work is SVM (Support Vector Machine). This method will be discussed further in the text.

The paper is organized as follows. Section 2 gives information on known algorithms that are used in my coursework. In section 3 an algorithm of solution of the problem is described step by step. Finally, section 4 concludes the paper.

2. Components of the proposed algorithm

2.1. Color Structure Code (CSC)

In computer vision *segmentation* refers to the process of partitioning a digital image into multiple segments (sets of pixels, also known as superpixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics.

The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image (see edge detection). Each of the pixels in a region are similar with respect to some charac-

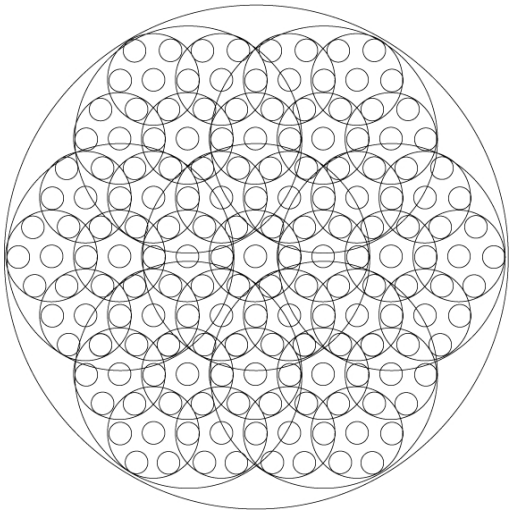


Figure 1. The hexagonal topology used in the Color Structure Code segmentation. It makes the algorithm very accurate and well parallelizable. See section 2.1 for details. Image is taken from [8].

teristic or computed property, such as color, intensity, or texture. Adjacent regions are significantly different with respect to the same characteristic(s).

The Color Structure Code (CSC) color segmentation which is used in my course paper was first introduced in [16]. Next few paragraphs will give a short information on the method.

Hexagonal structure The CSC segmentation is based on a special data structure so-called hexagonal island structure (see fig. 1). Let the smallest circles be pixels and seven pixels in a specified structure (fig. 2, left) compose an island. These small islands are so-called "islands on level 0". Each seven islands compose another island on level 1 in the same structure and seven of these new islands construct another one on level 2. Continue this procedure until one island covers the whole image. Notice that an island on level N contains seven islands at most from level $N-1$ or when $N=0$ it contains seven pixels instead.

Notice that two neighboring islands on level N are overlapped and they share one and only one sub-island; on level 0 this means they share a single pixel. To apply this structure to an orthogonal bitmap can bring up some difficulties. One simple method for this transformation is to shift every other row theoretically by half of a pixel either left or right. Fig. 2 (center) illustrate an island on level 0 and level 1 over an orthogonal bitmap. By the help of fig. 2 (rightmost), an island is shown on level 2 but let us point out that the overlapping part of its sub-islands are not precise. It is confined to one shared sub-island and not other additional pixels.

This hierarchical structure would seem to be complicated but it worth while to build up because this will make the algorithm so fast and accurate later.

Creating code elements On the described hexagonal hierarchical structure each island has one or more so-called code elements. On level 0 a code element simply means linked pixels which are similar in color. Each island on level 0 has up to seven code elements depending on the image and the applied color similarity. It is also important that the whole island is "covered" by code elements. On an island on level N where $N > 0$ a code element is defined as liked code elements of its sub-islands which are connected. A code element can be implemented as an array of pointers to other code elements and on level 0 as an array of image coordinates. The code elements are stored on the appropriate islands together with other attributes like area (number of pixels) and mean color.

In implementation used in the coursework this creation procedure is a simple request to create all code elements of the "top level island" and it does everything else due to the following recursive algorithm. First of all an island on level $N > 0$ calls the procedure to create code elements on all of its sub-islands if necessary (recursion). After all the code elements of its sub-island have been computed we can create some new ones on the island itself and link the appropriate sub-code elements to their lists. Two sub-code elements must be attached to the same code element if they are connected and must not if they are not. Whether two elements are connected or not, can be detected easily by comparing their list of sub-code elements. If they have an identical entry they are connected. This part of the algorithm is also called the linking phase. Trivially, the code element creation is different on level 0 from that on any other level. Here we create code elements for the pixels itself on each island. Two pixels are connected to the same code element if they are similar in color as mentioned earlier in this section. Furthermore during the linking process it is unnecessary to create code elements with only one sub-code element because if the algorithm did not find any connected element on the same level means that this would be a so-called "root code element" and it can be collected globally in an array.

This "hierarchical region growing" part of the algorithm would not have been as powerful and accurate as it is without the following splitting phase [16] which extend the segmentation with a global view. That means, before two connected sub-code elements are to be linked we check the color similarity between their mean colors. If they are not similar they will not be liked; moreover they must be separated by this method because they share a region on a previous level. The subregions of this common region will be associate to the one which is closer in color. Therefore some additional splitting could be required on the lower level and in these cases we should act in the same way which means an elegant recursion in the implementation.

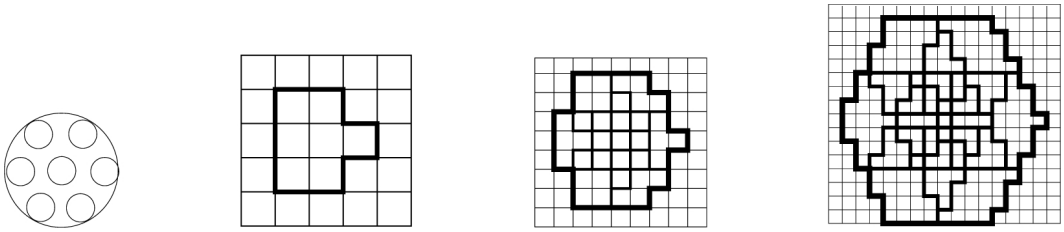


Figure 2. An illustration of hexagonal topology in the Color Structure Code segmentation and its adaptation for the bitmap. From left to right: islands on level 0 on hexagonal structure; orthogonal structure of the islands on level 0 (*i.e*. on the bitmap); island on level 1, island on level 2 over an orthogonal bitmap. See section 2.1 for details. Image is taken from [8].

Final steps After the previous phases the results (the regions) are represented hierarchically. Each region has a root element which is stored either as an "immediate" root or as a simple code element on the "top level island". Usually this hierarchical information cannot be used directly. The final phase is responsible for converting them into an appropriate format. This can be a label image or a set of chain codes. The other important task of this phase is to clear up some problems and difficulties of the original segmentation algorithm.

2.2. Graph-Based Image Segmentation

The second segmentation algorithm that was used in my work is described in [9].

A graph-based approach to segmentation is taken. Let $G = (V, E)$ be an undirected graph with vertices $v_i \in V$, the set of elements to be segmented, and edges $(v_i, v_j) \in E$ corresponding to pairs of neighboring vertices. Each edge $(v_i, v_j) \in E$ has a corresponding weight $w((v_i, v_j))$, which is a non-negative measure of the dissimilarity between neighboring elements v_i and v_j. In the case of image segmentation, the elements in V are pixels and the weight of an edge is some measure of the dissimilarity between the two pixels connected by that edge (*e.g*., the difference in intensity, color, motion, location or some other local attribute). In the graph-based approach, a segmentation S is a partition of V into components such that each component (or region) $C \in S$ corresponds to a connected component in a graph $G' = (V, E')$, where $E' \subseteq E$. In other words, any segmentation is induced by a subset of the edges in E. There are different ways to measure the quality of a segmentation but in general one wants the elements in a component to be similar, and elements in different components to be dissimilar. This means that edges between two vertices in the same component should have relatively low weights, and edges between vertices in different components should have higher weights.

Authors of the algorithm define a predicate, D, for evaluating whether or not there is evidence for a boundary between two components in a segmentation (two regions of an image). This predicate is based on measuring the dissimilarity between elements along the boundary of the two components relative to a measure of the dissimilarity among neighboring elements within each of the two components. The resulting predicate compares the inter-component differences to the within component differences and is thereby adaptive with respect to the local characteristics of the data.

The internal difference of a component $C \subseteq V$ is defined as the largest weight in the minimum spanning tree of the component, $MST(C, E)$. That is,

$$Int(C) = \max_{e \in MST(C,E)} w(e). \tag{1}$$

The difference between two components $C_1, C_2 \subseteq V$ is defined as the minimum weight edge connecting the two components. That is,

$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w(v_i, v_j). \tag{2}$$

The region comparison predicate evaluates if there is evidence for a boundary between a pair of components by checking if the difference between the components, $Dif(C_1, C_2)$, is large relative to the internal difference within at least one of the components, $Int(C_1)$ and $Int(C_2)$. A threshold function is used to control the degree to which the difference between components must be larger than minimum internal difference. The pairwise comparison predicate is defined as,

$$D(C_1, C_2) = \begin{cases} \text{true} & \text{if } Dif(C_1, C_2) > MInt(C_1, C_2); \\ \text{false} & \text{otherwise} \end{cases} \tag{3}$$

where the minimum internal difference, $MInt$, is defined as,

$$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2)). \tag{4}$$

The threshold function τ controls the degree to which the difference between two components must be greater than their internal differences in order for there to be evidence of a boundary between them (D to be true). For

small components, $Int(C)$ is not a good estimate of the local characteristics of the data. In the extreme case, when $|C| = 1$, $Int(C) = 0$. Therefore, a threshold function based on the size of the component is used,

$$\tau(C) = \frac{k}{|C|} \tag{5}$$

where $|C|$ denotes the size of C, and k is some constant parameter. That is, for small components a stronger evidence for a boundary is required. In practice k sets a scale of observation, in that a larger k causes a preference for larger components.

The segmentation algorithm is as follows. The input is a graph $G = (V, E)$, with n vertices and m edges. The output is a segmentation of V into components $S = (C_1, \ldots, C_r)$.

1. Sort E into $\pi = (o_1, \ldots, o_m)$, by non-decreasing edge weight.
2. Start with a segmentation S^0, where each vertex v_i is in its own component.
3. Repeat step 4 for $q = 1, \ldots, m$.
4. Construct S^q given S^{q-1} as follows. Let v_i and v_j denote the vertices connected by the q-th edge in the ordering, *i.e.*, $o_q = (v_i, v_j)$. If v_i and v_j are in disjoint components of S^{q-1} and $w(o_q)$ is small compared to the internal difference of both those components, then merge the two components otherwise do nothing. More formally, let C_i^{q-1} be the component of S^{q-1} containing v_i and C_j^{q-1} the component containing v_j . If $C_i^{q-1} \neq C_j^{q-1}$ and $w(o_q) \leq MInt(C_i^{q-1}, C_j^{q-1})$ then S^q is obtained from S^{q-1} by merging C_i^{q-1} and C_j^{q-1}. Otherwise $S^q = S^{q-1}$.
5. Return $S = S^m$.

Segmentation S produced by the algorithm obeys the global properties of being neither too fine nor too coarse (according to the terminology used in [9]) when using the region comparison predicate D.

2.3. 2D Pair-wise Geometrical Histogram (PGH)

The pairwise geometric histogram (PGH) is a powerful shape descriptor that is applied to polygonal shapes [1]. It can be applied also to an irregular shape if the shape is first approximated with a polygon.

Consider a polygon defined by its edgepoints $(\tilde{x}(t), \tilde{y}(t)) \in \mathbb{R}^2$. Now successive edgepoints define the line segments the polygon consists of. The PGH is calculated using the following strategy. Let each line segment be a reference line on its turn. Then the relative angle $\theta \in [0, \pi)$ and the perpendicular minimum and maximum distances (d_{min} and d_{max}) are calculated between the reference line and all the other lines, as shown in fig. 3 (a). The histogram values are increased by one on the indexes corresponding to the angle θ and the line segment from the d_{min} to d_{max} (fig. 3 (b)).

2.4. Speeded Up Robust Features (SURF)

Speeded Up Robust Features (SURF) is a robust image detector and descriptor, first presented by Bay *et al.* in 2006 [2], that can be used in computer vision tasks like object recognition or 3D reconstruction. It is partly inspired by the SIFT descriptor. The standard version of SURF is several times faster than SIFT and claimed by its authors to be more robust against different image transformations than SIFT.

Fast-Hessian Detector The detector is based on the Hessian matrix because of its good performance in computation time and accuracy. However, rather than using a different measure for selecting the location and the scale (as was done in the Hessian-Laplace detector), SURF relies on the determinant of the Hessian for both. Given a point $x = (x, y)$ in an image I, the Hessian matrix $H(x, \sigma)$ in x at scale σ is defined as follows

$$H(x, \sigma) = \begin{pmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{pmatrix}, \tag{6}$$

where $L_{xx}(x, \sigma)$ is the convolution of the Gaussian second order derivative $\frac{\partial^2}{\partial x^2} g(\sigma)$ with the image I in point x, and similarly for $L_{xy}(x, \sigma)$ and $L_{yy}(x, \sigma)$.

A simpler alternative to Gaussians is used for SURF. As Gaussian filters are non-ideal in any case, and given Lowes success with LoG approximations, the authors of SURF push the approximation even further with box filters. These approximate second order Gaussian derivatives, and can be evaluated very fast using integral images, independently of size. It is shown that the resulting performance is comparable to the one using the discretized and cropped Gaussians.

In order to localize interest points in the image and over scales, a non-maximum suppression in a $3 \times 3 \times 3$ neighbourhood is applied. The maxima of the determinant of the Hessian matrix are then interpolated in scale and image space with the method proposed by Brown *et al.* [5]. Fig. 4 (left) shows an example of the detected interest points using our Fast-Hessian detector.

SURF Descriptor The SURF descriptor of the interest point is calculated in two stages.

The first one is orientation assignment. In order to be invariant to rotation, a reproducible orientation for the interest points is identified. For that purpose, the Haar-wavelet responses in x and y directions are calculated, and this in a circular neighbourhood of radius $6s$ around the interest point,

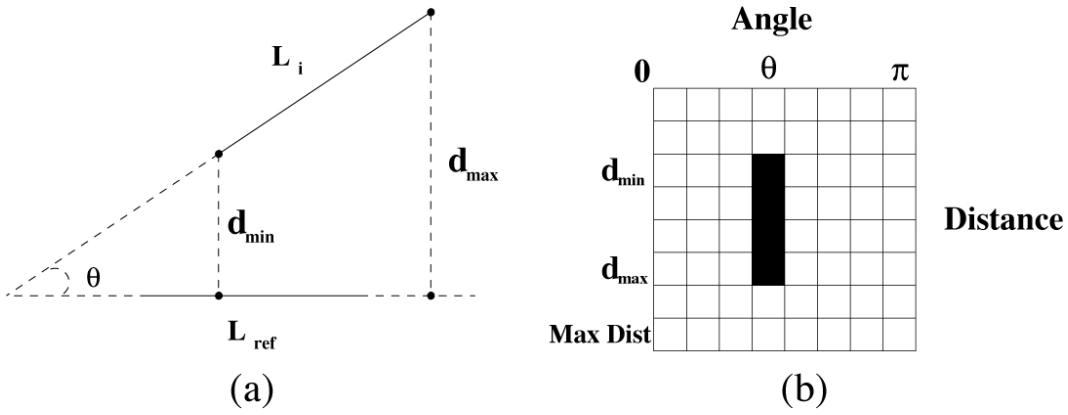


Figure 3. (a) Relative angle and perpendicular distances between two lines that are calculated for the PGH. These values gathered for all contour's segments make up a good shape descriptor. (b) The pairwise geometric histogram itself and bins that are incremented by the corresponding values for two lines in (a). See section 2.2 for details. Image is taken from [13].

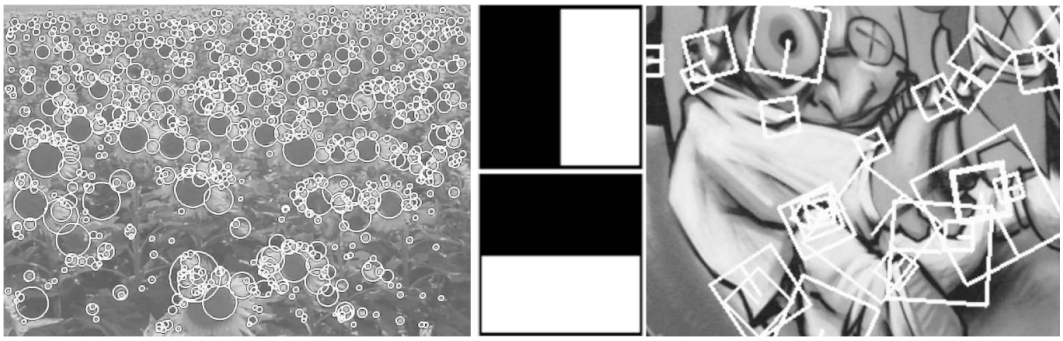


Figure 4. Left: Detected interest points for a Sunflower field. This kind of scenes shows clearly the nature of the features from Hessian-based detectors. Middle: Haar wavelet types used for SURF. Right: Detail of the Graffiti scene showing the size of the descriptor window at different scales. See section 2.3 for details. Image is taken from [2].

with s the scale at which the interest point was detected. Also the sampling step is scale dependent and chosen to be s. In keeping with the rest, also the wavelet responses are computed at that current scale s.

Once the wavelet responses are calculated and weighted with a Gaussian ($\sigma = 2.5s$) centered at the interest point, the responses are represented as vectors in a space with the horizontal response strength along the abscissa and the vertical response strength along the ordinate. The dominant orientation is estimated by calculating the sum of all responses within a sliding orientation window covering an angle of $\frac{\pi}{3}$. The horizontal and vertical responses within the window are summed. The two summed responses then yield a new vector. The longest such vector lends its orientation to the interest point. The size of the sliding window is a parameter, which has been chosen experimentally.

The second stage is the extraction of the descriptor. A square region centered around the interest point, and oriented along the selected orientation is constructed. The size of this window is $20s$. The region is split up regularly into smaller square sub-regions. This keeps important spatial information in. For each sub-region, a few simple features based on Haar wavelet response are computed and a feature vector is calculated. All these vectors form the resulting descriptor of the interest point which has 64 components. The wavelet responses are invariant to a bias in illumination (offset). Invariance to contrast (a scale factor) is achieved by turning the descriptor into a unit vector.

2.5. Bag-of-words

The bag-of-words in natural language processing is a popular method for representing documents, which ignores the word orders. This model allows a dictionary-based modeling, and each document looks like a "bag", which contains some words from the dictionary. Computer vision researchers uses a similar idea for image representation.

To represent an image using bag-of-words model, an image can be treated as a document. Similarly, "words" in images need to be defined too. However, "word" in images is not the off-the-shelf thing like the word in text documents.

To achieve this, it usually includes following three steps: *feature detection, feature description and codebook (vocabulary) generation*. A definition of the bag-of words model can be the "histogram representation based on independent features".

Given an input image, feature detection is an extraction of several local patches (or regions) which are considered as candidates for basic elements, *i.e*. "words". That can be done in many different ways (regular grid, random sampling, segmentation methods to name a few). The SURF interest point detector is used in my coursework.

After feature detection, each image is abstracted by several local patches. Feature representation methods deal with how to represent the patches as numerical vectors. These methods are called feature descriptors. A good descriptor should have the ability to handle intensity, rotation, scale and affine variations to some extent (*e.g*. SURF descriptor).

The final step for the bag-of-words model is to convert vector represented patches to "codewords" (analogy to words in text documents), which also produces a "vocabulary" (analogy to a word dictionary). A codeword can be considered as a representative of several similar patches. One simple method is performing k-means clustering over all the vectors. Codewords are then defined as the centers of the learned clusters. The number of the clusters is the vocabulary size (analogy to the size of the word dictionary).

Thus, each patch in an image is mapped to a certain codeword through the clustering process and the image can be represented by the histogram of the codewords.

2.6. Support Vector Machine (SVM)

Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. A binary (two-class) classification problem that is solved by the most simple SVM mechanism can be described as follows: given a set of labeled points $\mathcal{D} = \{(\mathbf{x}_i, c_i)|\mathbf{x}_i \in \mathbb{R}^p, c_i \in \{-1, 1\}\}_{i=1}^n$, where $\mathbf{x}_i$ are vectors of features and c_i are class labels, construct a rule that correctly assigns a new point x to one of the classes.

The vectors $\mathbf{x}_i$ in this formulation correspond to objects, and the dimensions of the space are the features or characteristics of these objects (*e.g*. superpixel in the image has an average color expressed as vector (r, g, b), an area, a perimeter *etc*.).

Formalization We are given some training data $\mathcal{D}$. Each $\mathbf{x}_i$ is a p-dimensional real-number vector. We want to find the maximum-margin hyperplane that divides the points having $c_i = 1$ from those having $c_i = -1$. Any hyperplane can be written as the set of points $\mathbf{x}$ satisfying:

$$\mathbf{w} \cdot \mathbf{x} - b = 0\,. \tag{7}$$

The vector $\mathbf{w}$ is perpendicular to the hyperplane. The parameter $\frac{b}{\|\mathbf{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector $\mathbf{w}$.

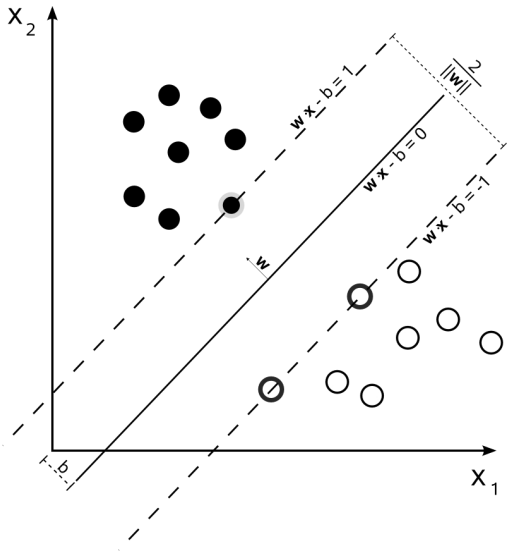


Figure 5. Maximum-margin hyperplane ($\mathbf{w} \cdot \mathbf{x} - b = 0$) and margins ($\mathbf{w} \cdot \mathbf{x} - b = 1$ and $\mathbf{w} \cdot \mathbf{x} - b = -1$) for a SVM trained with samples from two classes (white and black points on the image). Samples on the margin are called the support vectors. For the SVM on the image every point that falls into the halfplane (relatively to the line $\mathbf{w} \cdot \mathbf{x} - b = 0$) with black points is calssified as black, otherwise as white. See section 2.5 for the details. Image is taken from [21].

The $\mathbf{w}$ and b are chosen to maximize the margin. In other words we want to maximize distance between the parallel hyperplanes that are as far apart as possible while still separating the data. These hyperplanes can be described by the equations:

$$\mathbf{w} \cdot \mathbf{x} - b = 1 \tag{8}$$

and

$$\mathbf{w} \cdot \mathbf{x} - b = -1. \tag{9}$$

The distance between these two hyperplanes is $\frac{2}{\|\mathbf{w}\|}$, so we want to minimize $\|\mathbf{w}\|$. To prevent data points falling into the margin, the following constraint is added: for each i either

$$\mathbf{w} \cdot \mathbf{x}_i - b \geq 1 \qquad \text{for } \mathbf{x}_i \text{ of the first class} \tag{10}$$

or

$$\mathbf{w} \cdot \mathbf{x}_i - b \leq -1 \qquad \text{for } \mathbf{x}_i \text{ of the second.} \tag{11}$$

This constraint can be rewritten as:

$$c_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \quad \text{for all } 1 \leq i \leq n. \tag{12}$$

The resulting optimization problem is:

Minimize (in $\mathbf{w}, b$)

$$\|\mathbf{w}\| \tag{13}$$

subject to (for any $i = 1, \ldots, n$)

$$c_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1\,. \tag{14}$$

Primal form The optimization problem presented above is difficult to solve because it involves a square root operation. It is possible to alter the equation by substituting $\|\mathbf{w}\|$ with $\frac{1}{2}\|\mathbf{w}\|^2$ without changing the solution. This is a quadratic programming (QP) optimization problem:

Minimize (in $\mathbf{w}, b$)

$$\frac{1}{2}\|\mathbf{w}\|^2 \tag{15}$$

subject to (for any $i = 1, \ldots, n$)

$$c_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1\,. \tag{16}$$

The previous constrained problem can be expressed as

$$\min_{\mathbf{w},b} \max_{\boldsymbol{\alpha}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i [c_i(\mathbf{w} \cdot \mathbf{x_i} - b) - 1] \right\} \tag{17}$$

that is we look for a saddle point. In doing so all the points which can be separated as $c_i(\mathbf{w} \cdot \mathbf{x_i} - b) - 1 > 0$ do not matter since we must set the corresponding α_i to zero.

The solution can be expressed by terms of linear combination of the training vectors as

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i c_i \mathbf{x_i}\,. \tag{18}$$

Only a few α_i will be greater than zero. The corresponding $\mathbf{x_i}$ are exactly the *"support vectors"*, which lie on the margin and satisfy $c_i(\mathbf{w} \cdot \mathbf{x_i} - b) = 1$. This leads to:

$$\mathbf{w} \cdot \mathbf{x_i} - b = 1/c_i = c_i \iff b = \mathbf{w} \cdot \mathbf{x_i} - c_i \tag{19}$$

which allows one to define the offset b. In practice, it is more robust to average over all N_{SV} support vectors:

$$b = \frac{1}{N_{SV}} \sum_{i=1}^{N_{SV}} (\mathbf{w} \cdot \mathbf{x_i} - c_i)\,. \tag{20}$$

Soft margin If there exists no hyperplane that can split the training data, the "Soft Margin" method (1995, Corinna Cortes and Vladimir Vapnik [6]) will choose a hyperplane that splits the examples as cleanly as possible, while still maximizing the distance to the nearest cleanly split examples. The method introduces slack variables, ξ_i, which measure the degree of misclassification of the datum x_i

$$c_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1 - \xi_i \quad 1 \leq i \leq n\,. \tag{21}$$

The objective function is then increased by a function which penalizes non-zero ξ_i, and the optimization becomes a trade off between a large margin, and a small error penalty. For linear penalty function, the optimization problem becomes:

$$\min_{\mathbf{w},\xi} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i \right\} \tag{22}$$

subject to (for any $i = 1, \ldots n$)

$$c_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1 - \xi_i, \quad \xi_i \geq 0\,. \tag{23}$$

This constraint in (21) along with the objective of minimizing $\|\mathbf{w}\|$ can be solved using lagrange multipliers as done in *Primal form* subsection.

Non-linear classification In 1992, Bernhard Boser, Isabelle Guyon and Vapnik [4] suggested a way to create non-linear classifiers by applying the kernel trick (originally proposed by Aizerman *et al.*) to maximum-margin hyperplanes: every dot product is replaced by a non-linear kernel function in the original algorithm. This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. The transformation may be non-linear and the transformed space high dimensional. This yields a classifier that may be non-linear in the original input space.

If the kernel used is a Gaussian radial basis function (RBF) ($k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma\|\mathbf{x_i} - \mathbf{x_j}\|^2)$, for $\gamma > 0$), the corresponding feature space is a Hilbert space of infinite dimension.

The kernel is related to the transform $\varphi(\mathbf{x_i})$ by the equation $k(\mathbf{x_i}, \mathbf{x_j}) = \varphi(\mathbf{x_i}) \cdot \varphi(\mathbf{x_j})$. The value $\mathbf{w}$ is also in the transformed space, with $\mathbf{w} = \sum_i \alpha_i c_i \varphi(\mathbf{x}_i)$. Dot products with $\mathbf{w}$ for classification can again be computed by the kernel trick, *i.e.* $\mathbf{w} \cdot \varphi(\mathbf{x}) = \sum_i \alpha_i c_i k(\mathbf{x}_i, \mathbf{x})$. However, there does not in general exist a value w such that $\mathbf{w} \cdot \varphi(\mathbf{x}) = k(\mathbf{w}', \mathbf{x})$.

Multiclass SVM Multiclass SVM aims to assign labels to instances by using support vector machines, where the labels are drawn from a finite set of several elements. Many methods exist for building a multiclass classifier system from binary classifiers (one-vs-all, one-vs-one *etc.*). Cramer and Singer [7] suggested a different approach. Suppose there are K classes. Then the optimization problem is as follows:

$$\min_{\mathbf{w}^j,\xi} \left\{ \frac{1}{2} \sum_{j=1}^{K} \|\mathbf{w}^j\|^2 + C \sum_{i=1}^{n} \xi_i \right\} \tag{24}$$

subject to (for any $i = 1, \dots n\,, j = 1, \dots K : \xi_i \geq 0\,, j \neq y_i$)

$$(\mathbf{w}^{y_i} \cdot \mathbf{x}_i - \mathbf{w}^j \cdot \mathbf{x}_i) \geq 1 - \xi_i\,. \tag{25}$$

The resulting decision function is:

$$f(x) = \arg \max_{j=1,\dots,K} \mathbf{w}^j \cdot \mathbf{x}\,. \tag{26}$$

3. Proposed algorithm

3.1. Training data and its conversion

The input data is a set of scenes which contain different types of objects (there are 8 types in total: cars, pedestrians, buildings, roads, sky, trees, sidewalks and bicycles). Every object is marked with a polygon and a corresponding label that are stored in a .xml file. The data should be converted to a form that is acceptable by the chosen classification method. My work uses multiclass SVM, so the training set is represented as multidimensional floating-point vectors. It is done with the following procedure.

There are two options for the initial extraction of a set of segments (connected subsets of pixels, superpixels). The first one is to extract them from each input image using the polygon information from the .xml file. The second one is to process all images through the segmentation algorithm. Each segment is then analyzed and a corresponding vector of features is extracted.

To store color information of the super-pixel its average color in the RGB color space is placed to the first 3 components of the feature vector.

The fourth component is the area of the segment (*i.e*. the number of pixels in the subset). The fifth component is a ratio of the area and a square of the perimeter (*i.e*. the number of border pixels) of the segment. The next 64 components are the linearized 2D PGH of the outer contour of the segment. These $1 + 1 + 64 = 66$ values describe the shape and the size of the super-pixel.

The last B components are the bag-of-words histogram for the grayscaled interior of the segment. They give some information about the contents of the super-pixel (in terms of texture, interest points *etc*.).

The method of extracting the vector of features is pretty straight-forward except the components which correspond to the bag-of-words histogram. Those are filled using the 2-pass procedure. The bag-of-words vocabulary is built during the first pass:

- Interest points with their SURF descriptors are calculated for each segment of each photograph in the training data set;
- All the obtained descriptors are stored in a general set;
- A general set is divided into clusters using the k-means algorithm (*i.e*. the centers of each cluster are found; B centers in total).

After that an interest point is said to be one the B words if the euclidean distance between its descriptor and the corresponding center of a cluster is the smallest.

During the second pass each interest point of a segment is labeled with the word and the bag-of-words histogram is built.

Thus, the training set of photographs is transformed into the set of vectors (one vector per segment).

The labels of all the vectors are obtained depending on the option chosen on the initial step. In the first case the labels are read from the corresponding .xml files. The second case is slightly more complicated. Let us consider a vector, a corresponding segment (with area α) and an image which contains this segment. An image is marked with the set of filled polygons (they are stored in the .xml). A segment is intersected with each polygon in turn and the area α_i of intersection is calculated. The label of a polygon with the greatest α_m is chosen for the segments feature vector (just in case if the ratio $\alpha_m/\alpha > \tau$, where τ is some threshold; the segment is removed from the training set otherwise).

3.2. SVM model training

One should scale the obtained general set of labeled vectors. Scaling before applying SVM is very important. Part 2 of Sarle's Neural Networks FAQ [18] explains the importance of this and most of considerations also apply to SVM. The main advantage of scaling is to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges. Another advantage is to avoid numerical difficulties during the calculation. Because kernel values usually depend on the inner products of feature vectors, *e.g*. the linear kernel and the polynomial kernel, large attribute values might cause numerical problems. So i-th component of features vector is scaled to the range $[0, 1]$ with the linear function f_i, where $i = 1 \dots 3 + 1 + 1 + 64 + B$.

Linear and RBF were chosen as the SVM kernels.

RBF kernel nonlinearly maps samples into a higher dimensional space so it, unlike the linear kernel, can handle the case when the relation between class labels and attributes is nonlinear. Furthermore, the linear kernel is a special case of RBF Keerthi and Lin [14] since the linear kernel with a penalty parameter C has the same performance as the RBF kernel with some parameters (C, γ). Moreover the RBF kernel has fewer numerical difficulties and reasonable number of parameters (*e.g*. comparing to the polynomial kernel).

However high dimensional input vectors are well separated with linear kernel. Moreover an SVM with such kernel works significantly faster.

(C, γ) selection is done by "grid-search" using cross-validation.

In v-fold cross-validation, we first divide the training set into v subsets of equal size. Sequentially one subset is tested

using the classifier trained on the remaining $v-1$ subsets. Thus, each instance of the whole training set is predicted once so the cross-validation accuracy is the percentage of data which are correctly classified. Such procedure can prevent the overfitting problem.

Various pairs of (C, γ) values are tried and the one with the best cross-validation accuracy is picked. As suggested by the developers of libsvm exponentially growing sequences of C and γ are used for the grid [12].

3.3. Test data labeling

The test data is a photograph that contains various objects of mentioned types. One must detect them and classify. Following procedure is proposed.

Firstly the image is converted to the set of features vectors pretty much in the same way how it was done for the training data. There is only two exceptions. The first one: segments are generated by the segmentation algorithm. The second one: and the bag-of-words histogram for each segment is built using the vocabulary that was formed on the stage of analyzing the training data (*i.e*. only second pass of mentioned 2-pass procedure is used).

The obtained vectors are scaled using f_i from previous section and fed to the input of multiclass SVM. The output labels are used for the classification of segments.

An optional improvement may be applied . The basic idea of it is borrowed from [17]. The test image can be segmented in several different ways. Each segmentation can be classified with the mentioned SVM. Let $\{S_i\}$ be the collection of these classifications. One can determine (manually or automatically) S_i which contains the most precise segments for the objects of the particular class. It can be done, for example, by using segmentation of the pre-classified images. So it is reasonable to consider S_i a better classification for that class. One can simply "sum up" all the $\{S_i\}$ with weights $\{C_i\}$ where C_i is greater than the other. This yields a classification which is more accurate for the particular class. The same applies to the other classes.

4. Results

The proposed algorithm was tested on The StreetScenes Challenge Framework [3]. A subset of the given images (about 100 images) was used to train the multiclass SVM. The training samples were generated using three methods. The first two are based on sets of segments obtained from segmentation algorithms (namely CSC and graph-based one), the third is based on a set that was read from file (handmarked by the authors of the framework). All the training samples were passed to the input of the SVM to check if there were mistakes in the implementation of the method and the accuracy was calculated. This step revealed that CSC algorithm (all range of parameters was tested) is inacceptable slow and produces too much segments (over 50k versus 5k by graph-based segmentation) which don't contain any whole objects. So it was decided to exclude CSC from tests.

Several examples of the resulting classifications of the test images can be viewed on fig. 6 - 10. Table 1 contains accuracies for several modifications of the proposed algorithm. The leftmost column describes the method of training segments' set extraction. The second one is number of bins in the bag-of-words histogram. The third column describes the type of kernel that was used for the SVM. The header of the table contains 7 different parameters (k) for the graph-based segmentation algorithm.

The accuracy of a particular modification is calculated as follows. A control set of images with corresponding ground-truth classification maps is chosen. Each image from the set is fed to the input of the algorithm. This results a "algorithm's version" of a classification map for each control photo. All these maps are compared to the ground-truth maps pixel by pixel. The accuracy is the number of matched pixels divided by the total number of pixels.

The shape-describing components are not used in the input vectors for the SVM. Early tests showed that the presence or absence of such descriptors does not significantly affect the proposed algorithm's performance. This is probably due to nature of superpixels produced by the segmentation (their shapes and boundaries).

It can be noticed that in general the proposed algorithm produces better (in terms of accuracy defined above) classification maps when the training set of segments is generated by the segmentation algorithm. Secondly, the accuracy reaches its maximum when graph-based segmentation algorithm parameter is about 800.0-1000.0 (see the highlighted value in the table). Too small values of the parameter cause the segmentation algorithm to produce non-discriminative superpixels (they contain no whole objects); too large values yield too large segments with several objects per one superpixel. The proposed algorithm woks better with bigger number of bag-of-words bins and linear kernel chosen for SVM. Be that as it may, the difference between linear and RBF kernels reduces when fewer bins are used.

One can note some misclassifications. Probable causes of them are discussed in the summary section.

5. Summary

An algorithm of detection and classification of objects on the image has been presented. It works quite acceptable on the test data. But it has a major drawback which can be fixed and thus effectiveness of the method will be significantly improved. That is the segmentation stage. Currently an image is separated into super-pixels that do not always correspond to the objects on it (some segments include more than one object, others represent non-discriminative parts of

			Graph segmentation parameter						
			500.0	800.0	1000.0	1300.0	1500.0	1700.0	2000.0
Training set of segments read from file	2048 bins	Linear	0.392638	0.453515	0.418384	0.408333	0.415309	0.475747	0.471167
		RBF	0.326291	0.331649	0.311874	0.382448	0.359752	0.419894	0.424669
	1024 bins	Linear	0.372686	0.396029	0.405263	0.394383	0.39246	0.393165	0.370746
		RBF	0.339466	0.346305	0.336582	0.322061	0.313437	0.306198	0.329108
	512 bins	Linear	0.41326	0.508071	0.475715	0.462077	0.395269	0.424606	0.44737
		RBF	0.319144	0.360606	0.34059	0.377863	0.377691	0.391715	0.367895
Training set of segments generated by segmentation	2048 bins	Linear	0.492704	**0.678004**	0.624416	0.645422	0.661948	0.674248	0.630718
		RBF	0.601622	0.60931	0.626852	0.518661	0.548584	0.576272	0.500707
	1024 bins	Linear	0.621804	0.677689	0.622303	0.52346	0.632322	0.564095	0.664618
		RBF	0.542109	0.636555	0.592278	0.632291	0.608653	0.555219	0.591704
	512 bins	Linear	0.591347	0.522702	0.64467	0.578042	0.593019	0.597141	0.651216
		RBF	0.584374	0.569661	0.673198	0.616584	0.579944	0.626553	0.517795

Table 1. Results for several baselines. The leftmost column describes the method of training segments' set extraction. The second one is number of bins in the bag-of-words histogram. The third column describes the type of kernel that was used for the SVM. The next 7 columns contain accuracies that correspond to 7 values of segmentation parameter. The maximal accuracy is highlighted. See section 4 for more details.

the objects). Selection and testing of various segmentation methods in order to make the proposed algorithm perform better in experiments is subject to further research.

References

[1] A. Ashbrook, N. Thacker, and P. Rockett. Pairwise geometric histograms: A scalable solution for the recognition of 2d rigid shape. In *SCIA95*, pages 271–278, 1995. 4

[2] H. Bay, T. Tuytelaars, and L. J. V. Gool. Surf: Speeded up robust features. In A. Leonardis, H. Bischof, and A. Pinz, editors, *ECCV (1)*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer, 2006. 4, 5

[3] S. Bileschi. *StreetScenes: Towards scene understanding in still images*. PhD thesis, Massachusetts Institute of Technology, 2006. 9

[4] B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *COLT*, pages 144–152, 1992. 7

[5] M. Brown and D. G. Lowe. Invariant features from interest point groups. In P. L. Rosin and A. D. Marshall, editors, *BMVC*. British Machine Vision Association, 2002. 4

[6] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. 7

[7] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001. 7

[8] G. Dorko, D. Paulus, U. Ahlrichs, and L. fur Mustererkennung. Color segmentation for scene exploration. In *Workshop Farbbildverarbeitung*, 2000. 2, 3

[9] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004. 3, 4

[10] S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller. Multi-class segmentation with relative location prior. *International Journal of Computer Vision*, 80(3):300–316, 2008. 1

[11] X. He, R. Zemel, and M. Carreira-Perpinán. Multiscale conditional random fields for image labeling. 2004. 1

[12] C. Hsu, C. Chang, C. Lin, et al. A practical guide to support vector classification, 2003. 9

[13] J. Iivarinen, M. Peura, J. Sarela, and A. Visa. Comparison of combined shape descriptors for irregular objects. In *Proceedings of the 8th British Machine Vision Conference*, volume 2, pages 430–39. Citeseer, 1997. 5

[14] S. Keerthi and C. Lin. Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural computation*, 15(7):1667–1689, 2003. 8

[15] P. Kohli, L. Ladickỳ, and P. Torr. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009. 1

[16] V. Rehrmann and M. Birkhoff. Echtzeitfähige objektverfolgung in farbbildern. *Fachberichte Informatik*, 15(1):13–16, 1995. 2

[17] B. Russell, W. Freeman, A. Efros, J. Sivic, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, 2006. 9

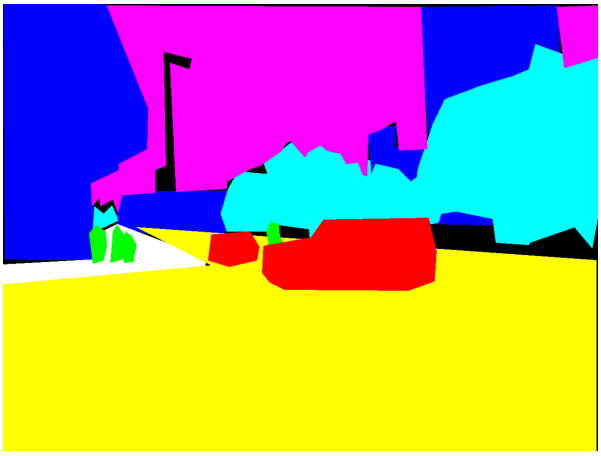
[18] W. S. Sarle. Neural Networks FAQ, 1997. `http://www.faqs.org/faqs/ai-faq/neural-nets/`. 8

[19] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. *Computer Vision–ECCV 2006*, pages 1–15. 1

[20] A. Torralba, K. Murphy, and W. Freeman. Contextual models for object detection using boosted random fields. Citeseer, 2004. 1

(a) Image

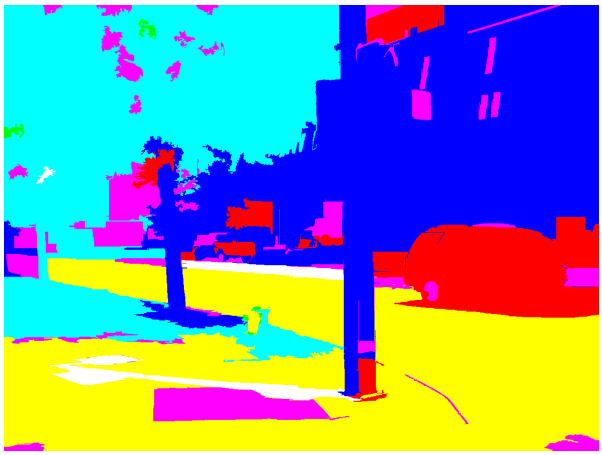
(b) Resulting segmentation



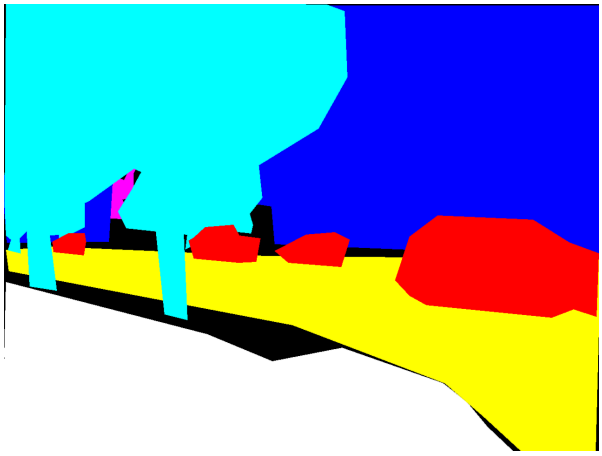
(c) Ground truth

Figure 6. Example output of the proposed algorithm. Different colors correspond to different classes of objects. See sections 4 and 5 for details.

(a) Image



(b) Resulting segmentation



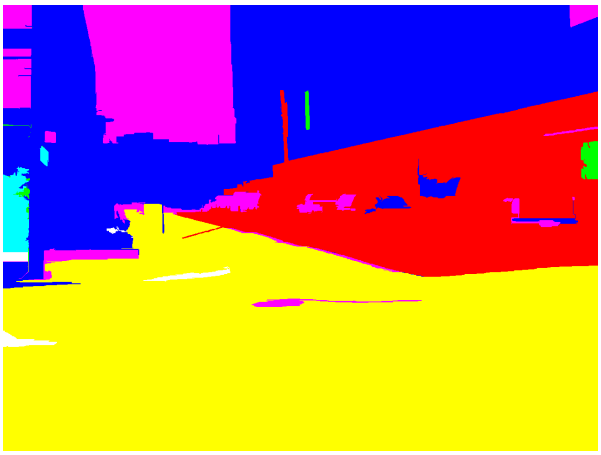
(c) Ground truth

Figure 7. Example output of the proposed algorithm.

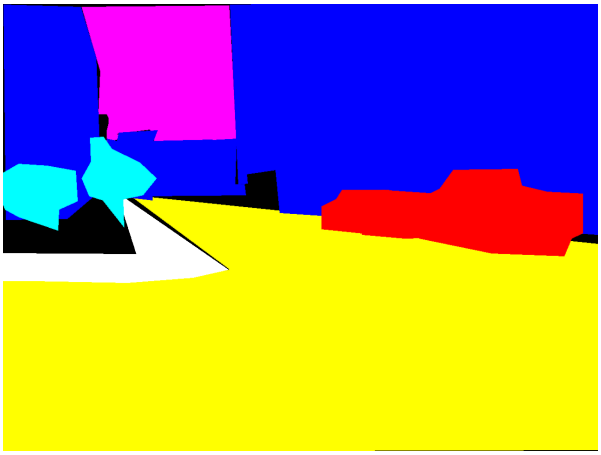
1, 6

[21] Wikipedia. Support vector machine. http://en.wikipedia.org/wiki/Support_vector_machine.

(a) Image



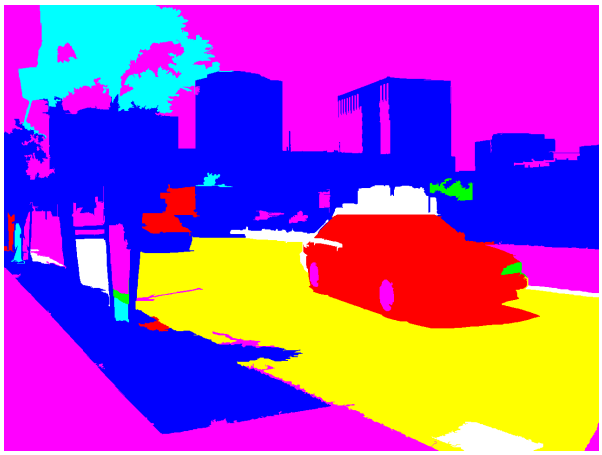
(b) Resulting segmentation



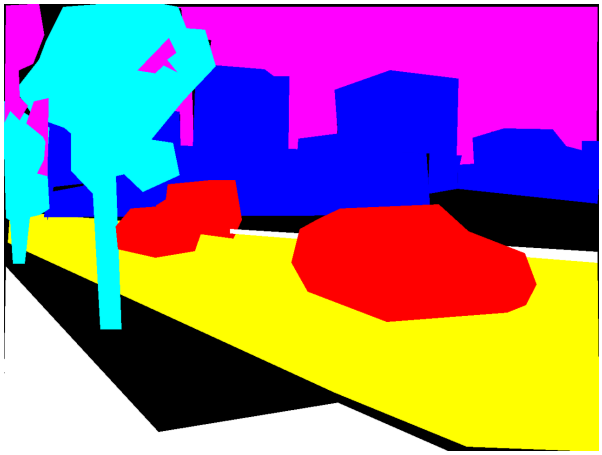
(c) Ground truth

Figure 8. Example output of the proposed algorithm.

(a) Image



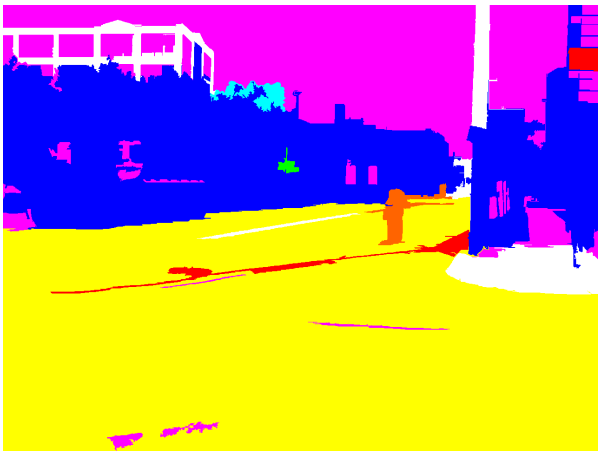
(b) Resulting segmentation



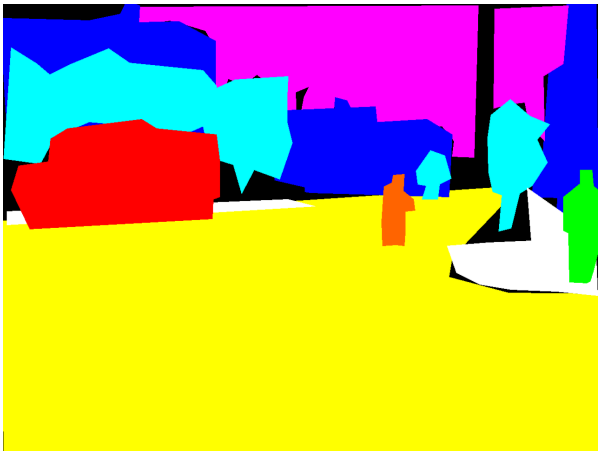
(c) Ground truth

Figure 9. Example output of the proposed algorithm.

(a) Image



(b) Resulting segmentation



(c) Ground truth

Figure 10. Example output of the proposed algorithm.