Diss. ETH No. 18410

Efficient Multi-Class Object Detection

A dissertation submitted to
ETH ZURICH

for the degree of
Doctor of Sciences

presented by

PHILIPP ZEHNDER
Dipl. El.-Ing. ETH
born 8th March 1977
citizen of Dübendorf

accepted on the recommendation of

Prof. Dr. Luc Van Gool, examiner
Prof. Dr. Tinne Tuytelaars, co-examiner

2009

Abstract

Detection of objects plays an important role in the semantic analysis of images. One of the long sought-after goals in computer vision is to reliably and efficiently detect objects of many different classes. This could be used for example to automatically generate a textual description for an image or to annotate a collection of images with sophisticated tags that not only indicate the presence of objects but also their location and extent. Furthermore, it is helpful for other image processing tasks by providing meaningful context-information. It could support for example the recognition other objects or certain parts of them. Or it could be used to infer other information like for example scene or object geometry.

This work improves the efficiency of multi-class object detection by exploiting similarities between different classes. Most previous work on object detection has focused on the acceleration or performance improvement of detection algorithms, but has treated multiple classes on a per-class basis. This does not scale well with the number of classes. To keep the computational complexity at a feasible level for higher numbers of classes, it is essential to treat similar classes jointly by using shared features. Our method targets that very aspect. We show that a considerable efficiency improvement can be achieved when class similarities are exploited. Our detector is structured as a graph that represents efficient decision sequences. Objects are detected in a sequence of splits and detection steps, whereas classes are treated as groups up to some point. The structure is learned completely automatically based on object examples. Thus, no manual specification of class relations is required.

We investigate three different approaches, two *decision trees* and a so-called *shared cascade*. All three approaches use Haar wavelets as features but differ in the type of classifiers, the training strategy or the graph structure. Both decision trees are grown by recursive addition of stumps, i.e. simple decision trees consisting of a split node and a layer of detection nodes. In the first approach, SVM classifiers are used and different variants of splits are investigated to find an efficient solution. In the second approach, AdaBoost classifiers are used and a stump is trained in a joint boosting manner. The evaluation of both approaches shows an efficieny improvement compared to standard approaches, but at the same time it points out certain disadvantages. As a better solution, the concept of the shared cascade

is developed. Contrary to the idea of splitting, this concept starts with separate cascade nodes and merges them based on a measure of similarity. The shared cascade is compared to other standard cascade approaches and shows considerably better efficiency, especially for higher numbers of classes. The algorithm has a low training complexity and scales well with the number of classes. We also show an extension that allows to add a class to an already trained shared cascade. This reduces the training effort to a fraction of that required for a full training from scratch, but still ensures efficient run-time operation.

Zusammenfassung

Detektion von Objekten spielt eine wichtige Rolle in der semantischen Analyse von Bildern. Eines der langersehnten Ziele des maschinellen Sehens (Computer Vision) ist, Objekte vieler verschiedener Klassen zuverlässig und effizient zu detektieren. Dies könnte zum Beispiel benutzt werden, um automatisch eine Text-Beschreibung für ein Bild zu generieren, oder um eine Kollektion von Bildern mit differenzierten Etiketten zu versehen, die nicht nur die Präsenz von Objekten, sondern auch deren Ort und Ausdehnung anzeigen. Weiter ist es hilfreich für andere Bildverarbeitungs-Aufgaben, indem bedeutsame Kontext-Informationen geliefert werden. Es könnte zum Beispiel die Erkennung von anderen Objekten oder gewissen Teilen davon unterstützen. Oder es könnte benutzt werden, um andere Informationen herzuleiten wie zum Beispiel Szenen- oder Objekt-Geometrie.

Diese Arbeit verbessert die Effizienz von Mehr-Klassen Detektion indem Ähnlichkeiten zwischen verschiedenen Klassen ausgenutzt werden. Die meisten früheren Arbeiten bezüglich Objekt Detektion fokusieren auf die Beschleunigung oder Treffsicherheits-Verbesserung von Detektions-Algorithmen, aber behandeln mehrere Klassen jeweils auf Einzelklassen-Basis. Dies skaliert nicht gut hinsichtlich der Anzahl Klassen. Um die Berechnungskomplexität auf einem praktikablen Niveau zu halten, ist es notwendig, ähnliche Klassen vereint zu behandeln durch Ausnutzung gemeinsamer Merkmale. Unsere Methode zielt genau auf diesen Aspekt ab. Wir zeigen, dass eine beträchtliche Effizienz-Verbesserung erreicht werden kann, wenn Klassen-Ähnlichkeiten ausgenutzt werden. Unser Detektor ist strukturiert als Graph, der effiziente Entscheidungs-Folgen darstellt. Objekte werden detektiert in einer Folge von Aufteilungs- und Detektions-Schritten, wobei Klassen bis zu einem gewissen Grad als Gruppen behandelt werden. Die Struktur wird komplett automatisch gelernt basierend auf Objekt Beispielen. Das heisst, es wird keine manuelle Vorgabe von Klassen Beziehungen benötigt.

Wir untersuchen drei verschiedene Ansätze, zwei *Entscheidungs-Bäume* und eine sogenannte *gemeinsam benutzte Kaskade*. Alle drei Ansätze benützen Haar Wavelets als Merkmale, aber unterscheiden sich in der Art der Klassifikatoren, der Trainings-Strategie oder der Graph-Struktur. Die beiden Entscheidungs-Bäume werden erzeugt durch rekursives Hinzufügen von Stümpfen, d.h. einfachen Entscheidungs-

Bäumen bestehen aus einem Aufteilungs-Knoten und einer Schicht aus Detektions-Knoten. Im ersten Ansatz werden SVM-Klassifikatoren benutzt und verschiedene Varianten von Aufteilungen untersucht, um eine effiziente Lösung zu finden. Im zweiten Ansatz werden AdaBoost-Klassifikatoren benutzt, und ein Stumpf wird in einem gemeinsamen Boosting-Verfahren trainiert. Die Auswertung beider Ansätze zeigt eine Effizienz-Verbesserung verglichen mit Standard Ansätzen, aber sie zeigt gleichzeitig gewisse Nachteile auf. Als bessere Lösung wird das Konzept der gemeinsam benutzten Kaskade entwickelt. Im Gegensatz zur Aufteilungs-Idee startet dieses Konzept mit separaten Kaskaden-Knoten und verschmilzt diese basierend auf einem Ähnlichkeits-Mass. Die gemeinsam benutzte Kaskade wird verglichen mit anderen Standard-Kaskaden-Ansätzen und zeigt eine beträchtlich bessere Effizienz, im speziellen bei einer höheren Anzahl Klassen. Der Algorithmus hat eine tiefe Trainings-Komplexität und skaliert gut mit der Anzahl Klassen. Wir zeigen auch eine Erweiterung, die es ermöglicht, eine Klasse zu einer bereits trainierten gemeinsam benutzten Kaskade hinzuzufügen. Dies verringert den Trainings-Aufwand auf einen Bruchteil dessen was für ein volles Training von Grund auf benötigt wird, aber es gewährleistet trotzdem effizientes Laufzeit-Verhalten.