

Annotating Digital Images of Insects

Automatic Generation of RDF files using Computer Vision Methods

Seminararbeit im Seminar

SEMANTIC MULTIMEDIA TECHNOLOGIES

Sommersemester 2015

Hasso-Plattner-Institut für Softwaresystemtechnik GmbH

Universität Potsdam

vorgelegt von

Leander Neiß
Friedrich Horschig
Clemens Frahnöw
Sten Ächtner

31th August 2015

Abstract

Inhaltsverzeichnis

1	Introduction	3
2	Related Work	4
2.1	Semi-automated Insect Analyzer	4
2.2	Approaches based on Machine-learning	4
2.3	Current state of Computer Vision	4
3	Requirements	5
4	Concept	6
4.1	Basic segmentation or something	6
4.2	Threshold-based Algorithm or something	6
4.3	Template Matching or something	6
4.3.1	Template Extraction or something	6
4.3.2	Sliding Window or something	6
4.3.3	Refinement of results or something	6
4.4	Comparing Effectiveness of Algorithms	6
5	Implementation	7
5.1	Algorithms	7
5.1.1	Threshold-based Analysis	7
5.1.2	7
5.2	Annotations	7
5.2.1	QR Code Analysis	7
5.3	Integrating Benchmarking	7
6	Conclusion and Discussion	8
6.1	Benchmarks	8
	Bibliography	9

1 Introduction

2 Related Work

2.1 Semi-automated Insect Analyzer

The Museum of Natural History Berlin (presenting us with our task) launched an own project to achieve the goal of digitalizing their insect photographs. The resulting tool is called “Inselect” [Gro14] and provides assistance for semi-automatic annotation of insects. During the time of this writing, their approach uses a simple edge detection from the library OpenCV (details about this library follow in the next subsection). The main part of annotation and segmentation is done by the user. They provide a possibility to scan the present QR codes for annotation proposals. There is a good coverage of tests but their test set consist of one image. Apparently, an evaluation of efficiency of their actual algorithm was not suitable so the projects are very hard to compare.

2.2 Approaches based on Machine-learning

During the last years, there was great process in image analysis. A lot of them are based on clustering algorithms which are very well suitable for either known numbers or sizes/shapes of objects [Pap92]. Others use learning algorithms which are usually trained with a large set of human-reviewed data to extract the most probable segmentations. One recent example for a rising technology is the use of neural networks [TMJ⁺10]. During chapter 4, we will explain why it would be very hard for our specific use case to gather a large training set.

2.3 Current state of Computer Vision

Luckily, there are very sophisticated approaches of static image analysis based on different features of a single image. The most famous, free and open-source collection of such computer vision algorithm is OpenCV [Bra]. This library contains very advanced methods like the “SURF” algorithm as well as basic steps for image processing for threshold-based contour detection algorithms.

One particularly interesting part of this library implements a template matching algorithm. It uses a predefined template that is applied on different positions to an image to find similar regions. The comparison of those regions to the template image can base on different factors like oriented gradients or distribution in luminosity/color.

3 Requirements

4 Concept

TODO: Explain why machine learning is a hard-to-implement approach. Huge set of human-controlled training data and stuff.

4.1 Basic segmentation or something

4.2 Threshold-based Algorithm or something

4.3 Template Matching or something

4.3.1 Template Extraction or something

4.3.2 Sliding Window or something

4.3.3 Refinement of results or something

4.4 Comparing Effectiveness of Algorithms

To have an objective measurement of the effectiveness of changes in the algorithm, we will consider two established factors for data quality. They are based on a gold standard. The gold standard will be a manually selected and annotated subset of the given data. All algorithms will be tested against the same set of data and compared against the standard. Some data will be rightfully found (true positives), some will be wrongfully found (false positives) and some insects won't be found even if they should have been (false negatives)

The first factor is the recall that measures how many relevant results were retrieved. This is achieved by dividing the true positives by all relevant results (true positives and true negatives).

The second factor is precision of the results how many of the found results were relevant. This is achieved by dividing the the true positives by all positives (true and negative).

To raise the meaning of each factor, they are usually combined in a harmonic mean called F-Measure. The resulting value can only be maximized by maximizing both values. A very low value in one of the factors will result in an overall low factor. The F-Measure is calculated as follows:

$$F - Measure = 2 * \frac{Precision * Recall}{Precision + Recall}$$

5 Implementation

5.1 Algorithms

5.1.1 Threshold-based Analysis

5.1.2 ...

5.2 Annotations

5.2.1 QR Code Analysis

5.3 Integrating Benchmarking

Due to the availability of each algorithm as single step in the pipeline, it was easy to execute them in the exact same environment with the exact same parameters. Different from the usual pipeline, the step of selecting the examples was automated and the step of writing out RDF files was replaced by analyzing the discovered bugs.

To have a reliable set of data (a “gold standard”), we created annotations for some photos manually. The photos were mainly taken randomly. In addition, we chose some photos which we found hard to analyze (due to transparent wings, difficult shapes or different sizes).

The actual gold standard was not a set of RDF files but a collections of comma-separated values. These CSV files were easier to create (with a helping tool we wrote to manually annotate the bugs) and easier to analyze than RDF documents with same contents. The actual results, their effect on our process and how we could optimize the benchmarking process will be discussed in chapter 6.

6 Conclusion and Discussion

6.1 Benchmarks

FMeasure -> value, optimization with weights Discussion of test set

Literatur

- [Bra] BRADSKI, G.: In: *Dr. Dobb's Journal of Software Tools*
- [Gro14] GROUP, Natural History Museum's Biodiversity I.: *Inselect Automated segmentation of scanned specimen images*. <http://naturalhistorymuseum.github.io/inselect/>. Version:2014
- [Pap92] PAPPAS, T.N.: An adaptive clustering algorithm for image segmentation. In: *Signal Processing, IEEE Transactions on* 40 (1992), Apr, Nr. 4, S. 901–914. <http://dx.doi.org/10.1109/78.127962>. – DOI 10.1109/78.127962. – ISSN 1053–587X
- [TMJ⁺10] TURAGA, Srinivas C. ; MURRAY, Joseph F. ; JAIN, Viren ; ROTH, Fabian ; HELMSTAEDTER, Moritz ; BRIGGMAN, Kevin ; DENK, Winfried ; SEUNG, H S.: Convolutional networks can learn to generate affinity graphs for image segmentation. In: *Neural Computation* 22 (2010), Nr. 2, S. 511–538