CoasterDex

software for detection and recognition of beer-coasters from images

Authors
David Pažout & Dagur Elinór Kristinnson
RU
19.12.2023

Contents

1 Introduction **3**
1.1 Problem Definition 3
1.2 Relevant Work 3

2 Materials and Methods **3**
2.1 Finder - Coaster Detection 4
2.2 Matcher - Instance Recognition 4
2.3 Finder&Matcher Integration 4

3 Results **4**
3.1 Dataset 4
3.2 Finder 4
3.3 Matcher 4
3.4 Finder&Matcher 4

4 Discussion **4**

5 Conclusion **4**

Abstract

CoasterDex is a software for detection and classification of beercoasters in images and video. It combines fine tuned Yolov5 based CNN with custom SIFT based instance level recognition algorithm to achieve high accuracy detection of beercoasters across varied environments.

1 Introduction

1.1 Problem Definition

This problem is inspired by my friend who collects beercoasters. They have a collection of more then 200 unique beercoasters. Every time I'm in a pub and see a beercoaster I don't know if they already have it or not. They aren't always available to be asked or they sometimes don't remember and bringing them a duplicate coaster is suboptimal.

Formal definition is as follows, we have a collection $C = \{c_1, ...c_n | c_i \neq c_j;\ \forall i, j;\ i \neq j\}$ of images of unique beercoasters or coaster-scans from frontal view. Given an input image img, the goal is to find all beercoasters $B = \{b_1...b_m\}$ present in the image and determine if a beercoaster $b \in B$ is also present in C. If b isn't present in the collection we should add it to the collection ($C \leftarrow C \cup b$).

1.2 Relevant Work

We found a few papers tackling a similar or adjacent problems. The oldest is a paper from 2011 solving postage stamp recognition from photo-scans [**?**]. Since the paper was written before wide adoption of CNN it mainly uses custom, hand-crafted features and basic template matching for classification of relatively small number of stamps. The authors in HotSpotter [**?**] try to match individual animals (not species) with past photos in a larger database with multiple instances of each individual. In [**?**] CNN is used to tackle grocery product recognition. The authors approach the problem as classification of images into 100 plus classes.

Most of the approaches for the instance recognition were taken from two throughout overviews on the subject [**?**] [**?**], where authors compare older SIFT based and newer CNN based approaches to this problem.

Majority of the previous approaches work with a static databases, e.g. the whole pipeline/algorithm have to be retrained if we wand to introduce a new type of sample. They often have multiple scans/instances available for retrieval and aren't limited by the time it takes to detect and retrieve the relevant object. We have only single instance of the coaster available in the collection and want to use our system in real-time environment where prolonged waiting times are inconvenient. System also should easily scale to larger collections (thousands of items).

2 Materials and Methods

We decomposed the problem to two main parts: Finder and Matcher. Finder is an image classification and localization algorithm and its goal is to find all occurrences of beercoasters $B = \{b_1...b_m\}$ in the image img with their appropriate bounding boxes. Matcher is an instance level recognition/retrieval algorithm and its goal is to match a beercoaster from the collection $c \in C$ with a beercoaster b present in the image img or to determine that no matching beercoaster is present in the image.

2.1 Finder - Coaster Detection

2.2 Matcher - Instance Recognition

2.3 Finder&Matcher Integration

We first input the image into the Finder, as output we receive a list of bounding boxes. If the list is empty we input an unaltered image to the matcher. Otherwise we do for each bounding box separably the following: we extract part of the image inside the bounding box, stretch the cropped image to a square - this helps with feature extraction since most coasters are of square-ish shape - and input the stretched image into the Matcher. As the output we have a list of best matched coasters for each bounding box or the whole image.

3 Results

3.1 Dataset

3.2 Finder

3.3 Matcher

3.4 Finder&Matcher

4 Discussion

5 Conclusion

References