Crowdsourcing Part Annotations for Visual Verification

by
BART VREDEBREGT
6229778

September 15, 2017

42 EC
January 2016 - September 2017

Supervisor:  Dr J.C. van Gemert
Assessor:  Dr T Gevers
Abstract

Many maintenance and repair tasks involve first verifying the condition of parts of an object e.g. bicycle maker checking for other defects before starting to repair defects. It involves first detecting the object, followed by locating each of the parts and finally judging the state of each part. We call this process Visual Verification. Many popular datasets already contain object annotations, datasets containing part annotations are rare and there is no dataset that provides part state judgments. As a result state-of-the-art object detection algorithms are only evaluated on detecting relatively large objects and not on the often much smaller parts. Thus there is a need for a new dataset. In this thesis we created an unique crowdsourced dataset consisting of 10,000 images of bicycles in various settings to fill this gap. For each bicycle 22 parts were annotated using the crowdsourcing platform CrowdFlower. Resulting in a total of 220,000 bounding box annotations with median relative area size of 60.10% to 0.40% covering a wide range of object sizes where PASCAL VOC is limited to median relative area sizes of 46.15% to 1.8%. Additionally each part in the dataset also was judged to determine its state (intact, broken, occluded or absent) allowing future research into Visual Verification. For this purpose 220,000 state judgments are made available in addition to the bounding boxes. In our experiments we show that, unlike most crowd sourcing campaign, only a single judgment is sufficient to create annotations of sufficient quality. We show under which condition this is a reasonable trade-off between cost and quality. We apply three state-of-the-art detectors to show the relationship between relative part size and performance, which shows part detection and therefor Visual Verification are yet to be solved.
Acknowledgments

I would like to thank Jan van Gemert for his continuous supervision, especially for his supportive and relax attitude during the many meetings where it became clear that again all my time went into Aiir instead of my thesis.

I would like to thank my girlfriend for her support and patience in good and bad times. I would like to thank my loved ones for their support during my study and I would like to thank my friends for the awesome distractions!

I would like to thank the Aiir team for motivating and supporting me in the process of finishing my thesis and the many moment where they asked this question causing me to refocus:

“Is your thesis done yet?”

- Aiir team

I am proud to finally be able to answer this question with “Yes!”. 
## Contents

1. **Introduction** ............................................. 4

2. **Related Work** ........................................ 6
   2.1 Crowdsourcing ......................................... 6
   2.2 Semantic Parts ....................................... 6

3. **Background** ........................................... 8
   3.1 Crowdsourcing ........................................ 8
       3.1.1 Terminology .................................... 8
       3.1.2 Platforms ...................................... 8
   3.2 Object Detection ................................... 9
       3.2.1 Selective Search ................................ 9
       3.2.2 R-CNN .......................................... 10
       3.2.3 Fast R-CNN .................................... 11
       3.2.4 Faster R-CNN .................................. 11
       3.2.5 YOLO ........................................... 12
       3.2.6 SSD .............................................. 13
       3.2.7 YOLOv2 .......................................... 14

4. **Method** ................................................ 15
   4.1 Images Collection .................................. 15
   4.2 Crowdsourcing ....................................... 15
       4.2.1 Instructions .................................... 16
       4.2.2 Task ............................................ 17

5. **Dataset** ............................................... 20
   5.1 State Annotations .................................. 20
   5.2 Bounding Box Annotations ......................... 25
       5.2.1 Relative Area .................................. 25
       5.3 Practical details .................................. 28

6. **Experiments** .......................................... 30
   6.1 Judgments ............................................ 30
       6.1.1 Experimental data set .......................... 30
       6.1.2 Combination of Judgments ..................... 31
       6.1.3 Number of Judgments ......................... 33
   6.2 Object Detectors .................................... 34

7. **Conclusion & Discussion** .......................... 37
   7.1 Conclusion ........................................... 37
   7.2 Discussion ........................................... 37
   7.3 Future Work .......................................... 38

8. **Appendices** .......................................... 43
   A IoU Graphs ............................................. 43
   B State distributions .................................... 49
   C Example Images ....................................... 51
   D State correlations .................................... 63
   E Average Bikes ......................................... 79
1 Introduction

In this thesis we use crowdsourcing to create the first benchmark dataset for Visual Verification. In the modern society many jobs & tasks involve visual inspections of objects to see whether they are complete and intact. Ranging for a car mechanic checking whether all the parts are correctly assembled to a consumer making sure all the parts of their new wardrobe are in the correct place. This is what we call Visual Verification. To thoroughly perform visual inspections many skills are required, first we need to recognize the parts, then we determine whether parts are placed correctly and finally we need to determine whether every part is intact. Humans rely on domain knowledge to perform these inspection with relative ease, however they take a lot of time to do so and mistakes are easily made. So automating some of these tasks will make our lives easier and safer.

If we look at the current state-of-the-art in Computer Vision locating object become more feasible every year. However most these objects detectors often work on clearly defined objects, work on locating parts of objects so far has mainly been focused on human body parts. We believe much work is needed to bring Computer Vision to the level were automating visual inspections becomes possible. To enable this process not only research interest into the topic is needed, also data and benchmarks are necessary. Which results in a chicken and egg situation, without datasets research interest is limited, without research interest no datasets are created.

To resolve the chicken and egg situation for the automation of visual inspection we introduce the first Visual Verification dataset. Many objects qualified to be the main subject for this dataset, in modern society many complex machine consisting of many parts are used on a regular basis, some more visible than others. We however want to use an object to which people could easily understand and relate to. We ended up on creating the dataset using *Bicycles*, according to the Dutch Central Bureau for Statics there are more than 22 million bikes (1.3 per person) in the Netherlands in 2016 making it the perfect subject for our dataset.

Our dataset contains 10,000 images of bikes for across the Netherlands in all kind of colors, shapes and models varying from brand-new to almost wrecked bikes. For each bike a total of 22 parts have been annotated using crowdsourcing campaigns in which not only the part location was annotated but also the part state divided in 4 types: intact, broken, occluded and absent. During the crowdsourcing campaign hundreds of workers helped to create this dataset on crowdsourcing platform CrowdFlower. Unlike other works only one judgment per part was used, we test how large the negative effect of this decision is by testing with up to ten judgments per part.

To test the current state-of-the-art we apply three well-known object detection models to our dataset. Additionally we attempt to show the relation between relative object area and detection quality, which determines the usability of such detectors for Visual Verification.

In this thesis we will first introduce related work both on crowdsourcing and datasets in Computer Vision.
Vision. Next we attempt to provide background information on crowdsourcing, followed by an summary of the evolution of object detection. Then we explain how our dataset was created and provide a detailed explanation about the dataset. This is followed by an elaboration on our experiments with both crowdsourcing and object detection. Finally we conclude with the conclusion and discussion.
2 Related Work

In this section we will provide a short overview of related work in crowdsourcing and crowdsourcing in Computer Vision. Finally we discuss the work so far on semantic parts.

2.1 Crowdsourcing

Crowdsourcing has become an essential tool for labeling large datasets in almost all fields ranging from Psychology to Economics and Biomedical Sciences [1][2][3]. Within Computer Science crowdsourcing has been used for almost any type of data like images [4][5][6], videos [7], 3D point clouds [8], taxonomies [9], clustering [10] and audio [11][12]. In some of the fields in which crowdsourcing was used regularly, like Speech Recognition, crowdsourcing is no longer relevant as the state-of-the-art methods [13] are now on par with professional transcribers, making crowdsourcing redundant for this topic. Within Computer Vision however crowdsourced data sets are still much needed, in many fields (e.g. Human Activity Understanding [14][7], Material Recognition [15][16]) new crowdsourced data sets are introduced to challenge the community to reach better results in increasingly challenging conditions.

One of the first standardized image classification datasets was Caltech 101 [17] followed by Caltech 256 [18], both of which were annotated by students or “Vision dataset users” making it difficult to scale. Alternatively [19] was able to generate large quantities of labels by employing the crowd to annotate the images as part of the ESP game. Another approach was used in the TinyImages dataset [20] which was able to create a dataset consisting of 79,302,017 tiny (32 × 32) images by searching image search engines using over 75,000 nouns from Wordnet [21]. As the results were not verified by humans the set contains many errors. ImageNet [4] is also based on Wordnet and contains 14,197,122 labeled images of which 1,034,908 contain bounding box annotations. Unlike TinyImages, ImageNet used crowdsourcing using Amazon Mechanical Turk (AMT) to ensure good data quality making it one of the most used datasets in Computer Vision.

Where ImageNet used crowdsourcing to annotate their images, PASCAL VOC [5] used a “single annotation party” instead. The reason for doing so was the consistency of the annotations, yet this significantly limited the scale of the dataset. For VOC2007 9,963 images with in total 24,640 objects were annotated. MS COCO [6] therefore went back to crowdsourcing through AMT, as it allowed them to label over 2.5 million objects. To still ensure high data quality a custom annotations pipeline was used consisting of several stages.

As crowdsourcing for Computer Vision is not straightforward, research has also been done to optimize the quality of annotators and the crowdsourcing process [22]. A good example of this type of research is LabelMe [23] whose contribution is twofold, a dataset and an annotation tool. The dataset got known for the fact that the labels for the images were not standardized, a lesson newer datasets all took into account. The annotation tool on the other hand is still used by many papers to quickly gather annotations. Other works focuses on handling crowdsourcing campaigns [24][25] and analysis of the annotators behavior [26][27][28][29]. Beside this alternative usages of crowdsourcing are being explored e.g. active learning [30][31] or even more extreme work like [32] that pushed for completely letting go of the classic crowdsourcing model of annotations images using bounding boxes, instead they propose to use the crowd to verify predictions of an object detector in effort to reduce the annotation time significantly.

2.2 Semantic Parts

Until recently object parts were mostly a tool to assist with object detection. For example by matching superpixels to templates [33] or deformable part models [34]. Semantic parts were also used for the fine-grained recognition. A example of this is the recognition of birds and butterflies in [35] crowdsourcing was used to gather part annotations for the CaltechUCSD Birds-200-2011 [36] dataset in order to train a classifier. With the introduction of R-CNNs [37] parts fell out of favor.
for object detection. Modern object detectors like SSD [38], YOLO [39] and Faster R-CNN [40] nowadays achieve state-of-the-art results without using any parts annotations.

In [41] the PASCAL-Part dataset was introduced as an extension on PASCAL VOC 2010, they suggest parts could be used to counter occlusion, deformations and low resolution images. The PASCAL-Part dataset contains just under 20,000 images of 20 different object classes in which on average 10 parts are annotated. In [42] the PASCAL-Part dataset is used to research whether semantic parts inherently appear in CNNs, which they concluded was only partially the case. Another part based dataset is the VehicleSemanticPart [43] which contains roughly 10,000 images and approximately 100 semantic parts. But as this dataset is not made publicly available, it has only been reused in work of the original author [44]. Thus at this moment no publicly known large part datasets exists that not only provide many annotated part but also state judgments for each part.

\footnote{{Previously known as "ImageNetPart"}}
3 Background

In this section we will provide a theoretical background for the rest of this work. First we will discuss crowdsourcing after which we will end with overview of the evolution of object detection.

3.1 Crowdsourcing

Crowdsourcing is often used in research, it allows easy access to an large group of willing participants which can fill in surveys, annotate data and much more. In this section we will discuss the essential points of crowdsourcing, namely the terminology and the platforms.

3.1.1 Terminology

Within crowdsourcing there many terms that are often used but never fully explained. Therefor this section attempts to cover the most common terms that might require clarification.

Crowdtask or Microtask are often simple straight forward tasks that can be completed in a short period of time. Most crowdtasks are tasks for which a efficient automated solution is not yet available but can easily be executed by a human. As there are many different names circulating for this type of task we use the most general version Crowdtask (CT).

Workers in crowdsourcing are members of the public that executes tasks often for a monetary reward. In the case of a monetary reward it is recommended to make sure the worker can earn at least $1 per hour [22].

Fraudulent Workers are workers that are purposely trying to minimize time and effort needed to earn the reward.

Requesters in crowdsourcing are the parties that supplies tasks for the workers to execute, most requesters are researchers or large companies.

Test Questions or Golden Questions, are (part of) tasks for which the correct outcome is known to the requester. These are often included into tasks to validate whether the worker is (still) performing the task correctly. For the maximal effect test questions should be indistinguishable from the a normal task to the worker.

Trust is an indication on how well the worker performed on all the test questions she has seen using the following accuracy formula:

$$\text{Trust} = \frac{\text{Test Question}_{\text{correct}}}{\text{Test Question}_{\text{answered}}}$$

Quality Assurances refer to the actions the requester takes to ensure the task is performed correctly and limit the reward for fraudulent workers. Ideally these ensure that performing the task correctly takes the same or less time as performing it incorrectly. By doing so an important incentive for fraudulent behavior is removed.

3.1.2 Platforms

There are many platforms available for requester to choose from, most used are Amazon Mechanical Turk (AMT) and CrowdFlower (CF). Below we will describe both platforms:

Amazon Mechanical Turk is a crowdsourcing marketplace launched by Amazon in November 2005. It is believed to be one of the first of it kind. MT has been applied to virtually every known purpose ranging from search-and-rescue missions [45] to social science experiments [46] [47]. AMT offer access to it own workforce of “Turkers” to work on Human Intelligence Tasks (HITs). Unlike CF, tasks can be hosted on any location giving the requester complete freedom in designing their
tasks. The cost of AMT start at a 20% fee on the workers reward, there are no academic plans available at this moment.

Until March 2017 only requesters with a US billing address were allow to post/request tasks on AMT which made it a less suitable solution for requesters outside the US. On the moment of writing (September 2017) a total 43 countries are supported including many European countries.

CrowdFlower is a crowdsourcing marketplace similar to AMT, in the sense it offers access to an online workforce, it was founded in 2007. CF distinguishes itself from other platforms like AMT by offering an end-to-end solution for labeling data and providing access to not only its own workforce of “Contributors” but also that of many other platforms.

Templates for many common annotation tasks are by default available on CF, if a template for a task is unavailable it can be created by the requester using a HTML variant (CML) in combination with JavaScript. But in every case the task itself will be hosted by CF. The cost of CF are a 20% fee on the workers reward. For commercial users also a monthly base fee is charged, for academic users within CF’s “Data for Everyone” plan there is no monthly fee.

Unlike AMT, CF has been accessible for European requester for a long time which made it a suitable alternative for many European requesters.

3.2 Object Detection

Recent advancements in Object Detection have greatly improved the speed and detection accuracy’s on traditional benchmarks like PASCAL VOC, MS COCO. State-of-the-art methods like Faster R-CNN, SSD and YOLOv2 reach mAPs between 73% and 78.6% on the PASCAL VOC 2007, some faster than others. These methods have one thing in common, they all predict the the object proposals internally, so without using an classic object proposal method like Selective Search[47]. In this section we will attempt to first provide some background about object detection based on Selective Search followed by a road map how the field evolved to end with explanations of three state-of-the-art object detectors.

3.2.1 Selective Search

![Figure 2: Example how Selective Search combines regions to find object proposals at different scales. On the top row the current set of regions, bottom row the resulting object proposals. As can be seen gradually the colored regions in the top row are combined into larger region till there is only a few regions left in the top left. Source: [47]](image)

In contrast to the other methods that will be discussed in this section Selective Search (SS) is not an object detection method, it is an object proposal method. Proposal methods were originally
used by many object detection methods as the first step in their pipeline. Instead of exhaustively searching every position and scale a object proposal method limits the search space to regions that have a high likelihood of being an object.

Roughly speaking SS works as follows, first a set of small regions (e.g. Superpixels) is collected that ideally only contain a single object. Next the most similar regions are combined into one bigger region, this process is repeated till only one region remains, see Figure 2. As this creates regions of ever increasing size the proposal cover all scales. As SS also need to account for color variations it repeats the process described earlier for a range of eight color space (e.g. RGB, Intensity and Hue). Beside color, an similarity measure is used take into account size, texture and shape of the regions. Finally the regions are converted in bounding boxes and duplicates are removed resulting in a set of region proposals that are likely to contain an object.

Even though the proposals generated by SS are good, most modern object detectors switch to alternative solution. The main reason being the computational cost, it takes about 2 seconds to complete. Faster alternatives following the same principals do exist, for example EdgeBoxes [48] which has a run-time of 0.25 seconds.

In order to use Selective Search for object detection it needs to be embedded in a classification framework. An classic example of this approach is using descriptors like SIFT to create features in a bag-of-words fashion. Which then later can be used to train a SVM. Using this approach a mAP of $35.1\%$ was reached on Pascal VOC 2010.

3.2.2 R-CNN

Unlike the framework described above R-CNNs [37] uses region proposals not to generate SIFT based features, instead it pioneered using CNNs to generate features for object proposals. A diagram of the framework used by R-CNNs can be found in Figure 3.

![R-CNN: Regions with CNN features](image)

**Figure 3:** Example of the R-CNN pipeline. First regions are extracted from the input image, these regions are then resize to a fixed size after which an CNN is used to create features for each regions and finally a SVM is used to classify to features. Source: [37]

Step by step R-CNNs work as follows, first a set of object proposals is generated for an image using any proposal method, in the original work Selective Search is used. For each object proposal a fixed size feature vector is computed by forward propagating it through a network consisting of five convolutional (conv) and two fully connected layers. To allow contextual information a padding is added to each object proposal. As the shape of the proposal are arbitrary the proposal plus padding is resized to $227 \times 227$ pixels disregarding the aspect ratio. The resulting 4096-dimensional features are then used to train a SVM.

At test time the features are used to score each proposal. As proposals tend to heavily overlap non-maximum suppression is applied, which rejects any scored proposal for which there is another higher scoring proposal that overlaps more than a given threshold.

By switching from a descriptor like SIFT to learned CNN based features, R-CNN reached a mAP of $54\%$ on Pascal VOC 2010 significantly out performing competing methods. But it also
3 Background

sparked a renewed interest in CNNs for object detection, many modern state-of-the-art methods are still using conv layers internally.

3.2.3 Fast R-CNN

As the original R-CNN uses a separate object proposal method and classified the results using a separate SVM performance both during training and testing was an issue. In the first improved version of R-CNN, Fast R-CNN\cite{faster_r-cnn} both train and test performance was improved by doing the complete classification using a single network which however still depends on object proposal methods for find regions of interest (RoI). See Figure 4.

![Figure 4: Example of the Fast R-CNN pipeline. Where R-CNN calculated features separately for each region Fast R-CNN calculates a single feature map on which the proposal are projected, from there a RoI pooling layer is used to create an fixed size feature for each proposal after which the class in determined using softmax layers for the object class and a regressor to determine the offset for the resulting bounding box. Source: \cite{faster_r-cnn}](image)

The original R-CNN used a pipeline consisting of three steps: proposal extraction, feature computation and classification. Fast R-CNN reduces this pipeline to just two steps: proposal extraction and object localization. In the Object localization step first an feature map is created using several convolutional and max pooling layers, after which for each RoI a fixed-length feature is created using a RoI pooling layer. The RoI pooling layer ensures an fixed size output by performing max pooling on a grid of size $H \times W$ across the input feature map resulting in an feature vector of $H \times W$ (e.g. for $H \times W = 7 \times 7$ a feature vector of length 49 is created). The resulting feature vector is then fed to a set of fully connected layers that branch into a softmax output layers that indicates the object class and a layer that outputs offsets that refine the original region proposal to create a better fitting bounding box.

As the feature map can be reused for all the region proposals processing images using Fast R-CNNs is significantly than R-CNN. Additionally training of a Fast R-CNN is $9\times$ faster than a R-CNN, this is the result of sharing resources inside batches during training. Using the improvements above Fast R-CNN reached a mAP of 66.1% on Pascal VOC 2010. The many of the improvements made in Fast R-CNN are still being used today, however Fast R-CNN was not capable of real-time parsing.

3.2.4 Faster R-CNN

Where R-CNN and Fast R-CNN both relied on proposal methods for find RoIs, Faster R-CNN\cite{faster_r-cnn} removed this slow process by incorporating the proposal generation in their network. For this purpose they introduce Region Proposal Networks (RPNs), see Figure 5. Similar to Fast R-CNN first a conv feature map is created using conv layers which will be used for both the object detection as well as the region proposals. In order to generate region proposal a small network is moved over the feature map. Taking a $n \times n$ window of the feature map as input the intermediate layer
outputs a fixed length lower-dimensional vector which is fed into two sibling fully connected layers: a bounding box regression layer and a objectness classification layer. The objectness classification predicts two values a probability of object / not-object. The box regression layer predicts offsets in regard to an anchor box, for each position of the sliding window multiple anchor boxes are used. In original work 3 scales and 3 aspect ratios are used for the anchor boxes.

![Figure 5: Example of the sliding window for a single position in a Region Proposal Network (RPN). Source: [40](#)](image)

As the RPN only predicts scored proposals, a Fast R-CNN is used is parallel to predict the object using the proposal predicted by RPN. This however complicates training, simply training the RPN and the Fast R-CNN separate would result in different weights in the conv layer resulting in a different feature map for each network. Instead the a four step pipeline is proposed: In step one, the RPN is trained, then a Fast R-CNN is trained using the proposals from the RPN but with different weights in the conv layers. Step three involves training the RPN using the conv layers from the Fast R-CNN and finally the Fast R-CNN is fine tuned using the proposal of the RPN trained in the previous step. Resulting in a network that uses the same conv layers for both generating object proposal as classifying those objects.

Without a slow proposal method, Faster R-CNN reaches a frame rate of 7 fps including the time needed for generating region proposals. At the same time the performance of Faster R-CNN is again better than its predecessor reaching a mAP of 73.2% on PASCAL VOC 2007.

### 3.2.5 YOLO

While Faster R-CNN attempted to improve performance and accuracy by improving upon the pipelines of its predecessors, YOLO [50] decided to throw out the existing pipelines and attempts to capture object detection in a single network. To do so YOLO approaches object detection as an regression problem instead, directly predicting bounding boxes and class probability from the pixels.

Figure 6 shows the essence of YOLO, first it resizes the input image to $448 \times 448$ and divides into $S \times S$ cells. Each of these cell will be responsible for any object center inside it, the cell does not have to cover the entire object, only the center. Each cell will predict $B$ bounding boxes that wrap the objects, returning for each bounding box a prediction containing: the offset relative to the cell, the size relative to the image and a confidence score which ideally represents the IoU with the ground-truth bounding box. At the same time each cell also predicts its class probability for each of the $C$ classes. At test time the confidence score of the bounding box and the class probability are combined to get class-specific scores for each bounding box.

In comparison to Faster R-CNN, YOLO is significantly faster. Where Faster R-CNN achieved 7 fps, YOLO achieves frame rates of 45 fps. The downside however is that YOLO has a lower
YOLO looks at the complete images for each detection by first resizing each image to a fixed square size and dividing it into $S \times S$ cells. For each cell a class probability is calculated and $B$ bounding boxes are predicted with a confidence that should be equal to the IoU of the actual object. The class probability and the confidence are then combined after which non-maximum suppression is used to get a single prediction per object. Source: [50]

accuracy, it reaches a mAP of 63.4\% on PASCAL VOC 2007. Another downside which is noted by the authors is that YOLO struggles to capture multiple small objects in the same region as each cell only can have one class and $B$ bounding boxes.

3.2.6 SSD

As can be seen when comparing YOLO and Faster R-CNN, speed and accuracy are a trade-off, significant improvement of the one often decreases performance on the other. SSD [38] follows a similar approach to YOLO however due to more scale invariant approach it is able to achieve significantly higher accuracy while maintaining a high speed.

Where YOLO uses only a single feature map, SSD uses a sequence of feature maps at multiple scales to capture object of different scales and improve detection quality for large objects. This is done by adding a series of conv feature layers to a base network, these layers decrease in size to capture objects at different size. While the size of the feature maps decreases the receptive field of cell increases allowing bigger objects to be captured. For each cell, a set of “default boxes” both a confidence score and offset is predicted, similar to Faster R-CNNs anchor boxes. Another advantage of using feature maps for prediction at different scales, is that it allows for small fine
grained feature to influence the detection of larger objects. There are two commonly used versions of SSD, SSD512 which is trained on an input size of 512 × 512 and SSD300 which is trained on input images of 300 × 300.

By using multiple scales of feature maps SSD300 managed to achieve significantly better results while maintaining the speed of YOLO. SSD300 achieved a mAP of 74.3% on PASCAL VOC 2007, while SSD512 achieved a mAP of 76.8% on PASCAL VOC 2007 it however run at 20 fps.

3.2.7 YOLOv2

As an improvement upon YOLO, YOLOv2 makes several changes based on earlier work like SSD. The most obvious change is that YOLOv2 uses anchor boxes instead of directly predicting bounding boxes as done in YOLO. This allows for an much higher number of predicted boxes which in turn improves the recall.

Unlike most other works, YOLOv2 uses a feature map with an odd number of cells in each direction, in this case 13 × 13 based on a 416 × 416 image and a downsample factor of 32. The reason behind this is that objects tend to be centered in the picture so it is preferable to capture these in a single cell instead of four.

![Figure 8: YOLOv2 uses k-means clustering to determine good priors based on the bounding boxes in the training set. On the left the IoU is plotted against the number of centroids k used during clustering. On the right the centroids of k = 5 are shown, in blue the centroids of MS COCO which contain greater variation in size than VOC (in white). Source: [39]](image)

Unlike SSD and Faster R-CNN which used hand picked priors for the bounding boxes, YOLOv2 actually uses k-means clustering to determine the optimal priors for a given dataset. Figure 8 shows the priors for PASCAL VOC and MS COCO using k = 5. As using euclidean distance would cause big boxes to cause larger errors an IoU based distance metric is used:

\[
d(B_i, B_c) = 1 - \text{IoU}(B_i, B_c)
\]

Where \(B_i\) is a bounding box from the train set and \(B_c\) a cluster centroid bounding box. The result are bounding box priors that are adapted to the dataset instead.

A lesson that YOLOv2 learned from SSD is that in order to improve detection quality for small objects multiple scales of feature maps are beneficial. So besides the features from the 13 × 13 feature map, also features from an earlier 26 × 26 feature map are used. This is done by first turning the 26 × 26 × 512 feature map into a 13 × 13 × 2048 which is then concatenated to the original 13 × 13 feature map. This allows the detector to also access the more fine grained features from the lower layers. This is however only done for one fine-grained layer and not with all feature map like in SSD.

Using these changes YOLOv2 highest resolution (544 × 544) version reaches a mAP of 78.6 on PASCAL VOC 2007 at 40 fps while the lowest resolution (288 × 288) version reaches a mAP of 69.0 at a frame rate of 91 fps.
4 Method

In this section we will describe the methods use for this project, first the image collection followed by an explanation of the crowdsourcing effort.

4.1 Images Collection

For the creation of the dataset used in this thesis images were scrapped from a website containing lost and found items from across the Netherlands. Only items that matched both the search term "Fiets" (Dutch for bike) and were in the category "Fietsen" (Dutch for bikes) were selected. This resulted in a collection of more than 20,000 images of bikes in various settings e.g. storage depot and streets.

As the goal of the source website is to enable users to recognize their lost bikes, most images show the bike clearly and in the center of the image. The images however tend to be far from professional photos and are representative of how an average person would take a picture of a bike. As the images were not taken with Computer Vision in mind the images were filtered to remove low quality images based on the following criteria:

- **Images have a sufficient resolution**, any picture that has a width smaller then 200 pixels is removed. These images were most likely thumbnails that have been uploaded as being full images.
- **Images should be sharp**, any picture that has a significant blur is removed due to e.g. incorrect focus of the camera or excessive compression. As these images cannot be used to judge the state of the bike.
- **Images should contain the complete bike**, any picture that does not show all parts of the bike is removed. As this makes it impossible to judge the state of all parts of the bike.
- **Images should not contain multiple bikes**, any picture that contains multiple bikes is removed. As this makes it hard for the annotator to decide which bike to annotate.
- **Images should contain at least one bike**, any focuses on another object than a bike is removed (e.g. pictures of tandems). As these object do not have all the parts a bike has or has multiple instances of the part.

After filtering just over 10,000 images remained after which the smallest photos were removed to create a set of exactly 10,000 images to be annotated through crowdsourcing.

4.2 Crowdsourcing

The annotation of 22 parts in 10,000 images is extremely time consuming, we therefore decide to use crowdsourcing to speed up the process. To do so we created a crowdsourcing campaign consisting of crowd tasks (CT) in which workers annotated the parts in each of the images.

Unlike most other works that involve crowdsourcing we decided not to use Amazon Mechanical Turk (AMT) for our CTs, instead we went for a less known competitor CrowdFlower (CF)\[51\]. The main advantage of CF compared to AMT is the complete framework CF offers. At AMT it is necessary for the researcher to provided their own annotation tool including any quality assurance methods on might need, most project therefore end up using standard annotation solution like LabelMe\[23\] or home-made annotation tools that often lack essential crowdsourcing tools e.g. quality assurance. At CF the basic structure is already handle for the researcher, it ensures that test questions are always included and it automatically collects relevant statistics about the campaign and the participating workers.

Beside the framework differences between AMT and CF also the user-bases are different, where AMT only allows workers registered through their own portal to work on CTs, CF also publishes
the tasks on a wide variety of sites/channels that offer users a reward in return for completion of CTs. As the quality of the channels greatly varies CF also provides statistics of the quality of the user-base of a channel, for this work only channels with a high quality where used.

Within the selected channels all the user groups, not only the “high level” workers were allowed to participate. The reasoning behind it was twofold. Firstly, it greatly increased the throughput of CTs annotating 10,000 images for a single part took on average 28 hours, the longest throughput period was around 50 hours. Secondly it allowed us to keep the cost low, “low level” workers often are willing to take on easy but low-paying CTs to build their reputations such that they will be promoted to the “high level” worker group.

At after a worker ended work on the CT they are presented with a small survey rating the CT on a 0-5 scale which 0 is the lowest possible score and 5 the highest. The survey consists on 5 topics:

- **Overall** the score workers give the CT in general.
- **Instructions Clear** the score how clear workers find the instructions.
- **Test Question Fair** the score how fair workers find the test questions.
- **Ease of Job** the score how easy worker find the CT.
- **Pay** the score how satisfied contributors are with the payment they receive for the CT.

Each of these aspects can be used to identify issues with the CT. The rest of this section will be focused on the implementation of the crowdsourcing campaign, first of the instruction followed by the task design and finally the results.

## 4.2.1 Instructions

For a CT it is vitally import to make sure the worker understands the task before she starts. In our task this is done in 3 steps: explaining, showing and testing.

### Step 1: Locate the Head Light

Look at the provided image and locate the position where you would expect the Head Light. The Head Light is located on the front of the bike.

### Step 2: Draw a bounding box

Draw a bounding box around the Head Light for its expected location, covering the complete Head Light without unnecessary extra space and press “Save”.

### Step 3: Determine state

Determine if the Head Light in the image is either of the following list, starting at the top and moving down the list to find a description that matches the state:

1. **Intact**
2. **Broken**
3. **Absent**

Parts that are not visible at all, or are partially occluded due to an external object, making judging the state of the part impossible.

**Figure 9:** Example of the step by step instructions provided to the annotator of the head light

First the worker is presented a simple overview explaining the 3 steps involved in the task, see Figure 9. In the first version of the instructions the steps and state descriptions were explained in great detail, explaining every possible scenario. Later on it became clear workers did not read the complete text if it was to long, therefor both the steps and the state description were simplified. This resulted in an increase of the Instructions Clear survey result from 2.5 to on average 3.5 out of 5.

After the instruction examples are used to show to the worker what exactly is expected, see Figure 10. These examples have two functions: for workers that have read the instructions it becomes more clear what they exactly need to do. For workers that did not or only partially read the instructions it gives them an easy way to see what the goal of the task is. In the first version these
also contained examples of poorly drawn bounding boxes, this was dropped in the simplification step to ensure that no confusion could arise about the proper way of executing the task.

Finally before the worker could start the CT she had to pass a quiz consisting of test questions only. The test questions validated both the amount of drawn bounding boxes and the state judgments. To pass the quiz a worker need to answer more than 80% of the test questions correctly (i.e. Trust). During the different campaigns on average roughly 60% of the workers that participated in the quiz passed. The exact percentage varied per part, one observation that could be made that some workers participated in CTs for several parts, passed the quiz more easily.

4.2.2 Task

While designing a CT there are multiple relevant aspects to consider: the desired results, cost, ease of use and quality assurances. For this project the desired output is to get annotations for each part of a bike, in the form of a localization and a state judgment. For the localization there are several options ranging from simple to complex, namely dot annotations in which the center of the target is annotated, bounding boxes in which a rectangle is drawn around the target, polygon tracing in which a polygon is drawn around the target or per pixel annotation in which every pixel of the object is marked. In the end we decide to go for bounding boxes as it is the desired input format for most object detection algorithms and it is relatively easy for a worker to learn and perform. For the state judgments a set of clearly defined states was created for the worker.
to choose from. The desired output was thus determined to balance both usability with in the dataset and practicality of creating the output.

To reach the desired output multiple CT designs are possible, the simplest solution is to present a worker with an image and let him annotate all the desired parts. Alternatively one could present the image and ask the worker to annotate just one part and its state. Or more extreme have one annotator draw the bounding box for a part and have another worker judge the part state. In the first example there are only 10,000 CTs (given 10,000 images and 22 parts in the set), in the second example this grows to 220,000 CTs and in the last example to 440,000 CTs. As the amount of CTs increase each CT becomes simpler and more straight forward, yet it also requires more intermediary processing and validation steps. For this project the second approach was chosen as the most suitable approach as it limited the amount of CTs and made the CT easy to explain, as the instructions (and the worker) can be focused on only one part.

An example of an CT as it was presented to the workers is shown in Figure 11. On the left side the image is shown, scaled to a width of 670 such that even the smaller images are easily visible. Alternatively the user has the option to view the full-scale image to better observe the state. In this image the user can draw a bounding box (using Javascript library Annotorious). On the right the steps from the instructions are repeated to help remind the worker and to structure the CT. When the worker draws a bounding box the number of bounding boxes shown at Step 2 is automatically increases, as can be seen in Figure 12.

![Figure 11: Example of the task shown to the annotator of the head light.](image)

On average a worker completed a single CT in 13 seconds for which the worker was paid $0.0012. The resulting hourly wage is therefor $0.33 per hour which is extremely low. One should aim for an hourly rate of at least $1.00 per hour, sadly this was not possible in this work due to budget restrains. As mentioned before, the most probable explanation why workers still participate in low-paying job like ours is the need to perform CTs to get to a higher ranked worker group.

Another aspect of design the CT is quality assurance, before the quiz was already mentioned this is the first of several filters that are designed to remove low quality workers. After the quiz the worker is presented with a page containing 10 CTs, of which one is a so-called test or golden questions which is used to validate whether the worker is still performing the task correctly. It is essential it is unknown to the worker which CTs are the tests questions as this will render the test question useless, as workers in that case only will complete the task question. In our campaign on average 7% of the workers was removed for failing to many test questions i.e. trust fell below a set threshold, all the annotations of these workers were discarded. For every question it was validate that there was exactly one bounding box drawn and that a state judgment was given. As a final check a random sample of annotations was manually checked to ensure the quality of annotations, during this process only 2 workers across all parts where removed, indicating that the quality assurances were sufficient.
**Figure 12:** Example of the task with the bounding box drawn and state filled as it would be seen by the annotator of the head light.

In conclusion, the CT was designed to gather data that could easily be used for Computer Vision purposes, while keeping it simple for the workers. The quality assurances and checks used filtered out a large portion of fraudulent workers ensuring a good annotation quality.
5 Dataset

The main contribution of this work is to provide the community with a dataset that can be used for training systems for part detection and visual verification. Therefore the dataset contains not only images of bikes, but also bounding box annotations for 22 parts including a label describing the state of the annotated object. In this section we will first discuss the state annotation, followed by the bounding box annotations and finally a discussion of the general datasets statistics.

5.1 State Annotations

For each image in the dataset 22 parts were annotated as described in Section 4.2. The description of the parts can be found in Table 1. The selected parts cover most of the parts are usually present on a Dutch city bike, only one notable part was left out: The Frame. It was not included as defining the border of the frame is extremely difficult and the absence of the frame would make the bike a collection of parts.

For each of the parts we required the annotators to draw a bounding box and judge its state. The states which the annotator could select were defined as follows:

- **Intact** The part is clearly visible and doesn’t show any sign of damage.
- **Broken** The part is shows signs of damage (for example rust, dents) or is missing parts.
- **Absent** The part is missing completely and is not occluded.
- **(Partially-) Occluded** The part is not visible at all, or is partially occluded due to an external object, making judging the state of the part impossible.

The states are prioritized and mutually exclusive meaning that each part will always have only one state label even when a combination could be justified e.g. a saddle that is clearly damaged but also partially occluded (see Figure 14) will be marked as broken instead of the combination broken/occluded. Furthermore the (Partially-) Occluded state excludes partially occlusion by other parts of the bike, this because of the fact that some parts are always partially occluded by other parts.
<table>
<thead>
<tr>
<th>Part Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Back Hand Break</td>
<td>The Back Hand Break is located on the steer, farthest away from the camera.</td>
</tr>
<tr>
<td>Back Handle</td>
<td>The Back Handle is located on the steer, farthest away from the camera.</td>
</tr>
<tr>
<td>Back Mud Guard</td>
<td>The Back Mud Guard is located near the rear wheel of the bike.</td>
</tr>
<tr>
<td>Back Pedal</td>
<td>The Back Pedal is the pedal that is farthest away from the camera.</td>
</tr>
<tr>
<td>Back wheel</td>
<td>The Back wheel is the wheel underneath the saddle.</td>
</tr>
<tr>
<td>Bell</td>
<td>The Bell is located on the steer of the bike and is often absent.</td>
</tr>
<tr>
<td>Chain</td>
<td>The Chain is often occluded by the gear case.</td>
</tr>
<tr>
<td>Dress Guard</td>
<td>The Dress Guard is located near the rear wheel.</td>
</tr>
<tr>
<td>Dynamo</td>
<td>The Dynamo is located near the top of the front wheel of the bike.</td>
</tr>
<tr>
<td>Front Hand Break</td>
<td>The Front Hand Break is the hand break closest to the camera and is located</td>
</tr>
<tr>
<td></td>
<td>on the steering wheel.</td>
</tr>
<tr>
<td>Front Handle</td>
<td>The Front Handle is located at the steer, nearest to the camera.</td>
</tr>
<tr>
<td>Front Mud Guard</td>
<td>The Front Mud Guard is located near the front wheel of the bike.</td>
</tr>
<tr>
<td>Front pedal</td>
<td>The Front pedal is the pedal that is closest to the camera.</td>
</tr>
<tr>
<td>Front wheel</td>
<td>The Front wheel is the wheel underneath the steer.</td>
</tr>
<tr>
<td>Gear case</td>
<td>The Gear case is located near the rear wheel of the bike and is often occlud</td>
</tr>
<tr>
<td>Head Light</td>
<td>The Head Light is located on the front of the bike.</td>
</tr>
<tr>
<td>Kickstand</td>
<td>The Kickstand is located near the rear wheel of the bike.</td>
</tr>
<tr>
<td>Lock</td>
<td>The Lock is located near the rear wheel of the bike. We only care about dis</td>
</tr>
<tr>
<td></td>
<td>k locks, so not about chains and cable locks!</td>
</tr>
<tr>
<td>Rear Light</td>
<td>The Rear Light is located at the rear of the bike, on the mudguard</td>
</tr>
<tr>
<td>Rear Reflector</td>
<td>The Rear Reflector is located at the rear of the bike, at the back of the</td>
</tr>
<tr>
<td>Saddle</td>
<td>The Saddle is located at the top of the bike, it is where the user sits.</td>
</tr>
<tr>
<td>Steer</td>
<td>The Steer starts at the frame and ends and the handles.</td>
</tr>
</tbody>
</table>

Table 1: Parts annotated in the dataset including the description that was given to the annotator

parts (e.g. the back pedal), meaning that it is actually a property of the part instead of relevant occlusion. Still complete occlusion by other parts is included in the (Partially-) Occluded state as we can no longer judge visually identify the part or its state.

As can be seen in Figure 15 the distribution of states greatly varies per part, roughly speaking they can be divided in three main categories:

- **Essential Parts** are parts that are critical for the usability of the bike. These parts are characterized with a high percentage of intact states and have low amounts of absence and damage e.g. front pedal and steer.

- **Hidden Parts** are parts that tend to be outside of the direct line of sight. These parts are characterized with a high percentage of occluded states and have low amounts of absence and damage e.g. back pedal and chain.

- **Optional Parts** are parts that not every bike has and tend to have limited influence on the usability of the bike. The parts are characterized with a relative high percentage of absent states.

In Figure 54 the correlation between the intact state of parts is shown, in Appendix D the correlations of all other combinations of states can be found. A positive correlation indicates that the two part states likely to occur together, a negative correlation indicates that the two part states are unlikely to occur together. For example the intact chain and intact gear case have a strong negative correlation, which makes sense because if the gear case is intact you shouldn’t be able to see the chain inside it. Another example is the positive correlation between the front- and back hand break, which indicates that they if the front hand break is intact also the back hand break is intact.

The correlation is calculated by first determining the covariance matrix for the observed states $X$. 

5 Dataset
Figure 14: Example of a Saddle that is damaged (reversed) but also occluded which is marked a broken due to the prioritization of the states.

Figure 15: State distribution per part, a larger version of this image can be found in Appendix B.
and $Y$ of part $i$ and $j$ as follows:

$$C_{ij} = \frac{\sum_{i=1}^{N}(X_i - \bar{X})(Y_i - \bar{Y})}{N - 1} \quad (3)$$

After which the correlation was calculated using the following formula in which $C$ is the covariance matrix:

$$R_{ij} = \frac{C_{ij}}{\sqrt{C_{ii} \cdot C_{jj}}} \quad (4)$$

An important note is that the correlations are based on 1 vs all states signals, so for example the intact state each part is classed as intact or not intact. As a consequence when comparing different states the diagonal is not perfect positive correlated as one might expect, instead it will always be a negative correlation in which a high negative correlation indicates that the two states appear more than remaining two states. To illustrate, the signal used for calculating the correlation between the intact saddle $S^{(intact, saddle)}$ and the occluded saddle $S^{(occluded, saddle)}$ of one part could be as follows:

$$S^{(intact, saddle)} = \{0, 1, 0, 0, 1, 0, 1, \ldots, 0, 1\}$$
$$S^{(occluded, saddle)} = \{1, 0, 0, 0, 1, 0, 0, \ldots, 1, 0\} \quad (5)$$

Note that for each position (image) only one of the signals is one. This is because only one part is annotated per image and each part can only have one state. The result is that a perfect negative correlation $C = -1.0$ indicates that the two signals are exactly opposite, which happens when either one signal is always 1 and the rest always 0 or when the two states are the only ones ever to be activated.
Figure 16: Correlation matrices for the combination of the Intact state with all states including Intact. In each plot the vertical axis contains all intact parts, the horizontal axis the other state for each part. A positive correlation indicates that the parts in the given state(s) tend to appear together, for example the Intact Chain and Absent Gear Case. A negative correlation means that the parts in the given state(s) tend not to appear together for example Intact Back Mudguard and Occluded Dress guard.
5.2 Bounding Box Annotations

In each of the images a bounding box was drawn around each of parts. The instruction to the annotator was to draw the box no matter the state and leave as little extra space around the object. The result is a total of 220,000 bounding boxes which can be used for various purposes.

As bounding boxes were drawn independent of state, parts that are labeled *(Partially-)* Occluded and *Absent* do have a bounding boxes, but not around the part itself but around the location the annotator would expect the part to be located. This has practical reasons (see 4.2.2) but also practical applications, it can be used to get an feeling for the context of the object. For example for an absent head light the bounding box is drawn a the front of the frame where it is usually located.

In Figure 17 for each part an ellipse was plotted sized based on the mean width and mean height of the bounding boxes and positioned based on the mean position relative to the saddle. Resulting in a figure in which the shape of a bike is easily recognizable, which can be used to see how the different parts relate to each other not only in size but also on position. Most interesting is the overlapping between the gear-case and chain in which as expected the gear-case is slightly larger as it is supposed to cover the chain.

![Figure 17: Visualization of the structure of the different parts](image)

5.2.1 Relative Area

In many object detection datasets the target objects cover a large part of the images. For example in PASCAL VOC 2010 the *cat* class on average covers 47.09% of the image, while the *bottle* class covers 1.78%. To calculate the coverage the relative area is calculated between image $I$ and annotation $A$ as follows:
<table>
<thead>
<tr>
<th>Part Class</th>
<th>Median Relative Area</th>
<th>Mean Relative Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bike</td>
<td>60.10%</td>
<td>61.44%</td>
</tr>
<tr>
<td>Back Wheel</td>
<td>13.14%</td>
<td>13.50%</td>
</tr>
<tr>
<td>Front Wheel</td>
<td>12.42%</td>
<td>12.68%</td>
</tr>
<tr>
<td>Back Mudguard</td>
<td>11.17%</td>
<td>11.72%</td>
</tr>
<tr>
<td>Front Mudguard</td>
<td>7.01%</td>
<td>7.57%</td>
</tr>
<tr>
<td>Gear Case</td>
<td>5.51%</td>
<td>5.86%</td>
</tr>
<tr>
<td>Chain</td>
<td>5.06%</td>
<td>5.41%</td>
</tr>
<tr>
<td>Dress Guard</td>
<td>4.25%</td>
<td>4.85%</td>
</tr>
<tr>
<td>Steer</td>
<td>3.95%</td>
<td>4.36%</td>
</tr>
<tr>
<td>Kickstand</td>
<td>1.66%</td>
<td>2.08%</td>
</tr>
<tr>
<td>Front Hand Break</td>
<td>1.17%</td>
<td>1.59%</td>
</tr>
<tr>
<td>Front Pedal</td>
<td>1.16%</td>
<td>1.47%</td>
</tr>
<tr>
<td>Saddle</td>
<td>1.12%</td>
<td>1.28%</td>
</tr>
<tr>
<td>Lock</td>
<td>0.98%</td>
<td>1.22%</td>
</tr>
<tr>
<td>Back Hand Break</td>
<td>0.98%</td>
<td>1.25%</td>
</tr>
<tr>
<td>Front Light</td>
<td>0.75%</td>
<td>1.15%</td>
</tr>
<tr>
<td>Dynamo</td>
<td>0.73%</td>
<td>1.08%</td>
</tr>
<tr>
<td>Front Handle</td>
<td>0.63%</td>
<td>1.20%</td>
</tr>
<tr>
<td>Back Pedal</td>
<td>0.57%</td>
<td>0.95%</td>
</tr>
<tr>
<td>Back Light</td>
<td>0.57%</td>
<td>1.03%</td>
</tr>
<tr>
<td>Bell</td>
<td>0.53%</td>
<td>1.24%</td>
</tr>
<tr>
<td>Back Reflector</td>
<td>0.49%</td>
<td>0.80%</td>
</tr>
<tr>
<td>Back Handle</td>
<td>0.40%</td>
<td>0.53%</td>
</tr>
<tr>
<td>All</td>
<td>5.84%</td>
<td>6.27%</td>
</tr>
</tbody>
</table>

Table 2: Relative area of part annotations in our dataset. The relative area of an part is determined by dividing the area of the part bounding box by the total image area. For example an bounding box of size 50 × 50 in an 800 × 600 image would have an relative size of 0.5%. To put the numbers into perspective also the Relative size of the complete bike is reported, this is created by combining the bounding boxes of all parts.

$$\text{Relative Area}(I, A) = \frac{A_w * A_l}{I_w * I_h}$$

In Table 2 the relative areas for our dataset is shown, in Table 3 the relative areas for PASCAL VOC 2010. As can be seen our dataset contains significantly smaller parts than PASCAL VOC, this property makes it more difficult for the object detector to complete the task as the amount of pixels available for detection is lower. Current state-of-the-art detectors are validated based on sets like PASCAL VOC that mainly contain relatively large annotated objects, which can potentially bias detectors toward larger objects. An example of this is the behavior of the original YOLO which performed well on PASCAL VOC but had big issues with detecting groups of small objects (see Section 3.2.5) something that does not appear often in PASCAL VOC. We believe that our data sets collection of small parts in combination with larger parts ensures a fair benchmark for object detection quality.
<table>
<thead>
<tr>
<th>Object Class</th>
<th>Median Relative Area</th>
<th>Mean Relative Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>bottle</td>
<td>1.78%</td>
<td>7.35%</td>
</tr>
<tr>
<td>car</td>
<td>3.07%</td>
<td>15.63%</td>
</tr>
<tr>
<td>pottedplant</td>
<td>3.50%</td>
<td>12.40%</td>
</tr>
<tr>
<td>sheep</td>
<td>3.86%</td>
<td>12.87%</td>
</tr>
<tr>
<td>boat</td>
<td>4.11%</td>
<td>15.47%</td>
</tr>
<tr>
<td>tvmonitor</td>
<td>6.39%</td>
<td>14.32%</td>
</tr>
<tr>
<td>chair</td>
<td>6.50%</td>
<td>12.96%</td>
</tr>
<tr>
<td>cow</td>
<td>7.53%</td>
<td>18.80%</td>
</tr>
<tr>
<td>bird</td>
<td>8.10%</td>
<td>17.78%</td>
</tr>
<tr>
<td>person</td>
<td>8.95%</td>
<td>17.71%</td>
</tr>
<tr>
<td>bicycle</td>
<td>16.78%</td>
<td>28.73%</td>
</tr>
<tr>
<td>bus</td>
<td>22.19%</td>
<td>29.26%</td>
</tr>
<tr>
<td>motorbike</td>
<td>22.79%</td>
<td>32.07%</td>
</tr>
<tr>
<td>aeroplane</td>
<td>23.31%</td>
<td>28.35%</td>
</tr>
<tr>
<td>diningtable</td>
<td>23.42%</td>
<td>29.21%</td>
</tr>
<tr>
<td>horse</td>
<td>24.02%</td>
<td>30.16%</td>
</tr>
<tr>
<td>dog</td>
<td>30.16%</td>
<td>36.33%</td>
</tr>
<tr>
<td>sofa</td>
<td>31.14%</td>
<td>39.35%</td>
</tr>
<tr>
<td>train</td>
<td>31.50%</td>
<td>35.27%</td>
</tr>
<tr>
<td>cat</td>
<td>46.15%</td>
<td>47.09%</td>
</tr>
<tr>
<td>All</td>
<td>16.26%</td>
<td>24.06%</td>
</tr>
</tbody>
</table>

**Table 3:** Relative area of annotations in PASCAL VOC 2010. Note that the big difference between Median and Mean in small object classes indicates the existence of much large annotations for these object classes.
5.3 Practical details

In this section we will describe the more practical information about the dataset, all resource mentioned in this section will be made publicly available.

The annotations are provided as a JSON file structured around the images, each image is a key in the main dictionary. For each image the entry starts with the filename followed by the dimensions of the image (in image) and finally a dictionary of all the annotated parts, like this:

"48678.jpg": {
    "filename": "48678.jpg",
    "image": {
        "channels": 3,
        "height": 480,
        "width": 640
    },
    "parts": {
        # The part entries, see below
    }
}

It was decided to add the image dimension directly in the annotations as it might be preferable for some applications to be allow for easy filtering based on the dimension, as the image sizes are not normalized as can be seen in Table 4.

<table>
<thead>
<tr>
<th>Width</th>
<th>Height</th>
<th>Megapixels</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>640</td>
<td>480</td>
<td>0.3</td>
<td>5028</td>
</tr>
<tr>
<td>640</td>
<td>360</td>
<td>0.23</td>
<td>4305</td>
</tr>
<tr>
<td>640</td>
<td>427</td>
<td>0.27</td>
<td>333</td>
</tr>
<tr>
<td>512</td>
<td>384</td>
<td>0.2</td>
<td>139</td>
</tr>
<tr>
<td>352</td>
<td>288</td>
<td>0.1</td>
<td>81</td>
</tr>
<tr>
<td>640</td>
<td>426</td>
<td>0.27</td>
<td>45</td>
</tr>
<tr>
<td>640</td>
<td>478</td>
<td>0.31</td>
<td>16</td>
</tr>
<tr>
<td>640</td>
<td>640</td>
<td>0.41</td>
<td>11</td>
</tr>
<tr>
<td>640</td>
<td>384</td>
<td>0.25</td>
<td>8</td>
</tr>
<tr>
<td>640</td>
<td>359</td>
<td>0.23</td>
<td>7</td>
</tr>
<tr>
<td>622</td>
<td>492</td>
<td>0.31</td>
<td>3</td>
</tr>
<tr>
<td>640</td>
<td>362</td>
<td>0.23</td>
<td>3</td>
</tr>
<tr>
<td>328</td>
<td>246</td>
<td>0.08</td>
<td>3</td>
</tr>
<tr>
<td>302</td>
<td>240</td>
<td>0.08</td>
<td>2</td>
</tr>
<tr>
<td>640</td>
<td>512</td>
<td>0.33</td>
<td>2</td>
</tr>
<tr>
<td>640</td>
<td>383</td>
<td>0.25</td>
<td>2</td>
</tr>
<tr>
<td>640</td>
<td>385</td>
<td>0.25</td>
<td>1</td>
</tr>
<tr>
<td>640</td>
<td>361</td>
<td>0.23</td>
<td>1</td>
</tr>
<tr>
<td>640</td>
<td>482</td>
<td>0.31</td>
<td>1</td>
</tr>
<tr>
<td>519</td>
<td>425</td>
<td>0.22</td>
<td>1</td>
</tr>
<tr>
<td>480</td>
<td>640</td>
<td>0.31</td>
<td>1</td>
</tr>
<tr>
<td>640</td>
<td>358</td>
<td>0.23</td>
<td>1</td>
</tr>
<tr>
<td>640</td>
<td>461</td>
<td>0.3</td>
<td>1</td>
</tr>
<tr>
<td>360</td>
<td>480</td>
<td>0.17</td>
<td>1</td>
</tr>
<tr>
<td>420</td>
<td>311</td>
<td>0.13</td>
<td>1</td>
</tr>
<tr>
<td>640</td>
<td>481</td>
<td>0.31</td>
<td>1</td>
</tr>
<tr>
<td>480</td>
<td>360</td>
<td>0.17</td>
<td>1</td>
</tr>
<tr>
<td>640</td>
<td>513</td>
<td>0.33</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4: Dimensions of the images in the dataset

For each of the parts there is an entry in the parts section of the image entry, the consist of
Dataset

four parts. Firstly, the part_name, secondly the bounding box as absolute_bounding_box and relative_bounding_box followed by the object_state both as human readable text and a integer and finally the trust score of the annotation (see 3.1.1) like this:

"back_hand_break": {
    "absolute_bounding_box": {
        "height": 29,
        "left": 263,
        "top": 95,
        "width": 43
    },
    "object_state": "absent",
    "object_state_class": 2,
    "part_name": "back_hand_break",
    "relative_bounding_box": {
        "height": 0.0604166666666667,
        "left": 0.4109375,
        "top": 0.19791666666666666,
        "width": 0.06874999999999998
    },
    "trust": "0.8947"
},

The decisions to include both the absolute_bounding_box and relative_bounding_box is again based on providing easy ways of filtering, it is not unlikely that one might wants to exclude parts based on their size, in that case having the absolute_bounding_box is easiest. While the relative_bounding_box provides the most accurate version of the bounding box when images are resized.
6 Experiments

In this section we will first experiment with crowdsourcing, by looking at the effect of multiple judgements and the way these can be combined. Then we move on to experiment with three state-of-the-art object detectors to determine the current capabilities of object detection methods to detect parts.

6.1 Judgments

In Crowdsourcing it is common practice to use multiple judgments for any given task, as it would allow validation of each judgment based on the other judgments thus improving the overall quality of the resulting data. Yet the effectiveness of using multiple judgments depends on several key-factors like domain, combination method and quality assurance methods used. In this section we will discuss experiments that show the effects of these factors.

6.1.1 Experimental data set

The main dataset described before only has one judgment per image per part, so in order to experiment with multiple judgments an alternative dataset is needed. For this purpose 100 images with an intact front wheel were selected from the main data set.

The front wheel was selected as it is the least ambiguous part and the relatively big size of the part making the task as simple as possible, while similar to the domain of our main data set.

From there 3 crowd sourcing campaigns with different quality assurance methods were started:

- **No quiz** In this campaign the quiz and test question were turned off completely, so anybody could execute the task and there was no way for a contributor to banned from the task. A total of 21 contributor started this task of which 0 were removed.

- **Easy quiz** In this campaign the quiz and test question were selected to be obvious examples of a intact front wheel, broken front wheel or absent front wheel. See Figure 18. A total of 35 contributor started this task of which 1 was removed during the quiz.

- **Hard quiz** In this campaign the quiz and test question were selected to be hard examples of a intact front wheels, broken front wheels or absent front wheels. See Figure 19. A total of 37 contributor started this task of which 7 were removed. (5 during the quiz and 2 due to failing test question).

All judgments made by removed contributors were excluded from the dataset to ensure the best possible results in each set.

![Figure 18: Examples of the images used for the test questions in the "Easy quiz" set. As can be seen these examples are really easy to distinguish for a human, making it almost impossible to miss for a human paying even the slightest amount of attention to the task.](image)
6 Experiments

Figure 19: Examples of the images used for the test questions in the “Hard quiz” set. As can be seen these examples are significantly more difficult to spot, requiring the contributor to pay attention to the task.

Ground truth labels for each images were created by the author. Each image was manually annotated two times, after which the best fitting bounding box for each image was stored. This way the boxes are fitted as tightly as possible ensuring a proper ground truth for evaluation in the experiments to follow.

6.1.2 Combination of Judgments

When working with multiple judgments it is often required to combine them into a single result. The combination methods available heavily depend on the domain. For example in sentiment analyses majority voting works quite well while this is does not work when working with bounding boxes. In this section we will discuss and evaluate several methods that are suitable for combining bounding boxes.

For this experiments the following possible methods of combining bounding boxes are considered:

- **Random Bounding Box** selects a random judgment as the resulting bounding box. This method is the baseline combination method which creates the same effect as using a single judgment.

- **Naive Mean Bounding Box** creates the resulting bounding box by taking the mean corners of all the bounding boxes.

- **Naive Median Bounding Box** creates the resulting bounding box by taking the median of each corner coordinate. When an even number of bounding boxes is supplied the mean between the middle 2 coordinates is used.

- **IoU Mean Bounding Box** similar to Naive Mean Bounding Box, but with the main difference that only boxes that have an Intersection over Union (IoU) with another box higher then threshold $t$ are considered. In our experiments $t = 0.25$ is used.

- **IoU Median Bounding Box** similar to Naive Median Bounding Box, but with the main difference that only boxes that have a Intersection over Union (IoU) with another box higher then threshold $t$ are considered. In our experiments $t = 0.25$ is used.

- **Majority Voting on Proposals (1 vote)** uses a proposal method (in our case Selective Search) to generate proposals on which each bounding box votes for the proposal with the highest IoU.

- **Majority Voting on Proposals (3 votes)** same as Majority Voting on Proposals (1 vote) with the change that the votes are placed on the three proposals with the highest IoU.

From the considered methods *Random Bounding Box* is the baseline method, every properly functioning combination method should improve upon it. *Majority Voting on Proposals* was added to
provide a intelligent approach to combining bounding boxes, the downside of this method that it is significantly more expensive to calculate then the other methods.

In this experiment each method is evaluated 100 times on shuffled data. If the data is not shuffled the results of every method would be influenced by the order in which the judgments were given. For example if for unknown reasons the third judgment has a higher chance to be a junk judgment this will decrease the results for every method. Therefor the ordering of the judgments is shuffled each iteration.

As can be seen in Figure 20 the Majority Voting methods out perform the other methods. Interestingly enough the amount of votes seems to have a limited effect, only with 2 example there is a notable difference. Downside of the Majority Voting remains the computational cost, were the other methods one iteration on the experimental dataset took about 0.5 seconds, while Majority Voting took 75 seconds excluding the time needed to generate proposals.

Beside the Majority Voting methods only the Median methods were able to significantly improve upon the random baseline. This is most likely due to the fact that using the median junk judgments have a extremely limited effect upon the final results, while using the mean exactly also takes the junk judgments into account when calculating the final result. If we look at the effect of only using the IoU between the judgments we only see that there is a notable effect on the Median method below seven judgments, after that the positive effect disappears most likely due to the increased chance for each bounding box to overlap with another bounding box.

In conclusion the most suitable method for further experiments are the Median methods as it is gives good results. Also it is the most straight forward method and the computational costs are significantly lower then Majority Voting. For real world application were computation time is not an issue Majority Voting should be the preferred option.
6.1.3 Number of Judgments

For our data set it was decided to only use one judgment per image instead of the commonly assumed minimum of 3 judgments per CT. The reasoning behind this is two fold, firstly high amounts of judgments are costly and secondly the effect of multiple judgments is unclear.

To get a better feeling of the effect of multiple judgments we used the before mentioned subset of the data to show the effect of using multiple judgments. For this experiment we only used the Naive Median Bounding Box method due its performance and computational cost (see section 6.1.2). As can be seen in Figure 20 the Naive Median Bounding Box curve has a similar shape as the best performing method Majority Voting on Proposals (3 votes) therefor we assume the conclusions based on Naive Median Bounding Box will also hold for other (more sophisticated) methods.

![Figure 21: Mean Intersection over Union (IoU) for multiple judgments collected with different quality assurance methods. Number of judgments of 2 is left out as the implementation of median used applies the mean in that case. For the full size version see Appendix A.](image1)

![Figure 22: Benefit for extra judgments on the Intersection over Union (IoU) collected with different quality assurance methods. Number of judgments of 2 is left out as the implementation of median used applies the mean in that case. For the full size version see Appendix A.](image2)

As can be seen in Figure 21 there is a significant improvement in the IoU in each setting when multiple judgments are used. Which is to be expected as the amount of data it is based on is at least tripled, what is interesting however is the improvement in IoU caused by increase the amount of judgments, this is shown more clearly in Figure 22. Here it becomes clear that especially the increase from one to three judgments has a clear effect on the IoU, just under 0.03 in the Hard quiz setting and more then 0.05 in the No quiz setting.

Does that mean that it is always better to go with 3 judgments instead of just 1? No, because beside the quality of the information also the cost triple. So for the same budget we have the choice to have either a 2.5% to 5% increase in IoU or triple the amount of annotated images. In the time of Deep Learning and many parameters it might be a good idea to go for triple the amount of images, unless the amount of available raw data is limited of course.
In conclusion, the amount of judgments in crowdsourcing should be determined by the amount of available raw data, the required data quality and the budget. It is a balance that should carefully weighted, but for our data set the choice for one judgment seems validated by these results.

6.2 Object Detectors

Modern state-of-the-art detectors are tested on benchmark datasets like PASCAL VOC which tend to contain mostly objects with a large relative area, see Section 5.2.1. In this experiment we are testing three state-of-the-art detectors (YOLOv2, SSD and Faster R-CNN) on our dataset to answer the question whether the large area bias influenced these detectors.

As our dataset contains annotations for intact, absent, damaged and occluded parts unlike normal benchmarks which only contain intact objects we only use parts labeled intact in these experiments. Potentially the broken and occluded parts could also be added but for these states the severity heavily influences how much they look like intact parts. For example a broken front wheel can be a flat tire which is a minor change compared to an intact front wheel, while a broken front wheel can also be a completely folded wheel which no longer looks like an intact front wheel. Thus by only using the intact part the resulting dataset is similar to a normal object detection benchmark except for the reduced relative area, meaning the detectors should perform similar to the PASCAL VOC results.

The intact dataset was randomly divided into a training set contain 8,000 images, a validation set of 1,000 images and a test set of 1,000 images. For YOLOv2 the default Darknet implementation provided by the authors [39] was used with slight adjustments to enable to use of our 22 classes. For Faster R-CNN and SSD the Tensorflow implementations contained in the Tensorflow Object Detection API [52] were used, more specifically “Faster RCNN with Inception Resnet” and “SSD with Inception V2”. Also for the Object Detection API small adjustments were made to use our 22 custom classes.

All results are reported as mean Average Precision (mAP) which is a commonly used metric for reporting results in object detection. The mAP is calculated taking the mean of the Average Precision (AP) of all classes. The AP is meant to capture the shape of the precision & recall curve in a single value. It is calculated as follows using eleven equally spaced recall levels:

\[
AP = \frac{1}{11} \sum_{r \in \{0.0, 0.1, \ldots, 1\}} p_{\text{interp}}(r)
\]  

(7)

Where \( p_{\text{interp}}(r) \) is the precision interpolated by taking the maximum precision where the corresponding recall exceeds \( r \) as follows, where \( p(r) \) is the measured precision for recall \( r \):

\[
p_{\text{interp}}(r) = \max_{\tilde{r} \geq r} p(\tilde{r})
\]  

(8)

In Table 5 the mAP of the different models are shown. It is clear YOLOv2 performs significantly worse then the other models with a mAP of 0.2869. Only on the back wheel class YOLOv2 out performs the other models and for bell class YOLOv2 failed to detect any of the objects in the test set. Faster R-CNN often performs slightly worse than SSD, which is to be expected as SSD is a more advanced successor of Faster R-CNN.

If the detectors were unbiased towards larger objects there should be no correlation between the relative area and the mAP of the object classes. In Table 6 the correlations between mAP and relative area for each of detectors are shown, it is clear that there is a significant correlation between the relative area and the mAP. As it is an positive correlation this indicates that smaller relative areas co-occur with a lower mAPs and vice versa. It is important to note that a correlation not necessarily indicate causality. However a likely explanation is that optimizations were made to improve detection quality on the well-known benchmark datasets which decreased the detection quality for object types not covered by the benchmarks.
Table 5: Results from applying three state-of-the-art object detectors to the intact parts in our dataset. For all models an IoU of 0.5 was used to determine the quality of the detection.

<table>
<thead>
<tr>
<th>Part</th>
<th>YOLOv2</th>
<th>SSD</th>
<th>Faster R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Back Hand BREAK</td>
<td>0.0061</td>
<td>0.1133</td>
<td>0.1285</td>
</tr>
<tr>
<td>Back Handle</td>
<td>0.0728</td>
<td>0.3737</td>
<td>0.3479</td>
</tr>
<tr>
<td>Back Light</td>
<td>0.0916</td>
<td>0.2878</td>
<td>0.3632</td>
</tr>
<tr>
<td>Back Mudguard</td>
<td>0.7999</td>
<td>0.8534</td>
<td>0.8393</td>
</tr>
<tr>
<td>Back Pedal</td>
<td>0.0223</td>
<td>0.1584</td>
<td>0.1322</td>
</tr>
<tr>
<td>Back Reflector</td>
<td>0.1135</td>
<td>0.2280</td>
<td>0.1908</td>
</tr>
<tr>
<td>Back Wheel</td>
<td>0.9219</td>
<td>0.9022</td>
<td>0.7293</td>
</tr>
<tr>
<td>Bell</td>
<td>0.0000</td>
<td>0.0809</td>
<td>0.0488</td>
</tr>
<tr>
<td>Chain</td>
<td>0.5859</td>
<td>0.6807</td>
<td>0.3956</td>
</tr>
<tr>
<td>Dress Guard</td>
<td>0.0588</td>
<td>0.6080</td>
<td>0.3851</td>
</tr>
<tr>
<td>Dynamo</td>
<td>0.0298</td>
<td>0.2630</td>
<td>0.2099</td>
</tr>
<tr>
<td>Front Handbreak</td>
<td>0.0012</td>
<td>0.1130</td>
<td>0.1055</td>
</tr>
<tr>
<td>Front Handle</td>
<td>0.0283</td>
<td>0.2120</td>
<td>0.1655</td>
</tr>
<tr>
<td>Front Light</td>
<td>0.0610</td>
<td>0.2003</td>
<td>0.1858</td>
</tr>
<tr>
<td>Front Mudguard</td>
<td>0.5752</td>
<td>0.8029</td>
<td>0.7195</td>
</tr>
<tr>
<td>Front Pedal</td>
<td>0.0416</td>
<td>0.2413</td>
<td>0.1770</td>
</tr>
<tr>
<td>Front Wheel</td>
<td>0.8761</td>
<td>0.9215</td>
<td>0.8503</td>
</tr>
<tr>
<td>Gear Case</td>
<td>0.7052</td>
<td>0.8031</td>
<td>0.7008</td>
</tr>
<tr>
<td>Kickstand</td>
<td>0.0757</td>
<td>0.5500</td>
<td>0.4018</td>
</tr>
<tr>
<td>Lock</td>
<td>0.2386</td>
<td>0.5087</td>
<td>0.4748</td>
</tr>
<tr>
<td>Saddle</td>
<td>0.4286</td>
<td>0.6754</td>
<td>0.6748</td>
</tr>
<tr>
<td>Steer</td>
<td>0.5782</td>
<td>0.6925</td>
<td>0.6726</td>
</tr>
<tr>
<td>Mean</td>
<td>0.2869</td>
<td>0.4668</td>
<td>0.4045</td>
</tr>
</tbody>
</table>

Table 6: Pearson Correlation coefficients between the relative area and the mAP of the tested detectors. The overall column shows the correlation between the average mAP of the three detectors and the relative area of the parts.

<table>
<thead>
<tr>
<th></th>
<th>YOLOv2</th>
<th>SSD</th>
<th>Faster R-CNN</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Median Relative Area</td>
<td>0.8981</td>
<td>0.8328</td>
<td>0.7982</td>
<td>0.8684</td>
</tr>
<tr>
<td>Mean Relative Area</td>
<td>0.8912</td>
<td>0.8266</td>
<td>0.7904</td>
<td>0.8613</td>
</tr>
</tbody>
</table>
In Figure 23, the median relative area is plotted against the mAP for each object class for each detector. Trend lines were added to show the trend in the data, if the detectors were completely unbiased to size one would expect these lines to be close to horizontal. As can be seen neither of the trend lines is close to horizontal, indicating the detectors do not generalize to small object area’s. Interestingly enough Faster R-CNN, the oldest of the three methods, has the most flat trend line.

![Graph showing relative area vs mAP for object detectors]

**Figure 23:** The relative area is correlated with the performance of the object detectors. If an object detector is size invariant the trend line should be almost horizontal. For the tested detectors it is clear that Faster R-CNN is the most size invariant, followed by SSD and finally YOLOv2.

In conclusion, in our experiment we showed that there is an correlation between the relative area and the performance of the three state-of-the-art detectors. While SSD outperformed the other methods, Faster R-CNN shows the weakest correlation between mAP and relative area. YOLOv2 was by far the weakest method, only on large parts it was able to perform similarly or better than other models.
7 Conclusion & Discussion

In this section we will first summarize the outcomes from our research, followed by a short discussion of the work done and finally the future work in which we summarize potential next steps.

7.1 Conclusion

This work used crowdsourcing to create a new benchmark dataset which can be used for Object Detection, Part Detection and Visual Verification. We showed that it is possible to create an usable benchmark dataset within a limited budget. Based on our dataset we showed that state-of-the-art object detection models show signs of a bias toward larger objects making the less suitable for part detection and therefor Visual Verification.

Over the course of multiple crowdsourcing campaigns for 22 unique bicycle parts a total of 220,000 bounding box annotations were created. For each annotated part, workers were asked to provide a judgment of the object state limited to the general state classes intact, broken, occluded and absent. The result is an unique dataset allowing work not only in part detection but also classification of part states.

In crowdsourcing when using multiple judgments combining judgments is an essential step to get to usable data. We showed a variety of combination methods for bounding boxes some of which improved the IoU while other are actually decreasing the IoU. It was shown that using proposal methods we can actually improve the quality of bounding box annotations in our setting. However this is a computationally expensive and using proposal method to enhance ground-truth data might amplify any bias already present in the data on which the proposal method in trained.

In our experiments we show that the assumption that multiple judgments are always needed in crowdsourcing is not always true. The benefit of multiple judgments when using proper quality assurance methods is limited to an increase in IoU of 3% - 4% while the cost of crowdsourcing can increase up to ten fold. Making the trade off between quality and quantity even more relevant. In our case the usage of three judgments per part would mean instead of 10,000 images only 3,333 images could have been annotated within the same budget which would have greatly reduced the usability of our dataset for learning methods.

Additionally we showed in our experiments applying three state-of-the-art detectors on our dataset that for some parts reasonable results already can be achieved. By combining statistics about our dataset and the results from the experiments we showed there is a correlation between the relative area of the objects and the quality of the detection measured using the mAP. Interestingly enough the most modern object detection model, YOLOv2, performed worst of the three tested detection models both in absolute mAP and in the correlation between relative area & mAP. During our experiments SSD proved to be the best performing object detection model for detecting intact parts. Faster R-CNN however did show the lowest correlation between relative area and mAP.

With our crowdsourcing campaigns we created a unique dataset and showed that a commonly made assumption in crowdsourcing is not necessarily true. By applying detectors we showed that not all state-of-the-art detectors perform as expected on our dataset.

7.2 Discussion

In this work a simple crowdsourcing pipeline was used to cost efficiently create a benchmark dataset, for this purpose many trade-offs were made between cost and quality. The main trade-off was in the design of the crowdsourcing campaign. We limited the crowdsourcing pipeline to a single crowdsourcing task while a higher annotation quality could have been reached by implementing a multi-stage pipeline. In our experiments we showed that a single annotation is sufficient to reach an overall good annotation quality. It is important to note that this is overall annotation quality, some of the annotations are still extremely low quality (IoU = 0.0). As a result it can still
be preferable to implement either another validation stage during the crowdsourcing campaign in which the results are checked or use more judgments to reduce the chances of incorrect annotations.

Another trade-off which was made early on during the crowdsourcing process was the part selection. Originally 35 parts were selected to be annotated including the later removed classes panniers, basket, child seat seat post and crank-arms. Based on the limited budget it was decided to exclude classes that were present only in a small subset of the images e.g. panniers, basket, child seat. Also structural classes that were always present if other parts were present were removed e.g. seat post and crank-arms. This decision reduced the complexity and cost of crowdsourcing drastically however it does mean our dataset does not cover all the bike parts commonly present on Dutch bikes.

Another thing that was considered during the setup of the crowdsourcing campaign is pricing. During the current campaign on average $0.0012 was paid per annotations. With an average 13 seconds spent on each annotation this means the workers only got paid $0.33 per hour of work. It is recommend in the literature to aim for at least $1 per hour. In our case the lower price most likely caused a slower throughput rate. However by being a relatively easy task in which workers can easily score points toward better paying jobs, the throughput was still high enough to complete the tasks in a reasonable amount of time. In a commercial or academic setting were sufficient budgets are available however one should still aim for an hourly rate of at least $1 per hour.

In our experiments with multiple judgments we showed that increasing the number of judgments while using hard quality assurance methods only increase the IoU by 3% - 4% improve the IoU from 77% to 81% with 10 judgments. To put this number in perspective, in Object Detection predictions with an IoU of 50% are seen as correct predictions. So if the ground truth annotations are 3% - 4% more accurate predictions with and IoU of more46% would also be marked as correct predictions, we feel this effect is limited enough to justify not using multiple judgments.

For the experiments with the object detectors, we used standard implementations. This has two advantages, firstly it makes implementation easy. Secondly it reduces the risks of fine-tuning the detectors variables to perfectly fit the dataset. However the risk of this decision is that some of the implementations are more sub-optimal then others, one could argue for example that the YOLOv2 results are caused by not optimizing YOLOv2’s hyper-parameters.

In our reported experiment we only used the intact parts, this made the task similar to normal object detection conditions. Broken parts however can have an similar appearance to an intact part, which could distract the models during training. Same holds for Occluded parts which sometimes only have minor occlusions. During an early test a model was trained on all states which at test time was able to pinpoint the location of parts like the Lock even when it was Absent. This shows that in some cases that context of part was learned instead of the appearance of the actual part. As we wanted to test the detectors capability to detect the parts we believe our experiment gives a correct representation.

When looking at the correlations of relative area and mAP it seems that the more recent detectors have an increasing bias towards larger objects. We believe that this is caused by the benchmarks used, they contain more large objects than small objects meaning that to improve upon the state-of-the-art further improving large object detection is the way to go. We believe there are three solutions for this: firstly, using different benchmark dataset which have an more equal distribution of relative area. Secondly, keep using the same benchmarks and wait till improvement on large objects becomes harder then improvements on small objects. Finally, adapting the metrics used incorporates any size bias that might be present. Which one or combination of solution is the best solution is something only time will tell.

### 7.3 Future Work

As stated before our crowdsourcing pipeline was single stage and single judgment, leaving room for improvements in the directions of multi stage pipelines and smart usage of multiple judgments. In the post crowdsourcing cleaning in this work we did not use a technique that could potentially have identified incorrect annotations, namely structure analyses. When multiple parts on an object
are annotated it is possible to identify the average internal structure of the object allowing the removal of extremely unlikely annotations, for example if in an image the back wheel is annotated as being underneath the steering wheel and the head light it is likely that the back wheel annotation is incorrect. Thus leveraging the relatively fixed structure of objects could allow easy validation of annotations provided by workers.

In many applications data quantity is as important as data quality, above we described a possible improvement of the data quality. We believe that increasing the data quantity can also help improve the usability of the dataset. As images of bikes are easy to come by we believe running more crowdsourcing campaigns should be considered. During future campaigns it is possible to experiment with different types of annotations, bounding boxes are straightforward but for some parts it is more sensible to use the more precise pixel annotations or the faster less precise dot annotations.

Based on our experiments with state-of-the-art object detection models it is clear there are still much improvement is needed in the fields of object & part detection before visual verification become reality. Using our dataset it is possible to further improve detection quality for smaller objects enabling more in-depth research into the usability of part annotations.

In the experiments it became clear that Faster R-CNN had the weakest correlation between relative area and mAP. This could indicate that older algorithms, like R-CNN and Deformable Parts Models, might have an even smaller bias toward large objects. Further research into this could show whether recent improvements are indeed biased toward large objects. As mentioned before research into a new evaluation metric which incorporates the presence of size bias as well as detection quality could help provide a more fair indication of detection quality.

Additionally the state annotations in our dataset allows research into what we called Visual Verification. Within the scope of this work it was not yet possible to set train models for this purpose, we therefor challenge the community to use our dataset to further develop this field. We believe work into this topic can lead to applications in industries like, but not limited to, medicine, maintenance and production. An possible research subject could be learning to recognize occluded parts by their surroundings, for example a human can easily determine whether the saddle of an bike is located even when the saddle is not visible while most object detection algorithms will fail to do so.
References


Appendices

A IoU Graphs

Figure 24: Mean Intersection over Union (IoU) for multiple judgments collected with in the hard quiz setting
Figure 25: Mean Intersection over Union (IoU) for multiple judgments collected with in the easy quiz setting
Figure 26: Mean Intersection over Union (IoU) for multiple judgments collected with in the without quiz setting
Figure 27: Benefit for extra judgments on the Intersection over Union (IoU) collected with in the hard quiz setting
Figure 28: Benefit for extra judgments on the Intersection over Union (IoU) collected with in the easy quiz setting.
Figure 29: Benefit for extra judgments on the Intersection over Union (IoU) collected with in the without quiz setting
B State distributions

Figure 30: State distribution per part (1/2).
Figure 31: State distribution per part (2/2).
C Example Images

![Intact](image1)

(a) Intact

![Damaged](image2)

(b) Damaged

![Absent](image3)

(c) Absent

![Occluded](image4)

(d) Occluded

**Figure 32:** Examples of Dynamos in the different states
Figure 33: Examples of Dress guards in the different states

Figure 34: Examples of Steers in the different states
Figure 35: Examples of Front mudguards in the different states

Figure 36: Examples of Locks in the different states
(a) Intact

(b) Damaged

(c) Absent

(d) Occluded

**Figure 37:** Examples of Back handles in the different states

(a) Intact

(b) Damaged

(c) Absent

(d) Occluded

**Figure 38:** Examples of Back lights in the different states
Figure 39: Examples of Back mudguards in the different states

Figure 40: Examples of Front wheels in the different states
Figure 41: Examples of Back wheels in the different states

Figure 42: Examples of Front handbreaks in the different states
Figure 43: Examples of Front pedals in the different states

Figure 44: Examples of Saddles in the different states
Figure 45: Examples of Bells in the different states

Figure 46: Examples of Kickstands in the different states
Figure 47: Examples of Front handles in the different states

Figure 48: Examples of Gear cases in the different states
Figure 49: Examples of Back reflectors in the different states

Figure 50: Examples of Front lights in the different states
Figure 51: Examples of Chains in the different states

Figure 52: Examples of Back pedals in the different states
Figure 53: Examples of Back hand breaks in the different states
Figure 54: Coorelation matrix for Intact vs Intact
Figure 55: Coorelation matrix for Intact vs Damaged
Figure 56: Coorelation matrix for Intact vs Absent
Figure 57: Coorelation matrix for Intact vs Occluded
Figure 58: Correlation matrix for Damaged vs Intact
Figure 59: Coorelation matrix for Damaged vs Damaged
Figure 60: Correlation matrix for Damaged vs Absent
Figure 61: Correlation matrix for Damaged vs Occluded
Figure 62: Coorelation matrix for Absent vs Intact
Figure 63: Correlation matrix for Absent vs Damaged
Figure 64: Coorelation matrix for Absent vs Absent
Figure 65: Coorelation matrix for Absent vs Occluded
Figure 66: Coorelation matrix for Occluded vs Intact
Figure 67: Coorelation matrix for Occluded vs Damaged
Figure 68: Coorelation matrix for Occluded vs Absent
Figure 69: Coorelation matrix for Occluded vs Occluded
E  Average Bikes

Figure 70: Mean of all images centered around the Back Handle.

Figure 71: Mean of all images centered around the Back Hand Break.
Figure 72: Mean of all images centered around the Back Light.

Figure 73: Mean of all images centered around the Back Mudguard.

Figure 74: Mean of all images centered around the Back Pedal.
Figure 75: Mean of all images centered around the Back Reflector.

Figure 76: Mean of all images centered around the Back Wheel.

Figure 77: Mean of all images centered around the Bell.
Figure 78: Mean of all images centered around the Chain.

Figure 79: Mean of all images centered around the Dress Guard.

Figure 80: Mean of all images centered around the Dynamo.
Figure 81: Mean of all images centered around the Front Handbreak.

Figure 82: Mean of all images centered around the Front Handle.

Figure 83: Mean of all images centered around the Front Light.
Figure 84: Mean of all images centered around the Front Mudguard.

Figure 85: Mean of all images centered around the Front Pedal.

Figure 86: Mean of all images centered around the Front Wheel.
Figure 87: Mean of all images centered around the Gear Case.

Figure 88: Mean of all images centered around the Kickstand.

Figure 89: Mean of all images centered around the Lock.
Figure 90: Mean of all images centered around the Saddle.

Figure 91: Mean of all images centered around the Steer.