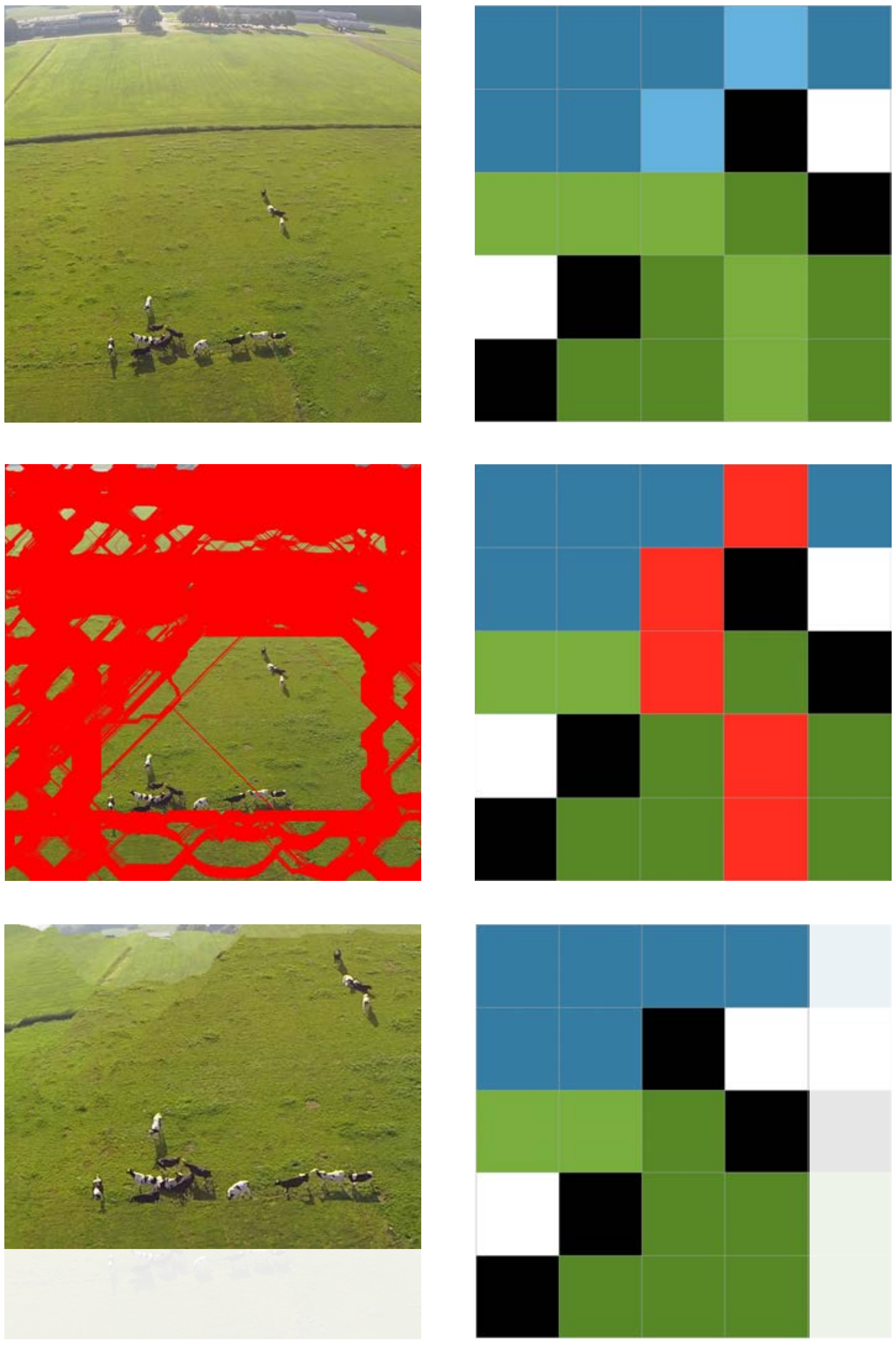


Resizing Images with CAIR

Content-Aware Image Resizing for Fast Object Detection on a Nature Conservation Drone

by

A.E.M. Visser

UNIVERSITY OF AMSTERDAM

MSc Artificial Intelligence

Master Thesis

Resizing Images with CAIR

Content-Aware Image Resizing for Fast Object Detection on a Nature Conservation Drone

by

A.E.M. Visser

6277209

October 22, 2016

42 ECS

Supervisors:	*Assessor:*
Dr Jan van Gemert	Dr.ir. Leo Dorst
Dr Arnoud Visser	

ABSTRACT

Unmanned aerial vehicles (UAVs) are an efficient surveillance tool; by flying over the game reserve rangers are able to cover much more ground than on their traditional foot patrol. However, the aerial imagery gathered by the UAV contains a lot of irrelevant information. In fact, less than 1% of the pixels contain the information the rangers require to protect their game reserve. Current methods for object detection are not fast enough to process the high resolution aerial imagery on a nature conservation drone.

Resizing the image results in a higher detection rate, but by resizing the image without considering the content we might remove the important pixels containing the object the ranger is looking for. Resizing images increases the detection rate in two ways. First the reduction in image size allows the proposal method to generate candidate boxes faster. Second, due to the reduction in image size the proposal method generates fewer candidate boxes that need to be evaluated by the object detector.

We use a content-aware image resizing method that avoids the removal of important pixels. We show that using our methods we can remove 75% of the pixels without negatively impacting detection performance. In fact, we find in some cases that due to the reduction in the number of candidate boxes, the detector produces fewer false positives resulting in better detection performance on the resized images.

The results of this thesis were published in The 28th Benelux Conference on Artificial Intelligence (BNAIC 2016) as:

Anouk Visser, Content-Aware Image Resizing to improve the Object Detection rate in Aerial Imagery. In The 28th Benelux Conference on Artificial Intelligence (BNAIC 2016).

CONTENTS

1 INTRODUCTION

To combat poaching or perform game counts, nature conservationists need to inspect areas that are very large and hard to reach by car or foot. To do so, nature conservationists have started to use unmanned aerial vehicles (UAVs) [1] [2]. Over the past few years, it has become increasingly simple to buy and fly a UAV, or drone. But, what do drones offer to conservation workers? This question is answered clearly in [3]: drones gather images efficiently. UAVs equipped with cameras enable conservation workers to monitor large areas more easily. Drones have proven to be useful for various nature conservation tasks including the counting of elephants without disturbing the animals [4] as well as looking for signs of animals such as orang-utan nests [5] which are often hard to see from the ground.

This thesis focusses on the application of drones as a tool for anti-poaching operations. Sadly, 2014 was a record-breaking year for the number of rhinos poached in South Africa. A total of 1215 rhinos were poached, a 21% increase over the previous year [6]. The declining rhino population has now become a major cause for concern for many nature conservationists, game reserves and governments. Many different measures are taken to combat the illegal poaching of rhinos, one of the most notable ones being the use of drones.

Currently, rangers patrol the game reserve by foot to their best ability, looking for tracks of endangered animals and poachers. However, patrolling on foot only allows for surveilling a limited area. To combat poaching more efficiently, a different solution is needed. In [7] the application of UAVs to anti-poaching operations is described and evaluated thoroughly. The authors list three ways in which UAVs could be useful for anti-poaching operations. First, the UAV can serve as an efficient surveillance tool. By flying over the game reserve rangers are able to cover much more ground than on their traditional foot patrol. Second, the UAV can be used as a tool to use during poaching incidents, where it could for example relay the position of the poachers in real-time to the rangers to assist during an arrest or catch the poachers red-handed. And lastly, the mere presence of a UAV might deter poachers from entering the game reserve at all.

RHINOS	**training**	**testing**
Dimensions	3000×2250	3000×2250
Number of images	169	168
Number of rhinos	626	613
Average rhino dimensions	34×36	35×36

Table 1: Overview of the Birds.ai Rhinos Dataset, which we will refer to as RHINOS.

COWS	**training**	**testing**
Dimensions	1920×1080	1920×1080
Number of images	141	127
Number of cows	1383	1227
Average cow dimensions	40×43	41×44

Table 2: Overview of the Verschoor Aerial Cow Dataset [9], which we will refer to as COWS.

There are, however, some limitations that prevent the widespread use of UAVs for anti-poaching purposes. In most cases it is not be possible to establish a real-time high quality data stream between the UAV and the ground station that is required for effective surveillance of the reserve. This means that the gathered images can only be analyzed after the UAV has landed. In order to transfer the images to the ground station in a timely manner, the amount of information should be reduced. To do this, an image compression algorithm may be used.

Image compression algorithms are able to encode an image such that the number of bits needed to present the image is reduced. The goal of image compression is not to make any changes to the image, but just reduce the file size. This is achieved by applying a combination of redundancy reduction (removing repeating patterns from the image) and irrelevance reduction (removing parts of the image that are irrelevant) [8].

The idea of irrelevance reduction is an interesting one when applied to the usage of UAVs for anti-poaching operations. The vast majority of data captured by a UAV is mostly irrelevant. During an anti-poaching mission the ranger is interested in finding two objects: poachers and rhinos. These objects do not appear frequently in the captured aerial imagery and when they do, take up a minimal amount of pixels compared to the background.

To illustrate that the majority of data is irrelevant, we show some statistics of the two datasets that will be used throughout this thesis. The Birds.ai Rhino Dataset contains a total of 337 images with a total of 1,239 rhinos that need to be detected. We will refer to this dataset as the RHINOS dataset. In Table 1 we show some statistics of RHINOS. If we look at the training set, we find that an image contains on

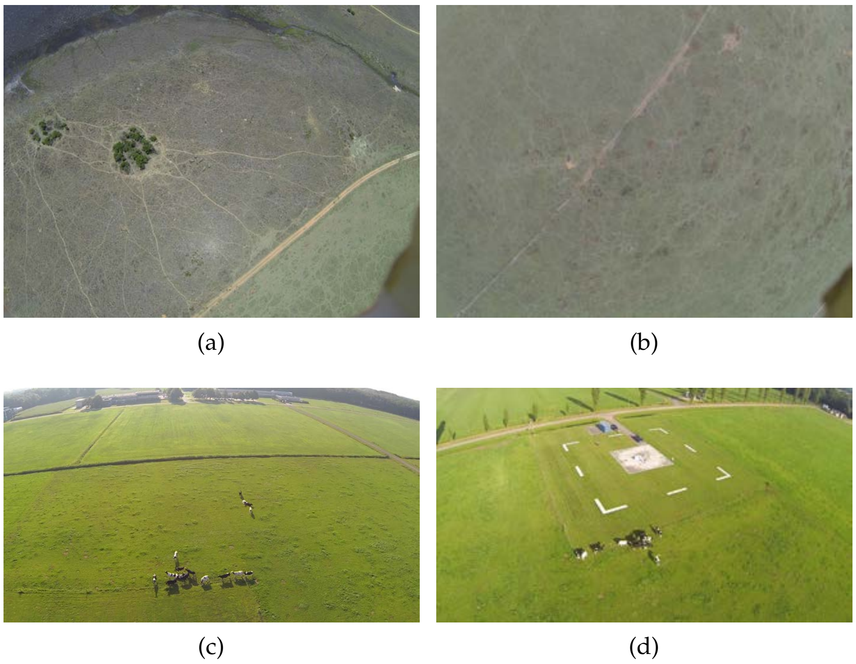


Figure 1: (a) (b) Images taken from the RHINOS testing set.
(c) (d) Images taken from the COWS testing set.

average 3.7 rhinos that cover 34×36 pixels each. We can estimate that in an average image from the RHINOS training set, only 0.07% of the pixels contain a rhino. These numbers could be even lower in practice, as this dataset was recorded specifically to obtain aerial images of the rhino. In Figures 1a and 1b we show two example images taken from the RHINOS testing set. Additionally, we use the Verschoor Aerial Cow dataset [9], which we will refer to as COWS. This dataset was used to perform an initial evaluation of the applicability of drones for the task of nature conservation. In Table 2 we provide the statistics for the COWS dataset along with two images taken from the testing set in Figures 1c and 1d. Again, we find that only a small percentage of the pixels, 0.81% in this case, contain a cow.

Ideally the UAV is capable of reducing the image to only show the objects that are relevant to the user. As we saw above, in the case of aerial imagery this means that over 99% of all pixels are irrelevant and could potentially be removed. Standard image compression techniques are not capable of applying this much irrelevance reduction, to do so the algorithms would require a better (visual) understanding of the world.

Once the data is available another challenge arises: processing the data. According to [7] reviewing only 500 images manually can take up to 45 minutes. A lot of efficiency can be gained if the data were

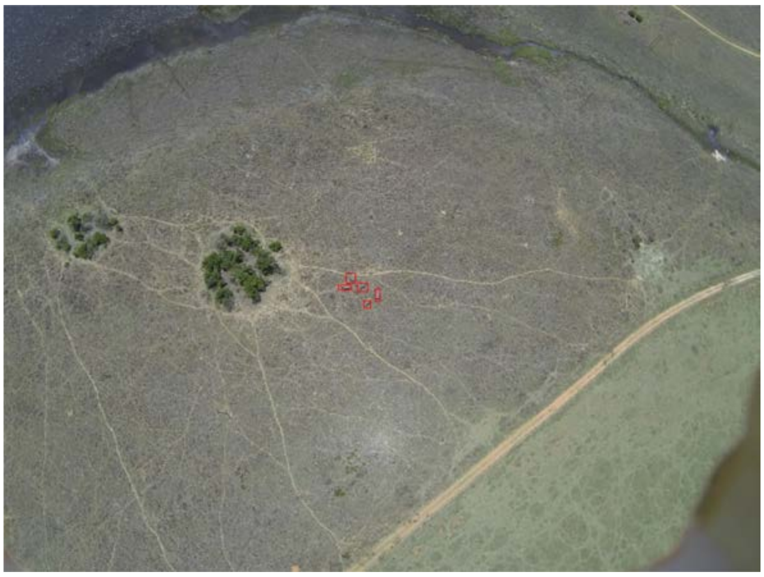


Figure 2: Example output of an automatic object detection algorithm on an image taken from the RHINOS dataset.

processed automatically. Automatic object detection algorithms reduce the time spent looking for the object of interest. Example output of such algorithms can be found for an image from the RHINOS testing set in Figure 2. Even though UAVs are often packed with on-board intelligence such as an autopilot, camera stabilization gimbal or object avoidance capabilities, automatic object detection algorithms on-board a UAV are still rare. State-of-the-art object detection algorithms may be computationally expensive or are not trained for detecting objects in aerial imagery.

To detect objects in an image a proposal method first generates a number of candidate boxes that are likely to contain an object [10]. These boxes are then evaluated by a proposal-based object detector which classifies each of these boxes to produce the final detections. An initial evaluation of using object detection algorithms for the task of nature conservation is performed in [9]. The authors find that the results of applying object detection algorithms on aerial imagery look promising for the task of nature conservation. However, the Selective Search [11] proposal method does not seem suitable to use on a conservation drone. This is due both to its detection rate (31 seconds per image) and the number of candidate boxes needed to achieve reasonable recall.

Proposal methods are used to speed up the detection rate of the object detector. However, with 31 seconds per image, Selective Search is significantly slower on aerial imagery than the 10 seconds per image reported by [12]. As can be seen in Table 2, the dimensions of images needed for the task of nature conservation are much larger than images found in popular benchmarks for object detection. For example, a typical image as provided by the PASCAL Visual Object Classes (VOC) Challenge [13] has dimensions 500×375. We expect that if we can reduce the image size, the detection rate of any proposal method will increase resulting in an overall speedup in the object detection pipeline.

This thesis concerns the problem of content-aware image resizing on a nature conservation drone. Content-aware image resizing is the task of resizing an image without compromising on the content. To do so, the content-aware image resizing methods removes pixels from the image that it finds irrelevant to the image content.

Applying a content-aware image resizing method to aerial imagery on a UAV has three major advantages. First the reduction in image size should allow any proposal method to generate candidate boxes faster. Second, due to the reduction in image size we expect the proposal method to generate fewer candidate boxes that need to be evaluated by the object detector resulting in another speedup in the object detection pipeline. Third, the resized image may be transferred to the ground station in a more timely manner, allowing the rangers to review the relevant data while the UAV is still in the air.

The main research question of this thesis than becomes: **How can we resize an image on a drone to increase the detection rate of the object detection pipeline without affecting detection performance?** In order to answer this question we will answer the following subquestions:

- How can we remove pixels from the image without compromising on the image content?
- Can we keep the recall of proposal methods constant while reducing the image size using a content-aware image resizing method?
- Can we increase the detection rate of the proposal method by resizing the image?

This thesis is organized as follows. In chapter 2 we provide an overview of previous work on content-aware image resizing and object detection. We then describe our solution to detect objects in aerial imagery on a nature conservation drone after resizing the images using our content-aware image resizing method in chapter 3. We perform several experiments to verify that using our method increases the detection rate without affecting the detection performance in chapter 4. Finally, we conclude with some final remarks and a direction for future work in chapter 5.

2

THEORY

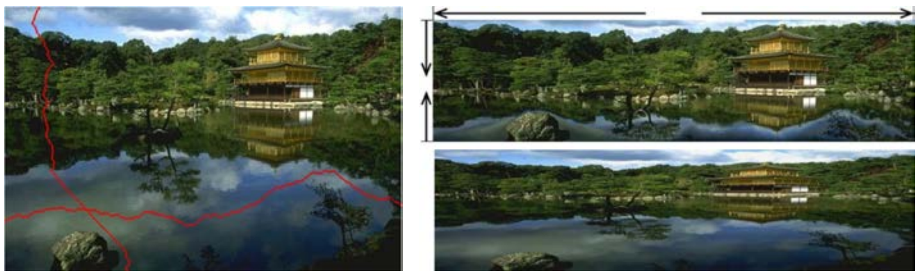
2.1 CONTENT-AWARE IMAGE RESIZING

With content-aware image resizing an image can be resized while preserving the image content. An example of content-aware image resizing using seam carving on an image from [14] can be found in Figure 3. Using content-aware image resizing parts of the image (in this case parts of the water and sky) are removed to preserve the most important image content. Content-aware image resizing may be applied to an image when the dimensions of the image that needs to be displayed are not known in advance. An example of this is the use of images on mobile websites, where the target dimensions differ across different mobile devices. Most of the work in content-aware image resizing (or: image retargeting) is focused on resizing the image for human viewers.

Various methods for content-aware image resizing are evaluated in [15]. In this study three objectives for a content-aware image resizing method are identified. First, the resizing method should preserve the content. Second, resizing the image should result in a new image with as few visual artifacts noticeable to a human viewer as possible. And finally, the method should preserve internal structures.

The methods reviewed in [15] can be grouped into two main categories. The first category contains continuous methods that warp an image to the target dimensions. This type of content-aware image resizing method does not seem interesting to the problem of resizing images on a conservation drones. The other category of content-aware image resizing consists of discrete methods that remove pixels from an image to resize it to the target dimensions. In section 2.1.1 we describe one of the most popular discrete content-aware image resizing methods: seam carving [14].

One of the outcomes of the evaluations in [15] is that discrete methods work best on images where irrelevant content such as sky, water or grass is present. In chapter 1 we established that most pixels in aerial images are irrelevant, the majority of pixels make up the landscape. Based on this observation discrete methods seem suitable for the task of resizing aerial imagery used for nature conservation. However, aerial imagery is very different from images such as the one



(a) (b)

Figure 3: Example of content-aware image resizing using seam carving from [14]. (a) Two seams that will be removed from the image by seam carving. (b) An image resized with seam carving (top) and the result of resizing the same image using regular rescaling (bottom).

shown in Figure 3a. To make content-aware image resizing suitable for aerial imagery, we make some modifications to seam carving resulting in our own content-aware image resizing method (which will be described in section 3.1).

2.1.1 *Seam Carving*

2.1.1.1 *Overview*

Seam carving [14] is a content-aware image resizing method that iteratively removes pixels from an image in order to resize it to a certain target size. The seam carving operator repeatedly removes 8-connected paths from the image that go from top to bottom or left to right. To preserve the image content, these paths, which are referred to as 'seams' should not contain pixels that are of importance to the image content. Figure 3a shows an example of two seams. We can determine the optimal seam by using an energy function that expresses the importance of pixels. Pixels that are important to the image content should result in high energy, whereas pixels that are not should result in low energy. In every iteration, the optimal seam to remove is the seam along which the total energy is lowest. The process of finding the optimal seam can be solved efficiently using dynamic programming. We will elaborate further on this in section 3.1.2. Figure 3b shows an example of an image resized with seam carving (top) and an example of resizing the image to the same aspect ratio using regular rescaling (bottom).

In [14] seam carving is proposed as a method of resizing images for human viewers. Therefore, it is important that any visual artifacts are minimized. The authors note that the total energy (the sum of the energy of every seam that was removed) removed from one image is affected by the order in which seams are removed. Even though a seam that has the lowest energy might seem like the optimal choice, by removing this seam from the image the energy of consecutive seams might turn out to be higher. Seams with higher energy naturally contain more important pixels and removing this seam will result in visible artifacts. To solve this problem the authors use dynamic programming to evaluate the total energy removed for every possible sequence of removing seams.

In [14] the authors tested several energy functions and concluded that there is no single best energy measure for all images, but different types of images require different energy functions. The authors initially use gradient magnitude as a simple example of an energy function that might be used. In other experiments they use various saliency methods as an energy function, including the output of face detectors to avoid removing faces from the image. In this thesis we propose and evaluate several energy functions in order to find the one that is most suitable to use on aerial imagery.

2.1.1.2 *Optimizations*

Seam carving can also be used for video retargeting as proposed in [16]. The two most important contributions of [16] are the introduction of forward energy and the move from using dynamic programing to graph cuts.

The first contribution of [16] is the introduction of forward energy. In [14] the cost of removing a seam was described as the total energy along that seam. Forward energy is a new method to compute the cost of a seam that eliminates the need to evaluate all possible orders in which seams can be removed. Removing a seam using forward energy means that the optimal seam to remove is the seam for which the total energy in the image after removing that seam, is lowest.

The second contribution of [16] concerns finding the optimal seam to remove efficiently. In videos a 3D seam should be removed, where the three dimensions are width, height and time. To do so efficiently, the authors substitute dynamic programming with graph cuts. The constructed graph connects all pixels to its neighboring pixels along the dimensions of the image, as well as throughout the different frames (the temporal direction). To find the optimal seam, we now look for the minimum cut through the graph.

Using graph cuts an efficient near real-time implementation of seam carving can be achieved. Graph cuts can be performed efficiently using a GPU [17]. Several variations of seam carving as described in [16] have been re-implemented using GPU parallelization [18] [19], indicating that real-time seam carving to retarget either images or videos can be achieved.

2.2 OBJECT DETECTION

Object detection is the task of classifying and localizing all objects in an image. The number and size of these objects may vary between images and in the image itself. To detect an object, proposal-based object detectors [20] [21] [22] evaluate a set of candidate boxes at different positions and different sizes. The object detector itself essentially is a classifier that classifies each of these candidate boxes and assigns a confidence score to every candidate box.

Object detection is often considered one of the hardest tasks of computer vision because it requires correct classification and localization of an arbitrary number of objects (which may also be absent) [23]. Additionally, it is computationally expensive because of the need to evaluate candidate boxes at every possible position and scale in the image. Dense sliding window methods generate a large number of these candidate boxes in an attempt to cover as much of the image as possible. However, evaluating all of these boxes is expensive and decreases the detection rate (the number of seconds it takes the object detector to evaluate one image). To increase the detection rate, proposal methods are used to generate a limited number of candidate boxes which are more likely to contain an object.

We describe several of these proposal methods in Section 2.3. In section 2.4 we describe various object detectors.

2.3 PROPOSAL METHODS

To detect an object it is necessary that the proposal method generates at least one candidate box that denotes the object. A dense sliding window method does not consider the image content and it therefore has to produce a large number of candidate boxes to increase the probability of recalling all objects. Once an object is missed by the proposal method, it will not be recovered further along in the pipeline.

The number of candidate boxes are related to the detection rate of the object detector; because the object detector evaluates all candidate boxes, the larger the number of candidates the lower the detection rate. Dense sliding window methods are very inefficient because of the large number of candidate boxes they produce.

To increase the detection rate, the use of proposal methods has become increasingly popular. In contrast to dense sliding window methods, proposal methods do consider the image content. These methods are build on the assumption that objects in general share the same or similar characteristics that make them stand out from the background. For example, an object has a closed boundary and it appears different from its surroundings [24]. Different proposal methods use different methods of deciding what makes an object.

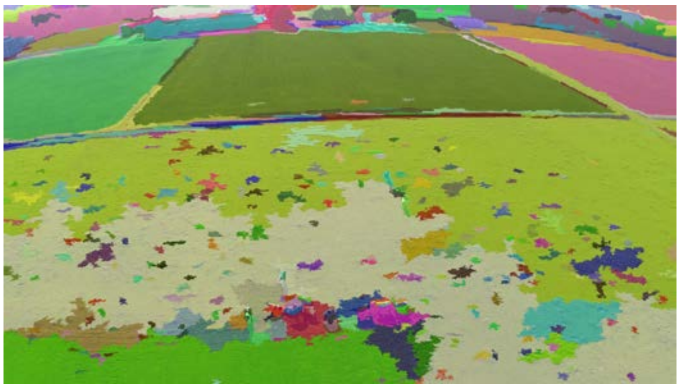


Figure 4: Visualization of the output from the segmentation algorithm proposed in [25] on an image from the COWS dataset (image is visible underneath the segmentation for clarity). Selective Search uses these regions to generate candidate boxes, it will continuously evaluate neighboring regions and merge them to create candidate boxes that correspond to different levels of the hierarchies that can be found in the image.

This enables them to reduce the number of candidate boxes needed to achieve good recall and only produce candidate boxes ('object proposals') that are likely to contain an object. In addition to this, the proposal method might produce a score along with the candidate boxes indicating the likeliness that the box contains an object.

A thorough evaluation and comparison of various proposal methods is provided by [10] and [12]. In this work ten proposal methods are analyzed by evaluating their performance through repeatability experiments. In these experiments the performance of the methods are monitored under different image transformation such as resizing, rotation, illumination changes and blurring. The experiments shows that most proposal methods suffer from even small scale changes. Amongst the top performing methods across all experiments are Selective Search and Edge Boxes, which are described in more detail below. Although the performance of BING strongly resembles a dense sliding window approach throughout most experiments, we also evaluate and describe BING because it is the only method that does not suffer significantly from changes in image size.

2.3.1 *Selective Search*

Selective Search is build on the observation that images are hierarchical: the appearance of multiple objects (sweater, arm, head) make up another object (person). To generate candidate boxes, Selective Search starts by segmenting the image using the fast segmentation algorithm proposed in [25]. The initial regions obtained by segmenting

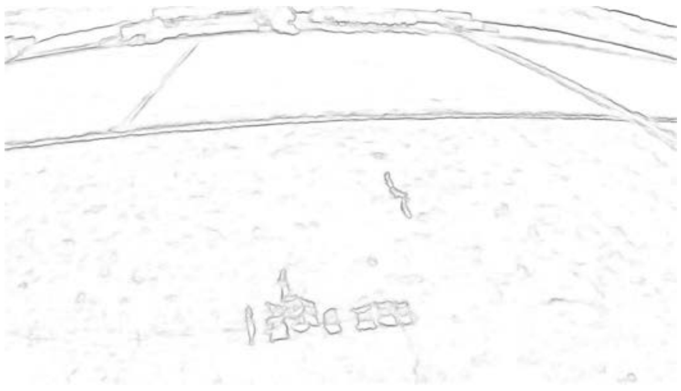


Figure 5: Output for the Structured Edge Detector [26] on an image from the COWS dataset. The Structured Edge Detector is used by Edge Boxes to evaluate candidate boxes produced by a sliding window approach.

the image represent the smallest objects or object parts in the image. Figure 4 provides a visualization of the output from the segmentation algorithm proposed in [25] on an image from the COWS dataset. The algorithm then proceeds to group these regions (based on similarity) to create bigger regions resulting in a number of different regions at different scales. The image regions at different scales are used to generate the candidate boxes. Additionally, the authors propose a 'quality' and a 'fast' setting which can be used depending on whether high quality candidate boxes are required, or quality can be traded for computational efficiency for a specific application.

On PASCAL VOC 2007 [13] the authors report a recall of 99% achieved with 2,134 boxes per image (generated while using the fast setting).

2.3.2 *Edge Boxes*

Edge Boxes [27] is a proposal method that generates candidate boxes from edges. The authors observe that edges often correspond to the boundaries of an object. Candidate boxes that contain an object thus contain a set of edges that are fully connected. The authors design an objectness score that favors boxes that enclose a set of connected edges over boxes enclosing disjoint edges. To generate a set of candidate boxes, this method first follows the sliding window approach. Boxes are scored by the sum of the strengths of all edges within the box minus the strength of the edges that extend across the bounding box. To compute the edges, the authors use the fast Structured Edge Detector [26]. Figure 5 shows the edges on images taken from the COWS dataset.

On PASCAL VOC 2007 the authors report a recall of 96% with an overlap threshold of 0.5 when using the 1,000 highest ranked candidate boxes. The average time per image in the PASCAL VOC 2007 dataset is reported as 0.25 seconds.

2.3.3 *BING*

BING [28] (Binarized Normed Gradients) is a proposal method that, like Edge Boxes, observes that objects are defined by having a closed boundary. To represent these boundaries, or edges, the authors use the gradient magnitude to assign an objectness score to a candidate box. The authors introduce the 64D normed gradient feature, that is obtained by scaling a candidate box to 8×8 and taking the norm of the gradient. In addition to this new feature, the authors propose an accelerated version; namely the binarized normed gradient. The use of the binarized normed gradient makes BING one of the fastest proposal methods. To produce the final objectness score, the normed gradient is combined with the location, size and the position of the candidate box.

On PASCAL VOC 2007 the authors report a recall of 96.2% with an overlap threshold of 0.5 when using the 1000 highest ranked candidate boxes. The average time per image in the PASCAL VOC 2007 dataset is reported as 0.003 seconds.

Even though BING delivers a minimal number candidate boxes with good recall within only 0.003 seconds per image, BING is not considered a good proposal method. In [29] it was shown that similar results can be achieved without looking at the image. BING assigns an objectness score based on the normed gradient, but also the location, size and position of the candidate box. It was shown that the influence of the normed gradient feature on the final objectness score is negligible.

2.4 DETECTION METHODS

We can divide object detectors into two categories: detectors that rely on carefully engineered features and detectors using features learned by Convolutional Neural Networks (CNNs, or ConvNets).

Early work in the field of object detection, such as the Viola Jones classifier [30] or detectors using HOG (Histogram of Oriented Gradients) features [31], fits the first category. The Viola Jones classifier uses a boosting algorithm to obtain a classifier using a large number of simple Haar filters. In [31], the authors were able to train a linear Support Vector Machine (SVM) on a grid of HOG features to classify pedestrians. The state-of-the art method for object detection using engineered features is the Deformable Part Model (DPM) [22]. This method views an object as a collection of parts, where each part is represented by HOG features.

In 2012 the ImageNet challenge [32] was won by Krizhevsky et al who applied a ConvNet to the task of image classification [33]. This deep ConvNet (with five convolutional layers) was trained on 1.2 million images, allowing it to learn very effective (hierarchical) features.

The features learned by a ConvNet can be used for several other tasks. In [23] a single ConvNet is used for the tasks of classification, localization and detection. To use one network for these different tasks, the authors introduce OverFeat: a feature extractor based on ConvNets. By using a feature extractor based on ConvNets, one could train a network on the task of image classification, but use the features (which we refer to as CNN features) that are learned by the convolutional layers for other tasks such as detection. The R-CNN [20] and Fast R-CNN [21] object detectors (both of which will be discussed below) are examples of object detectors using CNN features. The use of ConvNets for object detection has yielded an impressive increase in mean average precision (mAP) over the use of hand-engineered features: the mAP increased from 33.7 achieved by DPM to 66.9 achieved by Fast R-CNN.

2.4.1 *Regions with CNN features (R-CNN)*

Regions with CNN features (R-CNN) [20] is an object detector that uses convolutional features to evaluate regions proposed by Selective Search.

At test time, Selective Search is used to generate candidate boxes on the input image. For every one of these candidate boxes, CNN features are extracted. These features are used as input to an SVM that classifies the candidate box. For every object category that needs to be detected, the SVM outputs a score representing its confidence of classifying the candidate box as this object category. To produce the final detections, the authors apply non-maximum suppression (NMS) to the candidate boxes, meaning if two (or more) boxes overlap according to a given threshold, they choose to keep the box with the highest confidence score.

R-CNN is a supervised method, meaning that it should be trained using positive and negative examples. Because datasets where every object is annotated individually are scarce, the authors show that it is possible to train a ConvNet on the ImageNet classification task (that has 1.2 million images, with labels for 1000 categories) and fine-tune it on the task of object detection using PASCAL VOC 2007 that provides object annotations on 20 object categories [13]. Training the ConvNet on ImageNet has the advantage that the convolutional layers are able to learn effective features and fine-tuning afterwards increases performance on a specific task.

2.4.2 *Fast R-CNN*

R-CNN is succeeded by Fast R-CNN [21]. It improves upon R-CNN in two ways. First, it reverses the order in which CNN features are extracted. In R-CNN features were extracted for every candidate box separately. However, as many of these candidate boxes overlap, the same features would often be computed. Fast R-CNN first computes CNN features for the whole input image and then reuses these features for the different regions of interest (the candidate boxes).

The second improvement over R-CNN is the change from a multi-stage training to single-stage training. This is achieved by removing the SVM in favor of a softmax classifier and fine-tuning the network with a multi-task loss; learning the classification and the box regression at the same time. As a result of these two improvements, the time to train the VGG_CNN_M_1024 network [34] is reduced from 28 hours to 2 hours (14× faster) and the test time of 12.1 seconds per image is reduced to only 0.15 seconds per image (80× faster).

2.5 CLASS ACTIVATION MAPS

As mentioned in 2.1.1.1 various energy functions such as the gradient magnitude or the output of a face detector may be used by content-aware image resizing methods. In this thesis we use Class Activation Maps (CAMs) as an energy function. CAMs show which parts of the image a ConvNet uses to classify the image as a specific class.

To understand how a network classifies an image [35] proposes to visualize ConvNets. By visualizing different layers, they show that CNN features found in the higher layers are activated by specific (parts of) objects, such as faces and wheels. It seems that CNN features act as object detectors by itself. Based on this observation, a number of different methods started using ConvNets for weakly-supervised object localization. Supervised object localization needs a dataset where every object has been annotated separately. Using weakly-supervised learning, the object localization model learns to localize objects based only on image-level annotations.

In [36] it was shown that a ConvNet trained on image-level labels for the task of classification is able to localize objects as well. First the ConvNet is turned into a Fully Convolutional Network (FCN). An FCN does not contain any fully-connected layers which would normally use the CNN features to classify the image. After removing the fully-connected layers max pooling takes place on the feature activations to localize a single point on the object.

To improve upon the localizations of [36], [37] uses a global average pooling layer instead of a max pooling layer. While max pooling localizes the maximum activation, global average pooling localizes all activations. By using global average pooling the whole object can

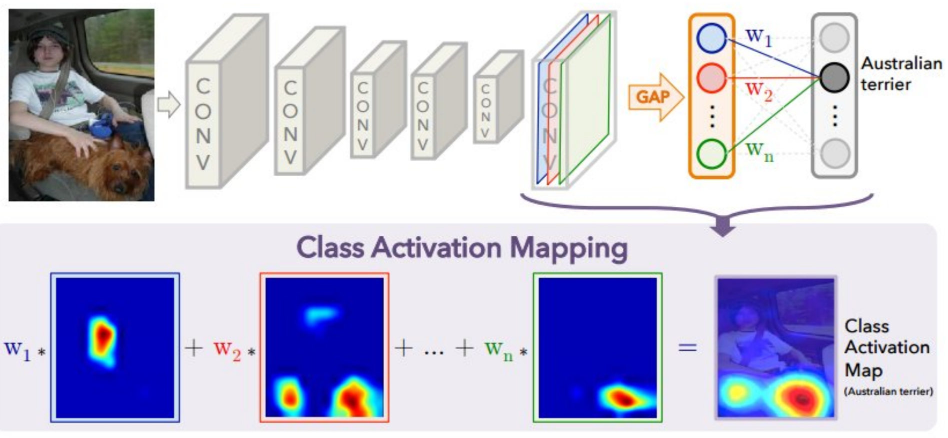


Figure 6: Illustration from [37] of the process to generate a Class Activation Map (CAM). CAMs show which parts of the image a ConvNet uses to classify the image as a specific class. It is generated by multiplying the weight for the specific class by the output of global average pooling layer (GAP) on the activations of the last layer.

be localized, instead of a single point on the object. To perform localization, a Class Activation Map (CAM) is generated. This map is generated by multiplying the weight for the specific class by the output of global average pooling layer on the activations of the last layer as illustrated in Figure 6. The map can be scaled up to cover the original image and shows what regions are most likely to contain the specific object class.

3 APPROACH

This chapter describes our solution to detect objects in aerial imagery on a nature conservation drone after resizing the images using our content-aware image resizing method. There are four steps that will be described in this chapter:

- **Content-aware image resizing** In section 3.1 we describe how we resize the image. This method removes the necessary pixels to scale the image down to a given scale factor.
- **Proposal Method** The resized image is used as input to a proposal method. This method generates a number of candidate boxes that are likely to contain an object. In section 3.2 we explore different proposal methods that could be used to generate candidate boxes on aerial imagery.
- **Reverse resizing candidate boxes** To be able to compare the candidate boxes with the ground truth, we reverse the resizing step. After we obtained candidate boxes from the resized image, we use the seams to transform the candidate boxes so we can find their place in the original image. The process of reverse resizing is described in section 3.3.
- **Object detection** The final candidate boxes are evaluated by the Fast R-CNN object detector [21] to produce the final detections as described in section 3.4.

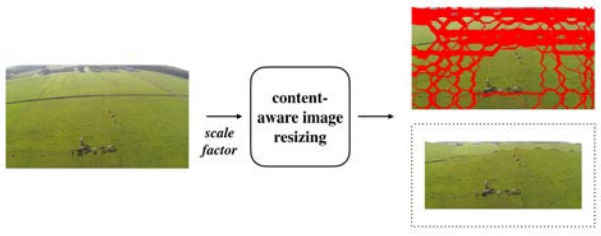


Figure 7: The content-aware image resizing method takes as input a scale factor and outputs the resized image.

3.1 CONTENT-AWARE IMAGE RESIZING

As can be seen in figure 7, this stage of the pipeline takes as input an image and a scale factor and outputs the resized image. The image is resized by applying a content-aware image resizing method inspired by seam carving [14]. This method removes the necessary number of pixels to scale the image down to a given scale factor.

The seam carving operator repeatedly removes 8-connected paths from the image that go from top to bottom, or left to right. To preserve the image content, these paths, which we refer to as 'seams', should not contain pixels that are of importance to the image content. To determine the importance of a pixel, we use an energy function that outputs high energy when the pixel is important and low energy when the pixel is irrelevant. This energy function is used to find the optimal seam to carve.

In this section we provide a detailed description of the content-aware image resizing method we use to resize the images. We first discuss what it means to resize an image to a certain scale factor followed by a description of how the optimal seams can be found. We then present several energy functions that we evaluated. Because our method is inspired by seam carving, we conclude by highlighting some important differences between our resulting implementation and the implementation described in [14].

3.1.1 *Scale Factor*

We want to be able to compare our content-aware image resizing methods to resizing methods that are not content-aware. Therefore, we will use the same definition of a 'scale factor' as used in regular image rescaling methods. To resize image I with scale factor r means that both the width I_w and height I_h will be scaled with r:

$$I_w^* = I_w \times r, \quad I_h^* = I_h \times r \tag{1}$$

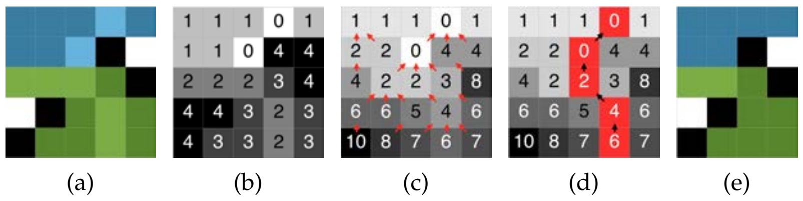


Figure 8: The seam carving algorithm: (a) original image (b) output of the energy function (c) accumulated energy (d) a seam to be deleted (e) result after one iteration.

where I_w^* and I_h^* are respectively the width and height of the image resized with scale factor r. The total number of pixels in the resized image I_p^* then becomes:

$$I_p^* = r^2 \times I_w \times I_h \tag{2}$$

The content-aware image resizing method will remove pixels from the image until the number of remaining pixels reaches I_p^*. This means that to resize an image with, for example, scale factor 0.5, 75% of the pixels need to be removed.

3.1.2 *Carving seams*

A seam is an 8-connected path that is not important to the image content. We can compute the cost of removing a seam s as:

$$c(s) = \sum_{(x,y)\in s} I_e(x,y) \tag{3}$$

where I_e denotes the energy image. The optimal seam s^* can then be defined as follows:

$$s^* = \arg\min_s c(s) \tag{4}$$

The optimal seam can be found efficiently by using dynamic programming. Dynamic programming is a method that reduces a large problem to a series of (smaller) sub-problems.

Let's first consider the case where we would like to remove vertical seams from an image. Figure 8 provides a visualization for the different steps needed to remove one vertical seam from the image depicted in figure 8a. After finding the energy image I_e (figure 8b), we continue by creating a table to hold the accumulated energy when going through the image from top to bottom. Starting at the second row, we define the accumulated energy for a pixel located at (x, y) as:

$$\begin{aligned} \text{acc}(x,y) = I_e(x,y) + \min(&\text{acc}(x-1,y-1), \text{acc}(x,y-1), \\ &\text{acc}(x+1,y-1)) \end{aligned} \tag{5}$$

We use dynamic programming and so the complex problem of finding the seam with the lowest accumulated energy, is reduced to finding small portions of the seam. In our case, this means that for every pixel (x, y), we find a pixel in the row above $(z, y-1)$ that results in the lowest accumulated energy at location (x, y). Following the dynamic programming method, we only store the lowest values in every row. In addition to the accumulated energy, we store a pointer to the previous pixel (the optimal sub-solution). When the table is completely filled every pixel in the last row holds the cost for the optimal seam *originating from that pixel*. Figure 8c shows a visualization of the accumulated energy table based on the energy image I_e as shown in figure 8b. The optimal seam s^* is found by backtracking starting from the lowest value in the last row of the accumulated energy table as shown in Figure 8d.

Because the most optimal seam does not necessarily have to be a vertical seam, we create two energy tables one as described above and one where energy is accumulated by going through I_e from left to right. This leaves us with one optimal vertical seam s_v^* and one optimal horizontal seam s_h^*. The length of a vertical seam will always equal the height of the image in this iteration and a horizontal seam's length equals the width of the image. To decide which seam to carve in this iteration, it is needed to compare the cost of vertical seam and horizontal seam. However, because the image width and the height may not be equal, simply comparing the cost of the seams might create a bias towards the shorter seam. Therefore, we normalize the costs for removing the horizontal and vertical seam by dividing the cost of both seams by the image height and width respectively:

$$c_v(s_v^*) = \frac{c(s_v^*)}{\text{height}}, \quad c_h(s_h^*) = \frac{c(s_h^*)}{\text{width}} \tag{6}$$

where $c_v(s)$ is the normalized cost of a vertical seam and $c_h(s)$ is the normalized cost of a horizontal seam. Now, the cost of the seams can be compared and the optimal seam s^* can be removed from the image. In addition to removing the seam from the image, it is also removed from the energy image I_e.

The process described above is repeated iteratively, until enough pixels were removed.

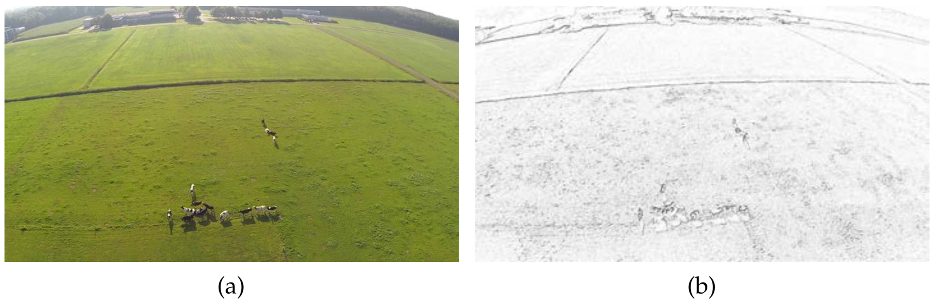


Figure 9: (a) The original image. (b) The gradient magnitude energy image, I_e. We find that the edges found in the original image become visible.

3.1.3 *Energy functions*

We apply an energy function to the image to capture the importance of a pixel to the image content. A good energy function assigns a higher value to important pixels. When an energy function is applied to an image, an energy image is created. The energy image is a visualization of the energy assigned to every image. In this section we provide a description of the three energy functions we evaluated for use on aerial imagery: gradient magnitude, ImageNet Class Activation Map and Aerial Class Activation Map.

3.1.3.1 *Gradient Magnitude*

The objective of seam carving as described in [14] is to remove pixels without leaving a noticeable effect. Among the many energy functions used to evaluate seam carving for content-aware image resizing is the gradient magnitude. The gradient magnitude of an image represents how much it changes. We can define importance of a pixel by the gradient magnitude. A pixel is important when at that point the image changes a lot. A pixel is unimportant if less change occurs in the image around that pixel.

To compute the gradient magnitude we first take the gradient of the image in both the x and y directions and sum the absolute value of these two gradients:

$$I_e = |\frac{\partial}{\partial x} I| + |\frac{\partial}{\partial y} I| \qquad (7)$$

where I is the image and I_e is the resulting energy image. Because the gradient magnitude of an image exposes how much an image changes, the resulting energy image I_e clearly shows the edges of the image. Figure 9 shows the gradient magnitude of an image from the COWS dataset.

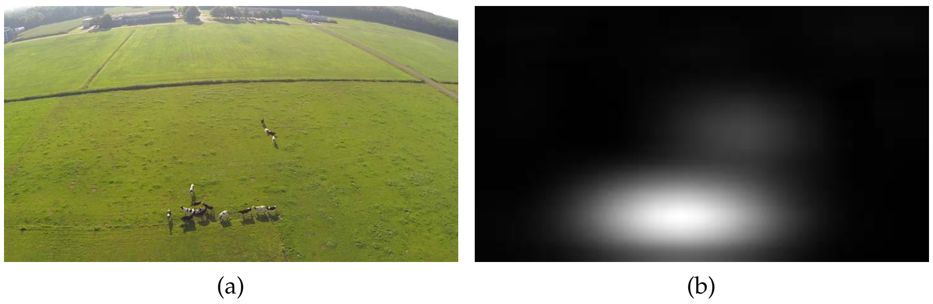


Figure 10: (a) The original image. (b) The ImageNet Class Activation Map energy image, I_e. We find that the areas around the cows light up, indicating these pixels contain the image content.

3.1.3.2 *ImageNet Class Activation Map*

A Class Activation Map shows which parts of the image the ConvNet used to classify the image as the specific class. Naturally, the image regions with the highest activation in the Class Activation Map are most likely to contain that specific object (if classification was done correctly). We use GoogLeNet-GAP from [37] that was trained on 1,000 ImageNet classes to generate a Class Activation Map for the top-1 prediction for the images in the COWS and RHINOS dataset. Neither cows nor rhinos are one of the 1,000 ImageNet classes, but the network could classify them as an animal that resembles a cow or rhino. We will use these Class Activation Maps as the energy images. Figure 10 shows the resulting energy image of using the ImageNet Class Activation Map as an energy function. We find that the areas around the cows light up, indicating these pixels contain the image content.

3.1.3.3 *Aerial Class Activation Map*

The GoogLeNet-GAP network was trained on 1,000 object categories. Neither cows or rhinos are amongst these objects. We expect to obtain better results using a network that predicts these objects. Therefore, we fine-tune the GoogLeNet-GAP network [37] for both of these datasets.

The GoogLeNet-GAP network was trained for the classification task on 1,000 ImageNet object categories. However, both our datasets only contain one object. After fine-tuning the network should distinguish between pixels containing that object (positive examples) or pixels containing the 'background' class (negative examples). Both the COWS and RHINOS dataset only contain images containing the objects. To be able to fine-tune this network we sample windows from the images. The sampled windows each have dimensions 224×224, the dimensions in which the network expects images. The bounding

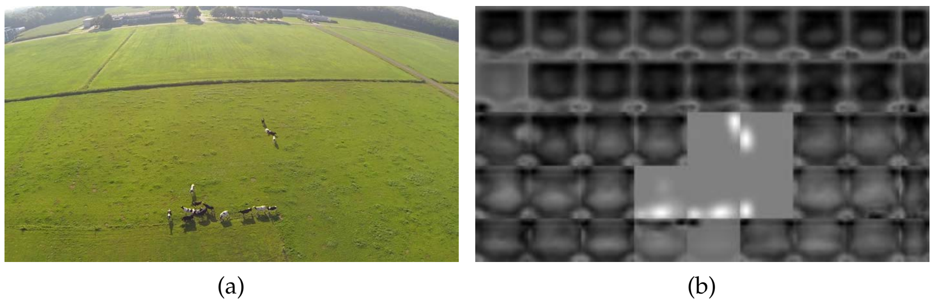


Figure 11: (a) The original image. (b) The Aerial Class Activation Map energy image, I_e. The image is divided into windows of 224×224. Each of these windows is then passed through the network to obtain the Class Activation Map for the top-1 prediction. Class Activation Maps for windows that were classified as 'background' are multiplied by -1. We find that the windows that contain cows light up as a whole compared to the windows containing the background.

box annotations are used to determine what windows contain the objects and represent a positive example, but after sampling the windows only image-level labels are used. We train a separate network for the two different datasets. We fine-tune the network for 1,600 iterations before it converges.

At test time we divide the image into windows of 224×224 (the same dimensions that were used to train the network). Each of these windows is then passed through the network to obtain the Class Activation Map for the top-1 prediction. We multiply the Class Activation Maps for windows that were classified as 'background' with -1. This ensures that pixels of which the network was most confident belong to the background get assigned the lowest energy. Figure 11 shows an energy image for an image from the COWS dataset generated with the Aerial Class Activation Map as described above. The different windows are still clearly visible. We find that the windows that contain cows light up as a whole compared to the windows containing the background.

3.1.4 *Modifications to the original seam carving algorithm*

In addition to using different energy functions than the original seam carving algorithm described in [14] we have made some other modifications.

First, we do not require the image to be resized to a certain aspect ratio. Instead, we merely want to reduce the number of pixels in the image. This means that instead of removing a fixed number of vertical and horizontal seams, we can remove the optimal seam regardless of its direction.

To avoid introducing visible artifacts into the image, the original seam carving algorithm searches for the optimal order of removing seams. However, in every iteration we remove one seam from the image, without considering the consequences. Because our method is not intended for human audiences the introduction of new energy into the image is not a problem. However, if the newly introduced energy is higher than the energy surrounding the important pixels, the algorithm might choose to remove important pixels rather than the pixels that produced a visible artifact. To counter this effect we make another modification to the original algorithm. Instead of re-computing the energy image after removing every seam, we simply remove the same seam from the energy image as well.

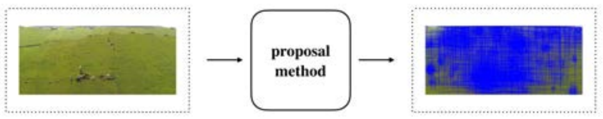


Figure 12: The resized image is used as input to a proposal method. This method generates a number of candidate boxes that are likely to contain an object.

3.2 proposal method

The resized image is used as input to a proposal method (figure 12). This method generates a number of candidate boxes that are likely to contain an object.

In choosing a proposal method for the task of object detection on a conservation drone we have three requirements. First, the proposal method must have high recall on aerial imagery, because objects that are missed in this stage will not be recovered further along in the pipeline. Second, we require the proposal method to have a potential to be used in real-time applications, using as little computational resources as possible. And third, the most important increase in detection rate by using proposal methods comes from the reduction of candidate boxes that need to be evaluated, thus we require the proposal method to achieve high recall with as little candidate boxes as possible.

[10] and [12] offer an extensive comparison of different proposal methods. However [9] finds that results on standard computer vision datasets do not necessarily translate to aerial imagery. To select a proposal method most suited to aerial imagery we perform some small experiments described in section 4.3.1.

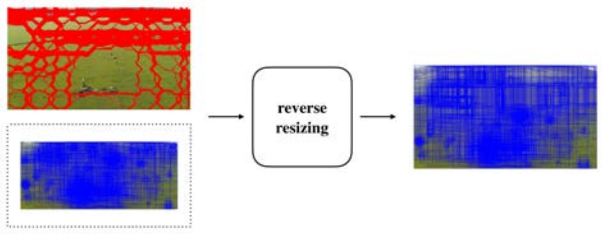


Figure 13: After we obtained candidate boxes from the resized image, we use the seams to transform the candidate boxes so we can find their place in the original image.

3.3 REVERSE RESIZING CANDIDATE BOXES

To be able to compare the candidate boxes with the ground truth, we reverse the resizing step. As shown in figure 13; after we obtained candidate boxes from the resized image, we use the seams to transform the candidate boxes so we can find their place in the original image.

The goal of the reverse resizing step is to transform the position of the candidate boxes from the resized image to a position in the original image. Doing so will allow us to compare the candidate boxes to the ground truth.

We reverse the resizing step by re-inserting seams that were removed from the image. Every time a seam is re-inserted this affects the position of some candidate boxes. By keeping track of the transformations caused by the re-inserted seams, we can compute the position of the candidate boxes in the original image. Just like when we were resizing the image, straight lines may not be preserved. As a result, the rectangular shape of the candidate boxes in the resized image may not be preserved after reverse resizing. Since the object detection step expects candidate positions to be rectangular boxes, we reconstruct the rectangular shape of the candidate box.

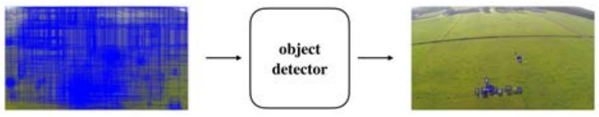


Figure 14: The final candidate boxes are evaluated by the Fast R-CNN object detector [21] to produce the final detections.

3.4 OBJECT DETECTION

The transformed candidate boxes are used as the input to the Fast R-CNN object detector [21] as can be seen in Figure 14. For both datasets we fine-tuned Fast R-CNN using candidate boxes generated by Edge Boxes [27] (no resizing was applied while training the models). In [21] the authors fine-tune small, medium and large pre-trained ImageNet models to perform object detection on PASCAL VOC 2007.

We selected the medium model, because it improves upon the small model, but does not require specialized hardware like the large model. The medium model is the VGG_CNN_M_1024 network from [34]. We used the fine-tuned network from [21] and fine-tuned it for both the COWS and RHINOS dataset for 2500 iterations (after which performance converged). The other parameters are identical to the parameters that were used to fine-tune the model for PASCAL VOC 2007.

4 RESULTS

In this chapter we present the experiments to answer the main research question of this thesis: **How can we resize an image on a drone to increase the detection rate of the object detection pipeline without affecting detection performance?** To answer this question we perform a number of experiments to verify that:

- although pixels are removed from the image, the ground truth stays more or less intact (section 4.2)
- the recall of the proposal method is minimally affected when resizing the image (section 4.3)
- using our content-aware image resizing method increases the detection rate both by increasing the speed with which the proposal method generates candidate boxes and by decreasing the number of candidate boxes that need to be evaluated by the object detector (section 4.3)
- the performance of the object detector remains stable under different scale factors (section 4.4).

4.1 EXPERIMENTAL SETUP

In this section we describe the datasets used to evaluate the effect of content-aware image resizing to detection performance, we discuss the different evaluation metrics that are used and recap the different methods that are evaluated.

4.1.1 *Dataset*

We perform experiments on two datasets. The fist dataset is the Birds.ai Rhinos Dataset, which we refer to as RHINOS. It contains 169 images in its training set and 168 images in its testing set. The images are 3000×2250. Table 1 shows more statistics of RHINOS. The second dataset is the Verschoor Aerial Cow Dataset [9], which we refer to as COWS. The dataset is split in a training set of 141 images and a testing set of 127 images. The images are 1920×1080. Table 2 shows more statistics of COWS.

4.1.2 *Evaluation Metrics*

This section provides a brief overview of the main metrics used to evaluate proposal methods and object detectors.

4.1.2.1 *True Positives, False Negatives and False Positives*

Detected objects are called *true positives* (TP), whereas objects that were missed by the proposal method or object detector are called *false negatives* (FN). The object detector might also produce detections that do not correspond to any object in the ground truth, which are called *false positives* (FP).

4.1.2.2 *Intersection over Union Threshold*

To determine whether an object is a true positive or a false negative the detection is compared to a ground truth box. If the intersection over union (IoU) of a detection with the ground truth box is greater or equal to a pre-determined IoU threshold, the detection is marked as a true positive. The intersection over union of a ground truth box B_{gt} and the box denoting the detection B_d is computed as:

$$\mathrm{IoU} = \frac{area(B_d \cap B_{gt})}{area(B_d \cup B_{gt})} \tag{8}$$

4.1.2.3 *Recall*

Recall represents the fraction of objects in the dataset that was detected by the proposal method or the object detector. Recall is computed as:

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{9}$$

4.1.2.4 *Precision*

Precision represents the fraction of detected objects that correspond to objects in the ground truth. Precision is computed as:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{10}$$

4.1.3 *Methods*

This section provides a brief overview of the four image resizing methods used throughout all evaluations.

4.1.3.1 *Baseline*

To assess the added value of considering the content when resizing the image, we compare our content-aware image resizing methods to an image rescaling method that is not content-aware. As a baseline, we use image rescaling with bilinear interpolation.

4.1.3.2 *CAIR GM*

We use CAIR GM to refer to the the content-aware image resizing method as described in section 3.1 using the gradient magnitude as the energy function described in section 3.1.3.1.

4.1.3.3 *CAIR ICAM*

We use CAIR ICAM to refer to the the content-aware image resizing method as described in section 3.1 using the ImageNet Class Activation Map as the energy image, as described in section 3.1.3.2.

4.1.3.4 *CAIR ACAM*

We use CAIR ACAM to refer to the the content-aware image resizing method as described in section 3.1 using the Aerial Class Activation Map described in section 3.1.3.3 as the energy image.

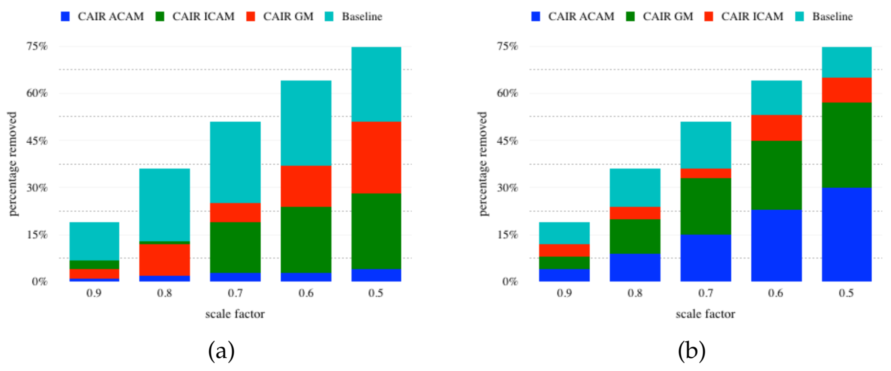


Figure 15: Comparison of the percentage of important pixels that were removed from the image by our methods to the percentage of important pixels removed by the baseline. All content-aware image resizing methods remove fewer important pixels from the image than the baseline. CAIR ACAM is the best performing method, removing fewest important pixels from images of both datasets. (a) Results on the COWS dataset. (b) Results on the COWS dataset.

4.2 EVALUATION OF SEAM CARVING

This section verifies that although pixels are removed from the image, the ground truth stays more or less intact. In section 4.2.1 we compare the percentage of important pixels removed from the image by our content-aware image resizing methods to the percentage of important pixels removed by the baseline. In section 4.2.2 we show some examples of images resized by all methods.

4.2.1 *Quantitative*

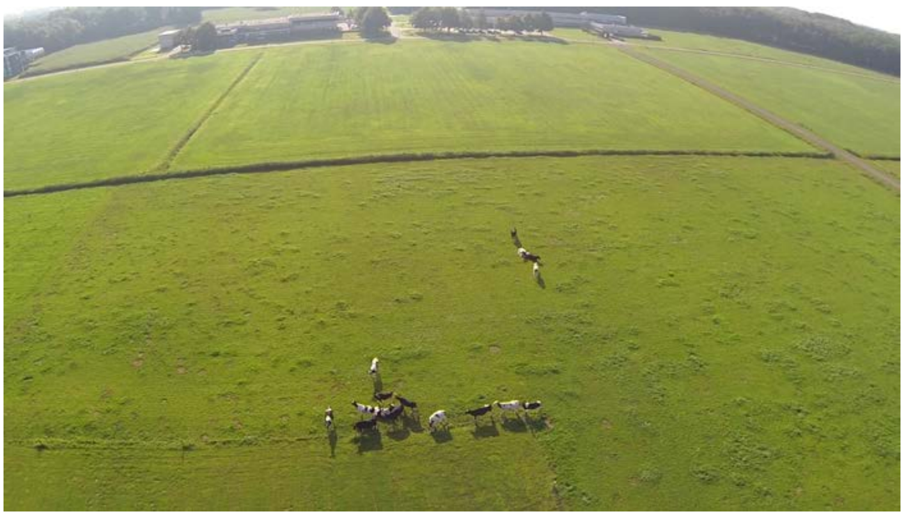
To evaluate to what extent the content-aware image resizing method is only removing irrelevant pixels, we compare the percentage of important pixels removed from the image by our methods to the percentage of important pixels removed by the baseline. In this case, 'important pixels' are pixels that are contained by one or more ground truth boxes. Figure 15a and 15b show the results for the COWS and the RHINOS dataset respectively. We find that all our methods remove fewer important pixels from the image than the baseline. This indicates that the content-aware image resizing methods are indeed avoiding the removal of pixels that are important to the image content. CAIR ACAM is the best performing method, removing fewest important pixels from images of both datasets. We observe that the RHINOS dataset appears to be more challenging than the COWS dataset.

More important pixels were removed for all scale factors from images from the RHINOS dataset than images from the COWS dataset.

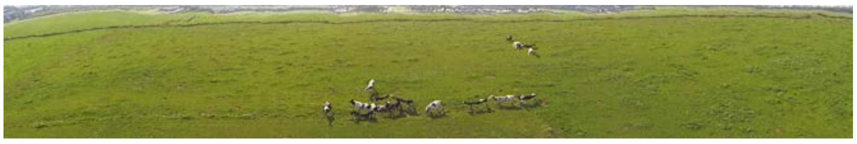
4.2.2 *Qualitative*

We can also verify the content-awareness of the resizing method by looking at the resized images themselves. Figure 16 shows an image from the COWS dataset resized with scale factor 0.5. Although 75% of the pixels were removed, the cows remain fully visible and seem largely unaltered. The results of CAIR GM are shown in Figure 16b, we find that large parts of the meadow were removed. CAIR ICAM removes most of the surroundings of the cows, except for two small patches of background clutter as can be seen in Figure 16c. CAIR ACAM removes the background as well, including the shadows of some cows as can be seen in Figure 16d.

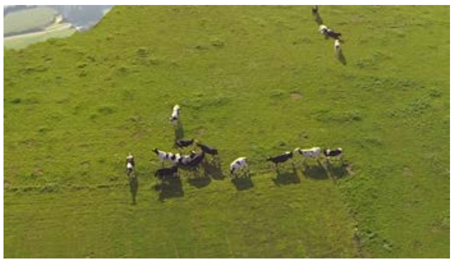
Figure 17 shows an image from the RHINOS dataset resized with scale factor 0.5. In the image from the RHINOS dataset multiple things stand out. First, in the original image (Figure 17a) the bottom right corner shows part of the UAV, this is completely removed in the resized image by CAIR GM (Figure 17b) and CAIR ICAM (Figure 17c). We also find that parts of the water were removed by CAIR GM, especially the stream that runs land inwards has been reduced significantly. CAIR ICAM even removes all water from the scene. Lastly, large portions of the road were removed. We find that due to the removal of pixels throughout the image by CAIR GM the direction of the road has been altered. When resizing the image with CAIR ACAM (Figure 17d) the road is removed almost completely.



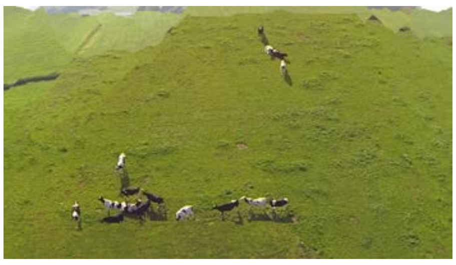
(a)



(b)

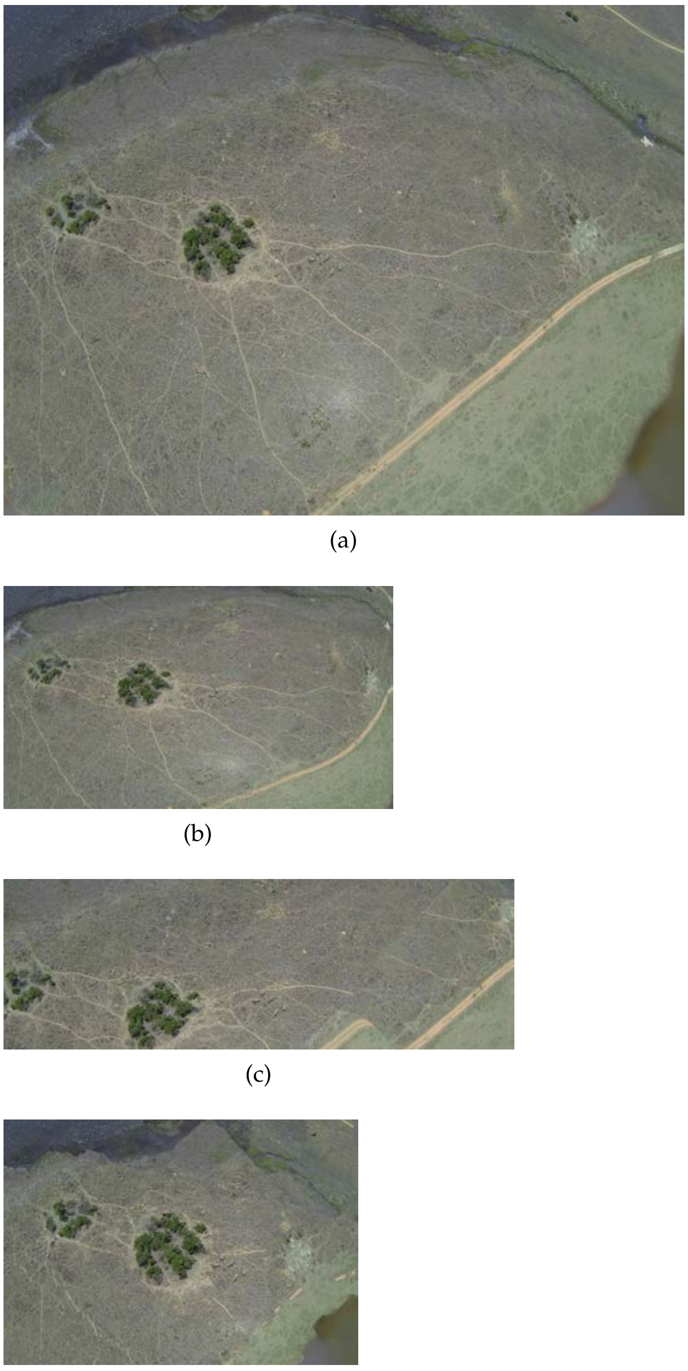


(c)



(d)

Figure 16: Image from the COWS dataset resized with scale factor 0.5. Although 75% of the pixels were removed, the cows remain fully visible and seem largely unaltered. (a) Original image. (b) Image resized with CAIR GM, large parts of the meadow were removed. (c) Image resized with CAIR ICAM, most of the surroundings of the cows are removed, except for two small patches of background. (d) Image resized with CAIR ACAM, most of the surroundings of the cows are removed including the shadows of some cows.



(a)

(b)

(c)

(d)

Figure 17: Image from the RHINOS dataset resized with scale factor 0.5. Although 75% of the pixels were removed, the rhinos remain fully visible and seem largely unaltered. (a) Original image. (b) Image resized with CAIR GM. Unimportant pixels such as the part of the UAV in the bottom right corner of the original image, were rightfully removed. (c) Image resized with CAIR ICAM, all water has been removed from the image. (d) Image resized with CAIR ACAM, most of the road has been removed.

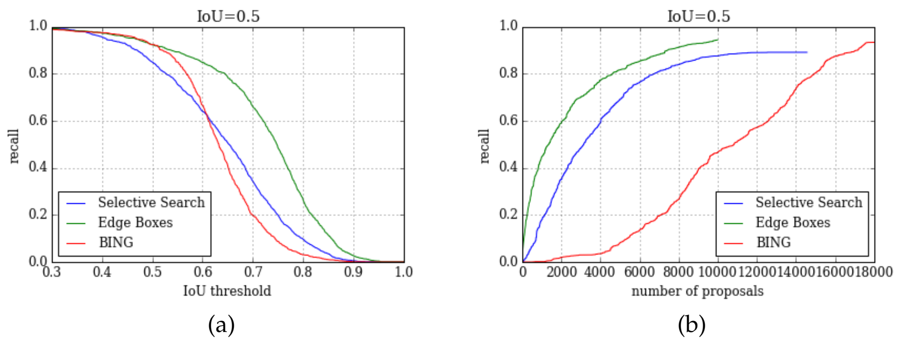


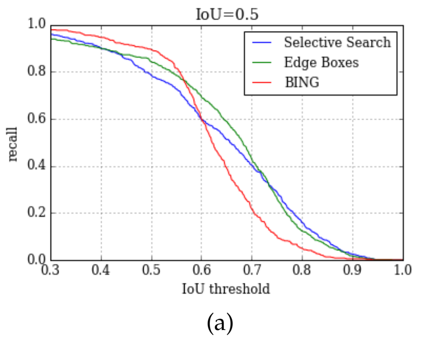
Figure 18: Comparing Selective Search (fast setting), Edge Boxes and Bing on recall on the COWS dataset. (a) Recall over IoU (intersection over union) thresholds for all candidate boxes. (b) Recall over the number of candidate boxes for an IoU of 0.5.

4.3 EVALUATION OF PROPOSAL METHOD

In this section we evaluate the proposal method that is used to generate candidate boxes that are likely to contain an object. In order to select a method we compare the performance of several proposal methods in section 4.3.1. We find that Edge Boxes outperforms Selective Search and BING on both the COWS and RHINOS dataset. In section 4.3.2 we then continue and evaluate the recall of Edge Boxes on resized images to evaluate to what extent resizing the image affects the recall. Finally, we report the speed with which Edge Boxes generates boxes on resized images and the number of candidate boxes that were generated in section 4.3.3.

4.3.1 *Proposal method*

We compared Selective Search (fast setting) [11], Edge Boxes [27] and BING [28]. Selective Search has been used by numerous object detectors [20] [21] because it produces high quality proposals. Edge Boxes, like Selective Search, offers great detection performance, but is much faster than Selective Search [10]. BING is an efficient proposal method that was found to be more robust under different image transformations than both Selective Search and Edge Boxes, however the proposals are of lesser quality. In Figures 18 and 19 we compare the performance of the proposal methods on COWS and RHINOS respectively. Figures 18a and 19a show the recall for different IoU thresholds for COWS and RHINOS respectively. Here, the recall was computed by using all candidate boxes. We find that Edge Boxes outperforms both Selective Search and BING. Based on these results



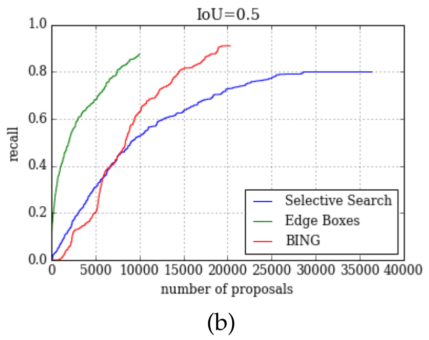


Figure 19: Comparing Selective Search (fast setting), Edge Boxes and Bing on recall on the RHINOS dataset. (a) Recall over IoU (intersection over union) thresholds for all candidate boxes. (b) Recall over the number of candidate boxes for an IoU of 0.5.

	COWS	**RHINOS**
Selective Search (fast setting)	27.05	111.07
Edge Boxes	3.62	6.74
BING	1.77	4.20

Table 3: Comparing the speed of Selective Search (fast setting), Edge Boxes and BING.

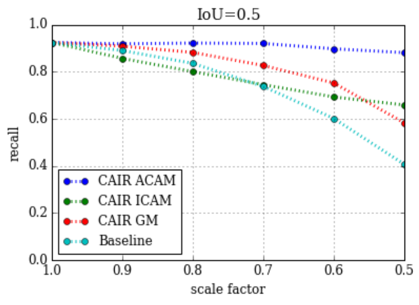
Edge Boxes makes the best choice of proposal method for the task of object detection in aerial imagery.

Figures 18b and 19b shows the recall over the number of candidate boxes for a fixed IoU of 0.5 for COWS and RHINOS. Selective Search and Edge Boxes offer better recall at a smaller number of candidate boxes. We also compare the evaluation time per frame for the three proposal methods. The results can be found in table 3. As expected, BING provides fastest performance, spending on average only 1.77 seconds per image. Like in [10] we find that Edge Boxes provides a good trade-off between efficiency and performance.

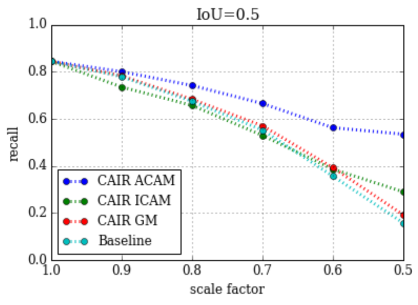
4.3.2 *Recall*

To assess the added value of considering the content when resizing the image, we compare our content-aware image resizing methods to the baseline which uses bilinear interpolation. Figure 20 shows the recall for different scale factors when the intersection over union is fixed at 0.5.

For the COWS dataset, we find that rescaling the image using CAIR ACAM results in only a slight drop in recall from 0.92 for the original image to 0.88 when resized with scale factor 0.5. This is a big improvement when we compare it to the recall achieved when resiz-



(a)



(b)

Figure 20: Recall of Edge Boxes after resizing the image with CAIR against the non content-aware baseline. (a) Results for the COWS dataset. The best performing method, CAIR ACAM, results in only a slight drop in recall. CAIR GM also offers an improvement over the baseline for all scale factors. However, the results for CAIR ICAM indicate the Class Activation Map of the top-1 prediction of the ImageNet object categories does not provide enough details to accurately describe the importance of the different pixels. (b) The RHINOS dataset is more challenging and shows only a slight improvement between resizing with CAIR ICAM or CAIR GM and the baseline. However, CAIR ACAM shows an improvement over the baseline. CAIR ICAM closely follows the baseline showing the ImageNet object categories do not provide to be accurate enough to generate a Class Activation Map that accurately identifies important pixels.

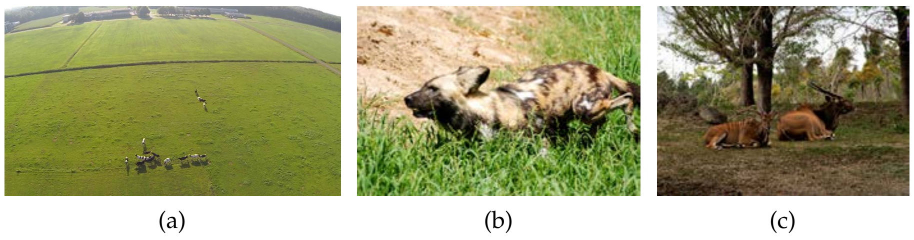


Figure 21: (a) Image from the COWS dataset. (b) Image from the 'wild dog' ImageNet sysnet. (c) Image from the 'gazelle' ImageNet sysnet.

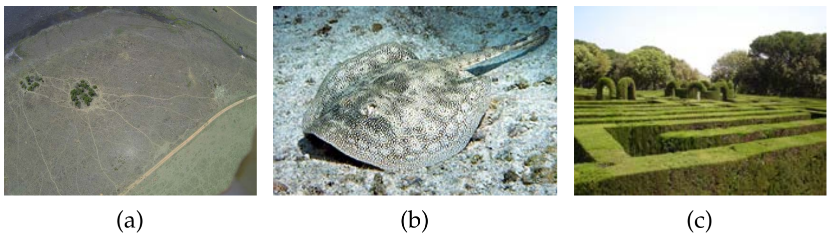


Figure 22: (a) Image from the RHINOS dataset. (b) Image from the 'stingray' ImageNet sysnet. (c) Image from the 'labyrinth' ImageNet sysnet.

ing the image with scale factor 0.5 using the baseline, which causes the recall to drop to 0.41. CAIR GM also offers an improvement over the baseline, with a final recall of 0.58 when resizing with scale factor 0.5. CAIR ICAM closely follows the baseline up until scale factor 0.7, after which it shows some improvement.

Although the initial energy images looked promising, using the Class Activation Map of the top-1 prediction of the ImageNet object categories does not provide enough details to accurately describe the importance of the different pixels. The top-1 predictions used to generate Class Activation Maps for images in the COWS dataset contain the 'wild dog' as well as the 'gazelle' classes. Both of these classes might resemble the cows we want to detect (Figure 21 shows a comparison of these classes), but the results show these predictions do not offer the accuracy needed to preserve the image content.

In contrast to the COWS dataset the predictions on images from the RHINOS dataset do not resemble rhinos. Figure 22 shows an image from the RHINOS dataset next to an example of the 'stingray' and 'maze' ImageNet classes that were used to generate the Class Activation Maps. As a result CAIR ICAM closely follows the baseline. The RHINOS dataset is more challenging and shows only a slight improvement between resizing with CAIR GM and the baseline. However, CAIR ACAM shows an improvement over the baseline. Using CAIR ACAM we observe a drop in recall from 0.85 for the original

	0.5	0.6	0.7	0.8	0.9	Original
Seconds per image	1.05	1.47	1.87	2.64	2.76	3.62
CAIR GM	3824	5033	6458	7664	8487	9333
CAIR ICAM	4304	5792	7335	8498	9052	9333
CAIR ACAM	5666	6712	7687	8237	8661	9333
Baseline	3641	4892	6317	7661	8605	9333

Table 4: Average number of candidate boxes generated by Edge Boxes on the COWS dataset and the average time it takes to generate the candidate boxes for images resized with different scale factors.

	0.5	0.6	0.7	0.8	0.9	Original
Seconds per image	1.60	2.53	3.74	4.71	5.91	6.75
CAIR GM	2994	4362	5845	7237	8249	8893
CAIR ICAM	4245	5340	6318	7290	8092	8893
CAIR ACAM	7149	7715	8157	8536	8787	8893
Baseline	4788	6091	7160	8020	8518	8893

Table 5: Average number of candidate boxes generated by Edge Boxes on the RHINOS dataset and the average time it takes to generate the candidate boxes for images resized with different scale factors.

image to 0.54 when resized with scale factor 0.5, which improves the baseline by 0.37.

4.3.3 *Detection rate*

We expect that applying content-aware image resizing increases the detection rate in two ways. First, we expect that on smaller images, the proposal method is able to generate candidate boxes faster. Second, we expect fewer candidate boxes to be generated on smaller images. A reduction in candidate boxes means a reduction of boxes to be evaluated by the object detector, immediately increasing the detection rate.

In Table 4 we show the number of candidate boxes (generated by Edge Boxes) for different image resizing methods and scale factors as well as the time it takes to generate these boxes for the COWS dataset. Resizing the image indeed leads to a reduction in the number of candidate boxes generated and the time it takes to generate them. Resizing the image with CAIR GM results in only 3823 candidate boxes when resized with scale factor 0.5, a reduction of approximately 60% compared to the original image (9333 candidate boxes). Even for CAIR ACAM, the number of candidate boxes can be reduced to 5666 boxes (approximately a 40% reduction). We observe

	0.5	0.6	0.7	0.8	0.9	Original
CAIR GM	0.34	0.40	0.45	**0.47**	**0.47**	0.42
CAIR ICAM	0.35	0.35	0.39	0.40	0.40	0.42
CAIR ACAM	**0.43**	**0.46**	**0.48**	0.46	0.46	0.42
Baseline	0.26	0.33	0.42	0.40	0.45	0.42

Table 6: Average precision of the Fast R-CNN object detector on detecting cows in the COWS dataset when the image was resized using different resizing methods and scale factors. CAIR GM and CAIR ACAM improve on the average precision achieved by the baseline for all scale factors. The average precision when resizing the image with CAIR ACAM shows an improvement over the original image, even when the image is resized with scale factor 0.5.

some variation between the number of candidate boxes generated by different methods for the same scale factor. Again, looking at scale factor 0.5 the number of candidate boxes generated can be as low as 3824 and as high as 5666. We expect this happens because the resized image clearly contains a number of objects and therefore the proposal method was able to 'make more sense' of the content. Edge Boxes takes only 1.05 seconds to generate candidate boxes for an image resized with scale factor 0.5, this is approximately 70% faster than on the original image.

Table 5 shows the results for the RHINOS dataset. Resizing the image with CAIR GM results in only 2994 candidate boxes when resized with scale factor 0.5, a reduction of approximately 65% compared to the original image (8893 candidate boxes). Even for CAIR ACAM, the number of candidate boxes can be reduced to 7149 boxes (approximately a 20% reduction). Edge Boxes needs 1.60 seconds to generate candidate boxes for an image resized with scale factor 0.5, this is approximately 75% faster than on the original image.

4.4 EVALUATION OF OBJECT DETECTION

Using the candidate boxes that were evaluated in the previous section, we now evaluate the performance of the Fast R-CNN object detector [21].

Table 6 show the average precision on the COWS dataset. CAIR GM and CAIR ACAM improve on the average precision achieved by the baseline for all scale factors. The average precision when resizing the image with CAIR ACAM shows an improvement over the original image, even when the image is resized with scale factor 0.5. Table 7 provides the results on the RHINOS dataset. Again: CAIR GM and CAIR ACAM either improve or do not affect the average precision achieved by the baseline for all scale factors. The average precision

	0.5	0.6	0.7	0.8	0.9	Original
CAIR GM	**0.15**	**0.16**	**0.16**	**0.16**	**0.15**	0.15
CAIR ICAM	0.10	0.12	0.13	0.13	**0.15**	0.15
CAIR ACAM	**0.15**	0.15	0.15	0.15	**0.15**	0.15
Baseline	0.12	0.14	0.15	0.15	0.14	0.15

Table 7: Average precision of the Fast R-CNN object detector on detecting rhinos in the RHINOS dataset when the image was resized using different resizing methods and scale factors. CAIR GM and CAIR ACAM either improve or do not affect the average precision achieved by the baseline for all scale factors. The average precision when resizing the image with CAIR ACAM with scale factor 0.5 equals the average precision achieved on the original image.

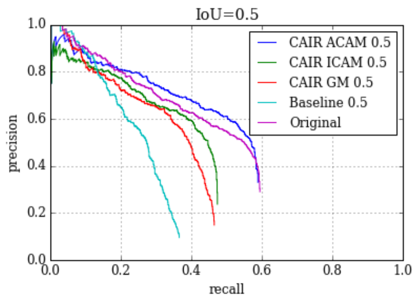
when resizing the image with CAIR ACAM with scale factor 0.5 is equal to the average precision achieved on the original image.

For both datasets we find the average precision achieved after resizing the image with CAIR GM and CAIR ACAM shows some improvement over using the original image for certain scale factors.

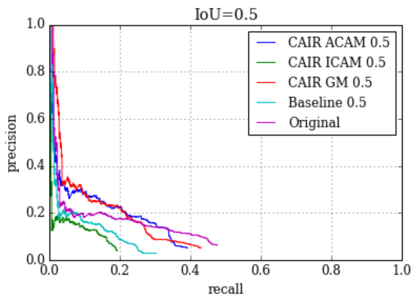
A possible explanation for the increase of performance of the object detector might be the fact that resizing the image leads to fewer candidate boxes. This means that there are fewer false positives amongst the candidate boxes that are evaluated by the object detector, resulting in better precision. This effect has been described in [10] as one of the benefits of using proposal methods over the dense sliding window method.

In figure 23 we provide the precision-recall curves for both datasets after resizing the image with scale factor 0.5 using different image resizing methods and Fast R-CNN as the object detector.

Like the average precision, when resizing images from the COWS dataset with CAIR ACAM the precision-recall curve also shows an improvement over using the original image (Figure 23a). Both CAIR ICAM and CAIR GM outperform the baseline, but suffer from a drop in recall as observed in section 4.3.2. Figure 24 shows true positive and false positives detections on the original image as well as the detections on the same image resized using CAIR ACAM with scale factor 0.5. In the resized image most of the cows were detected while the remaining false positives now appear much closer to the cows rather than at the top as was the case in the original image (Figure 24b). The remaining false positive detections are localization errors whereas the false positive detection in the original image is a more serious mistake.



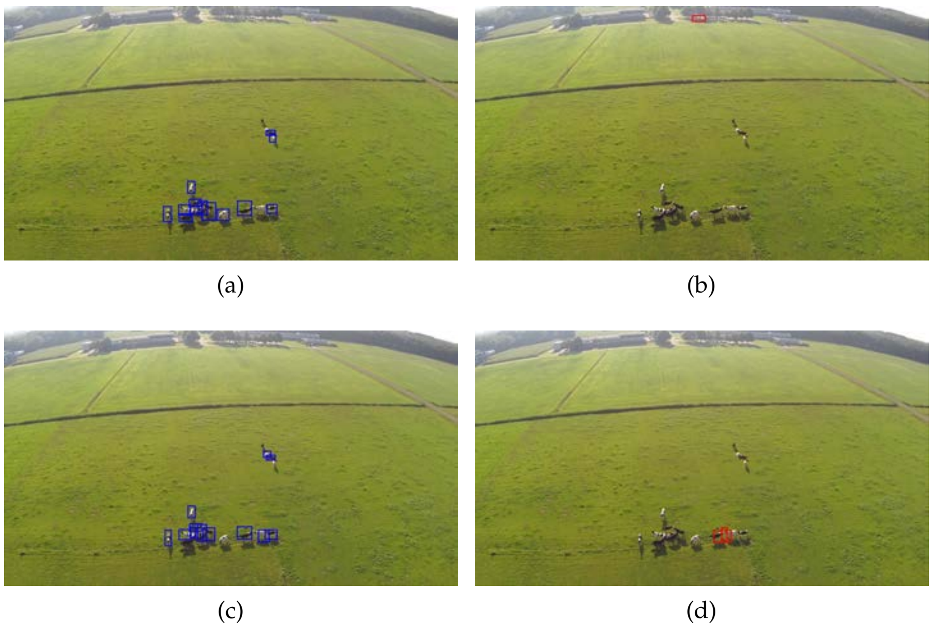
(a)



(b)

Figure 23: Comparison of precision-recall curves after resizing the image with scale factor 0.5 using different image resizing methods and Fast R-CNN as the object detector. (a) Results on the COWS dataset. Resizing images with CAIR ACAM shows an improvement over using the original image. Both CAIR ICAM and CAIR GM outperform the baseline, but suffer from a drop in recall as observed in section 4.3.2 (b) Results on the RHINOS dataset. CAIR GM and CAIR ACAM show similar performance and both outperform the baseline. However, the baseline outperforms CAIR ICAM which was not able to come up with a Class Activation Map for a class resembling the rhino.

The precision-recall curves of the more challenging RHINOS dataset are shown in Figure 23b. CAIR GM and CAIR ACAM show similar performance and both outperform the baseline. However, the baseline outperforms CAIR ICAM which was not able to come up with a Class Activation Map for a class resembling the rhino. Resizing an image from the RHINOS dataset sometimes leads to an improvement over using the original image. An example of detections on the original image as well as the same image that was resized with scale factor 0.5 prior to generating candidate boxes can be found in Figure 25. For this particular image the recall stays at 0.6 (three out of five rhinos were detected). In contrast, the precision turns out to be better on the resized image, where fewer false positives detections are detected (Figure 25d).



(a) (b) (c) (d)

Figure 24: Example detections from the Fast R-CNN object detector applied to the original image and the same image resized using CAIR ACAM with scale factor 0.5. (a) True positive detections on the original image. (b) False positive detections on the original image. (c) True positive detections on the resized image, most of the cows were detected. (d) False positive detections on the resized image. The remaining false positives appear close to the cows, however they do not have an IoU greater or equal to 0.5 with an actual cow. These false positive detections are localization errors whereas the false positive detection in the original image is a more serious mistake.

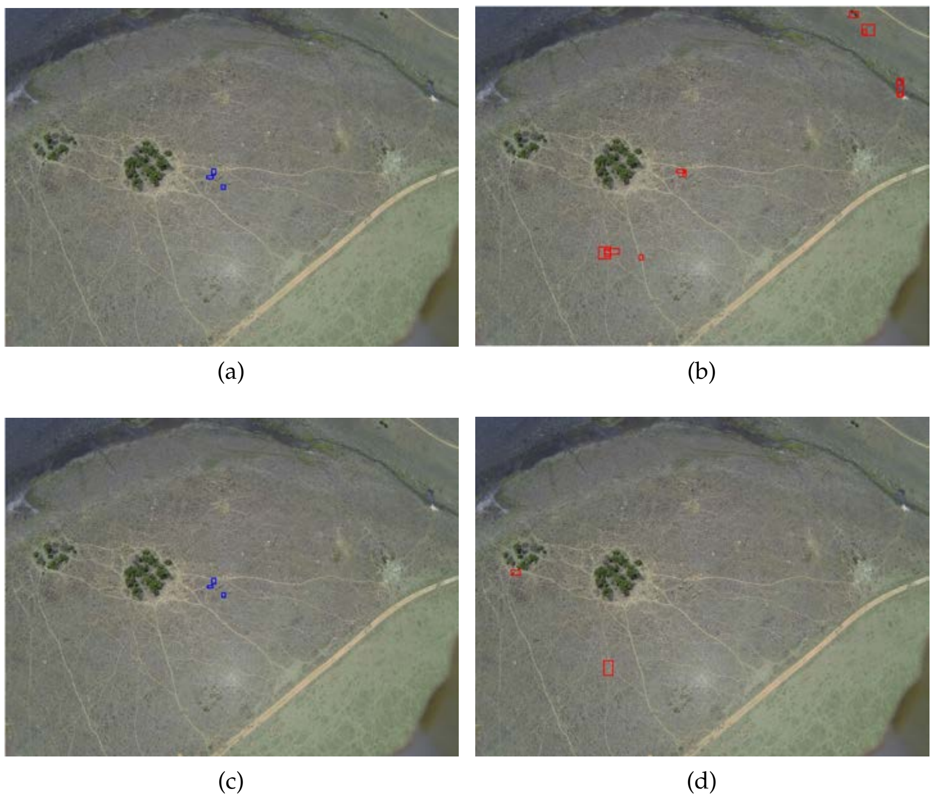

(a) (b) (c) (d)

Figure 25: Example detections from the Fast R-CNN object detector applied to the original image and the same image resized using CAIR ACAM with scale factor 0.5. (a) True positive detections on the original image. (b) False positive detections on the original image. (c) True positive detections on the resized image, in this image the recall of the object detector is not affected by resizing the image. (d) False positive detections on the resized image. We find less false positive detections on the resized image than on the original image.

5 CONCLUSION

To increase the detection rate of an object detector, we reduce the size of aerial imagery captured by a nature conservation drone. Resizing images increases the detection rate in two ways. First the reduction in image size allows the proposal method to generate candidate boxes faster. Second, due to the reduction in image size the proposal method generates fewer candidate boxes that need to be evaluated by the object detector.

To find out how we can resize an image on a drone to increase the detection rate without affecting detection performance we experimented with a content-aware image resizing method inspired by [14]. The content-aware image resizing method decides what pixels to remove based on an energy function. We evaluate the content-aware image resizing method using three energy functions: gradient magnitude (CAIR GM), ImageNet Class Activation Map (CAIR ICAM) and Aerial Class Activation Map (CAIR ACAM).

We perform several experiments to determine that resizing the images using a content-aware image resizing method increases the detection rate without affecting the detection performance. First, we need to verify the resizing methods do not remove important pixels. We then evaluate the recall of the proposal method, which should be minimally affected when resizing the image. Next, we report the reduction in time it takes the proposal method to generate candidate boxes along with the number of boxes generated when resizing the image. Finally, we perform an experiment to evaluate the average precision of the object detector on the resized images. We conclude that using CAIR ACAM an image can be resized with scale factor 0.5 resulting in a higher detection rate without impacting detection performance.

In the first experiment we measure the percentage of important pixels removed from the image when using different methods for resizing. We find all content-aware image resizing methods indeed avoid the removal of pixels that are important to the image content.

The second experiment evaluates the recall of the proposal method. Resizing the image using a content-aware image resizing method causes the recall to drop with only 0.04 points, whereas using the baseline which is not content-aware, causes the recall to drop with 0.51 points. The experiments confirm that recall is minimally affected when resizing the image using one of our methods.

The third experiment illustrates that resizing the image leads to an increase in the detection rate. Both the time it takes for Edge Boxes to generate candidate boxes as well as the number of candidate boxes generated decrease as the image size is reduced. In section 2.1.1.2 we describe how seam carving could run in real-time, using content-aware image resizing is a feasible method for increasing the detection rate of object detectors on a nature conservation drone.

To establish resizing the image does not have a negative effect on detection performance, we evaluate the Fast R-CNN object detector on images resized with different scale factors. We find that the average precision of the detector may benefit from resizing the images. Because fewer candidate boxes are generate by the proposal method, the candidate boxes contain fewer false positives, resulting in better average precision of the object detector.

5.1 FUTURE WORK

To increase the detection rate of Fast R-CNN its successor, Faster R-CNN, replaces the traditional proposal methods by using a 'Region Proposal Network' [38]. By using the Region Proposal Network Faster R-CNN is able to generate candidate boxes and assign a confidence score describing the likeliness this box contains an object simultaneously. Another example where a network is used to generate candidate boxes is HyperNet [39]. Like Faster R-CNN, HyperNet also generates candidate boxes and confidence scores simultaneously. A different approach is used by YOLO (You Only Look Once) [40], where proposal methods are no longer needed.

It is not immediately clear whether the detection rate of these methods would benefit from smaller images. In future work we would like to research whether the use of content-aware image resizing also increases the detection rate of these methods.

ACKNOWLEDGEMENTS

I would like to thank Jan van Gemert and Arnoud Visser for being my supervisors. Jan for believing I have the technical skills to make ConvNets work on a drone, for showing me 'The Science of Scientific Writing' (which was really helpful!) and for supervising my thesis even after changing universities. Arnoud for suggesting I would submit a summary of this thesis to BNAIC to get published (which got accepted), for continuously asking me whether I had started writing already (and being patient when I had not) and sharing our first initial and last name (which may be confusing at times).

Thank you to my colleagues at Birds.ai. Camiel for keeping the pressure on ('have you finished yet?') and providing me with (in his words) 'the cornerstone' of this work: [9]. Kitso for asking the right questions and telling me it's ok to take some time off sometimes. Nick for providing me (willingly or not) with plenty of food to keep me going and some much needed mental support during the last days of writing. Pieter for being an 'annotation king'; I never once had to worry about having too little data because pretty much all the data we collected in South Africa was annotated by him.

Thank you to my family and friends: Amber, Mama, Oma, Opa, Lars, Lisa, Luuk, Martina, Sanne and Papa. For providing me with some necessary distractions, their support, encouragement and patience.

BIBLIOGRAPHY

[1] Koh, L., Wich, S.: Dawn of drone ecology: low-cost autonomous aerial vehicles for conservation. Tropical Conservation Science **5**(2) (2012) 121–132

[2] Marvin, D.C., Koh, L.P., Lynam, A.J., Wich, S., Davies, A.B., Krishnamurthy, R., Stokes, E., Starkey, R., Asner, G.P.: Integrating technologies for scalable ecology and conservation. Global Ecology and Conservation **7** (2016) 262–275

[3] Kakaes, K., Greenwood, F., Lippincot, M., Dosemagen, S., Meier, P., Wich, S.: Drones and aerial observation: New technologies for property rights, human rights, and global development - a primer. Technical report, New America (2015)

[4] Vermeulen, C., Lejeune, P., Lisein, J., Sawadogo, P., Bouché, P.: Unmanned aerial survey of elephants. PloS one **8**(2) (2013) e54700

[5] Van Andel, A.C., Wich, S.A., Boesch, C., Koh, L.P., Robbins, M.M., Kelly, J., Kuehl, H.S.: Locating chimpanzee nests and identifying fruiting trees with an unmanned aerial vehicle. American journal of primatology **77**(10) (2015) 1122–1134

[6] Platt, J.R.: A record 1,215 rhinos were poached in 2014. http://blogs.scientificamerican.com/extinction-countdown/a-record-1-215-rhinos-were-poached-in-2014/ (January 2015)

[7] Mulero-Pázmány, M., Stolper, R., Van Essen, L., Negro, J.J., Sassen, T.: Remotely piloted aircraft systems as a rhinoceros anti-poaching tool in africa. PloS one **9**(1) (2014)

[8] Dhawan, S.: A review of image compression and comparison of its algorithms. International Journal of electronics & Communication technology **2**(1) (2011) 22–26

[9] van Gemert, J.C., Verschoor, C.R., Mettes, P., Epema, K., Koh, L.P., Wich, S.: Nature conservation drones for automatic localization and counting of animals. In: Workshop at the European Conference on Computer Vision, Springer (2014) 255–270

[10] Hosang, J., Benenson, R., Dollár, P., Schiele, B.: What makes for effective detection proposals? IEEE transactions on pattern analysis and machine intelligence **38**(4) (2016) 814–830

[11] Uijlings, J.R., van de Sande, K.E., Gevers, T., Smeulders, A.W.: Selective search for object recognition. International journal of computer vision **104**(2) (2013) 154–171

[12] Hosang, J., Benenson, R., Schiele, B.: How good are detection proposals, really? arXiv preprint arXiv:1406.6962 (2014)

[13] Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.htm

[14] Avidan, S., Shamir, A.: Seam carving for content-aware image resizing. In: ACM Transactions on graphics (TOG). Volume 26., ACM (2007) 10

[15] Rubinstein, M., Gutierrez, D., Sorkine, O., Shamir, A.: A comparative study of image retargeting. In: ACM transactions on graphics (TOG). Volume 29., ACM (2010) 160

[16] Rubinstein, M., Shamir, A., Avidan, S.: Improved seam carving for video retargeting. In: ACM transactions on graphics (TOG). Volume 27., ACM (2008) 16

[17] Vineet, V., Narayanan, P.: Cuda cuts: Fast graph cuts on the gpu. In: Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on, IEEE (2008) 1–8

[18] Chiang, C.K., Wang, S.F., Chen, Y.L., Lai, S.H.: Fast jnd-based video carving with gpu acceleration for real-time video retargeting. IEEE Transactions on Circuits and Systems for Video Technology **19**(11) (2009) 1588–1597

[19] Duarte, R., Sendag, R.: Accelerating and characterizing seam carving using a heterogeneous cpu-gpu system. In: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp) (2012)

[20] Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, IEEE (2014) 580–587

[21] Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 1440–1448

[22] Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. Pattern Analysis and Machine Intelligence, IEEE Transactions on **32**(9) (2010) 1627–1645

[23] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks. In: International Conference on Learning Representations (ICLR 2014), CBLS (April 2014)

[24] Alexe, B., Deselaers, T., Ferrari, V.: What is an object? In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE (2010) 73–80

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

[26] Dollár, P., Zitnick, C.L.: Fast edge detection using structured forests. IEEE transactions on pattern analysis and machine intelligence **37**(8) (2015) 1558–1570

[27] Zitnick, C.L., Dollár, P.: Edge boxes: Locating object proposals from edges. In: Computer Vision–ECCV 2014. Springer (2014) 391–405

[28] Cheng, M.M., Zhang, Z., Lin, W.Y., Torr, P.: Bing: Binarized normed gradients for objectness estimation at 300fps. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2014) 3286–3293

[29] Zhao, Q., Liu, Z., Yin, B.: Cracking bing and beyond. In: BMVC. (2014)

[30] Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. Volume 1., IEEE (2001) I–511

[31] Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). Volume 1., IEEE (2005) 886–893

[32] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision (2014) 1–42

[33] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. (2012) 1097–1105

[34] Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: Delving deep into convolutional nets. In: British Machine Vision Conference. (2014)

[35] Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European Conference on Computer Vision, Springer (2014) 818–833

[36] Oquab, M., Bottou, L., Laptev, I., Sivic, J.: Is object localization for free?-weakly-supervised learning with convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 685–694

[37] Zhou, B., Khosla, A., A., L., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. CVPR (2016)

[38] Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI). (2016)

[39] Kong, T., Yao, A., Chen, Y., Sun, F.: Hypernet: Towards accurate region proposal generation and joint object detection. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2016)

[40] Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016)