A Computer Vision System to Generate Image Captions

Ashwin Srinivasan

10th Grade

Pittsburgh Allderdice High School

Category: Computer Science/Math

Table of Contents

Abstract……………………………………………………………………………………...3

Introduction……………………………………………………………………………….....4

Goal………………………………………………………………………....…………........5

Design Criteria……………………………………………………………………………....6

Implementation…………………………………………………………………………........6

Testing……………………………………………………………………………..……..10

Discussion………………………………………………………………………...…...15

Conclusion……………………………………………………………………………………16

References……………………………………………………………………………….17

Abstract

Digital images are an essential component of the modern Internet as well as human interaction with computers. However, they require the user to physically view them, are much larger in size than text, and cannot be easily processed in bulk. Existing solutions to these issues have low success rates, are impractical, or require massive computing resources. In this project, an automated computer vision system to generate brief captions of images was developed, making images more accessible and easier to search for. The program was able to produce accurate captions for a wide variety of images, identifying the objects in the image, their background, and their positions within the image while being very fast and low-resource. Applications of this project include generating alternative text on slow connections where images cannot be loaded, enhancing online and local image search, and aiding vision-impaired users.

Introduction

Digital images are an essential component of the modern Internet as well as human interaction with computers. They are utilized in web pages, documents, software interfaces, and many others. Since the early 21st century, the volume of digital images has increased at a much faster rate than text, more than doubling in merely five years [1]. However, images impose several requirements that do not apply to text. First of all, they must be physically viewed by the user, rendering powerful existing text-to-speech software useless. For the visually impaired, the inability to view images seriously hinders understanding of documents and other media. Digital images are also much larger in size than text, and require far more bandwidth to transmit, regardless of the compression techniques applied. This is a major barrier for developing countries, where the Internet connection is often extremely slow and unreliable, severely restricting the consumption of visual content. Additionally, images, as well as audiovisual formats in general, are more difficult to process in bulk or by computers than text. In order to search images, they must be tagged appropriately with alternative text labels, which approximately 75 percent of images on the World Wide Web are not despite various efforts to reduce this figure. Currently available image tagging software is limited to uncomplicated images or only a few labels [2, 3].

The most widely available partial solution to these issues is **reverse image search**, offered by most popular search engines [4]. The two techniques used are image fingerprinting and feature detection. *Image fingerprinting* assigns a signature to images usually based on their color patterns and finds possible matches by searching for similar signatures. *Feature detection* attempts to match the edges, color patterns, and interest points of images [5, 6]. These methods are combined to locate a known matching image that has a human-written caption, ultimately producing a short text label of the given image. However, this description is elementary and solely based on cached human-written captions of images. No new text is generated to describe the image in natural language. Reverse image search also requires an extensive database of existing images in order to function, making it impractical to use locally or for less common images [4].

Another option is **crowdsourced image descriptions,** where human participants are tasked with writing captions of images in a game-like format. The participants must be compensated, however, and this solution opens potential for human error. In addition, crowdsourced captioning is much more time-consuming than any computer-based method and is ill-suited to the constant proliferation of new and significant images [3, 4].

While some fully automated solutions to these issues do exist, they generally do not exceed a success rate of 40 percent, which makes them unsuitable for real-world use [7, 8]. Additionally, they require massive computing resources and excessive prior training, negating much of their usefulness [9]. Therefore, it is important to develop a fast and accurate automated system to transcribe the contents of an image in plaintext [2].

Goal

The goal of this project was to develop an efficient, low-resource computer program to automatically generate brief captions of digital images that can be easily read, passed to existing text-to-speech software, or further processed by other software for analysis.

Design Criteria

The design criteria for the program were that (i) it should generate human-readable captions, (ii) the captions should convey the core content of an image, and (iii) the program should be fast and low-resource, unlike current alternatives.

Implementation

The program was written in Java, an object-oriented programming language, selected due to its vast documentation and library support. The program uses a simple command-line interface where the user enters an image path and the output is the computer-generated caption. The Java wrapper for the open-source OpenCV computer vision library and the free Aylien image tagging Application Programming Interface (API) are utilized in the captioning process [10]. Excluding these libraries, the program contained more than 1500 lines of code in total, all written only by the author of this paper. In order to generate a caption for the input image, the program uses a three-stage process: color based edge detection for to identify objects, OpenCV object detection, and image segmentation and object tagging. This is shown in Figure 1 on the next page.

The first step in the captioning process is loading the image into memory from the path provided to the user. The program supports most image file formats, and vector images are converted to bitmaps. At this time, the program also connects to the Aylien API and loads OpenCV libraries. After this, two object detection methods are run simultaneously to identify the regions of objects in the image. Multithreading this stage allows the program to run more efficiently and halves the execution time of the most intensive part of the process.

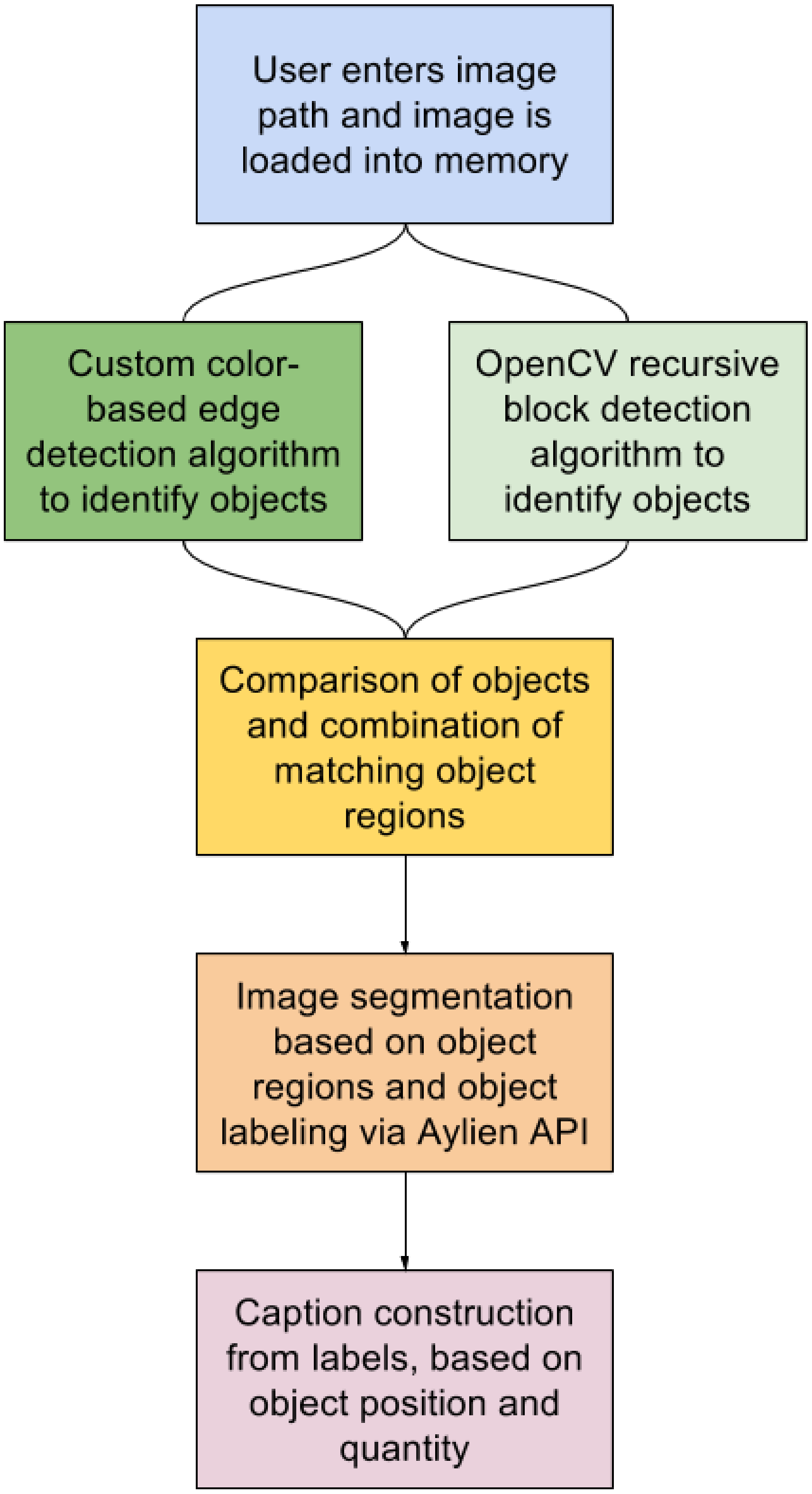
User enters image path and image is loaded into memory
Custom color-based edge detection algorithm to identify objects
OpenCV recursive block detection algorithm to identify objects
Comparison of objects and combination of matching object regions
Image segmentation based on object regions and object labeling via Aylien API
Caption construction from labels, based on object position and quantity

Figure 1: Diagram showing steps the program's image captioning process

The first object detection algorithm is a custom **color-based edge detection** method written for this program. A duplicate of the image is created and the contrast is boosted using OpenCV, which increases the intensity of light pixels and decreases that of dark pixels. Then, OpenCV's canny edge detection algorithm is applied, producing a map of outlines from the image representing the boundaries between objects [5]. The image is blurred to even its texture and smooth any imperfections, and the gradient of intensity is computed for groups of pixels in the image. Where this difference is highest, an edge is placed, resulting in an edge detection map populated with outlines [11].

Using this information, the program scans each edge for color change in square blocks. The square scanning frame follows the outlines and attempts to find a significant difference in color relative to the image between each of its corners. Where color change is detected, the process is recursively executed with smaller square blocks. After the color change detection algorithm returns its results, an object map is created where each object edge represents a change in color. A contiguous, closed set of object edges is stored as one object region defined by the furthest horizontal and vertical pixels on the edges for the object. Finally, the list of object regions found by the color-based edge detection algorithm is saved as its output.

OpenCV's object detection algorithm searches for object boundaries using a different method. It recursively creates cells of varying position and size and attempts to match them to the edge detection map. When a match is found, the algorithm recursively tries more blocks within the parent cell to identify object edges. Once this process is complete, the object region is saved as the group of pixels within the matching region on the edge detection map [12]. The final output is, much like the color-based edge detection algorithm discussed previously, a list of object regions detected by the algorithm.

Both of the above object detection algorithms typically return their results concurrently, and the object regions identified by each are compared. Since the color-based edge detection algorithm was more sensitive to minute or irrelevant objects when the program was tested during development, object regions produced by this algorithm that were within an object region produced by the OpenCV algorithm were combined and merged with the larger object. For other objects, regions identified by both algorithms that had similar boundaries were considered valid. After extensive testing and several iterations, this threshold was set at 5 percent. Object regions not detected by both algorithms are discarded by the program.

The last stage of the process is **image segmentation**, which uses the final set of object regions produced by all of the previous stages. For each object, the image is divided based on the size of the region, and pixels not within the object itself are converted into whitespace. After this is completed for all objects, there are pixels remaining in the image that constitute the background, which becomes its own segment. Object detection is run again on the background, possibly segmenting it further. All segments are saved as individual images and sent to Aylien's Image Tagging API for **object labeling**.

The Aylien API is a third-party resource that the program uses to determine the English text version of each object region. While this functionality requires an Internet connection, it can easily be implemented independently in the future to eliminate this requirement. The Aylien API uses feature detection to match each object's segment to one of hundreds of cached images that already have text labels. Unlike reverse image search, which simply does this for the entire image, the program in this project only uses this method to find the description of individual objects. The chance of finding an accurate match for just one object extracted from the image is much higher than matching the image itself, making this program superior in concept to the existing solution.

Once every object is assigned a text label by the API, the caption is constructed. A general template listing the object names and the background label, ordered by their left-to-right position in the image. If there are multiple objects with the same name, the quantity is included in the caption. The final sentence containing all of this information is displayed to the user, and the program is terminated.

Testing

The program was tested with dozens of images throughout development. In this paper, four sample images will be used to demonstrate the quality of the program's output:

- Collection of very simple stock photographs on white background (Figure 2)
- Actual photograph with an isolated subject (Figure 3)
- Close-range photograph with out-of-focus background (Figure 4)
- Photograph with multiple subjects and implied motion (Figure 5)

For each of these images, the origin human-written caption and the description produced by the program is shown, in addition to the program's detected object regions.

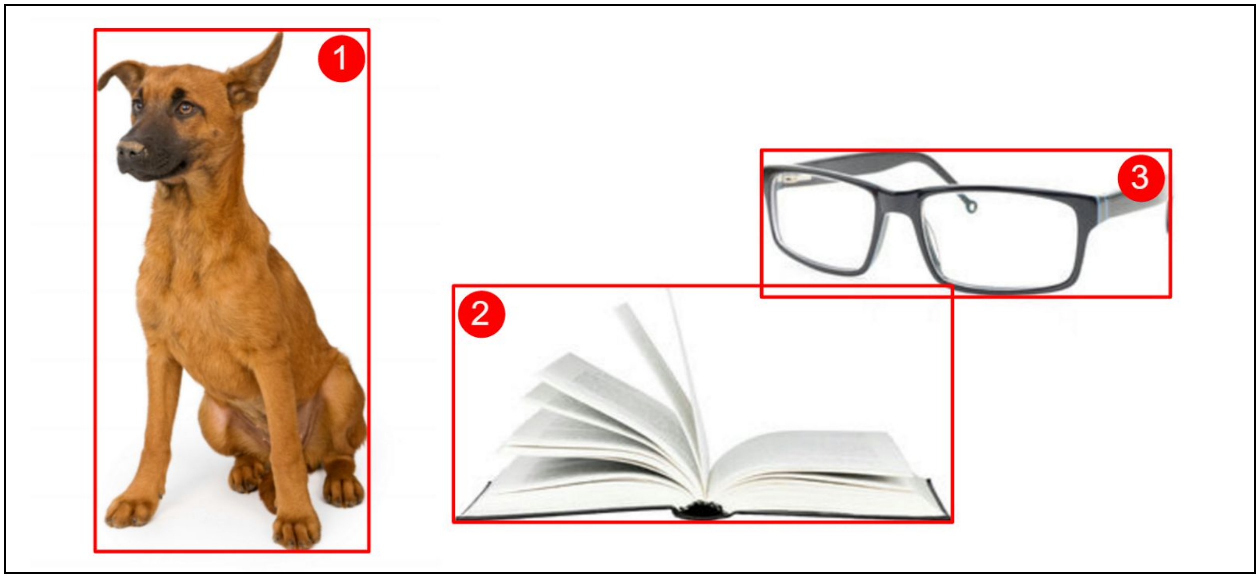

Figure 2: Collection of very simple stock photographs on white background (first sample image)

Original Caption: A dog, book, and glasses on a white background.

Program Output: Dog, book, reading glasses.

In the first sample image, shown above (Figure 2), the program detected all objects as expected. The color-based edge detection algorithm identified three objects within the dog due to its color variation, but they were merged as shown. The other two objects were detected nearly identically by both algorithms, and Aylien's tagging API correctly identified the objects and gave them proper text labels. While the description is not grammatically sound, it provides a very accurate account of the image content. The program also executed in less than half a second on the testing hardware, a major advantage over current solutions.

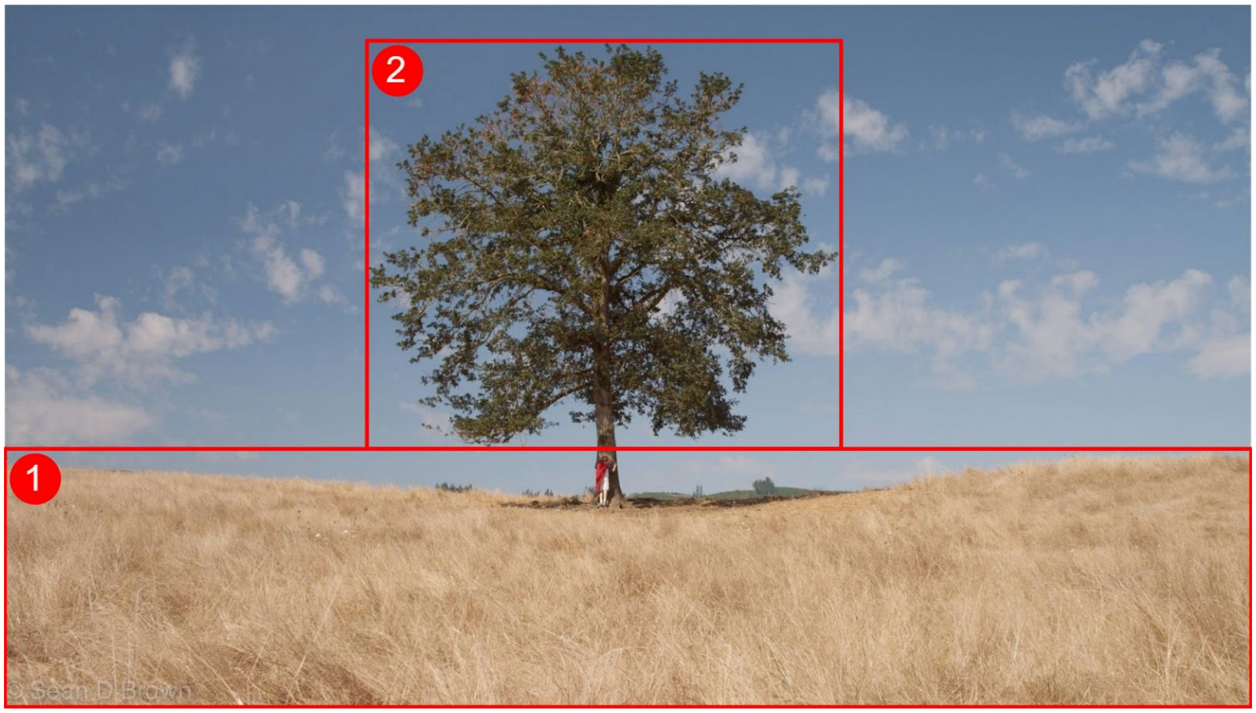

Figure 3: Actual photograph with an isolated subject (second sample image)

Original Caption: A child hugs a tall tree in a field.

Program Output: Wheat field, tree with sky in background.

The second sample image, shown above (Figure 3), is much more complex than the first sample image. Upon first glance, the only visible objects are the field, the tree, and the sky. The program's caption is very effective at describing these, but closer inspection reveals the child hugging the tree trunk, which is not included in the machine caption. During the object detection phases, the color-based edge detection algorithm detected the child and her cape separately, while OpenCV’s algorithm did not detect her at all due to the shadow of the tree trunk. This discrepancy resulted in those objects being excluded altogether. Overall, though, this caption still allows the user to understand the subjects of the image without needing to actually view it.

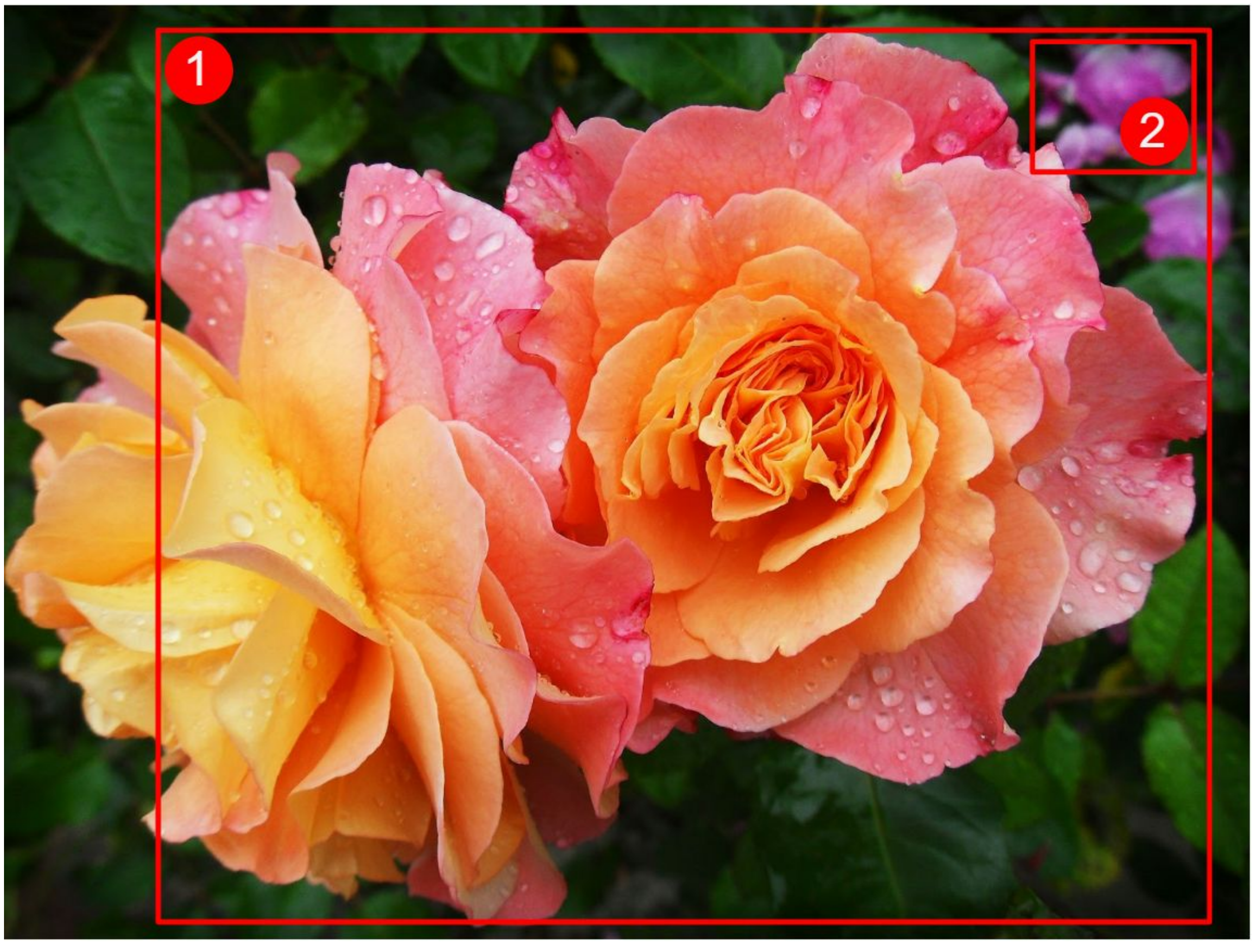

Figure 4: Close-range photograph with out-of-focus background (third sample image)

Original Caption: Two roses with water droplets on their petals.

Program Output: Rose, flower, with leaves in background.

In this close-range photograph (Figure 4), the program detected both roses as one object due to their color similarity and the low sensitivity of the OpenCV object detection algorithm. Interestingly, the purple flower in the out-of-focus background was also detected. This is due to the canny edge detection algorithm's requirement of a blurred image, which makes the program unable to differentiate between the photographical background and the subject. Despite these oddities, the program's output was very accurate. To a user not able to view the image, the description produced here would be an excellent illustration.

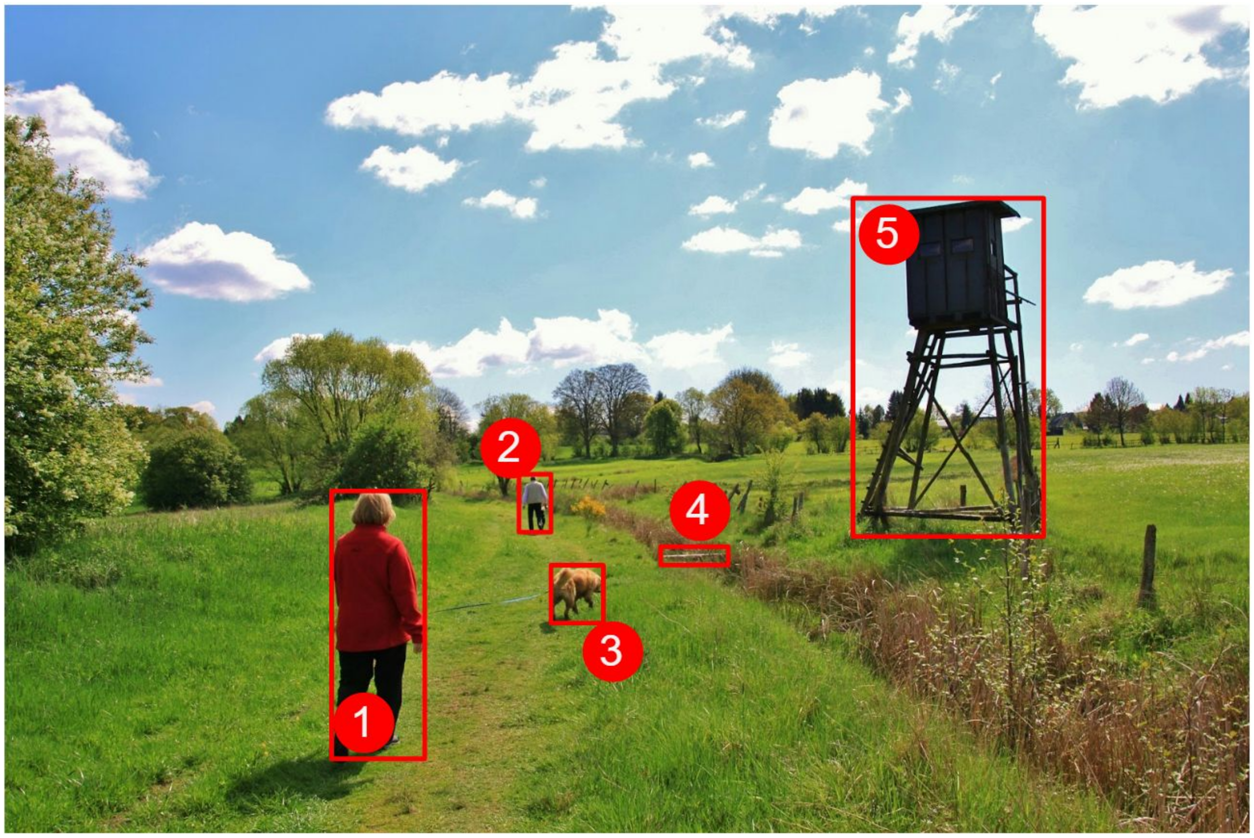

Figure 5: Photograph with multiple subjects and implied motion (fourth sample image)

Original Caption: Woman hiking through a grassy field and walking her dog.

Program Output: Person (x2), dog, tree (x4) with grass, sky in background.

The final sample image (Figure 5) is very complicated, with multiple subjects and implied motion. Foreground subjects were mostly detected accurately by object detection algorithms. Objects 4 and 5 were excluded from the caption since the Aylien API did not find any matches for them. For the most part, the program was able to identify the most significant objects in the image as well as a multi-part background. However, the crucial missing element is the action, which is the woman walking her dog. While the caption provides a suitable elementary description of the image, it does not include any mention of the verb that is clearly visible.

In all of the above sample images and throughout testing, the execution time of the program was under one second, which is a significant advantage over existing solutions, and the captions all provided fairly accurate descriptions of the content of the images.

Discussion

The program developed in this project was able to generate brief captions for wide variety of images, satisfying the design criteria. The **key strengths** of the program are the consistent detection of simple objects, accurate and reliable labels, and ability to effectively handle multiple-object images. Additionally, the captions produced by this program are much more descriptive than widely available reverse image search text, and the program itself is faster and much less resource intensive than existing automated solutions.

In general, image captions currently must be written manually by a human, which is not scalable to the vast number of images online and their rapid proliferation [2]. Commonly used image tagging software is limited to very simple images or only a few tags, necessitating a reliable automated image captioning solution [7, 8, 9], and this project fills this need.

This project has several **real-world applications** to mitigate the issues with images. The program can automatically generate alternative text on slow connections where images cannot be loaded, eliminating a major hinderance. It can be used to generate descriptions for images to drastically improve image search and indexing. Due to the program's low-resource nature, this can be applied to both online and local settings. Finally, it can be combined with widely-used text-to-speech software to read aloud captions of images to blind or vision-impaired users, allowing them to access visual content without human transcription.

The program does have some **limitations**, however. The custom color-based edge detection algorithm sometimes chooses incorrect object regions since not all objects in images are signified by a single unique color. In addition, the program cannot detect depth without a stereo image, and similarly, it cannot detect motion or action in images.

In the future, the object detection algorithms can be further refined to improve their consistency and reliability, as well as optimizations to increase speed. The program can be trained to detect objects in different states of motion so that it can add verbs to captions, putting the program's output much closer to an ordinary human-written caption. The Bilingual Evaluation Understudy (BLEU) score can be computed to measure the difference between the human-written caption and machine caption, and a user study can be conducted to evaluate the effectiveness of the program in real-world use.

Conclusion

In summary, digital images have inherent issues with file size and categorization that can be solved by automatically generating image captions. Existing solutions are time-consuming, require extensive computing resources, and are not practical or reliable. The computer program developed in this project is able to write brief captions involving objects and their position in an image while not requiring massive computing resources, a major advantage over current alternatives. The applications of this program include improving accessibility for low-bandwidth connections, enhancing image search, and aiding blind or vision-impaired users. In the future, the program can be improved to be more accurate and construct more sophisticated image descriptions.

References

1. "Seeing Is Believing: Why Using Visual Content in Your Marketing Makes Sense." *Curve*. Getty Images. Web. 24 Jan. 2016.
2. "Importance of Image Data and Image Processing Techniques for Digital Humanities." *DH101*. 21 Oct. 2014. Web. 17 Nov. 2015.
3. Ahn, Luis Von, and Laura Dabbish. "Labeling Images with A Computer Game." *CHI* (2004). Web.
4. Ahn, Luis Von, Shiry Ginosar, Mihir Kedia, and Manuel Blum. "Improving Image Search with PHETCH." *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2007* 4 (2007): 1209-212. Web. 17 Nov. 2015.
5. Szeliski, Richard. *Computer Vision: Algorithms and Applications*. Springer, 2010. Print.
6. Brown, Matthew. "Features and Image Matching." University of Washington. Web. 20 Jan. 2016.
7. Platt, John. "Rapid Progress in Automatic Image Captioning." *Machine Learning*. Microsoft Technet, 18 Nov. 2014. Web. 20 Jan. 2016.
8. Pan, Jia-Yu, Hyung-Jeong Yang, Pinar Duygulu, and Christos Faloutsos. "Automatic Image Captioning." *Proceedings of the IEEE International Conference on Multimedia and Expo* (2004). Web.
9. Vinyals, Oriol, Alexander Toshev, Samy Bengio, and Dumithru Erhan. "Show and Tell: A Neural Image Caption Generator." 20 Apr. 2015. Web.
10. "Image Tagging." *AYLIEN*. Web. 20 Jan. 2016.
11. Wang, Liming, Jianbo Shi, Gang Song, and I-fan Shen. "Object Detection Combining Recognition and Segmentation." *Proceedings of the ACCV Conference on Computer Vision* 4834 (2007): 189-99. Web.
12. "Java Documentation." *OpenCV*. Web. 20 Jan. 2016.