Teaching Computer Vision in the Computer Science Field Guide

Isabelle Taylor
University of Canterbury
Christchurch, NZ
iet11@uclive.ac.nz

Richard Green
Supervisor
University of Canterbury
richard.green@canterbury.ac.nz

Abstract—**This paper describes improvements made to the Computer Vision chapter of the Computer Science Field Guide. These improvements include the addition of an interactive that teaches students about face detection using the Viola-Jones approach. Feedback indicates the interactive effectively teaches the concept in a socio-constructivist way. The interactive works correctly on the latest browsers and on touch screen devices. Future work could include improving the interactive to include a few extra features, such as showing all Haar-like features or adjusting the parameters of the face detection algorithm.**

I. Introduction

The Computer Science Field Guide [1] is an open-source, interactive, online "textbook" that teaches a wide range of topics in computer science. The Computer Science Field Guide contains a large number of interactives that teach concepts in an engaging way. One of the topics taught in the Computer Science Field Guide is computer vision. This paper contains the proposed improvements to the computer vision chapter through the addition of an interactive that teaches students about how face detection using the Viola-Jones approach works. The goal is for the students to understand how face detection works, which includes understanding why it does not work correctly in certain situations.

II. Background

The Computer Science Field Guide computer vision chapter currently contains content related to filtering images and edge detection. It also has a hands-on activity in which the students can learn about face recognition. Last year there was work on improving the chapter and several JavaScript interactives were made that teach about mean, median and gaussian blur algorithms, thresholding and edge detection [2]. Though these interactives are yet to be included in the published version of the Computer Science Field Guide, it is important not to repeat work that has already been done.

The Computer Science Field Guide is designed to be engaging, cater to various learning styles, be platform independent and not require programming competency[3]. The improvements made to this chapter must follow this criteria. They also must cover a topic that is not yet included in the Computer Science Field Guide. The topic taught must be capable of being understood by high school students. This is particularly difficult in the field of Computer Vision, as many of the algorithms use complex maths.

Another criteria that the improvements should attempt to follow is the socio-constructivist approach to learning [4]. This approach involves providing scaffolding to help a student approach a problem, and allowing them to "invent" the solution themselves, rather than simply giving it to them. Students can grow in confidence and understand the solution, rather than merely memorising it. This will help them in the future when attempting to solve similar problems.

Following the constructivist approach, a student must be able to make a hypothesis, test it, and evaluate it for correctness. A wrong hypothesis (an error) will become apparent to the student through their testing and they will learn something from it [5]. This means we should encourage students to explore, rather than guiding them to the correct answer straight away.

The interactives in the Computer Science Field Guide help to teach concepts in this way. An example of this is the "Trainsylvania" interactive that teaches about finite state automaton by using the metaphor of trying to get to your destination and choosing between train A and train B. This interactive allows the student to explore and make mistakes and they end up with a deeper understanding of the concept being taught.

Though not all interactives in the Computer Science Field Guide can necessarily be called games, many are or have parts that are "game-like." This is usually to make them more engaging. A recent 2016 literature review into games for teaching computing did not find any games that teach about computer vision concepts [6].

There is a strong need to train students to become knowledgeable about image-related computation [7] as it can pique their interest in related areas such as mathematics. With cameras prevalent in almost all technical devices, it is important that people have an understanding of computer vision concepts and the way they can affect their lives. This includes use of computer vision for surveillance and new technology that controls self-driving cars and augmented reality applications. All of the interesting real world applications of computer vision also make it a good introduction to how computer science changes the world.

After investigation into various computer vision concepts, the following options were considered:

1) Face/object detection (Viola-Jones approach) [8]
2) Face recognition (Eigenfaces) [9]
3) Feature detection (Harris corner detection) [10]
4) Motion tracking (Diff based or using feature detection) [11]

While feature detection uses a reasonably simple algorithm, it may not be of interest to the students because it is generally used as parts of larger systems. This means any context that is discussed or used in the chapter will likely need to mention other Computer Vision concepts. It also involves matrix mathematics that, while possible for high schools students to understand, may make it less interesting to some students. Motion tracking can be taught in a very simple way by finding the difference between two images. There are also more complex ways of motion tracking, one of which uses feature detection, as discussed previously. This means that teaching motion detection at a reasonably advanced level will require teaching about feature detection.

Teaching about face detection or recognition should be immediately engaging and require less background knowledge. The Eigenfaces algorithm for facial recognition contains complex maths, including covariance matrices and calculation of eigenvectors. Though it may be possible to teach high school students these mathematical concepts, it will be difficult to teach in a socio-constructivist way. Because of this, the decision was made to focus on face detection using the Viola-Jones approach.

A. Viola-Jones approach to face detection

This method has 3 main parts that combine to allow efficient detection of many faces in a scene [8]. This approach can also be applied to objects other than faces.

1) Identifying features: Features of images are identified by first calculating the integral image. This can be done efficiently and is then used to find Haar-like features in the image. These are the patterns of intensity in images. Generally similar objects will have similar patterns. Figure 1 shows the eye area of a face is generally darker than the part below it.

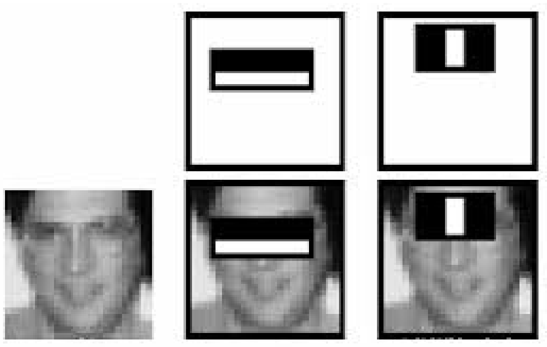


Figure 1. Haar-like features from [8]

2) Training a classifier using boosting: These weak features are not enough to classify objects on their own. Instead they need to be weighted correctly. This is done through a method called boosting, which adjusts the weighting incrementally. This requires a lot of training data and can take a long time, but would only need to be done once and then it can be used everywhere.

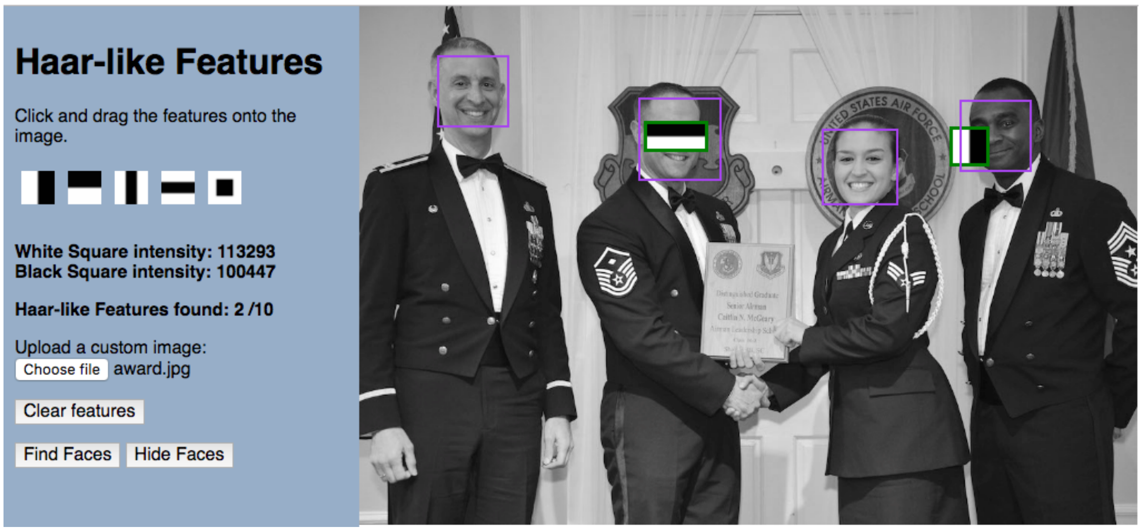
3) Cascading: Checking whether a part of an image has a face can take a long time if it is checked against every feature. Instead, it is easier to identify parts of an image that are definitely not faces by checking them against the features with the highest weighting. For example, the highest weighted feature may be the horizontal dark area with a lighter part below it, as see in Figure 1. Only parts of the image that are identified as containing this feature are checked against other features to confirm if there is a face. This approach makes the algorithm far more efficient and is the reason this it can be used to detect faces and other objects in video. At the time of publication this was something that had never previously been accomplished.

B. Viola-Jones Teaching Resources

There are currently no resources that aim to teach this concept in an interactive way. If students perform a Google search, it will lead to academic papers that are often not accessible to high school students. The most accessible resource available is the Udacity course on Computer Vision [12]. It has a chapter about face detection that discusses the Viola-Jones approach. Udacity does not follow a constructivist approach - It teaches a concept through videos and then gives an exercise for the student. The exercise in the relevant chapter is a programming exercise, which means it is definitely catering to a different audience than the Computer Science Field Guide.

III. Method

In order to teach students how face detection works using the Viola-Jones approach, we created



Haar-like Features

Click and drag the features onto the image.

White Square intensity: 113293
Black Square intensity: 100447

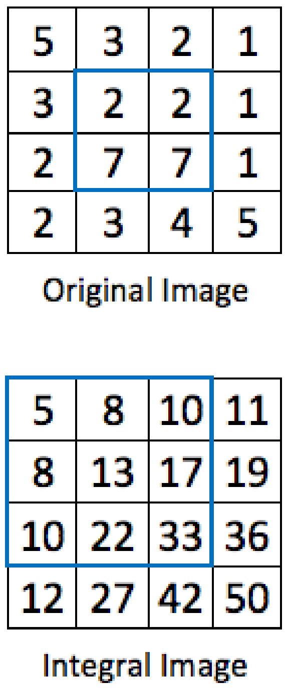
Haar-like Features found: 2 /10

Upload a custom image:

Figure 2. Screenshot of interface showing faces detected and some Haar-like features

a interactive for placing Haar-like features on an image. The interactive was creating using Javascript because it is platform independent and will run on all common browsers. Figure 2 shows a screen shot of the final interactive. The student picks from the Haar-like features on the right and drags them onto the image. When the feature is in a correct position (i.e. where the intensity of the white area is greater than the intensity of the black area) a green border will be shown and the number of features found will be incremented. The goal is for the student to find 10 features. After finding some features the student can then run the face detection algorithm and squares will be drawn around the faces. If there is a face that is not found, or a face detected in a space where no face exists, the student can use the interactive to figure out the reason for this. For example, there might be poor lighting in that area and the expected Haar-like features do not exist. In this way the interactive encourages a contructivist style of learning because the student can make a hypothesis and test it. If a student sees that a face in profile is not detected by the face detection algorithm they can test and understand that a face in profile will have some different Haar-like features than that of a forward-facing face. They will understand the algorithm and it's limitations more deeply. To help them understand when how the algorithm works with different types of faces and other images, the student can upload their own image and find the Haar-like features on it.

The interactive takes advantage of two open source Javascript libraries: tracking.js [13] and interact.js [14]. tracking.js provides an implementation of face detection using the Viola-Jones method and also provides some of the helper methods, including a method to calculate the integral image of an image. This was very useful because the integral image is needed to be able to quickly calculate the intensity of any rectangle in the image. The integral image calculation returns an array that



5	3	2	1
3	2	2	1
2	7	7	1
2	3	4	5

Original Image

5	8	10	11
8	13	17	19
10	22	33	36
12	27	42	50

Integral Image

Figure 3. The integral image is the sum of the intensity of the pixels in the rectangle to the left and above every pixel in the original image.

contains the intensity of the rectangle above and to the left of every pixel of the image. This can be seen in Figure 3. If we want the total intensity of the blue rectangle from the original image, we could add every pixel intensity from the original image together. However this computation is $O(n)$ as it will become larger as the number of pixels increases. The better option is to use the computed integral image. Figure 4 shows the important points when calculating the intensity of any rectangle in a image. It can be done with a simple calculation:

$$I = C - B - D + A$$

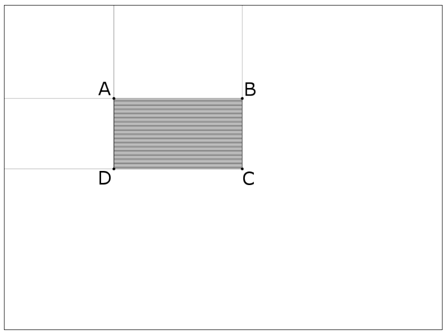


Figure 4. How the intensity of pixels within any rectangle in the image can be computed with the integral image.

Using the example from Figure 3, this calculation is:

$$I = 33 - 10 - 10 + 5$$

$$I = 18$$

This is consistent with summing all the pixels from the original image, but will only ever use 3 additions/subtractions no matter how many pixels are included. This calculation is done for each rectangle in a Haar-like feature and can be completed very quickly as the user is dragging the feature around the image. interact.js was useful for the dragging and dropping of the features. It also allowed for the resizing of the features and let us limit the features to stay within the image. Because there will be multiple of the same feature in the image, the interactive clones each feature when it is dragged onto the image. This cloning was not a feature of interact.js so had to be implemented manually.

IV. Results

The goal was to produce an interactive that would teach high school student about a computer vision concept. It should be easy to use and aim to follow the socio-constructivist approach. To discover whether this interaction achieved these goals we showed it to 30 teachers and gathered their feedback. In the future it may be useful to gather feedback from the students themselves. This interactive is not intended to stand alone i.e. it will be a part of the computer vision chapter of the Computer Science Field Guide and will have a short explanation of the concepts to accompany it. Since this text is not yet written, instead a aural explanation was given and a short demonstration of the interactive was given.

The interactive made it very easy to explain how face detection works. The teachers were able to intuitively understand that the algorithm would have problems when a face was badly lit or turned to the side. The success of the interactive in explaining concepts to the teachers in a great indicator of its future success with students. The teachers had a lot of positive feedback to give and a few suggestions. Some of these suggestions are discussed in the Future Research section of this paper. I received feedback from, Tim Bell, the head of the Computer Science Education research group. Bell believed it would be a very engaging interactive for students and especially appreciated that you could upload your own images. I also requested feedback from some of the other developers who have worked on the Computer Science Field Guide for a long time. Jack Morgan thought it clearly allowed experimentation with Haar-like features and would interest students with the real-world applications of face detection.

An important indicator of the success of this interactive is whether it is appropriate for NCEA students studying Digital Technologies to use for Achievement Standard 3.44 [15]. Not all students using the Computer Science Field Guide will be studying NCEA, but it is useful nonetheless. This standard requires students to demonstrate understanding of an area of computer science. One of the areas listed is "graphics and visual computing". The Viola-Jones algorithm for facial detection fits within this area. To achieve a good grade in this standard the student discuss practical applications of the algorithm and evaluate its effectiveness. The interactive will allow the student to figure out in which situations the algorithm might become ineffectual. Therefore, it can definitely help students to achieve this standard.

The interactive has been tested in the latest versions of all major browsers. It has also been tested on a touch screen. The interactive works as intended on these systems. The performance was tested on a 2014 Mac Book Pro with a 2.6Ghz processor and 16GB of RAM using the Google Chrome browser. This computer has better specifications than many computers that are expected to be running the interactive. When dragging a Haar-like feature around it consistently took 0.001 milliseconds to compute the intensity of any rectangle in the image. This low number means that the program will not lag, even on a much slower computer.

Another computation that the interactive uses is the tracking.js method to track faces and draw rectangles around them. This happens whenever the user clicks the 'Find Faces' button. This calculation was timed with the same specifications described above and it took an average of 578.0 milliseconds on 10 different images with different numbers of faces in them. This number is high enough to be a noticeable lag, and would be even more pronounced on a slower computer. It is not possible to improve on this time, unless I chose to implement the algorithm myself. This would take a significant amount of time and there is no guarantee it could be made more efficient. Instead, in future work it may be beneficial to give the user some feedback when they click the button, for example, a progress indicator.

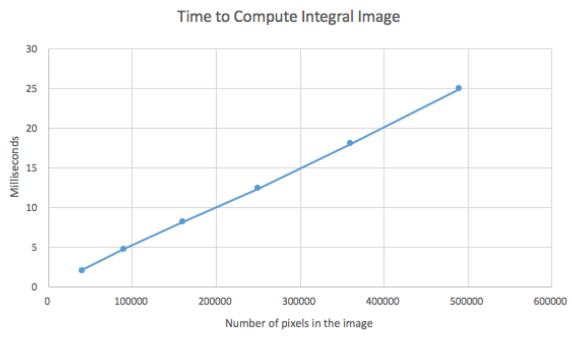


Figure 5. Time to compute the integral image

Figure 5 shows that the calculation of the integral image (which is done once for every image uploaded) is linear, relative to the number of pixels in the image. Since the image is resized if it is larger than 700 pixels wide, we can be sure that this calculation should never take longer than 30 milliseconds on this computer. On a slower computer this calculation will take longer but since it is done in the background, after the image has been uploaded, it will have no effect on the user.

V. Conclusion

The results indicate that this interactive accomplishes the goal of teaching about facial recognition in an engaging way. It follows the socio-constructivist approach which allows students to discover how the Viola-Jones algorithm works themselves, and also discover its limitations. This will also allow NCEA students to complete the Digital Technologies standard 3.44.

One limitation of the interactive is that it does not include all possible Haar-like features. There are many extended sets of Haar-like features that include ones that have been rotated 45 degrees. It did not seem necessary for the students' learning to include them all.

This paper discusses a unique approach to teaching about the Viola-Jones algorithm. Though interactive games exist that teach about other areas of computer science, and even a few for other areas of computer vision - this is the only one to attempt to teach facial recognition. There are limited resources available that teach about the Viola-Jones algorithm in a way that will be accessible to high school students. Most resources are research papers or complex discussions about code. Even the most accessible resource, the Udacity course on Computer Vision, assumes the students have programming skills.

A. *Future Research*

In the future it would be useful to complete a study into the effect of this interactive (and others) on the learning outcomes of high school students. There are also many areas in which this interactive could be extended, to make it even more useful as a learning tool.

One of these is fitting it seamlessly into the computer vision chapter by writing some text to explain the concepts and even suggest some things for students to try. For example, students should place a Haar-like feature over the eyes and area below and see that all faces with have this feature. Writing this text and suggestions was outside the scope of this paper.

Another part that could increase the benefit from this interactive is by showing more parts of the process. For example, currently it is showing how a computer learns what faces look like. However, it might also be helpful to show the part of the process where a computer tests an image to see if it contains a face. This would display the cascading part of the algorithm, by showing that you can check the most highly weighted feature against the image first and anywhere it does not exist - there must not be a face. It would also be useful to allow the student to manually adjust the parameters for the face detection algorithm. This will allow the student more insight into why a face is detected or why it might not be detected.

A suggestion by one of the teachers who tried this interactive was that it might be useful to show all of the Haar-like features of an image by clicking a button. Another suggest was that a slider that amplified the contrast in the image might help the students to visualize where features might exist. Other developers of the Computer Science Field Guide suggested that the user should be able to choose between multiple photos that have already been uploaded. These photos could included faces that might not be recognised (profile or wearing a hat and glasses) which will encourage the students to figure out why this is happening. Future work could look into the benefits of these suggestions and decide if the implementation of any of them might help the students to understand the concept.

For future work in the computer vision chapter of the Computer Science Field Guide, focusing on one of the other topics discussed earlier would be beneficial. Face recognition would be the obvious next choice but further research would be needed to find a way to teach it without focusing too much on the complex maths behind it.

References

[1] T. Bell and J. Morgan, "Computer science field guide - computer vision." http://www.csfieldguide.org.nz/en/chapters/computer-vision.html. Accessed: 22/03/2017.

[2] A. Bell, "Github - pixel play." https://github.com/andybelltree/cs-field-guide/pull/1. Accessed: 23/03/2017.

[3] T. Bell, C. Duncan, S. Jarman, and H. Newton, "Presenting computer science concepts to high school students," 2014.

[4] T. Bell and H. Newton, *Improving Computer Science Education*, ch. Unplugging Computer Science, pp. 66–81. 2013.

[5] C. T. Fosnot and R. S. Perry, *Constructivism: Theory, perspectives, and practice*, ch. Constructivism: A psychological theory of learning, pp. 8–33. 1996.

[6] P. Battistella and C. v. Wangenheim, "Games for teaching computing in higher education–a systematic review," *IEEE Technology and Engineering Education*, vol. 9, no. 1, pp. 8–30, 2016.

[7] G. Bebis, D. Egbert, and M. Shah, "Review of computer vision education," *IEEE Transactions on Education*, vol. 46, no. 1, pp. 2–21, 2003.

[8] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Computer Vision and Pattern Recognition*, 2001.

[9] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pp. 586–591, IEEE, 1991.

[10] C. Harris and M. Stephens, "A combined corner and edge detector.," in *Alvey vision conference*, vol. 15, pp. 10–5244, Citeseer, 1988.

[11] L. W. Kheng, "Computer vision and pattern recognition." http://www.comp.nus.edu.sg/ cs4243/lecture/motion.pdf. Accessed: 26/03/17.

[12] Udacity, "Computer vision." https://classroom.udacity.com/courses/ud810/lessons/3554318679. Accessed: 26/03/17.

[13] E. Lundgren, T. Rocha, Z. Rocha, P. Carvalho, and M. Bello, "tracking.js." https://trackingjs.com/. Accessed: 30/04/17.

[14] T. Adeyemi, "interact.js." interactjs.io. Accessed: 30/04/17.

[15] NZQA, "Achievement standard as91636 - digital technologies 3.44," 2017.