

AR Bridge Builder: Real-time Vectorisation of Freehand Sketched Structures for an AR Application

F. Petzold¹, J. Pfeil¹, C. Riechert¹ and R. Green¹

¹Dept. Computer Science, University of Canterbury, Christchurch, New Zealand.

Web: <http://arbridgebuilder.grundeis.net>

Abstract

In this paper the idea of an augmented reality game involving the overlay of information on freehand bridge drawings is presented. This is supported by a proposed algorithm which merges vectorisation and corner detection, followed by a post-processing of the resulting bridge graph. Different implemented approaches to recognise the sketches are discussed. The Hough transform is found not to be well suited to cope with the problem. Whereas simple vectorisation gives better results but still has some drawbacks in terms of precision and reliability. This paper proposes a hybrid algorithm to overcome these drawbacks by using corner detection in addition to vectorisation.

Keywords: vectorisation, freehand sketches, skeletonisation, hough transform, thinning, augmented reality

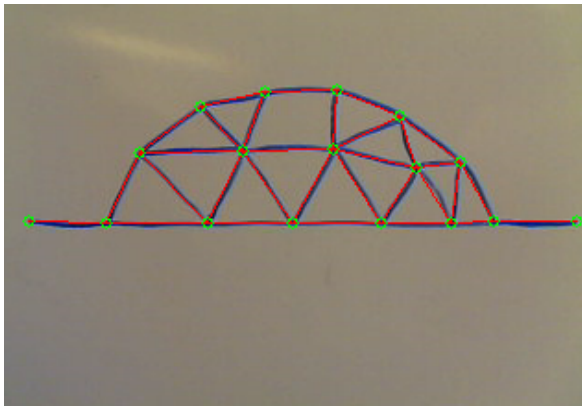


Figure 1: Recognised bridge

1 Introduction

Sketching is a very straight forward way to express and test out ideas. Recognised structures and features of sketches can be used to generate additional information which is overlayed on the video image. The idea presented in this paper is to overlay physics and game information on real world drawings. The proposed algorithm forms the base of an Augmented Reality (AR) game that can be played on a normal white board or on paper. It could be used for education, giving learners instant feedback on the validity of their ideas.

The game is about building bridges that can support itself and traffic. A bridge is drawn on a whiteboard or on paper. The user points the camera at the sketch and starts the simulation – revealing the statics and game information which is

overlayed in real time on the video. A selection tool is used to choose a track that a simulated train then runs over. This allows the user to playfully discover more about the statics of the drawn bridge.

As a first step, the elements of the drawn bridge have to be recognised in the video image. In the game a bridge only consists of connected straight lines. Thus, as a final result the bridge sketch should be represented as a graph containing the joint positions of the bridge and the trusses between them.

In this paper several possible algorithms to achieve this are investigated. The first approach uses the Hough transform to detect the trusses. Due to several draw-backs concerning robustness to noise and bad recognition of slightly curved lines, the Hough transform approach is found inappropriate. It also fails to extract the main information about the bridge structure, which does not lie in the trusses, but rather in joint positions and their connectivity.

A second approach using thinning and vectorisation was developed, which could directly extract the joint positions. Thinning and vectorisation reliably preserve joint connectivity but may alter joint positions due to known issues of several thinning algorithms, which will be explained. To overcome the inaccurate extraction of joint positions, we decided to implement a hybrid approach, which combines vectorisation with better joint position recognition using the Harris corner detector.

The final algorithm was realised using C++ and the Open Computer Vision Library (OpenCV) [1].

1.1 Related Work

The proposed AR game is heavily inspired by a PC game called BridgeBuilder [2] which allows the player to construct a bridge with small straight trusses which is then physically simulated and tested by a train running over it.

The illustration of physics information using free-hand sketches has been demonstrated by Microsoft Research's Physics Illustrator [3]. This system needs a special input device to capture the strokes. It can either be run on a tablet PC or use a digital whiteboard combined with a projector.

Algorithms for freehand sketch recognition exist but usually the interest lies in labelling strokes according to some model in order to extract meaning from a sketch (e.g. [4]). For our application the exact position and connectivity of structures is important and the underlying model is rather simple. Also the most robust sketch recognition has been achieved with special input devices capturing the process of drawing (e.g. [5, 6]). At least in the general case we do not have this additional time information which makes reliable detection much harder.

Recognition of scanned documents like maps and technical drawings is another area where robust vectorisation is required [7, 8]. But here the resolution is usually high and the lighting is ideal.

The novelty of the proposed application is the use of a simple camera which does not give the same amount of information as a tablet PC or digital whiteboard. In order to be able to use it in an AR application the computer vision algorithm has to be very robust under different lighting conditions, varying line thickness and discontinuities in the drawn lines.

2 Hough Transform

The Hough transform [9] can be used for straight line detection. It transforms the image into Hough space where lines are represented as points. The coordinates of every point in the Hough space correspond to the distance and the angle of a normal representing one line in the original image. The structure that is to be recognised is ideally a straight line. Every white pixel in the original image votes for every possible line that could go through it.

There are several variations of the Hough transform. The Probabilistic Hough transform, for example, uses a selected subset of the input points to speed up the algorithm [10].

The Hough transform has some known disadvantages and problems. If there are too many lines,

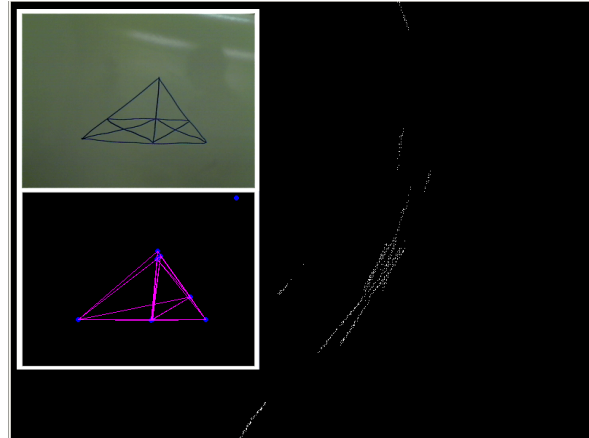


Figure 2: Big image: Hough space, Small images: Original image and coarsely recognised bridge

they can no longer be extracted from the Hough space. Longer lines outvote shorter ones. Parallel lines which lie close together cannot be distinguished.

For pre-processing the images, the Canny edge detector is used. It enables further processing to be fairly independent of varying lighting conditions [11].

The Hough algorithm is very noise prone. The noise of a fixed camera is enough to make stable line recognition on live video impossible. Lines disappear due to noise making it necessary to lower the recognition threshold. This results in line bundles instead of single lines. A clustering algorithm was implemented which groups lines by their proximity in the Hough space. From every group the line with the highest number of votes is chosen.

With this clustering algorithm it was possible to get a stable real-time recognition of structures consisting of a small number of lines. These lines were intersected to obtain a list of candidate junction points. If a rectangular area of the thresholded input image between two candidate points contains a number of white pixels that is higher than the minimum number of pixels a line through this area would have, the two candidate points are considered to be connected by a line in the real image.

2.1 Problems

As soon as the drawings become more complex or the variety of line lengths too wide, the algorithm is no longer able to extract all lines (see fig. 2). The problem with the standard Hough transform lies in the fact that pixels vote globally. The more lines are in an image the harder it becomes to extract single lines from the Hough space.

A possible solution could be a modified version of the Hough transform that divides the image into small regions. Each of these regions is transformed

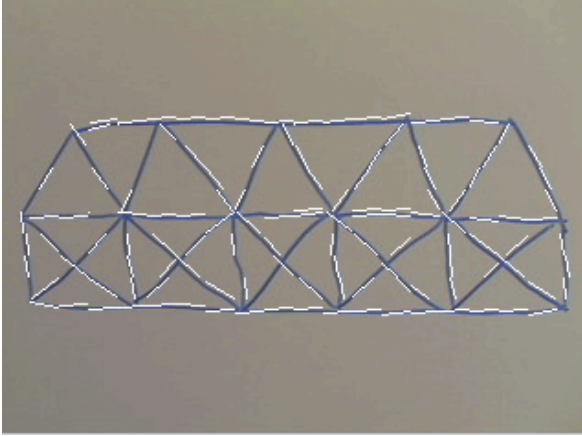


Figure 3: Visualisation of Hough transform applied to image subsections

independently. In every of these Hough spaces the global maximum defines a line segment (fig. 3). These local line segments could then be joined together. This way, long lines cannot outvote shorter ones.

Even if the problems inherent to the Hough transform could be overcome, it proved not to be the best possible approach to the problem at hand. Lines drawn are often slightly curved resulting in imprecise junction point detection. It could also be desirable to recognise the exact shape of drawn elements. An alternate approach overcoming the shortcomings of the Hough transform is discussed in the next section.

3 Vectorisation

Another possible method is to use skeletonisation to obtain a thin line which can then be vectorised. This also enables the detection of arbitrary curved structures. The skeletonisation is done with a thinning algorithm proposed by Zhang [12] after applying an adaptive threshold to the original image (see fig. 6). A great advantage of thinning over other kinds of skeletonisation algorithms is its guaranty of a connected skeleton.

Since Zhang’s thinning algorithm does not always result in an 8-connected line, an algorithm to remove staircases in diagonal line segments, as proposed by Holt [13], is applied. This results in a thinned 8-connected line which is ideal for vectorisation. Although thinning is potentially slow, it does not pose a problem in our case. The bridge structure does already consist of relatively thin lines, which constrains the maximum number of iterations necessary for the algorithm to converge.

The vectorisation is performed by simply adding each line pixel as a node to a graph structure. If two line pixels are adjacent, their corresponding nodes are connected by an edge. The output of

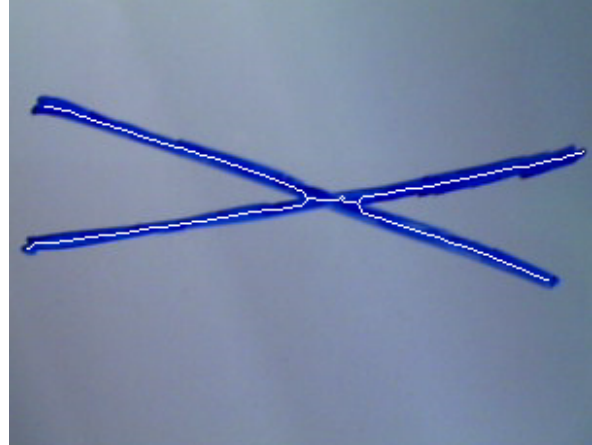


Figure 4: Necking artefact produced by the thinning algorithm

the vectorisation algorithm is a graph representing the drawing. It obviously contains nodes which do not correspond to junctions in the sketch. To remove non-critical points a polygonal approximation algorithm is applied.

An error function, shown in equation (1), calculates the cost of removal for every point that has two edges. The cost is the distance between each node and the line connecting its neighbours. At each iteration the point with the lowest cost is removed and its neighbours are directly connected. The algorithm terminates if there are no more points undercutting a certain threshold.

$$C_n = \left| \vec{P}_n - \vec{P}_{n-1} \right|^2 - \frac{(\vec{P}_n - \vec{P}_{n-1}) \cdot (\vec{P}_{n+1} - \vec{P}_{n-1})}{\left| \vec{P}_{n+1} - \vec{P}_{n-1} \right|^2} \quad (1)$$

Polygonal approximation yields a graph whose nodes are either junctions or sharp bends in a line (e.g. triangle).

3.1 Problems

There are two major drawbacks of this approach, concerning the thinning algorithm. Firstly, necking artefacts appear at junctions where lines meet at an acute angle (fig. 4). Secondly, extra line segments are often created if the underlying shape is not regular. These artefacts result in the introduction of additional junction points.

Furthermore the adaptive threshold used in the creation of the binary image can result in gaps in the centre of the junction of thick lines (fig. 5). This can make correct vectorisation impossible.

Extra line segments introduced by the thinning algorithm can be addressed by removing nodes with a single edge, whereas the other artefacts cannot be as easily dealt with. Necking artefacts and

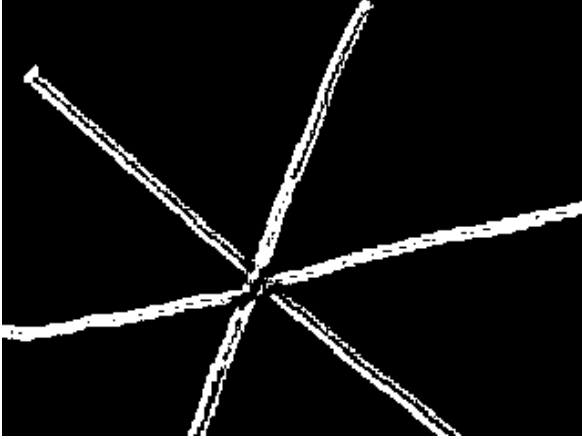


Figure 5: Artefacts caused by adaptive thresholding

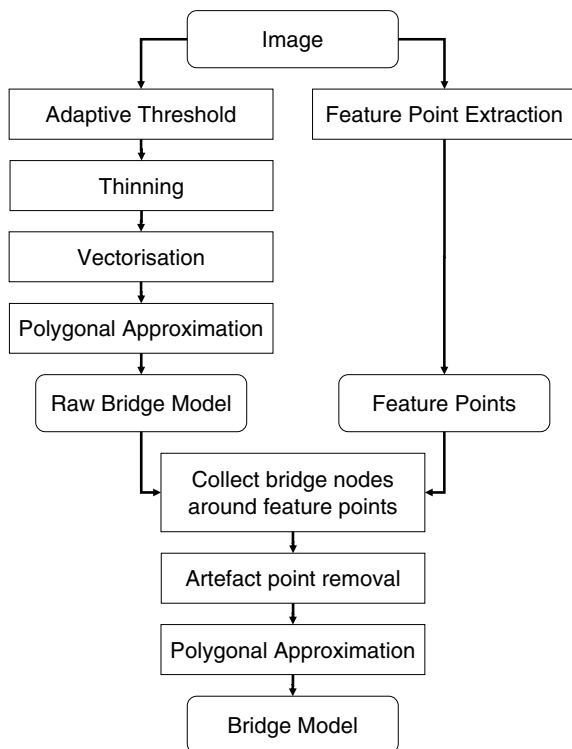


Figure 6: The final algorithm

gaps introduced by adaptive thresholding result in decreased reliability at junctions. Furthermore this crude vectorisation algorithm can result in imprecise junction positioning. To overcome these drawbacks, we propose a hybrid solution, which uses Harris Corner detection in addition to vectorisation.

4 Final Algorithm

Junctions in the drawing can be recognised very reliably using corner detection. The final algorithm (see fig. 6) uses the Harris corner detector [14] to obtain a set of feature points. These features are then used to collect bridge nodes in their neighbourhood (fig. 7). For each feature point a new

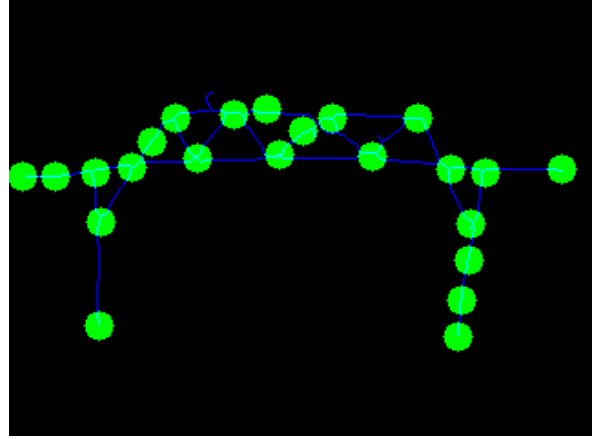


Figure 7: Thinned drawing with overlaid feature points

node is created and all edges attached to nodes within a certain range are transferred to the newly created node. The now unconnected nodes are removed.

At this state, tightly connected groups of nodes can still exist. They are caused by holes in the lines between feature points and cannot be removed by polygonal approximation, because they have more than two edges. To break up these clusters a processing step called "artefact point removal" is applied. Every node not created by a feature point is removed if another node is within a certain distance. The removed node's edges are transferred to the nearest neighbour. After breaking up these clusters, the polygonal approximation algorithm is invoked again.

4.1 Corner Detection

Harris corner detection is a very common way to obtain distinct features in an image. To obtain the possible corner points the structure tensor T shown in equation (2) has to be calculated for each point in the image. I_x and I_y are the derivatives of pixel intensity in x and y direction, respectively.

$$T = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix} \quad (2)$$

Corners are detected by searching for local maxima of R which is calculated by equation (3) with $\kappa \approx 0.04$.

$$R = \det(T) - \kappa \text{trace}^2(T) \quad (3)$$

5 Discussion

The described algorithm was created to solve the special problem of recognising bridge sketches on a whiteboard. An algorithm suitable for our problem

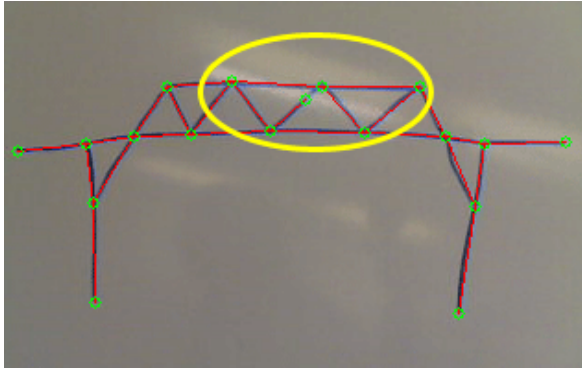


Figure 8: Reflections

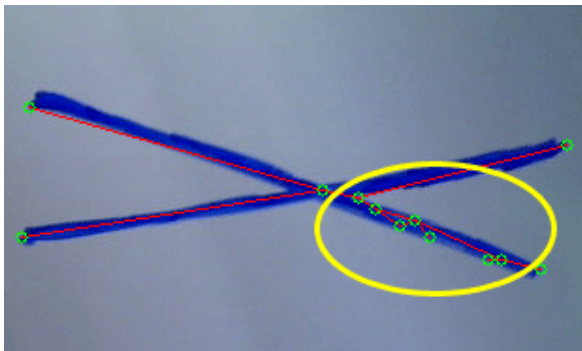


Figure 9: Thick lines

does not only have to be fast but also more robust than algorithms working on still images from scanners. Video camera images are usually of low resolution and contain significant noise. Furthermore, fast movements of the camera yield motion blur on the image. These constraints make reliable recognition a challenging problem.

To test the algorithm we used a fixed set of still images exhibiting critical properties like poor lighting and marginal cases such as short and long lines, acute and obtuse angles and thin and thick lines. In addition we continuously tested with live video.

The algorithm can deal with light reflections on the whiteboard. But there are also cases like in fig. 8 where reflections are too strong for proper recognition. By decreasing the kernel size of the adaptive threshold the algorithm becomes less affected by brightness variations. But smaller kernels can cause holes in thick lines. The kernel size has to be chosen in correspondence to the thickness of the lines and the distance to the drawing plane. This should be done dynamically, which is not yet implemented. As a consequence we can currently only detect 1-5 pixel wide lines. Fig. 9 shows that thick lines cause the algorithm to fail.

Corner detection fails for small distances to the drawing plane. This could probably be overcome by using scale spaces. Furthermore the threshold used to collect nodes around the corner points implies a minimal distance between junctions. This

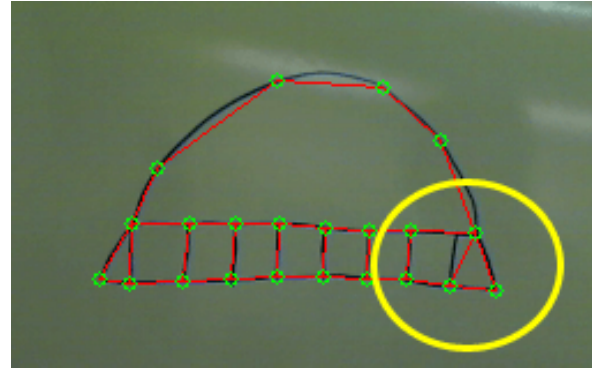


Figure 10: Small structures

effectively prevents the recognition of any structures smaller than this minimal distance. A marginal case can be seen in fig. 10.

The mentioned limitations prevent the proposed algorithm from being a general solution for straight line sketch recognition. But it works accurately, reliably and in real time within the constraints of the project it was built for.

6 Conclusion and Outlook

In this paper we presented the first step in the design of an augmented reality game involving the overlay of static information on freehand bridge drawings. This first and most important step was the development of an algorithm for the recognition of bridge sketches. The Hough transform proved to be inappropriate to extract the wanted information. Simple vectorisation gives better results but still has some drawbacks in terms of precision and reliability. The presented hybrid algorithm overcomes these drawbacks by using corner detection in addition to vectorisation.

The proposed solution is able to recognise bridge sketches for a broad scale of lighting conditions and distances and works in real time. Further improvements can be achieved by adding variable thresholds and scale spaces. This would allow for a large range of distances, differently sized or even mixed-sized structures.

The result of the presented algorithm is a model of the sketched bridge which is used by the AR Bridge Builder game to simulate the bridge's physics and overlay the simulation results on live video.

References

- [1] Intel, "OpenCV Open Source Computer Vision Library," <http://sourceforge.net/projects/opencvlibrary/>, visited on 4/10/2007.

- [2] Chronic Logic, “Bridge Builder,” <http://www.chroniclogic.com/>, visited on 9/9/2007.
- [3] Microsoft Research, “Microsoft Physics Illustrator,” <http://www.microsoft.com/downloads/details.aspx?familyid=56347faf-a639-4f3b-9b87-1487fd4b5a53&displaylang=en>, visited on 9/9/2007.
- [4] J. Mahoney and M. Fromherz, “Three main concerns in sketch recognition and an approach to addressing them,” in *AAAI Spring Symposium on Sketch Understanding*, 2002, pp. 105–112. [Online]. Available: citeseer.ist.psu.edu/mahoney02three.html
- [5] B. Yu, “Recognition of freehand sketches using mean shift,” in *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*. New York, NY, USA: ACM Press, 2003, pp. 204–210.
- [6] T. M. Sezgin and R. Davis, “Hmm-based efficient sketch recognition,” in *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*. New York, NY, USA: ACM Press, 2005, pp. 281–283.
- [7] R. Janssen and A. Vossepoel, “Adaptive vectorization of line drawing images,” *CVIU*, vol. 65, no. 1, pp. 38–56, January 1997.
- [8] Jinyang and Yumei, “Automatic extraction of contour lines from scanned topographic map,” in *Geoscience and Remote Sensing Symposium, 2004. IGARSS '04. Proceedings.*, vol. 5. IEEE International, 2004, pp. 2886–2888.
- [9] P. Hough, “Method and means for recognizing complex patterns,” US Patent No. 3069654, 1962.
- [10] J. Matas, C. Galambos, and J. Kittler, “Progressive probabilistic hough transform.” 1998. [Online]. Available: <http://dblp.uni-trier.de/db/conf/bmvc/bmvc1998.html#MatasGK98>
- [11] J. Canny, “A computational approach to edge detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, 1986.
- [12] T. Y. Zhang and C. Y. Suen, “A fast parallel algorithm for thinning digital patterns,” *Commun. ACM*, vol. 27, no. 3, pp. 236–239, 1984.
- [13] C. M. Holt, A. Stewart, M. Clint, and R. H. Perrott, “An improved parallel thinning algorithm,” *Commun. ACM*, vol. 30, no. 2, pp. 156–160, 1987.
- [14] C. Harris and M. Stephens, “A combined corner and edge detector,” in *Proceedings of The Fourth Alvey Vision Conference*, Manchester, 1988, pp. 147–152.