Mr Painting Robot

Samuel Grant Dawson Williams
Computer Science and Software Engineering
University of Canterbury
Christchurch, New Zealand
Email: samuel@oriontransfer.org

Richard Green
Computer Science and Software Engineering
University of Canterbury
Christchurch, New Zealand
Email: richard.green@canterbury.ac.nz

***Abstract*—This paper discusses the identification of outlines, and how these can be used for input into a creative painting algorithm. It builds on several existing algorithms including the bilateral filter and laplacian edge detection, and develops several new algorithms including intensity scanning and brush stroke formation. The benefit of this approach is that the output format is vector based (and thus suitable for robotic reproduction), rather than pixel based as in the majority of prior research on painterly rendering. Synthetic tests show that the entire process can reproduce sumi-e style images with over 90% accuracy.**

I. Introduction

Extracting significant details from images is an important field of computer vision. The human vision system is very good at this, but computers don't have the body of knowledge required for complex separation and classification of visual features. "Mr Painting Robot" is an idea that a robot can be created to interpret and paint a scene creatively, including four primary elements: colours, outlines, shapes and textures.

A painting robot would ideally be a robotic arm with both a camera and a paint brush attached. The arm would use the camera to look around its environment and choose something to paint. The painting materials would be manually supplied to the robot in pots which would then be applied to a canvas using a brush.

The successful implementation of a painting robot will depend primarily on robust brush stroke formation. This research will focus on the computer vision challenge of extracting significant edges and mapping these to brush strokes artistically.

II. Background

Painting is a fundamental form of expression which encompasses many discrete actions. Painting a picture is a process (to some degree) of rationalising the human visual system into philosophical decisions and physical motions. As artists, we interpret what we see, and create something new using our experiences and aspirations.

The difficulty of establishing an empirical model for this process lies with the ambiguity of the task. A wide range of techniques have been explored, including water-colour rendition [1], simulated brush strokes [2], [3], [4], pen and ink rendering [5], [6]; there is no right or wrong way to paint a picture [7] - it is a highly subjective act where the results may delight some and frustrate others. The choice of colour, or the direction of brush strokes may be the difference between perceived success or failure.

Fig. 1. The sample image.

In this research, we will be considering only a single tool - a paint brush. This means that given any input image, the output must be a set of strokes, each stroke being made up of a sequence of connected points with associated thickness and colour. While the discussed techniques relate to painterly rendering in general, we are not overly concerned with producing an image-based result typical of other research in this area.

A. Detecting Edges

There are many methods to extract shapes and lines from an image. In order to discuss several existing approaches, a sample image (see figure 1) will be used. There are several elements of this image which make it notable: significant textured and non-textured areas, same colour surfaces which change based on lighting conditions and surfaces with the same intensity value but differing colour.

The Canny algorithm [8] presents a robust method to detecting edges in an image. This can be useful when we are interested in the shape of an object, but information such as line thickness and intensity is lost (see figure 2a).

The Hough transform [9] can identify known features in an image. It can be used to detect straight lines, even when they lack continuity. However, it does not help us in the situation where we want to detect lines of unknown curvature, and thus not so suitable for extracting brush strokes.

(a) Canny edge detection

(b) Grey-scale Laplacian edge detection

(c) 3-Channel Laplacian edge detection

Fig. 2. Various edge detection algorithms.

Fig. 3. The bilateral algorithm applied to the original sample image.

The Laplacian operator [10] can be used to analyse an image for edges in a way that preserves thickness and intensity (see figure 2b). This gives a useful result for further analysis of such edges and their conversion to brush strokes.

To reduce the amount of small details extracted by edge detection, the bilateral filter [11] can be applied to the colour image to reduce texture (see figure 3). Large contrasting edges which typically represent the most important visual features remain unchanged.

Edges may occur as both changes in colour and changes in intensity. By applying the Laplacian operator to each colour channel separately, far more information about visual features can be extracted, including edges that exist between colours of equal intensity (see figure 2c). With this approach, a strong edge is white (detected in all three colour channels), and a weaker edge might only be detected in one or two colour channels.

B. Extracting Lines

To paint a line, we start from a single point and extend the line based on direction, thickness and curvature. In typical artistic reproductions, we cannot rapidly adjust these properties, and so we need to apply some cost function to the search space of brush strokes. This is a slightly different goal from existing image vectorisation techniques [12], [13], [14], [15].

The Snakes algorithm [16] can fit a line to a contour using a cost function. Randomly generated brush strokes could be matched to the processed image, however for large images with many lines this approach may be inefficient and give many overlapping lines with no guarantee of complete coverage.

The A* [17] algorithm can also be used to compute connectivity information. However, it requires a given start and end point for heuristic cost calculation, and can also have significant performance issues as it back-tracks through the search space.

C. Artistic Interpretation

In an abstract artwork, pertinent shapes and objects are often emphasised, while background shapes and context may be ignored completely. The human visual system is supremely well adjusted to the environment that we exist in and thus classification is not an arduous task; however computer vision algorithms have no intrinsic ability to classify objects in a scene and to rank their overall importance. Semantic classification of objects can be used for brush stroke selection and other stylistic decisions [18], but these techniques are not a primary focus of this research. Therefore, we will avoid any kind of local classification and focus on producing brush strokes accurately for all features in a given image.

In pictures painted using only a small number of colours, we have a limited ability to separate objects based on colour and texture alone. Brush stroke contour, thickness and direction can separate objects, but without the ability to recognise discrete objects, isolating brush strokes to individual objects is challenging. Computer vision algorithms are typically unable to identify one object from another without advanced knowledge and classification systems, which can make it difficult to establish accurate line connectivity. Therefore in this respect we will simply pay particular attention to the form of important edges, while ignoring irrelevant texture and detail as much as possible.

In all painted artworks, a sense of depth can be created by the artist by understanding the relative positions of

Fig. 4. An example of the intensity scanning process with R = 5.

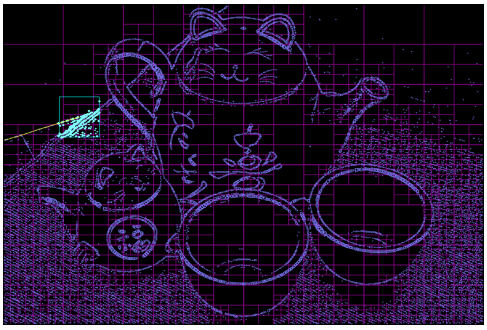


Fig. 5. An example of the quad-tree containing a set of intensity circles. An example of a line being formed can be seen.

objects in a scene. Lighting and shadows pay a large part in establishing the correct interpretation of an artwork, as well as geometric effects such as occlusion. Some computer vision systems allow for the analysis of depth, but robust systems for analysing lighting and shadows do not yet exist. In general, we will rely on our analysis of the source image to provide sufficient visual cues for depth related phenomenon, however further interpretation could assist with isolating important visual elements such as shadows and continuous segments which have been occluded.

III. Method

A brush stroke is a sequence of points, where each point specifies an (x, y) coordinate and the width of the brush stroke at that point. With this information, we can create a basic painting robot. Additional information such as colour can be computed from the source image or generated dynamically.

A. Intensity Scanning

The image is initially processed by firstly applying the bilateral filter, and then applying the 3-channel Laplacian operator to extract edge information.

An intensity scanning algorithm performs a sequence of evenly distributed vertical and horizontal scans on the image. The distance between scanlines is dictated by R (the inverse scan resolution). Each pixel is tested for intensity using a normalised Euclidian distance function:

$$Intensity(R, G, B) = \frac{\sqrt{R^2 + G^2 + B^2}}{\sqrt{3}}$$

We then extract a set of ranges of high intensity (see figure 4). A range consists of the start of a high intensity pixels, and the length that this high intensity sequence continues for.

For each extracted range, a circle is constructed with centre point x, y from the middle of the range, and diameter being the length of the range. This circle is then checked for coverage by calculating the number of high intensity pixels.

$$\frac{PixelsInCircle(image, circle, tolerance)}{AreaOf(circle)}$$

If the coverage is not greater than a given minimum percentage, we reduce the size of the circle by a small amount and repeat this test. We continue until the coverage is satisfactory; we discard circles that become too small.

B. Forming Lines

After processing the input as above, we have a set of circles covering areas of high intensity (see figure 5). We could simply paint each circle an appropriate colour, and we would be done. However, we wouldn't have achieved much more than a dot matrix printer style reproduction. Joining the circles together to form lines allows us to start producing an artistic result.

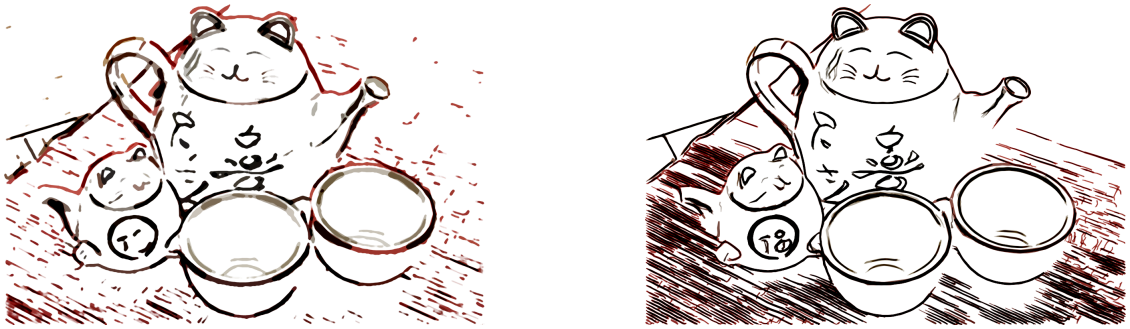
There are a number of criteria to consider depending on the desired output:

- We might have a preference for longer strokes rather than short strokes. Many short brush strokes might create unwanted texture.
- It might be difficult or unnatural to frequently move the brush at odd angles. Brush strokes potentially look better when they are straight or gently curved.
- Reduce overlapping brush strokes. If the output is a wet ink - we can only cover the same spot a certain number of times.

To build a line, we use a greedy search. We pick any circle at random, and then build a set of potential candidates for connection based on Euclidian distance. We then feed these candidates through a cost function, and insert the results into a heap. After processing all possible candidates, we select the best choice from the top of the heap. We continue to build the line until there are no further candidates within a certain cost threshold.

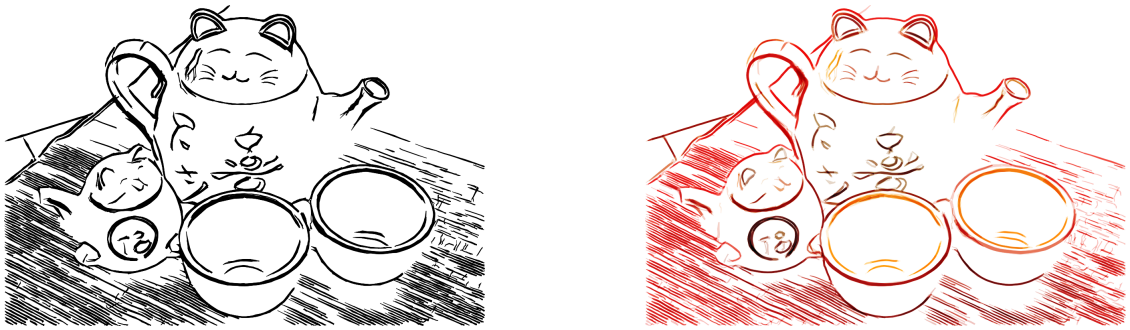
After processing the line as far as we can in one direction, we reverse the line and continue processing. This has the effect that the line ultimately ends up being as long as possible in both directions from the starting point.

A quad-tree is used to improve the speed of space queries. It can also be used to perform pruning of the initial data-set, which can reduce the generation of lines that overlap and improve performance.



(a) Round Brush Strokes (b) Stylised Brush Strokes

Fig. 6. Various Brush Styles



(a) Sumi-e Colour Rendition (b) Colour Boost Rendition

Fig. 7. Various Colour Renditions

C. Cost Functions

The line generation algorithm is highly sensitive to the cost function used. There are several criteria calculating the cost between an existing line and a candidate point:

- Keep the line as straight as possible.
- Ensure that the line contains all points along the given contour.
- Avoid crossing colour boundaries.
- Avoid crossing low intensity edges (i.e. gaps).
- Avoid rapidly changing line thickness.

Based on these criteria, we can formulate many different cost functions. An example of a simple cost function is given:

procedure SIMPLECOST(*image*, *line*, *point*)
  $d \leftarrow$ DistanceBetween(EndOf(*line*), *point*)
  if Size(*line*) ≥ 2 **then**
    $a \leftarrow$ AreaOfTriangle(LastSegment(*line*), *point*)
  else
    $a \leftarrow 1$
  end if
  $i \leftarrow$ IntensityBetween(EndOf(*line*), *point*, *image*)
  return $d \times a + \frac{1.0}{i}$
end procedure

D. Simulated Painting

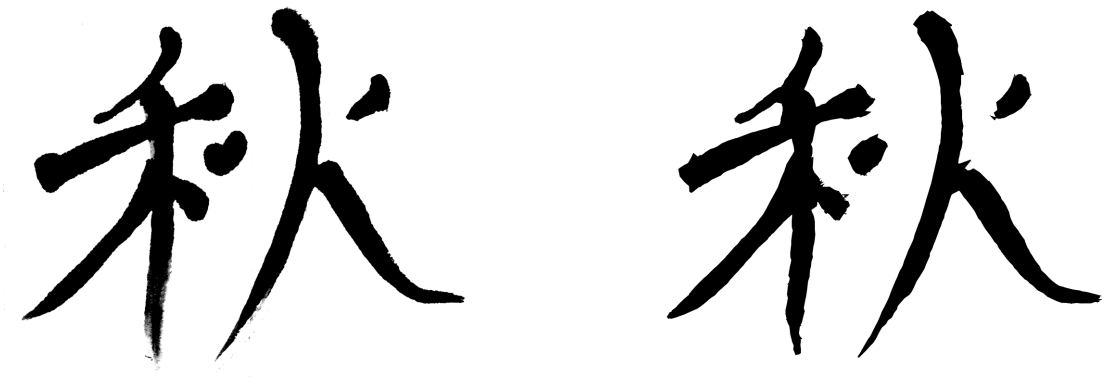
Due to lack of a suitable robot, painting has been simulated to test accuracy.

A water colour style [1] is difficult to simulate accurately. Thus, we will opt for a basic subtractive colour model with alpha accumulation and blending. This will be sufficient for a creative visualisation and testing purposes.

Two brush stroke algorithms were implemented (see figure 6): the circle brush uses polygons to render the sequence of circles based on the measured diameter at each point, while the spline brush uses Hermite spline interpolation for the edges of the brush stroke and a stylised stroke outline (tapered entry, rough exit) to more accurately simulate a simple brush.

There are several options for choosing colour (see figure 7). Traditional sumi-e renditions use a strong black ink. Other options include processing the colour at each point along the brush stroke i.e. average, median, or point sampled colour.

As an experiment to increase the liveliness of the rendition, a colour boost model was implemented, where approximately every 3rd brush stroke would have the saturation and brightness of its colour boosted. This was done by converting the average colour along the brush stroke into HSV, boosting the appropriate values, and then converting back into RGB for output.



(a) Original Image (b) Reproduced Image

Fig. 8. The input and result of running the intensity scanning algorithm during the synthetic accuracy test.

IV. Results

Although the overall intrinsic result qualitatively appears good, the challenge is to quantify artistic perception.

A. Performance

The general performance of the algorithm is highly dependent on both the efficiency of space queries and the cost function. The initial implementation did not use space partitioning and the performance was $O(N^2)$. The use of a quad-tree improves the general performance of the algorithm to $O(N \log N)$.

The scan resolution also heavily affects the performance of the scanning algorithm as more pixel data must be processed:

	Intensity Scan		Stroke Generation	
R	Circles	Time	Strokes	Time
1	40189	2.0s	2640	110s
2	20162	1.0s	1214	27s
4	10031	0.47s	570	8.6s
8	4995	0.27s	238	2.5s
16	2458	0.11s	67	0.67s

In this case, the number of strokes generated is roughly proportional to the number of circles generated.

B. Accuracy

The general accuracy of the intensity scanning algorithm is good. If we run the algorithm at $R = 2$ or $R = 4$ we get high accuracy as well as good performance. However, as we increase the scan resolution (e.g. $R = 1$), we also increase the chance of overlapping brush strokes.

Accuracy can be measured by running the algorithm on a black and white input image, and stroking the output paths using a stylistic brush stroke (see figure 8) . We can then measure the ratio of correctly coloured pixels:

	Spline Brush Accuracy		
R	White	Black	Difference
1	0.9858	0.9691	0.0167
2	0.9884	0.9648	0.0236
4	0.9904	0.9580	0.0324
8	0.9925	0.9347	0.0578
16	0.9940	0.8207	0.1733
	Circle Brush Accuracy		
R	White	Black	Difference
1	0.9871	0.9581	0.0290
2	0.9899	0.9473	0.0426
4	0.9913	0.9268	0.0645
8	0.9933	0.8893	0.1040
16	0.9946	0.7462	0.2484

In this test, we want to maximise both black and white accuracy, however as this algorithm is ultimately given some degree of creativity, 100% accuracy may also not be desirable.

V. Conclusion

The performance and accuracy of the algorithm are generally good. Sample images were processed in under a minute and produced pleasing output. In the synthetic case, accuracies in excess of 90% were achieved. In general, images can be systematically converted to brush strokes, and this data is suitable for input to a robotic painting system.

A. Limitations

In rendering the final set of brush strokes to a canvas, overlapping layers of ink may produce undesirable disfigurement of the painting surface. Several approaches to culling have been explored, but more work in this area is necessary to ensure that culling does not affect accuracy. Culling can also be useful as a way to improve the efficiency of the line formation process, by pruning the number of possibilities that need to be considered when forming lines.

Fig. 9. A processed photograph of a Bonsai tree.

Fig. 10. A processed photograph of a Buddha.

Due to the nature of artistic reproductions based on brush strokes, various visual features will be lost in the painting process. The goal of this algorithm is to retain the most important features in a way that is aesthetically pleasing, however due to the lack of a precise definition, there will always be some subjective elements in the quality of the results. Further work in the area of object identification could improve the ability of the algorithm to form brush strokes that match human expectations.

B. Future Work

The output of this algorithm is suitable for input to a robotic function, but such a robot has yet to be constructed and programmed.

It is hard to conceive of an ideal cost function, and thus experimentation in this area could continue to yield improvements to accuracy and artistic value.

The algorithm currently uses input from an intensity image created using edge detection. However, this is not the only kind of input that could be used to create artistic imagery. Several approaches have been considered for future experimentation:

- Use edge detection as currently implemented, but also fill in contiguous regions of specific colours for high intensity coverage (i.e. black).
- Use an image pyramid to detect large regions of consistent colour, and then use this information for 'washes' (large brush strokes which apply a small amount of colour).

References

[1] C. J. Curtis, S. E. Anderson, J. E. Seims, K. W. Fleischer, and D. H. Salesin, "Computer-generated watercolor," in *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997, pp. 421–430.

[2] M. Shiraishi and Y. Yamaguchi, "An algorithm for automatic painterly rendering based on local source image approximation," in *NPAR '00: Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*. New York, NY, USA: ACM, 2000, pp. 53–58.

[3] A. Hertzmann, "Painterly rendering with curved brush strokes of multiple sizes," in *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1998, pp. 453–460.

[4] M. Obaid, R. Mukundan, and T. Bell, "Enhancement of moment based painterly rendering using connected components," in *CGIV '06: Proceedings of the International Conference on Computer Graphics, Imaging and Visualisation*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 378–383.

[5] G. Winkenbach and D. H. Salesin, "Rendering parametric surfaces in pen and ink," in *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1996, pp. 469–476.

[6] M. P. Salisbury, M. T. Wong, J. F. Hughes, and D. H. Salesin, "Orientable textures for image-based pen-and-ink illustration," in *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997, pp. 401–406.

[7] R. M. Pirsig, *Zen and the Art of Motorcycle Maintenance: An Inquiry into Values*.

[8] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, pp. 679–698, Jun. 1986.

[9] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, pp. 11–15, January 1972.

[10] E. C. Hildreth, "Edge detection," Cambridge, MA, USA, Tech. Rep., 1985.

[11] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*. Washington, DC, USA: IEEE Computer Society, 1998, p. 839.

[12] X. Hilaire and K. Tombre, "Robust and accurate vectorization of line drawings," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 6, pp. 890–904, 2006.

[13] P. Selinger, "Potrace," [Online; accessed June-2010]. [Online]. Available: http://potrace.sourceforge.net/

[14] T. Xia, B. Liao, and Y. Yu, "Patch-based image vectorization with automatic curvilinear feature alignment," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 1–10, 2009.

[15] J. Sun, L. Liang, F. Wen, and H.-Y. Shum, "Image vectorization using optimized gradient meshes," *ACM Trans. Graph.*, vol. 26, no. 3, p. 11, 2007.

[16] A. W. Michael Kass and D. Terzopoulos, "Snakes: Active contour models," vol. 1, no. 4. Springer Netherlands, January 1988, pp. 321–331.

[17] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics SSC4 (2)*, pp. 100–107, 1968.

[18] K. Zeng, M. Zhao, C. Xiong, and S.-C. Zhu, "From image parsing to painterly rendering," *ACM Trans. Graph.*, vol. 29, no. 1, pp. 1–11, 2009.