Visualizing Batik

Brian Wyvill* University of Calgary
Kees van Overveld† Eindhoven University of Technology
Sheelagh Carpendale‡ University of Calgary

Figure 1: Simulated batik using our crack generation algorithm

Abstract

We present an algorithm for simulating the Batik wax painting and dyeing technique used to make images on cloth. The algorithm produces cracks similar to those found in batik due to the wax cracking in the dyeing process. The method is unlike earlier simulation techniques used in computer graphics, in that it is based on the Distance Transform algorithm rather than on a physically based simulation such as using spring mass meshes or finite element methods. Such methods can be difficult to implement and computationally costly due to the large numbers of equations that need to be solved. In contrast, our method is simple to implement and takes only a few seconds to produce convincing patterns that capture many of the characteristics of the crack patterns found in real Batik cloth.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms;

Keywords: Non-Photorealistic Rendering, cracking, distance transform, batik

1 Introduction

Batik painting is an ancient art probably originating in India or the Middle East as many as 2000 years ago. However, it is usually associated with the Indonesian islands of Java and Bali [Fraser-Lu 1991]. Liquid wax is painted onto cloth and the cloth is then dyed. The dye does not affect the cloth where it is painted with wax. This process can be repeated with new colours to add to the complexity of the image on the cloth. The wax can optionally be removed from the cloth by scraping and washing between the dye and paint cycles. During the process, cracks can form in the wax allowing dye to seep into the cloth. As the cloth undergoes successive dying procedures the cracks age and become wider. Other subtle effects take place such as cracks becoming wider at junctions with other cracks. These effects combine to form crack patterns that are immediately recognisable as batik. The motivation in this work was to produce batik like cracks in a 2D image as a post process operation. The algorithm has been implemented as a filter in a paint system, so that an artist may paint a simulated wax image and then apply colour and cracks simulating the dyeing and cracking process. In this way a series of layered images simulating wax painting can be combined to produce a final simulated Batik painting. Following this introduction the paper is organised into the following sections: previous work, visual characteristics of Batik, distance transforms, crack simulation, controlling the visualization process, dye simulation, results and conclusions.

2 Previous Work

The work we present here is part of the increasing number of new rendering approaches which have been loosely grouped as Non-Photo Realistic Rendering (NPR). Most of these have looked to either fine arts such as painting [Hertzmann et al. 2001], [Curtis et al. 1997] and drawing styles [Winkenbach and Salesin 1994], [Sousa and Buchanan 1999], [Kalnins et al. 2002] or cartooning [Kowalski et al. 1999]. Our work is in the same spirit in that we are interested in producing an expressive rendering method, however we differ in that we present a new NPR technique that is based on batik, a traditional craft approach. Woodcuts [Mizuno et al. 2000] and mosaics [Kim and Pellacini 2002] are also examples of crafts that have inspired rendering techniques.

In recent years there has been a great interest in modelling the crack formation process in various different media, usually for the purpose of animation. Previous work in the area of fracture formation have concentrated on either simulating the process of breaking an object or on the formation of a crack pattern. An example of shattered models was done by Norton et al. [Norton et al. 1991], in which a tea pot was discretized into cubes, the physical properties were simulated as a mass-spring mesh, and the springs would break when their elastic limit was exceeded. The actual crack patterns were quite coarse in this model since they were composed of fairly large sized cubes. Terzopoulos and Fleischer [Terzopoulos and Fleischer 1988], investigated the modeling of visco-elastic and plastic deformations. They used a finite differencing technique, and applied it to torn paper and ripped cloth. The method requires that a surface be discretized on a rectangular grid. The resulting fractures exhibited an aliasing effect caused by the imposed grid. Realistic cracks and material behaviour were produced by Metaxas and Terzopoulos [Metaxas and Terzopoulos 1992] using a finite element method for animating deformations. O'Brien and Hodgins [O'Brien and Hodgins 1999] also developed a method for fracture modelling based on continuum mechanics, in which the material

*e-mail: blob@cpsc.ucalgary.ca
†e-mail:k.van.overveld@wxs.nl
‡e-mail:sheelagh@cpsc.ucalgary.ca

deformation was expressed using strain tensors. The continuous model was then discretized into tetrahedral meshes and the finite element method applied.

Although these physically based methods can produce highly plausible visual results, they are difficult to program and solving the matrices associated with a larger finite element mesh can take considerable compute time, making fast response in an interactive situation difficult to achieve. O'Brien and Hodgins [O'Brien and Hodgins 1999], for example, report computing times of several minutes per frame; in contrast our crack formation process takes about 4 seconds on a resolution of 1000 squared. O'Brien and Hodgins point out in [O'Brien and Hodgins 1999] that engineering and computer graphics requirements are different in that engineers wish to use their simulations to make predictions about the real world whereas users of computer graphics are only concerned about how the cracks appear visually.

In our work we go a step further and use an algorithm that achieves good results but does not claim to follow physical simulation principles. Instead it uses only one very simple algorithm, the Distance Transform [Jain 1989]. Our approach should therefore be considered more in the context of non-photorealistic rendering rather than in physically based modelling.

The interface to the software was written so that artists could use this technique to add a crack texture to an existing battery of filters. For this reason we built the "crack filter" as a Photoshop plugin.

Figure 2: Real Batik. Image Courtesy the Batik Institute.

3 Visual Characteristics of Batik

Figure 2 shows an example of real Batik painting. The inset image is a close up taken from within the white rectangle, which shows

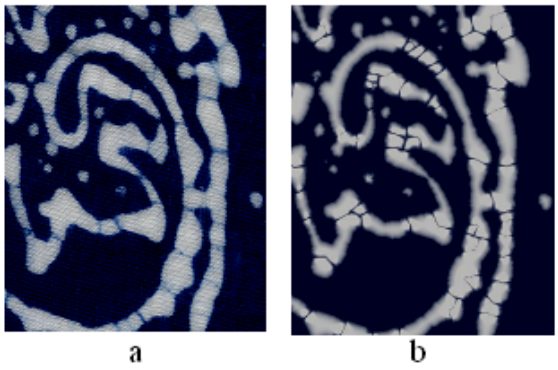

Figure 3: Image a. real batik cracks, b. generated cracks. Each crack tends to move towards the concave regions on the boundary.

the characteristic crack patterns found in Batik. There are a number of factors which affect the appearance of the cloth. Firstly the dye is absorbed non-homogenously by the cloth. Introducing some noise into the image can produce a reasonable simulation of this effect. Cracks are formed in time order, rather than all at once, so a crack will run until meeting the edge of the wax or another crack either of which will stop it. It can be observed in the close-up image that there are few places where one crack crosses another, but there are many T-junctions. Cracks often end in maximally concave regions of the wax border, as depicted in Figure 3, possibly because it is energetically cheaper for the cracks to be shorter. Figure 3*a* shows the real cracks and Figure 3*b* shows the simulated cracks. It can be seen that both exhibit this property. Cracks erode, therefore older cracks are wider than younger cracks. At T-junctions the wax becomes fragile at the edges and tends to break off, widening the cracks in this region. There are many types of wax used by the various different batik manufacturers, often the wax formulae used are a closely guarded secret. For example, unlike Indonesian batik, Chinese batik is made using mostly beeswax, which resists cracking when dyed, avoiding the crack patterns that the algorithms described in this paper attempt to reproduce.

In our simulation the user first generates one or more wax images, which will be combined to form the finished artwork. In our case they are digital images, which encode the characteristics of the wax in the colour channels (see section 7). A wax image is loaded into the simulator, cracks are produced, simulated dye is applied and the result is composited with the final image.

A visual simulation of these crack properties can be produced using variations of the distance transform algorithm.

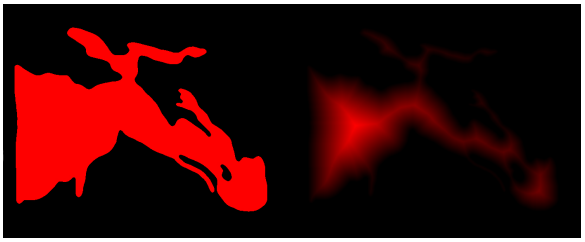

Figure 4: Sample wax image (left), distance transform (right)

4 Distance Transforms

In this and the following sections we use the following definitions: Ω defines the image domain. W defines the part of Ω that is covered with wax, and the set V consists of the border of W together with all the cracks.

In order to achieve a plausible distribution of cracks, we start from the following observations: (1) newer cracks run from one older crack to another older crack (where the border of the wax domain also counts as one or more cracks), and (2) cracks serve to alleviate the tension in the wax. Therefore tension is close to zero near an older crack, so new cracks should mainly run through regions that are far removed from older cracks.

In order to achieve (2), we need to know, in every point of Ω, how far the nearest older crack is. A simple way to do that, is by using a distance transform. Given a subset V of W, the distance transform $D(p)$ in a point $p \in W$ is defined as

$$D(p) = MIN(v : v \in V : |p - v|),$$

in other words: it is the distance to the point v in V that is nearest to p; $|p-v|$ is the distance between p and v according to some metric.

The value of $D(p)$ depends on the distance metric chosen. Using an L^2 metric is preferred in many applications as it provides a rotationally invariant measure. Brute force methods to calculate a rotationally invariant distance transform require $O(|\Omega||V|)$ calculations. Various work has been done on such methods to increase the efficiency [Gavrilova and Alsuwaiyel 2001].

For our application however, experiments show that we don't really need rotational invariance. A simple implementation using L^1 metric, based on Jain's algorithm [Jain 1989] gives results that are visually just as appropriate with the additional advantage of much simpler code and considerably more efficient execution.

The Jain algorithm is a two-pass algorithm with complexity $O(|\Omega|)$ as follows:

```
for(pixel p in V) D(p)=0;
for(pixel p in Omega-V) D(p)=infinity;
for(pixel p: scan from top left to bottom right){
  for(pixel n in upper left half Neighbourhood(p))
    if(D(n)+|n-p|>D(p))D(p)=D(n)+|n-p|;
  }
  for(pixel p: scan from bottom right to top left){
    for(pixel n in lower right half Neighbourhood(p))
      if(D(n)+|n-p|>D(p))D(p)=D(n)+|n-p|;
  }
```

For some applications, such as modeling the widening of older cracks, not only do we need to know the distance to the nearest feature but also the identity of this feature. (A "feature" here means a crack or the border of W). To this aim, we use an extended version of the distance transform [Jain 1989]; we call this the *identity transform* (IDT). Assume that every point p in Ω has a label, $\lambda(p)$. The labels for points in V are taken from a set $\Lambda = \{\lambda_0, \lambda_1, \lambda_2, \ldots\}$. The algorithm for IDT is as follows:

```
for(pixel p in V) {
  D(p)=0;
  lambda(p)= element from Lambda;
}
for(pixel p in Omega\V) {
  D(p)=infinity;
  lambda(p)= 'undefined';
}
for(pixel p: scan from top left to bottom right){
  for(pixel n in upper left half Neighbourhood(p))
    if(D(n)+|n-p|>D(p)) {
      D(p)=D(n)+|n-p|;
      lambda(p)=lambda(n);
  }
}
// and similar for the second pass
```

The IDT effectively computes a discrete sampling of the Voronoi diagram for the currently used metric, where the seeds may be arbitrarily shaped subsets of Ω.

We use the following features of the distance transform in our crack simulator:

- $D(p)$ is the shortest distance between p and any earlier feature in Ω;
- $\lambda(p)$ is the identity of the nearest feature to p;
- $D(p)$ and $\lambda(p)$ allow local updating in case of local modifications of V (in other words, as soon as the next younger crack is formed, we only have to re-compute $D(p)$ and $\lambda(p)$ in a subdomain of Ω that is roughly equal to the bounding box of the recent crack);
- For closer approximations to the L^2 metric, we can simply adjust the definition of "Neighbourhood". Larger neighbourhoods give better approximation to circular (D(p)=const)-contours in case of a point-shaped subset V (at the expense of slower execution).
  - 3 x 3, 4-connected: square;
  - 3 x 3, 8-connected: 8-gon,
  - 5 x 5, 24-connected: (16-gon),
  - 7 x 7, 48-connected: (32-gon, etc.)

5 Crack Simulation

In this section we describe the process of forming a pattern of cracks in a layer of wax. A wax-processing step of the algorithm is as follows:

```
crack initialisation;
do{
  find suitable starting point;
  for(both halves of the crack){
    do{
      find propagation direction;
      apply random perturbation;
      make a step;
    } until arrival on earlier feature
  }
  update D(p) and lambda(p) near recent crack;
} until amount of cracks set by user.
```

In the initialization phase a distance transform and an IDT are applied to the areas which have been covered by wax, W. The non-covered area, $\Omega - W$, is assigned $D(p) = 0$ and $\lambda(p) =$ "undefined".

A sample wax mask (in red) and its distance transform is shown in Figure 4, the distance values in the right-hand image have been scaled and assigned to the red channel.

In our approach to crack propagation, we assume that cracks serve to alleviate the stress distribution in the wax. The energetically cheapest way to do this, is to have cracks of minimal length. So there are two boundary conditions: (1) as explained in section 4, a crack should pass through a point that is at (locally) maximal distance from any earlier crack, since there the stress is (locally) maximal; and (2) a crack should propagate as fast as possible to the nearest feature (i.e. earlier crack or border of the wax). The latter will ensure that begin and end points of new cracks are likely to begin and end at a point of local maximal curvature of older cracks, which conforms to the physical behaviour of cracks.

In order to steer cracks in the direction of the steepest descent of $D(p)$, we need the gradient of $D(p)$. As well as storing $D(p)$ and $\lambda(p)$ at each pixel, the gradient $\vec{\nabla}D(p) = (\frac{\partial D(p)}{\partial x}, \frac{\partial D(p)}{\partial y})$ of the distance transform is also calculated and stored. The derivatives are

computed on the grid locations using finite differences; to mitigate aliasing, we first low-pass filter $D(p)$.

Once distance values and their gradients have been computed the crack propagation algorithm is initiated by first choosing a seed point inside the wax region. A random point q inside W is chosen (see also section 6). From q, we follow a path in the direction of increasing $D(p)$ to find a local maximum of $D(p)$. Let q' be this local maximum.

From the point q', the crack is propagated simultaneously in two (normalized) directions $\vec{d}, -\vec{d}$. We find $\vec{d}$ as the direction of the steepest descent of $D(p)$.

Let the current crack point be p_c. p_c is either on the horizontal or vertical edge between two adjacent pixels, so p_c has either an integer x-coordinate and a floating point y-coordinate, or vice versa. From p_c downhill, $\vec{\nabla}D(p)$ is used to determine the direction of propagation, say $\vec{r}$, by following the gradient. To do this, $\vec{r}$ is evaluated in p_c at sub-pixel resolution by linearly interpolating the values of $\vec{\nabla}D(p)$ in the two pixels adjacent to p_c. We intersect the line $p_c + \mu\vec{r}$ with the four sides of the square shown in figure 5, consisting of p_c's neighbouring pixels and their appropriate opposites. The intersection point that is closest to p_c will be the next point on the crack.

Propagating the crack as outlined above, would lead to very straight cracks. To get the characteristic wrinkled appearance, we perturb $\vec{r}$ with a filtered Brownian motion noise. Both the noise amplitude and the filtering strength are put under user control. The noise amplitude is locally controllable (see section 6); the filter strength is a global parameter.

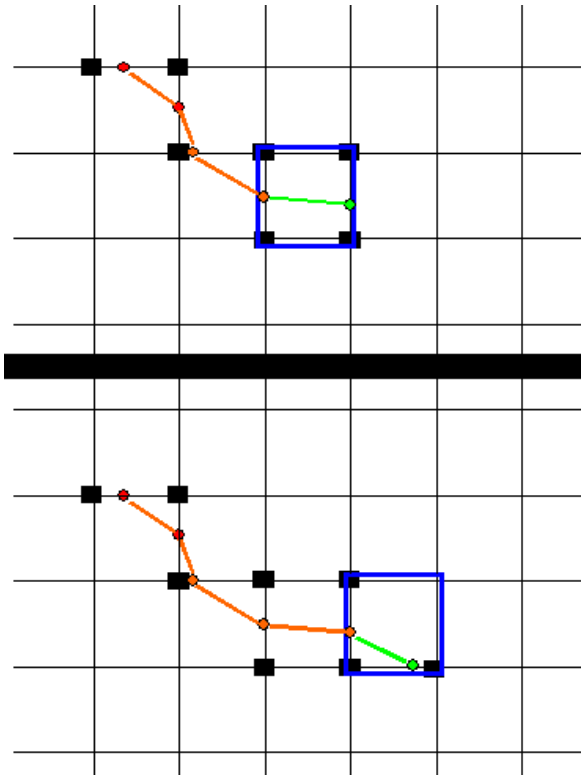

Figure 5: Calculating the direction of crack propagation.

In Figure 5 the pixels indicated with black squares are labelled as "on a crack". Their λ-values receive a serial number that encodes the age of the crack. If there are n cracks required, the λ-value in the pixels that are "on" crack number i are set to $n-i$. After completion of this crack, the IDT propagates these λ-values to the environment. So, after the IDT, all pixels have the age of the nearest crack set in their λ-value.

The distance values $D(p)$ of points "on" the crack are sub-pixel accurate; in Figure 5, the existing crack is shown as the red polyline with sub-pixel coordinates; the blue squares indicate pixels used for computing $D(p)$ and the next positions of p_c. So the presence of all subsequent cracks is entirely encoded in the maps $D(p)$ and $\lambda(p)$.

In section 7 we explain how the age of the crack, as encoded in $\lambda(p)$, is used to account for an age-dependent width of the dye track.

We account for one further effect that is the widening of the T-junction between a crack and an older crack. To this aim, every point on a crack should also know its distance to the two features (typically: T-junctions with earlier cracks) that cause it to terminate. In order to compute this distance, we use the fact that we know (roughly) in advance how long a crack is going to be. Indeed, the value of $D(q')$ in the starting point q' tells us this. During propagation over (one half of) a crack, we keep track of the distance travelled. In all the points p on a crack we administrate a value $d2j(p)$ (short for "distance to junction"), defined as :

$$d2j(p) = MAX(0, len(p,q') - D(q') + T).$$

Here, "len" is the distance travelled between q' and p. So for points p that are further than a certain threshold, say T, from their destination, $d2j(p) = 0$. If they are closer than T to the threshold, they have a linearly increasing value, with a maximum of T on the junction.

With a separate run of a distance transform algorithm, we also propagate the values of $d2j$ to the environment of cracks. This time, however, we decrease rather than increase the distance attribute. So after this distance transform, all pixels have a valid d2j-attribute which is 0 at locations far from T-junctions, which is maximal on the T-junctions, and which linearly decreases with the distance to T-junctions.

Once a crack has been propagated, aged and T-junctions administered, the cycle is repeated for the next crack.

6 Controlling The Process

To control the batik simulation the user supplies a series of images representing the wax masks and the simulation computes the cracks and composites the images into an output image. There are several controls that can be applied to the cracking process. Local control is achieved by defining the distribution of the values of crack width ($d(p)$), crack density ($\rho(p)$), and crack randomness (*wiggliness* $w(p)$) as functions of the location p. In order to provide these values the user can set the red ($d(p)$), green ($\rho(p)$), and blue ($w(p)$) distributions in the input image.

Figure 7 shows the effects of varying each of these parameters. The top map of Canada has a smooth red ramp varying from 100% red in the East to 0% in the West. The variation of $d(p)$ in the wax image can be seen in the top image of Figure 7: the cracks are wider in the East. The apparent width of a crack is not stored explicitly but is calculated as part of the dye application process (see 7).

The density parameter ($\rho(p)$) is used to control the probability that a crack is created. For example, in regions of low $\rho(p)$ fewer points are accepted and therefore fewer cracks generated. The algorithm to find the crack starting point q first selects a random point that lies within W. The density value, $\rho(q)$ ranging from 0 to 1, is compared against a uniform random number $r \in [0:1]$. If $\rho(q)^3 > r$ the point is accepted. The cube was found empirically to give better results than a linear or square relationship.

The parameter $w(p)$ is used to control the local direction taken by the crack. In Figure 7, the cracks appear straighter in the West and

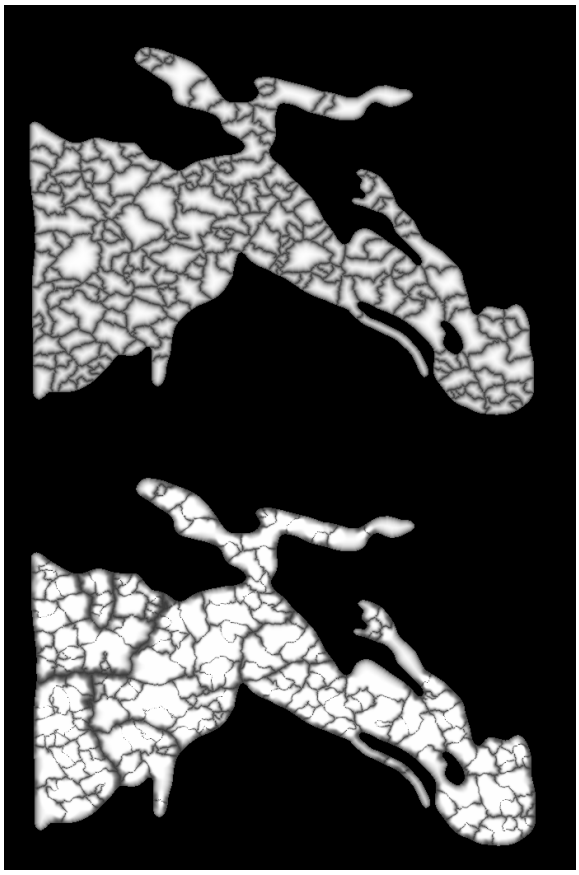

Figure 6: Crack before (top) and after (bottom) aging and widening due to T-junctions.

follow a more random (wiggly) path in the East as $w(p)$ increases from West to East.

There is also a global randomness control, which can be used to vary the direction of all the cracks. Details of how the path direction is calculated are given in section 5. Other global controls are provided by varying parameters such as overall crack width (i.e. a scaling factor affecting all cracks) and smoothness using an interactive user interface.

In real batik the wax is often left on the cloth until the final washing stage when all the wax masks are removed from the cloth. In some cases it may be desirable to remove a wax image from the cloth in between paint/dye cycles. In this case the area of cloth influenced by the wax would receive more than one dye application. We also simulate this process.

7 Dye Simulation

Applying the algorithm from section 5, using the control mechanisms as described in section 6, produces a map $D(p)$ that represents, in every point p, the distance to the nearest crack (or the border of W). It also produces a map $\lambda(p)$ that represents the age (i.e., the sequence number) of the crack nearest to p. In the present section we describe how a colour is assigned to p that corresponds to the application of dye of a given colour to the cloth with the present wax distribution.

There are two aspects to colour assignment: first, the computation of the colour that results from mixing the dye colour with the existing colour (which may be the original cloth colour or the colour that resulted from applying earlier dyes) at non-protected spots, and the computation of the intensity of the dye at a given location taking the wax distribution into account.

7.1 A Multiplicative Colour Model for Dye Application

We adopt a multiplicative colour model for dye colouring (see Figure 8), since it more closely models the interaction between dyes, as they are applied one over the other.

This means that we associate a set of spectral reflection coefficients to the cloth, say $r_i(p)$ for frequencies ν_i and location p, and a set of spectral transmission coefficients, say $t_{i;j}(p)$ to layer of dye number j. The resulting spectral distribution that results when illuminating the batik with a light source with spectral distribution s_i is given by

$$s_i r_i(p) \Pi_j {t_{i;j}}^2(p),$$

for all i. The square accounts for the fact that the light passes twice through each layer of dye. In our implementation, we sample the visible light spectrum with 3 samples: $i = 1,2,3$. In order to represent the simulated batik in standard RGB format, we choose pure spectral red, green and blue for our colour primaries ν_1, ν_2 and ν_3. In order to visualise the resulting batik, the pure spectral components are approximated by the red, green and blue of the current colour reproduction device.

The spectral transmission coefficients $t_{i;j}$ have a negative exponential dependency on the thickness of the dye layer (i.e., the amount of dye per unit area). We approximate this exponential dependency by a linear dependency on the dye concentration. Therefore we introduce an effective dye concentration distribution, say $c_j(p)$ for dye number j at location p. In the next section we explain how $c_j(p)$ is computed from $D(p)$ and $\lambda(p)$. For now we mention that $c_j(p)$ is everywhere a number between 0 (no dye) and 1 (maximal dye concentration). With $c_j(p)$, we compute

$$t_{i;j}(p) = 1 - c_j(p)(1 - t_{i;j0}),$$

where $t_{i;j0}$ is the spectral transmission coefficient for dye number j at frequency ν_i when it would be applied with maximal concentration.

7.2 Geometrical Aspects of the Dye Distribution

In order to compute $c_j(p)$, we adopt the assumption that dye is somewhat absorbed in cloth. This absorption gives rise to a roughly negative exponential distribution near the locations where $D(p) = 0$. We approximate the negative exponential decay by a truncated linear function (i.e., for distances between 0 and some threshold d from the side of the crack the concentration linearly decays from 1 to 0; for distances further than d, the concentration stays 0). As explained in section 6, d is allowed to vary as a function of the location p, hence we use $d(p)$.

We can compute $c_j(p)$ from $D(p)$ by setting:

$$c_j(p) = MAX(0, 1 - \frac{D(p)}{d(p)}).$$

With this formula, all cracks would give rise to dye tracks of equal width if $d(p)$ is constant. In order to take crack aging into account, we set

$$c_j(p) = MAX(0, 1 - \frac{D(p)\lambda(p)}{d(p)\lambda_0}),$$

where λ_0 is the scale parameter which globally controls crack width (see section 6). The age of the nearest crack is encoded in $\lambda(p)$ such that older cracks correspond to larger values.

Finally we need to account for T-junction widening. In section 5 it was shown how the value of $d2j(p)$ is computed to represent the distance to the nearest T-junction. T-junction widening is calculated in a similar way to crack aging, by introducing a further multiplication factor, $d2j(p)$.

8 Results and Conclusions

The example shown in Figure 9 was made by first drawing a series of six wax masks, which together form the outline of the moose head. Figure 9a. shows the first wax mask (inset) and the resultant image generated by running the crack simulation. A variation on Perlin noise [Wyvill and Novins 1999] has been used to simulate non-homogenous dye absorption. A new wax mask has been added in Figure 9b. and the mask used in Figure 9c. is an inverse mask providing cracks in the background. The final moose head is the result of combining all 6 masks in six passes through the crack simulation.

In the final image of Figure 9 some of the wax masks have been left in place until the end of the process, whereas others have been removed. For example the wax mask of Figure 9a., was not removed until the end of the whole process and the area under it has a complex crack pattern, the result of multiple passes through the crack simulation.

In section 3 it was mentioned that cracks often end in maximally concave regions of the wax border, as shown in Figure 3. The wax mask for Figure 3*b*, was generated by posterizing an image of the real batik shown in Figure 3*a*, and generating a uniform full intensity red as the wax mask. The generated and real cracks have an overall similarity in position. The tendency to end in concave regions can be observed in both images and many of the cracks in image *a* have very similar counterparts in image *b*.

One of the key elements in producing a convincing visual simulation of the batik cracks is the process of aging and manufacturing wider cracks close to T-junctions as can be seen in Figure 9. For a comparison of cracks generated with and without using these effects see Figure 6.

From our experiments with our crack generation techniques we conclude the following:

1. The crack algorithm gives convincing batik-like cracks in flat-coloured, segmented images.

2. The various mechanisms in the batik process can be implemented in terms of few elementary geometric operations and image processing operations on a purely local basis.

3. All required algorithms can be straightforwardly derived from the distance transform.

We have implemented these techniques as plug-in software for Photoshop, adding another filter to provide a new tool for artists.

References

CURTIS, C. J., ANDERSON, S. E., SEIMS, J. E., FLEISHER, K. W., AND SALESIN, D. H. 1997. Computer-generated watercolor. In *SIGGRAPH '97*, ACM.

FRASER-LU, S. 1991. *Indonesian Batik: Processes, Patterns and Places*. Oxford University Press.

GAVRILOVA, M., AND ALSUWAIYEL, M. 2001. Two algorithms for computing the euclidean distance transform. *International Journal of Image & Graphics, Special Issue on Image Processing 1*, 4, 635–646.

HERTZMANN, A., JACOBS, C. E., OLIVER, N., CURLESS, B., AND SALESIN, D. H. 2001. Image analogies. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, ACM, 327–340.

JAIN, A. 1989. *Fundamentals of Digital Image Processing, Chapter 2*. Prentice-Hall.

KALNINS, R. D., MARKOSIAN, L., MEIER, B. J., KOWALSKI, M. A., LEE, J. C., DAVIDSON, P. L., WEBB, M., HUGHES, J. F., AND FINKELSTEIN, A. 2002. Wysiwyg npr: Drawing strokes directly on 3d models. *ACM Transactions on Graphics 21*, 3 (July), 755–762.

KIM, J., AND PELLACINI, F. 2002. Jigsaw image mosaics. *ACM Transactions on Graphics 21*, 3 (July), 657–664.

KOWALSKI, M. A., MARKOSIAN, L., NORTHRUP, J. D., BOURDEV, L., BARZEL, R., HOLDEN, L. S., AND HUGHES, J. F. 1999. Art-based rendering of fur, grass, and trees. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, ACM, 433–438.

METAXAS, D., AND TERZOPOULOS, D. 1992. Dynamic deformation of solid primitives with constraints. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, vol. 26, ACM, 309–312.

MIZUNO, S., KASAURA, T., OKOUCHI, T., YAMAMOTO, S., OKADA, M., AND TORIWAKI, J. 2000. Automatic generation of virtual woodblocks and multicolor woodblock printing. *Computer Graphics Forum 19*, 3 (August).

NORTON, A., TURK, G., BACON, B., GERTH, J., AND SWEENEY, P. 1991. Animation of fracture by physical modeling. *The Visual Computer 7*, 4, 210–219.

O'BRIEN, J. F., AND HODGINS, J. K. 1999. Graphical modeling and animation of brittle fracture. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, ACM, 137–146.

SOUSA, M. C., AND BUCHANAN, J. W. 1999. Computer-generated graphite pencil rendering of 3d polygonal models. *Computer Graphics Forum 18*, 3 (September), 195–208.

TERZOPOULOS, D., AND FLEISCHER, K. 1988. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *Computer Graphics (Proceedings of SIGGRAPH 88)*, vol. 22, ACM, 269–278.

WINKENBACH, G., AND SALESIN, D. H. 1994. Computer-generated pen-and-ink illustration. In *Proceedings of SIGGRAPH 94*, Computer Graphics Proceedings, Annual Conference Series, 91–100.

WYVILL, G., AND NOVINS, K. 1999. Filtered noise and the fourth dimension. In *SIGGRAPH 1999 Conference Abstracts and Applications*, Association of Computing Machinery, Computer Graphics Proceedings, Annual Conference Series, ACM, 242–242.

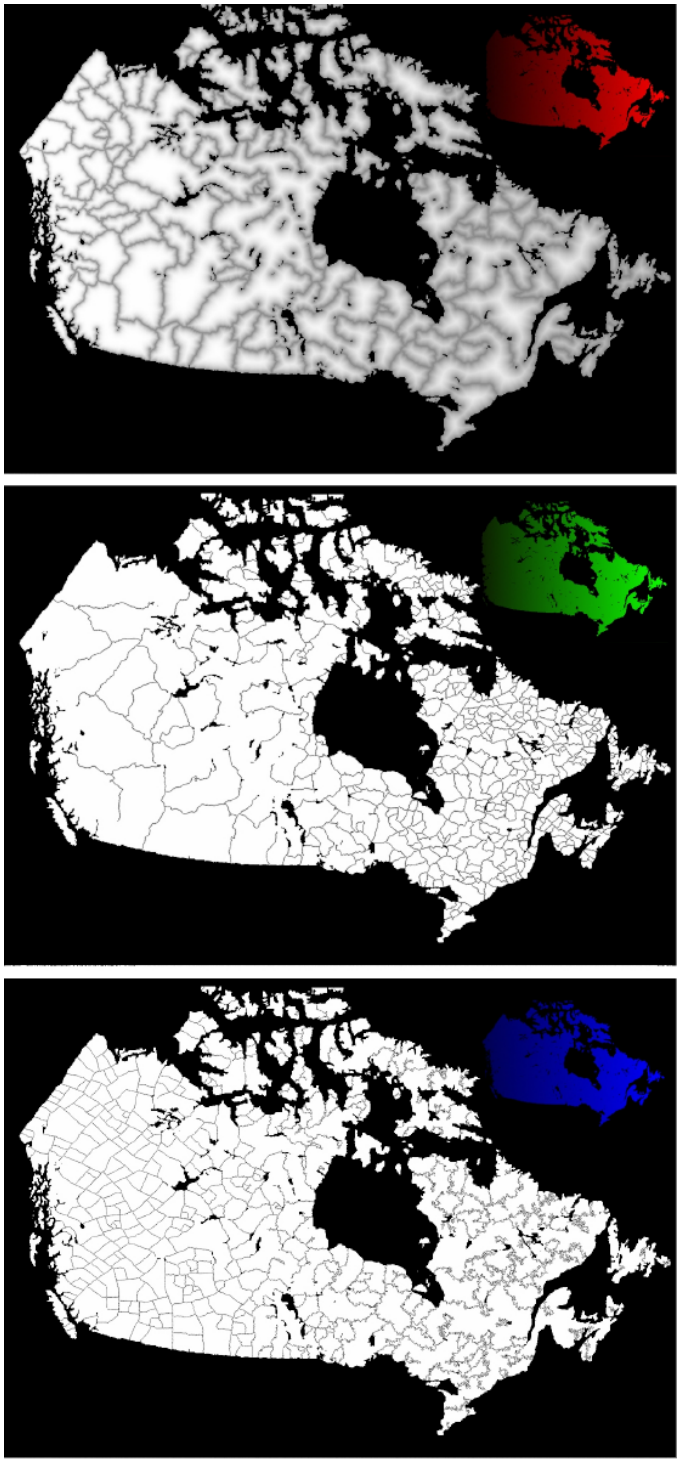

Figure 7: Each map of Canada is represented by a colour ramp going from West to East (0-255). Crack width (red), density (green) and randomness (blue) vary as the value of the input channel varies across each image.

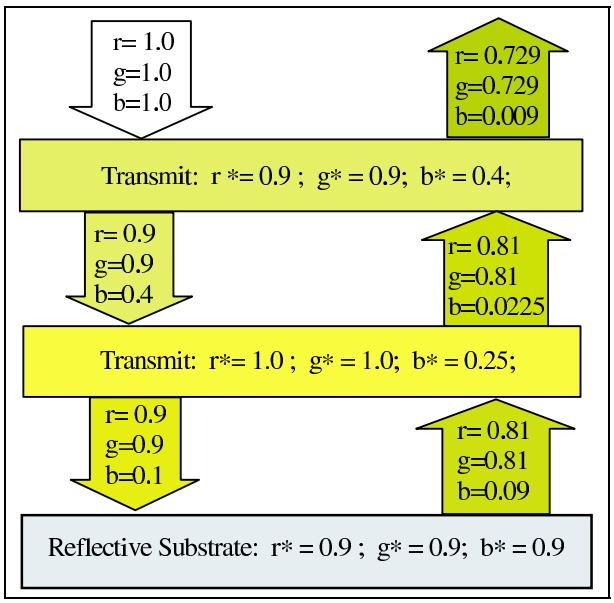

Figure 8: How Multiplicative Colour Works, dyes may be modelled as colour filters.

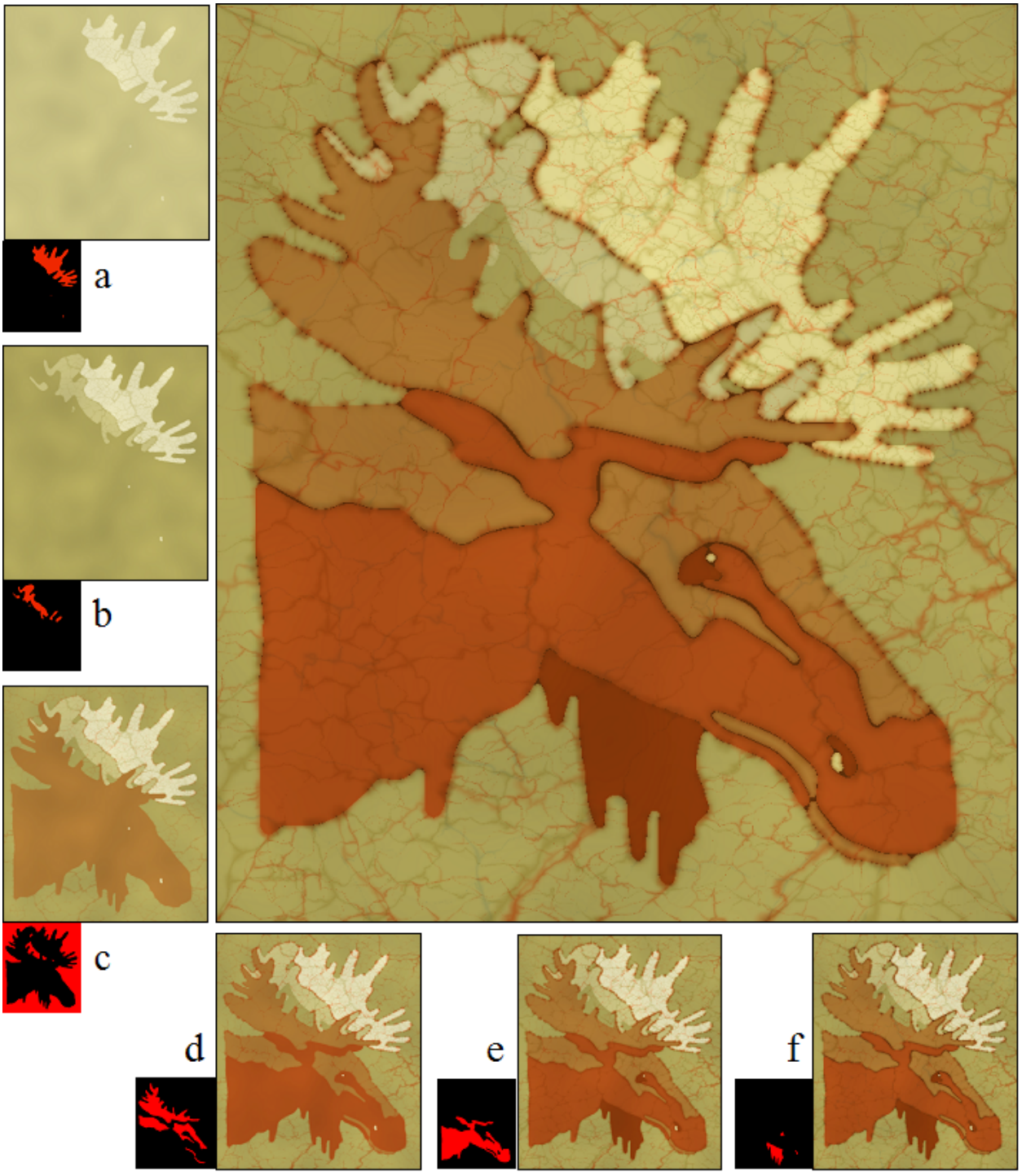

Figure 9: Figures 9a - f, show the stages in building the large image of the moose head. The other 6 images show how each part of the image is created after running the crack simulation on successive masks.