Sketch-Based Crowd Modelling

Li Guan **Burkhard C. Wünsche**

Graphics Group, Department of Computer Science
University of Auckland, New Zealand,
Email: lgua012@aucklanduni.ac.nz, burkhard@cs.auckland.ac.nz

Abstract

The creation of complex virtual worlds has expanded from the domain of designers and animators to that of general users with no background in computer graphics. Example applications are military simulations, urban planning, landscape design, search and rescue simulations, and social media technologies such as "Second Life". In many cases the user wants to create content containing hundreds or thousands of similar objects. Modelling and placing each individual object is infeasible and new ways must be found to allow users to easily specify the distribution of a large number of objects.

In this paper we introduce a sketch-based approach for crowd modelling, which is intuitive and suitable for different input devices such as mice, sketch pads, and touch screens (Windows 7). We derive design requirements by analysing real environments and by testing users' abilities to characterise crowds and collections/accumulations of objects. Based on these requirements we formulate a model-by-example approach in which users sketch a sample distribution of objects and our tool computes the complete "population" of objects over a domain specified with a sketched contour. In order to deal with different distribution patterns we first characterise the input and then use clustering and texture synthesis to replicate the characteristics over the domain. Initial results demonstrate that the tool gives plausible results for random, regular and clustered input and that it can be used in a wide variety of modelling applications.

1 Introduction

The use of virtual worlds and simulations is expanding rapidly and is now including such diverse applications as entertainment (games, movies), civil engineering, urban planning, visual impact studies, landscape design, social media, education and training, and military and civil defense simulations. In many of these applications large collections of objects are required such as crowds of people, forests of trees, or cities with thousands of houses and skyscrapers. Modelling and placing these objects by hand is time consuming and cumbersome and new tools for this process must be found.

Sketch-based interfaces for modelling are particularly attractive since they are intuitive (pen-and-paper metaphor), encourage creativity (Gross & Do 1996) and enable users to concentrate on the overall problems rather than details (Wong 1992). The past decade has seen a tremendous increase in the design and use of sketch-based interfaces, e.g., for 3D modelling (Igarashia et al. 1999, Joshi et al. 2010, Olsen et al. 2009), animation (Thorne et al. 2004, Li et al. 2006, Takahashi et al. 2005, Davis et al. 2003), gaming (Kloonigames Ltd. 2008), diagramming and interface design (Coyette et al. 2007, Schmieder et al. 2010, Plimmer et al. 2010), medical imaging (Ropinski et al. 2008), and robotics (Skubic et al. 2005, Sakamoto et al. 2009, Barber et al. 2010).

In this paper we investigate the use of sketch-based interfaces for crowd modelling. The term "crowd" is used here in a wider sense and refers to any large collection/accumulation/aggregation of objects. We investigate users' mental model of "crowds" and determine ways to characterise them. We then use these characteristics to model crowds by sketching example distributions over a user defined domain and then replicating the sketched characteristics using cluster analysis and texture synthesis techniques.

Section 2 reviews relevant previous work in sketch-based modelling, crowd modelling and texture synthesis. Section 3 derives the requirements for our application. Section 4 presents the design of the system. We evaluate our application in section 5 and conclude the paper with section 6, which also gives an overview of future work.

2 Literature Review

2.1 Crowd Modelling

A large variety of publications exists on crowd modelling and simulation. In general the authors are interested in the behaviour of a crowd given an initial configuration. The arguably most natural way to achieve this is to use an agent-based method where each character is given its own personal characteristics and goals. The concept has been initially used to simulate groups of animals (Reynolds 1987), but also proved effective for simulating human crowds (Funge et al. 1999, Sung et al. 2005). Agent-based methods are the key concept of the popular "MASSIVE" crowd simulator (Massive Software 2009), which is used among others in movie production, traffic simulations, and advertisements. In some situations more control of global behaviour is necessary and mathematical, physically and statistically based methods have been employed. Examples include Bayesian decision processes (Metoyer & Hodgins 2004) and the use of partial differential equations from continuum mechanics for describing local and global behaviour patterns of crowds (Treuille et al. 2006).

Very few resources are available on how to create the initial configuration of crowds and object collections. Professional crowd simulation tools usually offer interfaces for randomly generating crowds over a user defined domain by specifying the size and/or density of characters (WorldOfPolygons.com 2006). A spray interface for distributing grass, trees and other objects over a terrain has been presented by van der Linden (2001). Many applications define the positions of large groups of objects using application specific physically or statistically motivated techniques. For example, "Terragen" uses en-

vironmental parameters and directional controls to modify a fractal noise texture specifying the location of vegetation (Planetside Software, 2006). Procedural methods have been used for city simulations (Greuter et al. 2003). Diffusion-advection equations are useful for time-dependent processes with distance constraints such as traffic patterns (Garcia 2000).

2.2 Sketch-Based Modelling

Over the past decade a large variety of sketch-based modelling applications has been devised. We are using the crowd modelling software presented in this paper in "LifeSketch", a sketch-based framework for rapid prototyping of complex animated 3D environments (Yang & Wünsche 2010, Wünsche et al. 2010).

The principal problem in sketch-based modelling of 3D objects is that the user specified sketch is two-dimensional, whereas the modelled object is three-dimensional. This necessitates the introduction of geometric constraints limiting the number of possible 3D shapes corresponding to the 2D input sketch.

For the modelling of "blobby" shapes silhouettes, contours, cross-sections and skeletons have been used as geometric constraints. For example, for the popular "Teddy algorithm" users draw the outline of a 3D shape, which is subsequently generated by sampling and triangulating the contour, computing a skeleton, and rotating the sample points around the skeleton (Igarashia et al. 1999). Karpenko et al. use implicit surfaces to "inflate" contours to 3D bodies. As a result different sketched components can be easily blended together (Karpenko et al. 2002). Similar ideas are employed in ShapeShop (Schmidt et al. 2006) and MIBlob which uses implicit surfaces to inflate contours traced in medical images (de Araújo et al. 2004).

Two interesting application of silhouette-based algorithms are garment and tree modelling. For tree modelling the user sketches the outline of the crown of the tree and the algorithm computes a fitting branching structure based on existing templates and a probabilistic distribution (Chen et al. 2008). Garments can be modelled by sketching their outline and the algorithm automatically fits them to the body shape (Turquin et al. 2007). Gain et al. (2009) enable users to model 3D terrains by drawing the silhouette, spine and bounding curves of both extruding (hills and mountains) and embedding landforms (river courses and canyons). McCord et al. (2008) model orchids by sketching the cross sections, outlines and deformations of a flowers components.

The reviewed work demonstrates that sketch-based modelling is a powerful and popular approach for prototyping 3D scenes. We are not aware of any sketch-based interface for modelling crowds.

2.3 Texture Synthesis

The user study in the next section will demonstrate that a "model-by-example" technique is a promising approach for enabling users to design large and diverse crowds using just a few sketches. The underlying concept is similar to exemplar-based texture synthesis where a large image is generated from a smaller example image ("exemplar"). We will employ such texture synthesis techniques and hence present here a short overview of the most important methods.

A large variety of texture synthesis techniques exists, but none is suitable to generate all types of textures. Procedural techniques (Perlin 1985, Turk 1991, Witkin & Kass 1991, Worley 1996) are hard to control and, compared to exemplar-based methods, limited in the variety of materials that can be modeled. They often work well for stochastic textures and their applicability can be increased by combining them with statistical sampling e.g., (Guo et al. 2000).

Parametric exemplar-based methods, as proposed in (Heeger & Bergen 1995, De Bonet 1997, Dischler et al. 1998, Portilla & Simoncelli 2000, Bar-Joseph et al. 2001), rely on models of global statistical properties which serve as constraint function while matching statistics of the input and target texture. They are usually only successful in synthesizing homogeneous and stochastic exemplars.

Pixel-based methods (Efros & Leung 1999, Wei & Levoy 2000, Ashikhmin 2001) generate one pixel of the output texture at a time and hence offer a high level of control. The algorithm proposed in (Lefebvre & Hoppe 2005, 2006) performs an iterative optimization to minimize the difference of the synthesis result to the original exemplar, where the distance is measured using the sum of squared differences (SSD) of local neighborhoods. Most pixel-based methods consider only a local neighbourhood for generating new pixels in the output texture and hence are unable to capture global structures and semantics (e.g., a tomato having only one stem) (Manke & Wünsche 2010). Optimization-based approaches use local similarity measures of pixel neighborhoods to define a global texture energy function that is minimized (Kwatra et al. 2005). One example is histogram equalisation to ensure that input and output texture have the same colour distribution (Kopf et al. 2007). The algorithm generates very high quality results, but is considerable slower than comparable pixel-based techniques (Kopf et al. 2007).

Patch-based methods paste random patches of the exemplar into the output texture and optimize the transitions between overlapping patches (Praun et al. 2000, Efros & Freeman 2001, Kwatra et al. 2003). They can hence be considered an extension of pixel-based methods. Since in each step a patch is added to the texture they can often capture local semantics better, but they offer less control over the results (Wei et al. 2009). A special case of patch-based methods are tiling-based techniques, such as "Wang tiles" which uses carefully constructed square tiles with matching boundaries which can then be used to generate new textures (Cohen et al. 2003).

3 Requirement Analysis

3.1 Analysis of Real Environments

In order to find a suitable interface for specifying a large variety of crowds and collections of objects we evaluated hundreds of images obtained with Google using the keywords "crowd", "herd", "flock of birds", "forest", "park", "river boulders", "containers", "city" and "village". We found that in all cases the overall look of the crowd/collection could be characterised by the shape of its domain (the occupied region) and the pattern how components are distributed over it. The principal distribution patterns of objects are regular, random and clustered. Each pattern has numerous subpatterns, e.g., "regular" can mean a regular grid (soldiers standing in line) or objects lined up along a curve (houses along a street in a suburb). We also found more complex patters, which could be regarded as regular, and combinations of any of those patterns. An example are flocks of geese where the main shape formed by the geese is boomerang like, but some geese are distributed seemingly randomly among them.

Based on this analysis a tool offering predefined crowd patterns is too limited for out purposes and a "model-by-example" approach is most promising, where the users specifies an example distribution and the program replicates this distribution in a natural manner.

3.2 Pre-Design Study

In order to determine how users characterise large crowds and collections of objects with just a few strokes we performed a user study.

	Contour Type					Stroke Distribution Pattern				Undesired Inputs			Resulting Number of Objects		
	C1	**C2**	**C3**	**C4**	**C5**	**D1**	**D2**	**D3**	**D4**	**S**	**G3**	**G4**	Mean	Median	Standard Deviation
Task 1	17.65	41.18	0	0	35.29	82.35	17.65	5.88	0	17.65	11.76	29.41	100	219.82	269.54
Task 2	58.82	29.41	5.88	11.76	5.88	94.12	5.88	41.18	0	17.65	47.06	5.88	48	67.24	71.41
Task 3	82.35	11.76	5.88	0	0	5.88	94.12	5.88	17.65	5.88	0	17.65	110	170.88	184.56
Task 4	11.76	29.41	5.88	5.88	52.94	100	0	5.88	0	11.76	29.41	17.65	70	283.18	468.63
Task 5	47.06	29.41	11.76	5.88	11.76	17.65	82.35	0	0	5.88	17.65	17.65	120	198.12	202.95
Task 6	5.88	35.29	17.65	47.06	29.41	88.24	5.88	58.85	5.88	0	47.06	11.76	47.5	66.00	42.65
Task 7	35.29	29.41	11.76	0	23.53	70.59	29.41	11.76	0	52.94	35.29	11.76	19	48.65	78.16
Task 8	23.53	41.18	11.76	0	23.53	52.94	47.06	23.53	0	47.06	47.06	11.76	45	70.12	97.32
Task 9	41.18	35.29	17.65	5.88	5.88	47.06	52.94	29.41	0	23.53	41.18	5.88	42	180.00	472.41
Task 10	29.41	29.41	5.88	5.88	35.29	41.18	58.82	23.53	0	35.29	29.41	17.65	60	100.88	151.65
Task 11	17.65	41.18	5.88	5.88	35.29	100	0	17.65	0	23.53	29.41	0	75	146.47	197.30
Task 12	5.88	29.41	29.41	11.76	35.29	70.59	29.41	23.53	0	23.53	23.53	11.76	70	120.00	140.9
Task 13	5.88	47.06	17.65	0	29.41	88.24	11.76	17.65	0	35.29	52.94	17.65	12	24.18	30.08
Task 14	47.06	29.41	5.88	0	17.64	94.12	5.88	17.65	0	17.65	41.18	11.76	36	68.71	60.98
Task 15	52.94	17.65	17.65	5.88	11.76	11.76	88.24	11.76	0	29.41	35.29	11.76	73	119	114.49
Task 16	11.76	23.53	5.88	0	47.06	94.12	5.88	17.65	0	35.29	17.65	29.41	47	115.44	196.03
Task 17	23.53	11.76	5.88	0	52.94	88.24	5.88	17.65	0	52.94	29.41	11.76	32	61.31	69.34
Task 18	5.88	35.29	35.29	23.53	17.65	82.35	5.88	64.71	5.88	29.41	41.18	17.65	42.5	66.29	72.95

Table 1: Results of the pre-design user study. The rows represent the tasks 1–18 and the first twelve columns the percentage of participants using the type of contour, symbol and stroke distribution specified in subsection 3.2.2. The last three columns specify the mean, median and standard deviation of the estimated number of strokes which would fill the domain if continuing the user's example distribution.

3.2.1 User Study Design

The users were given eighteen tasks and were asked for each task to:

1. Indicate the area covered by the objects using a single sketch.
2. Indicate the distribution of objects using a small region filled with around 5-20 short strokes indicating how the whole area would be covered by the objects.

The users were told to do the sketching as fast as possible (less than 20 seconds per task) and they were told to imagine that somebody else would complete the placement of all objects for them based on their sketch. The eighteen tasks were:

1. Sketch a crowd of 1000s of people at a festival.
2. Sketch a group of 100s of student in the school yard during a break between classes.
3. Sketch an army of 1000s of soldiers marching in a parade.
4. Sketch a large natural forest with 10000s of trees.
5. Sketch a (human-planted) plantation forest with 10000s of trees.
6. Sketch a botanical garden with dozens of small areas of trees, where each area contains at most 10 trees.
7. Sketch 100s of farms distributed over a wide area, e.g., central Otago.
8. Sketch a small village with about 100 houses.
9. Sketch a large city area with 100s of Skyscrapers (e.g., Manhattan).
10. Sketch a modern suburb with hundreds of houses.
11. Sketch a flock of 100s of sheep.
12. Sketch dozens of flocks of geese migrating to warmer areas.
13. Sketch 100 tigers in a large jungle. Assume that tigers are territorial, i.e., they like to keep a large area for themselves.
14. Sketch 100s of toys lying in a child's room.
15. Sketch hundreds of containers stored in a port.
16. Sketch 1000s of sea shells on a beach.
17. Sketch 100s or large boulders in a river bed.
18. Sketch dozens of small island groups in the South Pacific Ocean. Each island group should have at most 8 islands.

3.2.2 Results

The study had 17 participants, 8 male, 8 female and for one participant the gender was not specified. Two of the participants were in the 15-20 age bracket, 13 between 20-30 years old, and two 30-40 years old. All but one of the participants were either university students or had completed a university degree. The fields of study/occupation were Computer Science (10), Chemistry (2), Mathematics (2), Commerce (1), Science (1) and Education (1).

For each question we recorded the type of contour specifying the domain and the distribution of strokes. The main types of contours we discovered were:

C1 Rectangular contour

C2 Circular/ellipsoidal contour

C3 No contour

C4 Contours for cluster

C5 Irregular contour

Note that these characterisations are neither complete nor mutually exclusive. For example, triangular contours, which were use once by one participant, are not covered by any of the above cases. Contours for clusters can be combined with any of the other contour types.

The main stroke distribution patterns were

D1 Random strokes/symbols

D2 Regular strokes/symbols

D3 Clustered strokes/symbols (regular/irregular recorded as above)

D4 Clusters, but no strokes indicating positions within clusters (regular/irregular recorded as above)

In addition we found that many participants used symbols (**S**) instead of strokes, especially when indicating large objects such as houses and farms. We also recorded when participants filled most (**G4**) or all of the domain (**G3**) with strokes/symbols. The results of the study are summarised in table 1.

It can be seen that most users prefer to sketch a rectangular or circular/ellipsoidal domain. The main exceptions were task 4, 16 and 17, which all refer to natural objects (natural forest, beach and river). In contrast human made objects, such as a crowd of people or a village were usually sketched with a rectangular or circular shape, even though in reality they inhibit highly irregular domains.

A potential problem is that many users also use contours to indicate clusters. The use of contours for this purpose varies widely between scenarios. For example, for the botanical garden almost 80% of partcipants sketching clusters surrounded them with extra contours. In other cases such as task 18 (island groups) less than 30% of participants sketching clusters surrounded them with extra contours. The main difference is that real garden beds are usually clearly differentiated from the surrounding area, whereas island groups have no boundary to the surrounding ocean.

In five instances users did not use closed contours. All of these cases occurred for task 17 (river). For task 16 (beach) one user only drew a single curve to outline the beach/water boundary as illustrated in figure 2 (i). Both cases are problematic since the domain of the intended collection of objects is not clearly defined and hence it would be difficult to complete for another user or computer algorithm. In two cases (task 1 and task 2) users drew a contour within another contour, e.g., to indicate a stage which is free of the audience.

In summary users do not seem to have a good mental model of the shape of the domain occupied by a collection of objects. Users created the most realistic contours for domains with clearly defined boundaries and scenes they are likley to have seen numerous times, e.g., an army of marching soldiers. In general users showed a reluctance to sketch irregular domains. Even for a natural forest more than 40% of users chose a rectangular or circular domain. In order to enable users to come up with the best crowd model an application should allow changes of crowd contours and recognise any erroneously drawn contours. At the very least clear visual feedback needs to be given on the recognised domain contour and the resulting object distribution.

In terms of sketched stroke distribution patterns table 1 suggests that in most cases users make appropriate choices between regular and irregular distributions. For example, 100% of users use an irregular pattern for the natural forst and 82% a regular pattern for the plantation forest. In contrast, the use of clusters seems to be less intuitive. Even in cases where clusters were explicitly specified, such as tasks 6 and 18, less than 65% os participants sketch clusters of strokes. However, for task 4 where no word indicates the use of clusters more than 41% intuitively use them. In 16 instances users drew only two clusters which is not sufficient to determine the 2D arrangement of clusters. There were also 16 instances of users drawing some clusters of size one, so any synthesis method devised by us must allow such clusters.

In summary most users are able to intuitively choose between regular and irregular distributions. The use of clusters seems to be less intuitive and hence, like before, it is important to give clear visual hints how a synthesized distribution is related to the sketched example distribution.

Observing the three rightmost columns of table 1 we can see that most participants do not have a good feeling for the size and density of an example distribution required to achieve a certain number of objects over a domain. The number of objects specified in the eighteen tasks varied between 100s and 10000s whereas the estimated number of objects resulting from user input usually varied between 50 and 300.

However, there is a strong correlation between the density of objects in reality and in the sketched distribution. Humans in a crowd and trees in a forests stand close together and consequently participants drew strikes close together resulting in a large estimated number of objects. In contrast tigers in a jungle or farms in New Zealand have large distances between individual objects, and consequently participants left large spaces between strokes resulting in much less objects than required. Note that from a logical perspective the sketch density should only depend on the required number of objects since a sketch only indicates a position in space and not the size of an object. We suspect that this problem can be partially alleviated by drawing points instead of strokes. The main reason for using strokes in the user study was that it is intuitive for pen-and-paper input and it is easier to recognise than small dots.

Many users filled most or all of the domain with the example distribution especially for task 6 (botanical garden), task 13 (tigers) and task 18 (small islands). Again it seems that a large size of objects or large distances inbetween objects encourages users to keep large gaps between strokes.

Another unexpected result was that many users used shapes and symbols to indicate objects, especially where the object had a large spatial extend such as in task 7 (farms), task 8 (houses), and task 17 (large boulders).

Figure 1 demonstrates examples of expected sketch input and figure 2 illustrates unexpected results.

3.3 Summary

The analysis of real environments and the pre-design study suggest that a feasible way to define a large variety of crowds and collections of objects is by defining its domain and an example distribution. The program must be able to differentiate between different types of distributions, such as regular, irregular and clustered, and must be able to replicate the characteristics of any such pattern without merely repeating it. Replicating the characteristics means that the relative positions, directions, inherent patterns, cluster sizes, and distribution densities should be similar, but not necessarily completely identical. For example, when modelling groups of students it would look unnatural if all groups have only, say, two different sizes.

The original input pattern must be part of the final distribution since the user defined point locations might have an important meaning and removing them might be confusing.

The example distribution is best defined by simple mouse clicks rather than strokes, since our pre-user study indicates that users associate stroke size with object size, which can influence the user's perception of scale (i.e., the distance between strokes). Clicking on points is also slightly more efficient and less cumbersome than drawing strokes. The program must be stable, i.e., not crash with unexpected input, and the visual feedback should indicate to the user how the input should be modified if the result is not the desired one.

4 Design

The problem of generating a distribution of objects over a domain based on an example distribution is similar to the exemplar-based texture synthesis problem, i.e., to algorithmically construct a large image from a given smaller image such that image characteristics are preserved. The current state of the art in that field is summarised in (Wei et al. 2009).

We can consider the example point distribution input by the user as a black-and-white input texture where black dots represent object locations. The problem then reduces to synthesising a new texture over the rest of the domain. Several difficulties arise:

- Most texture synthesis techniques are only good for certain types of textures, e.g., regular textures or stochastic textures.

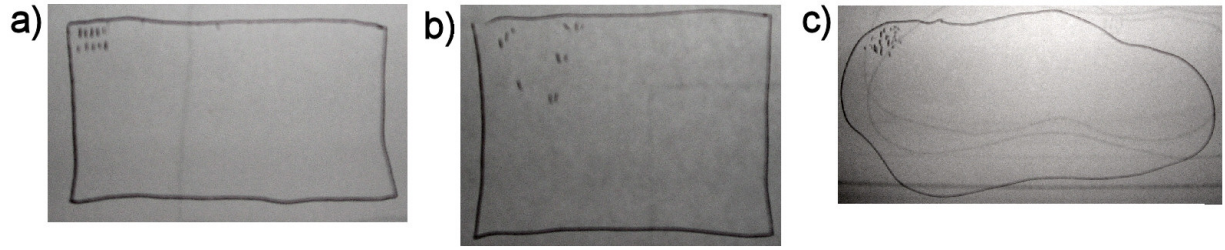

Figure 1: Examples of expected user input: (a) Task 3 - marching soldiers form a regular pattern within a rectangular shape. (b) Task 2 - students aggregate in small groups in a school yard with some individual students. (c) Task 4 - a natural forest has an irregular shape and trees are distributed densely and randomly within it.

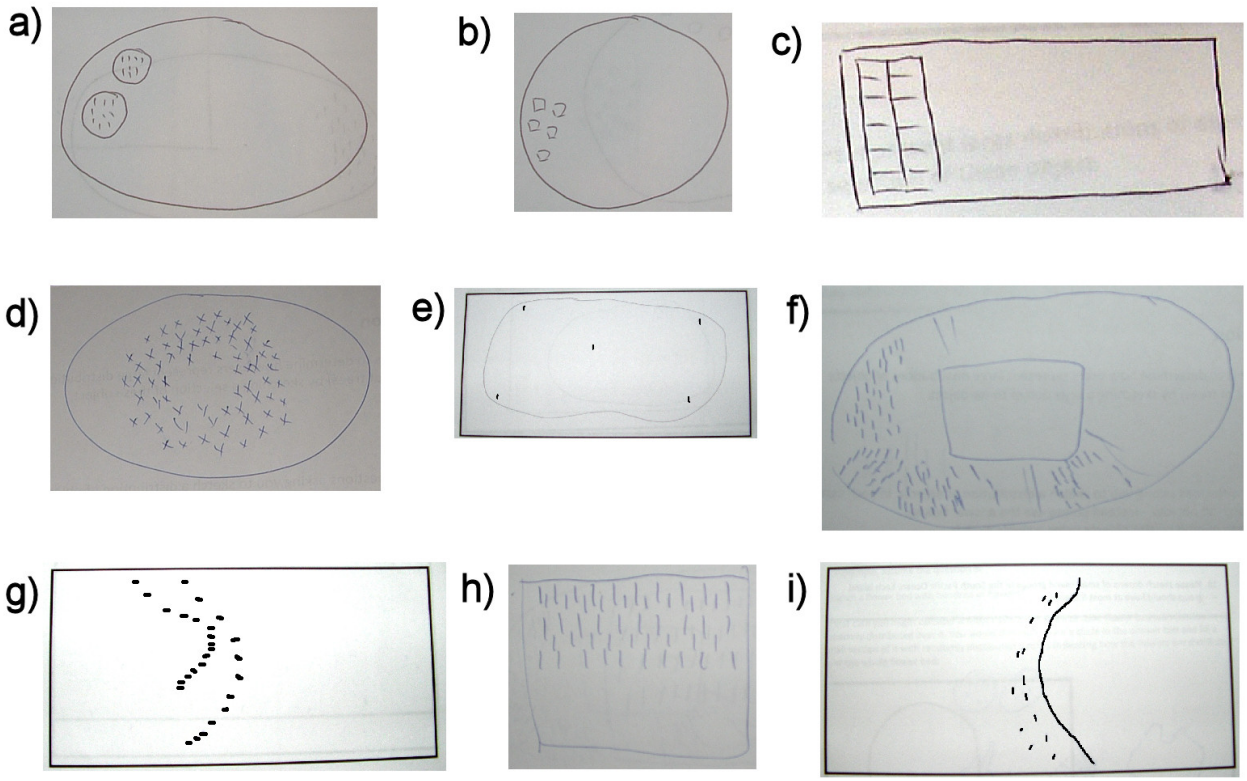

Figure 2: Examples of unexpected user input: (a) Task 6 - Extra contours for each cluster of trees. (b) Task 8 - A circular contour for a village and rectangular symbols for houses. (c) Task 15 - Connected symbols to indicate containers in a port (d) Task 1 - The sketched example distribution fills most of the region and its shape has an intended meaning (i.e., people aggregating around a stage). Objects are indicated by crosses instead of strokes. (e) Task 13 - Sparsely distributed strokes filling the whole region indicate tigers in a jungle [strokes have been redrawn digitally to enhance visibility]. (f) Task 1 - Shape of a crowd is described using a contour with a hole. (g) Task 12 - Regular stroke distribution along a curve and clusters overlapping in multiple coordinate directions [strokes have been redrawn digitally to enhance visibility]. (h) Task 5 - Intended regular pattern, but stroke number varies and rows are horizontally offset. (i) Task 16 - Beach region not clearly specified and the area to be occupied by seashells is undefined [strokes have been redrawn digitally to enhance visibility].

- It is difficult to reproduce the characteristics of an input texture with clusters since this would require a very large neighbourhood for any similarity measure used in the synthesis. This would dramatically slow down the synthesis process. Global optimization techniques might alleviate this problem, but are too slow (Kopf et al. 2007). In order to enable interactive modelling the texture synthesis techniques must be performed in near real-time.
- If we use an exemplar-based method, what is a suitable input texture?
- The texture synthesis must leave the original user input unchanged
- Most leading texture synthesis techniques are extremely complex and non-trivial to implement, so ideally we would like to employ an open source method.

In order to resolve these problems our algorithm uses the following steps:

1. Perform a cluster analysis and determine the number of clusters in the user's input
2. If the user input contains only one cluster then determine whether it is regular or stochastic, extract a suitable exemplar, and call a suitable exemplar-based texture synthesis method.
3. If the user input contains multiple clusters, then determine the properties of each cluster and the distribution of clusters and synthesise additional clusters with similar characteristics.

We now explain the varies steps of this algorithm in more detail.

4.1 Cluster analysis

In order to determine the number of clusters in the user input we employ a cluster analysis algorithms and calculate the size and distribution of clusters. A complication occurs due to the fact that the user input can contain clusters of size one (see figure 1 (b)).

4.1.1 K-means and K-means++ Algorithm

The arguably most popular algorithm used in scientific and industrial applications is the k-means algorithm (Arthur & Vassilvitskii 2007), which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest centre. We use the Lloyd-algorithm which initially chooses k points as the initial centres (means) of the clusters. In each iteration the remaining points are assigned to the closest centres and for each thus created cluster a new mean is calculated, usually as arithmetic mean of the clustered points. The iteration continues until the centres (means) do not change anymore. The resulting algorithm minimises the squared error function

$$J = \sum_{j=1}^{k} \sum_{X_j \in S_i} ||X j - \mu_i||^2$$

of the distances between cluster points and cluster centers. Here $S_i, i = 1, \ldots, k$ are the k-clusters and $||X_j - \mu_i||$ is the distance measure between a data point X_j and the cluster centre μ_i. For each cluster the new centre is computed

The results of the k-means algorithm are influenced by three key parameters: The number of clusters k, the distance metric employed (usually Euclidean distance) and the initial choice of cluster centres. When the amount of data is small, as in our case, the initial positions significantly effect outcomes. Another potential problem is that each point has the same weight factor and as a result the arithmetic mean is not robust when clusters are mixed with single points as illustrated in figure 1 (b). Possible improvements are discussed in (Xu & Wunsch II 2005).

The speed and accuracy of the k-means algorithm can be improved by carefully choosing centres in the initialisation using a probability function which takes into account the shortest distance of a potential centre point to the closest center already chosen (Arthur & Vassilvitskii 2007). This gives points far away from existing centres a higher probability to be chosen. The thus resulting K-Means++ algorithm has been proven to be superior both in theory and practice (Shindler 2008) and has been successfully applied in geography applications (Lee et al. 2008).

4.1.2 Determining the Correct Number of Clusters

Based on surveys of existing clustering algorithms (Xu & Wunsch II 2005, Shindler 2008) we decided that the k-means++ algorithm using the Lloyds k-means algorithm is the best solution to deal with differently sized and shaped clusters and isolated points.

An important question in our application is the number of clusters in the initial user input. This is not known a priori and, in fact, there might be no definite answer as indicated in some of the images of figure 2. Some references in the literature recommend to estimate cluster numbers using *k-Fold Cross-Validation* (Statsoft Ltd. 2010).

In our case we believe that this technique is not suitable based on the limited input size and, in most cases, low number of clusters. In particular, in many cases the input will consist of a single cluster and identifying such cases correctly is critical. We choose instead an information theoretical approach which is similar to Sugar & James (2003). In the original article the authors aim to be able to differentiate overlapping clusters with different density distributions. In our case we assume that users always sketch clusters such that their boundaries do not overlap. We hence compute the optimal number of clusters by computing the k-means++ algorithm for any number for $1 \leq k \leq n$ where n is the number of input points. For each clusterisation we define a minimum spanning tree. The optimal value of k is found if the tree edges between clusters are longer than the edges within clusters (see figure 3). While the resulting clustering might not be optimal in a mathematical sense, we found that it best reflects the way users draw clusters (see section 3). If the tree edges are all roughly similarly long for small values of k, then we consider the input to represent a single cluster.

4.2 Replicating an Input Point Distribution

After the correct number of clusters has been determined, it must be decided whether the user input is regular or stochastic. We do this by analysing the orientation of the edges of the minimum spanning tree. If the edges area aligned similarly, e.g., roughly horizontal and vertical, than the distribution is considered regular. If the tree edges have no predominant pattern than the distribution is considered irregular.

For a regular input distribution we choose the smallest enclosing square and use the thus created texture image as exemplar for a Wang tiling texture synthesis algorithm (Cohen et al. 2003). The algorithm finds the smallest repetitive texture, and simply tiles them together. We found that this algorithm preserves structures in the input texture well. A drawback reported in the literature is that repetitive patterns can be recognised for more stochastic input textures (Zhang & Kim 2007). Since we only employ the algorithm for near regular inputs this does not seem to represent a problem.

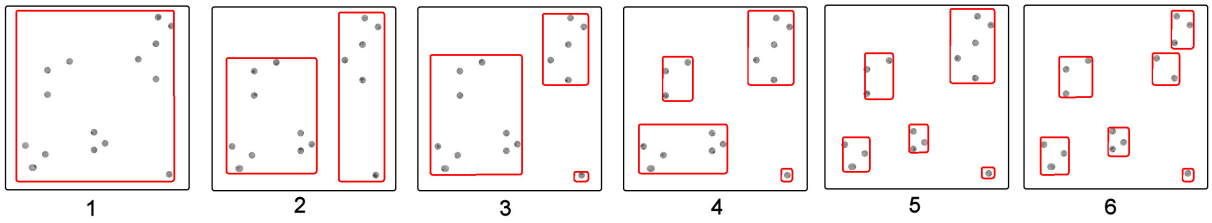

Figure 3: Illustration of the algorithm determining the optimal number of clusters. The k-means++ algorithm will be called with $1 \leq k \leq n$, where n is the number of input points. For each clusterisation we define a minimum spanning tree. The optimal value of k is found if the tree edges between clusters are longer than the edges within clusters. In the above figure the optimal value would be $k = 5$.

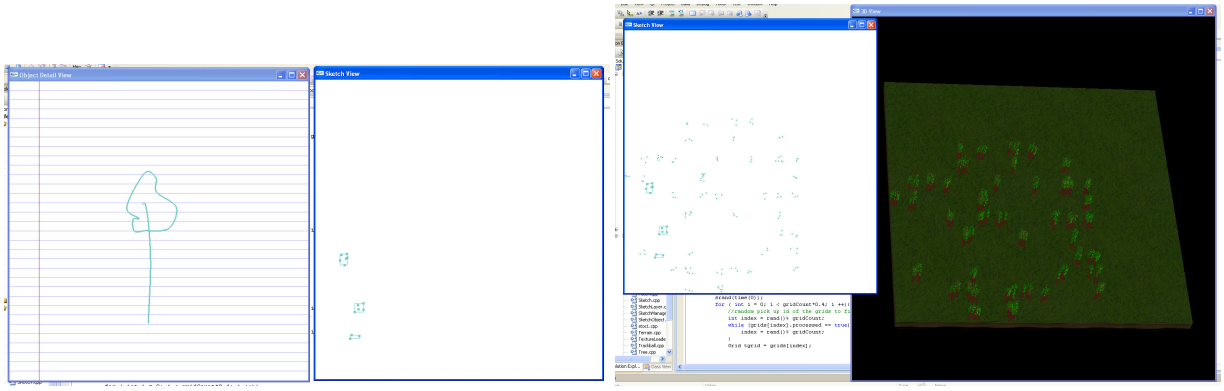

Figure 4: Left: Sketch of a tree and of a clustered point distribution. Right: The synthesised point distribution and the resulting scene of a landscape with small "patches" of trees.

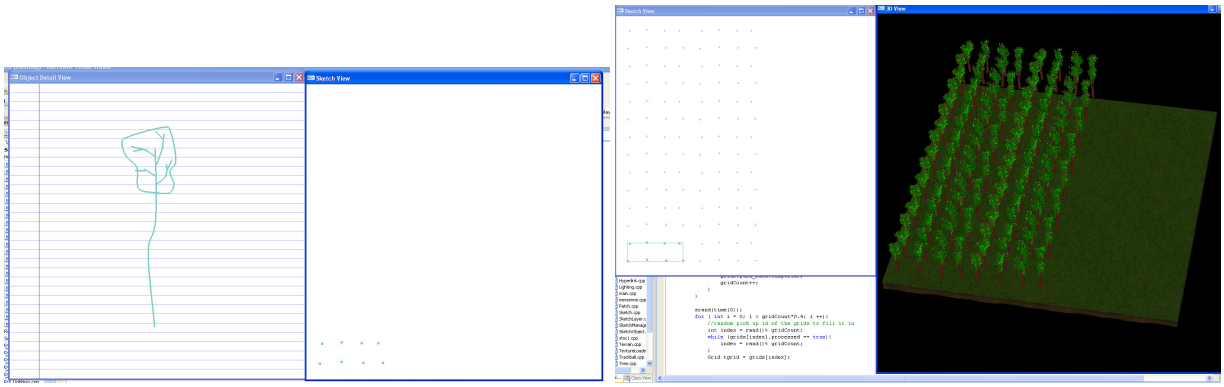

Figure 5: Left: Sketch of a tree and of a regular point distribution. Right: The synthesised point distribution and the resulting scene of a landscape with a forest plantation.

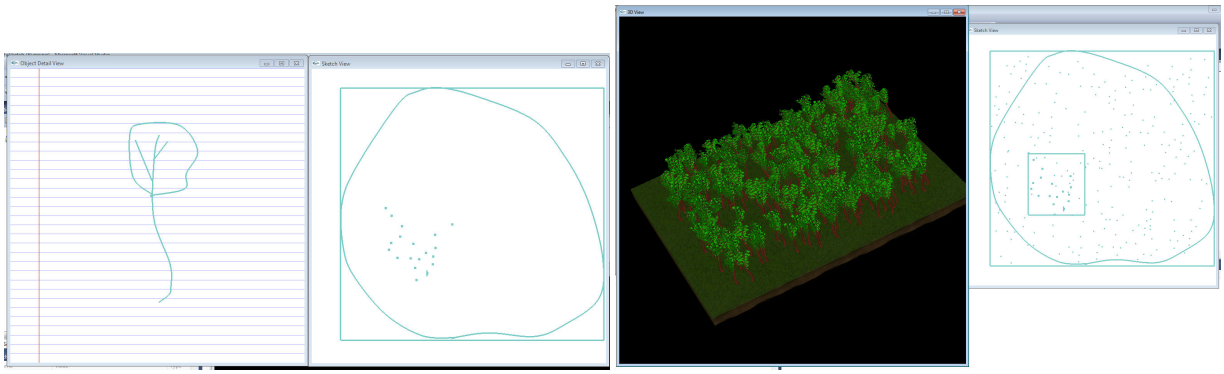

Figure 6: Left: Sketch of a tree and of an irregular point distribution and a domain for the synthesized points. Right: The synthesised point distribution and the resulting scene of a landscape with a natural forest.

For irregular distributions we choose the exemplar texture analogously, but then apply a Chaos Mosaic algorithm (Guo et al. 2000).

For a clustered input we compute the mean and standard deviation of the size of all clusters and of the distances of the points in them to the respective centers. We then generate new clusters based on these probability distributions. The clusters are then randomly placed subject to a minimum distance criterion. Currently we are unable to replicate regular clusters, but might be able to do that by utilising the previous described texture synthesis algorithms and then extracting appropriately sized patches.

5 Results

We have implemented the above described algorithms using Microsoft Visual C++ and OpenGL. So far we have only integrated the generation of sketched tree objects with our crowd generation software. Work is underway on the full integration of "LifeSketch" in order to allow generation of large collections of arbitrary objects such as buildings and characters. Figures 4–6 demonstrate that our application gives realistic results for regular, irregular and clustered input. All textures are synthesised in real-time.

More work is necessary to generalise the synthesis algorithms, i.e., to allow clusters with regular point distributions, and to deal with structured input which is not inherently regular as illustrated in image 2 (g). Also the structure of clusters should be considered, i.e., to allow regularly arranged clusters. This could be achieved by considering cluster centres as input texture and then first synthesising locations of new clusters and then their actual shape.

6 Conclusion and Future Work

We have presented a user study evaluating the representation of different types of crowds and collections of objects using sketch input. The results demonstrate that a differentiation into regular, irregular and clustered patterns is intuitive and effective. Problems exist with abstracting size of objects and their spatial distribution: when dealing with small objects which are widely spaced users tend to draw widely spaced sketches even though a sketch represents just a location in space. Our tool uses point input and further studies need to explore whether the same problem will exist when using our "LifeSketch" software.

Based on the results of the users study we designed a novel algorithm for synthesising large point distributions from sketched user input. The algorithm uses a combination of a minimum spanning tree algorithm, k-means++ algorithm, and probabilistic approach for characterising example point distributions. New point distributions are generated using different texture synthesis algorithms which take into account the inherent structure of the input textures. Preliminary results suggests that a wide variety of complex environments containing large collections of objects can be modelled that way.

More work needs to be done to increase the range of reproducible input distribution patterns. Also we want to fully integrate our crowd modelling software into "LifeSketch" and perform user testing to determine its effectiveness, intuitiveness and ease-of-use. We are keen to use the same tasks as in section 3 and to investigate whether users produce the same sketch input as on paper and how an interactive approach assists with generating the desired results.

References

Arthur, D. & Vassilvitskii, S. (2007), k-means++: the advantages of careful seeding, *in* 'Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms (SODA '07)', Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 1027–1035. http://ilpubs.stanford.edu:8090/778.

Ashikhmin, M. (2001), Synthesizing natural textures, *in* 'Proceedings of I3D '01', ACM Press, pp. 217–226.

Bar-Joseph, Z., El-Yaniv, R., Lischinski, D. & Werman, M. (2001), 'Texture mixing and texture movie synthesis using statistical learning', *IEEE Transactions on Visualization and Computer Graphics* **7**(2), 120–135.

Barber, C. M., Shucksmith, R. J., MacDonald, B. & Wünsche, B. C. (2010), Sketch-based robot programming, *in* 'Proceedings of IVCNZ 2010'. [in press].

Chen, X., Neubert, B., Xu, Y.-Q., Deussen, O. & Kang, S. B. (2008), Sketch-based tree modeling using markov random field, *in* 'ACM SIGGRAPH Asia 2008 papers', ACM, New York, NY, USA, pp. 1–9. http://graphics.cs.yale.edu/xuejin/SketchBasedTreeModeling_chenSiggraphAsia.pdf.

Cohen, M. F., Shade, J., Hiller, S. & Deussen, O. (2003), 'Wang tiles for image and texture generation', *ACM Trans. Graph.* **22**(3), 287–294.

Coyette, A., Kieffer, S. & Vanderdonckt, J. (2007), Multi-fidelity prototyping of user interfaces, *in* C. Baranauskas, P. Palanque, J. Abascal & S. Barbosa, eds, 'Human-Computer Interaction INTERACT 2007', Vol. 4662 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 150–164.

Davis, J., Agrawala, M., Chuang, E., Popović, Z. & Salesin, D. (2003), A sketching interface for articulated figure animation, *in* 'Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '03)', Eurographics Association, pp. 320–328. http://graphics.stanford.edu/papers/sketch_interface/Davis_149_SketchingAnimation_SCA2003.pdf.

de Araújo, B., Jorge, J., Sousa, M. C., Samavati, F. & Wyvill, B. (2004), MIBlob: a tool for medical visualization and modelling using sketches, *in* 'SIGGRAPH '04: Posters', ACM Press, p. 107.

De Bonet, J. S. (1997), Multiresolution sampling procedure for analysis and synthesis of texture images, *in* 'Proceedings of SIGGRAPH '97', ACM Press, pp. 361–368.

Dischler, J.-M., Ghazanfarpour, D. & Freydier, R. (1998), 'Anisotropic solid texture synthesis using orthogonal 2d views', *Computer Graphics Forum* **17**(3), 87–95.

Efros, A. A. & Freeman, W. T. (2001), Image quilting for texture synthesis and transfer, *in* 'Proceedings of SIGGRAPH '01', ACM Press, pp. 341–346.

Efros, A. A. & Leung, T. K. (1999), Texture synthesis by non-parametric sampling, *in* 'Proceedings of ICCV '99', IEEE Computer Society, pp. 1033–1038.

Funge, J., Tu, X. & Terzopoulos, D. (1999), Cognitive modeling: knowledge, reasoning and planning for intelligent characters, *in* 'Proc. of the 26th annual conference on Computer graphics and interactive techniques (SIGGRAPH '99)', ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, pp. 29–38.

Gain, J., Marais, P. & Strasser, W. (2009), Terrain sketching, *in* 'Proceedings of the 2009 symposium on Interactive 3D graphics and games I3D '09', ACM, pp. 31–38. http://people.cs.uct.ac.za/~jgain/publications/terrsketch.pdf.

Garcia, A. L. (2000), Physics of traffic flow, *in* 'Numerical Methods for Physics', 2nd edn, Prentice Hall, chapter 7.

Greuter, S., Parker, J., Stewart, N. & Leach, G. (2003), Real-time procedural generation of 'pseudo infinite' cities, *in* 'Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia (GRAPHITE '03)', ACM, New York, NY, USA, pp. 87–ff.

Gross, M. D. & Do, E. Y.-L. (1996), Ambiguous intentions: a paper-like interface for creative design, *in* 'Proceedings of the 9th annual ACM symposium on User interface software and technology (UIST '96)', ACM, pp. 183–192.

Guo, B., Shum, H., & Xu, Y.-Q. (2000), Chaos mosaic: Fast and memory efficient texture synthesis, Technical report MSR-TR-2000-32, Microsoft Research. http://research.microsoft.com/pubs/69770/tr-2000-32.pdf.

Heeger, D. J. & Bergen, J. R. (1995), Pyramid-based texture analysis/synthesis, *in* 'Proceedings of SIGGRAPH '95', ACM Press, pp. 229–238.

Igarashia, T., Matsuoka, S. & Tanaka, H. (1999), Teddy: a sketching interface for 3d freeform design, *in* 'Proceedings of SIGGRAPH '99', ACM Press, pp. 409–416.

Joshi, A., Robertson, G., Plimmer, B. & Wünsche, B. C. (2010), Bubbleworld builder - 3d modelling using two-touch and sketch interaction, *in* 'Proceedings of the 5th International Conference on Computer Graphics Theory and Applications (GRAPP 2010), Angers, France', pp. 116–122.

Karpenko, O., Hughes, J. & Raskar, R. (2002), 'Free-form sketching with variational implicit surfaces', *Computer Graphics Forum* **21**(3), 585–594. http://www.merl.com/papers/docs/TR2002-27.pdf.

Kloonigames Ltd. (2008), 'Crayon Physics Deluxe homepage'. http://www.crayonphysics.com.

Kopf, J., Fu, C.-W., Cohen-Or, D., Deussen, O., Lischinski, D. & Wong, T.-T. (2007), 'Solid texture synthesis from 2d exemplars', *ACM Transactions on Graphics (Proceedings of SIGGRAPH '07)* **26**(3), (2.1)–(2.9).

Kwatra, V., Essa, I., Bobick, A. & Kwatra, N. (2005), 'Texture optimization for example-based synthesis', *ACM Transactions on Graphics (SIGGRAPH '05)* **24**(3), 795–802.

Kwatra, V., Schödl, A., Essa, I., Turk, G. & Bobick, A. (2003), 'Graphcut textures: image and video synthesis using graph cuts', *ACM Transactions on Graphics (Proceedings of SIGGRAPH '03)* **22**(3), 277–286.

Lee, S. S., Won, D. & McLeod, D. (2008), Discovering relationships among tags and geotags, *in* 'Proceedings of the 2nd International Conference on Weblogs and Social Media (ICWSM 08)'. http://sir-lab.usc.edu/publications/2008-ICWSM2LEES.pdf.

Lefebvre, S. & Hoppe, H. (2005), 'Parallel controllable texture synthesis', *ACM Transactions on Graphics (Proceedings of SIGGRAPH '05)* **24**(3), 777–786.

Lefebvre, S. & Hoppe, H. (2006), 'Appearance-space texture synthesis', *ACM Transactions on Graphics (Proceedings of SIGGRAPH '06)* **25**(3), 541–548.

Li, Q. L., Geng, W. D., Yu, T., Shen, X. J., Lau, N. & Yu, G. (2006), Motionmaster: authoring and choreographing kung-fu motions by sketch drawings, *in* 'Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '06)', Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, pp. 233–241.

Manke, F. & Wünsche, B. C. (2010), Fast spatially controllable multi-dimensional exemplar-based texture synthesis and morphing, *in* A. Ranchordas, J. M. Pereira, H. J. Arajo & J. M. R. Tavares, eds, 'Computer Vision, Imaging and Computer Graphics. Theory and Applications', Vol. 68 of *Communications in Computer and Information Science*, Springer Berlin Heidelberg, pp. 21–34. http://www.cs.auckland.ac.nz/~burkhard/Publications/SpringerCCIS2009MankeWuensche.pdf.

Massive Software (2009), 'Homepage'. http://www.massivesoftware.com.

McCord, G., Wünsche, B. C., Plimmer, B., Gilbert, G. & Hirsch, C. (2008), A pen and paper metaphor for orchid modeling, *in* 'Proceedings of the 3rd International Conference on Computer Graphics Theory and Applications (GRAPP 2008)', pp. 119–124. http://www.cs.auckland.ac.nz/~burkhard/Publications/GRAPP2008McCordWuenscheEtAl.pdf.

Metoyer, R. A. & Hodgins, J. K. (2004), 'Reactive pedestrian path following from examples', *The Visual Computer* **20**, 635–649.

Olsen, L., Samavati, F. F., Sousa, M. C. & Jorge, J. A. (2009), 'Technical section: Sketch-based modeling: A survey', *Computers & Graphics* **33**(1), 85–103. http://pages.cpsc.ucalgary.ca/~olsenl/wiki/uploads/Papers/CnGsurvey.pdf.

Perlin, K. (1985), An image synthesizer, *in* 'Proc. of SIGGRAPH '85', ACM Press, pp. 287–296.

Planetside Software, (2006), 'Terragen'. http://www.planetside.co.uk/terragen/tgd/tg2faq.shtml#faq34.

Plimmer, B., Purchase, H. & Yang, H. Y. (2010), Sketchnode: Intelligent sketching support and formal diagramming, *in* 'Proceedings of OZCHI 2010'. [in press].

Portilla, J. & Simoncelli, E. P. (2000), 'A parametric texture model based on joint statistics of complex wavelet coefficients', *Int. Journal of Computer Vision* **40**(1), 49–70.

Praun, E., Finkelstein, A. & Hoppe, H. (2000), Lapped textures, *in* 'Proceedings of SIGGRAPH '00', ACM Press, pp. 465–470.

Reynolds, C. W. (1987), Flocks, herds and schools: A distributed behavioral model, *in* 'SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques', ACM, New York, NY, USA, pp. 25–34.

Ropinski, T., Praßni, J.-S., Steinicke, F. & Hinrichs, K. H. (2008), Stroke-based transfer function design, *in* 'IEEE/EG Volume and Point-Based Graphics', pp. 41–48.

Sakamoto, D., Honda, K., Inami, M. & Igarashi, T. (2009), Sketch and run: a stroke-based interface for home robots, *in* 'CHI '09: Proceedings of the 27th international conference on Human factors in computing systems', ACM, pp. 197–200. http://www-ui.is.s.u-tokyo.ac.jp/~takeo/papers/sakamoto_chi2009_sketchandrun.pdf.

Schmidt, R., Wyvill, B., Sousa, M. C. & Jorge, J. A. (2006), Shapeshop: sketch-based solid modeling with blobtrees, *in* 'ACM SIGGRAPH 2006 Courses', ACM, p. 14.

Schmieder, P., Plimmer, B. & Vanderdonckt, J. (2010), 'Generating systems from multiple sketched models', *Journal of Visual Languages and Computing* **21**, 98–108.

Shindler, M. (2008), Approximation algorithms for the metric k-median problem, Technical report, UCLA, Los Angeles, CA. http://cs.ucla.edu/~shindler/shindler-kMedian-survey.pdf.

Skubic, M., Anderson, D., Blisard, S., Perzanowski, D., Adams, W., Trafton, J. G. & Schultz, A. C. (2005), Using a sketch pad interface for interacting with a robot team, *in* 'AAAI'05: Proceedings of the 20th national conference on Artificial intelligence', AAAI Press, pp. 1739–1740.

Statsoft Ltd. (2010), 'Electronic Statistics Handbook'. http://www.statsoft.com/textbook/cluster-analysis/.

Sugar, C. A. & James, G. M. (2003), 'Finding the number of clusters in a dataset: An information theoretic approach', *Journal of the American Statistical Association* **98**(463), 750–763. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.107.9895.

Sung, M., Kovar, L. & Gleicher, M. (2005), Fast and accurate goal-directed motion synthesis for crowds, *in* 'Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '05)', ACM, New York, NY, USA, pp. 291–300.

Takahashi, S., Kato, Y. & Shibayama, E. (2005), A new static depiction and input technique for 2d animation, *in* '2005 IEEE Symposium on Visual Languages and Human-Centric Computing', pp. 296–298.

Thorne, M., Burke, D. & van de Panne, M. (2004), 'Motion doodles: an interface for sketching character motion', *ACM Transactions on Graphics (TOG)* **23**(3), 424–431.

Treuille, A., Cooper, S. & Popović, Z. (2006), 'Continuum crowds', *ACM Trans. Graph.* **25**(3), 1160–1168.

Turk, G. (1991), Generating textures on arbitrary surfaces using reaction-diffusion, *in* 'Proceedings of SIGGRAPH '91', ACM Press, pp. 289–298.

Turquin, E., Wither, J., Boissieux, L., Cani, M.-P. & Hughes, J. F. (2007), 'A sketch-based interface for clothing virtual characters', *IEEE Comput. Graph. Appl.* **27**(1), 72–81.

van der Linden, J. (2001), Interactive view-dependent point cloud rendering, *in* 'Proceedings of IVCNZ 2001', pp. 1–6. http://www.cs.auckland.ac.nz/~jvan006/papers/pointcloudrendering_final.ps.gz.

Wei, L.-Y., Lefebvre, S., Kwatra, V. & Turk, G. (2009), State of the art in example-based texture synthesis, *in* 'Eurographics 2009, State of the Art Report, EG-STAR', Eurographics Association. http://www-sop.inria.fr/reves/Basilic/2009/WLKT09.

Wei, L.-Y. & Levoy, M. (2000), Fast texture synthesis using tree-structured vector quantization, *in* 'Proceedings of SIGGRAPH '00', ACM Press, pp. 479–488.

Witkin, A. & Kass, M. (1991), 'Reaction-diffusion textures', *SIGGRAPH Computer Graphics* **25**(4), 299–308.

Wong, Y. Y. (1992), Rough and ready prototypes: lessons from graphic design, *in* 'Posters and short talks of the 1992 SIGCHI conference on Human factors in computing systems (CHI '92)', ACM, pp. 83–84.

WorldOfPolygons.com (2006), 'CrowdIT'. http://www.crowdit.worldofpolygons.com/.

Worley, S. (1996), A cellular texture basis function, *in* 'Proceedings of SIGGRAPH '96', ACM Press, pp. 291–294.

Wünsche, B. C., Keymer, D. & Amor, R. (2010), Sketch, Click, Plug and Play: Accelerated Design of Virtual Environments by Integrating Multimedia and Sketch Content into Game Engines, *in* 'Proceedings of the 11th Annual ACM SIGCHI NZ Conference on Computer-Human Interaction (CHINZ 2010)'.

Xu, R. & Wunsch II, D. (2005), 'Survey of clustering algorithms', *IEEE Transactions on Neural Networks* **16**(3), 645–678.

Yang, R. & Wünsche, B. C. (2010), LifeSketch - A Framework for Sketch-Based Modelling and Animation of 3D Objects, *in* 'Proceedings of the Australasian User Interface Conference (AUIC 2010)', pp. 1–10. http://www.cs.auckland.ac.nz/~burkhard/Publications/AUIC2010_YangWuensche.pdf.

Zhang, X. & Kim, Y. J. (2007), 'Efficient texture synthesis using strict wang tiles', *Graph. Models* **70**(3), 43–56.