USING AN AMORPHOUS COMPUTER FOR VISUAL DISPLAY APPLICATIONS IN INTELLIGENT ENVIRONMENTS

A.M.King, V.Callaghan, G.Clarke

University of Essex, United Kingdom, amking@essex.ac.uk / vic@essex.ac.uk / graham@essex.ac.uk

Keywords: Nanotechnology, Amorphous Computing, Pervasive Computing, Visual Display.

Abstract

In this paper we introduce the concept of an iSurface; a surface coated with a multitude of identical nano-scale computing devices called iCells. Based on earlier extensive simulation work, we describe the computational limits of amorphous computing for image display applications and explain how such surfaces could be used to create a variety of novel fashion applications ranging from electronic wallpaper, active jewellery to adaptive clothing.

1 Introduction

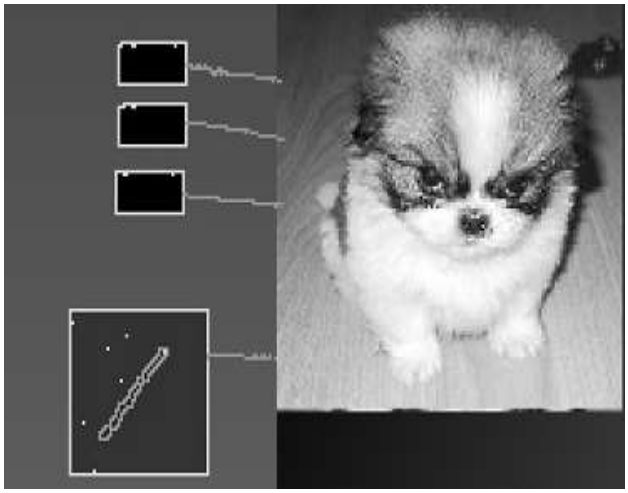
The field of nano-technology is at the cutting-edge of contemporary research (e.g. [1]). Its goal is the development of complex machines that are so small as to be invisible to the naked eye and to operate on a molecular or even an atomic scale.
In the future, this miniaturisation will inevitably include computing devices and, it is envisaged, these will be manufactured in such quantities that we will see the pervasive use of "Smart Matter" [2], nano-technology with computational ability. These tiny devices could be used to coat the surfaces of everyday environments and deliver novel functionality. Such surfaces, it is argued, could be used as video displays, user interfaces, and sensor arrays. The implication is that with 'a coat of paint' significant functionality can be added to any surface. We call this development of an 'intelligent surface' an iSurface, and the individual computing devices it is comprised of are called iCells. These iCells can host a variety of mobile agents and these provide its functionality.

The functionality required for an iSurface involves coordinated activity across surfaces that might be as large as the wall of a room. There are considerable problems involved since these surfaces are made up of literally thousands of microscopic devices communicating over short range. One possible way to realise the predicted functionality is by using an Amorphous Computing [3] approach. An Amorphous Computer is a multitude of identical tiny computing devices. These devices have limited processing capability, are only capable of communicating with their immediate neighbours, and can be unreliable. These 'particles', as they are known, could be painted onto a surface to form an ad hoc network, as envisioned for the iSurface. The central problem is how does one obtain useful global behaviour from this collection of hundreds or thousands of locally interacting particles? The field of Amorphous Computing attempts to solve this via the development of organisational principles and programming languages for Amorphous Computers.

With the iSurface paradigm, users would be able to deploy distributed and spatially located applications to perform various tasks. The computing devices, or iCells, would contain the instructions and data necessary for these applications and act in concert to realise the specific application. Individually the iCells would be unlikely to perform all of the necessary actions for the applications to succeed, but when many iCells cooperate (or just interact) the desired behaviour could arise from their interactions; one of the underlying challenges then is how to engineer this emergent behaviour. The code for these applications would be mobile agents; self-interested bits of code that can migrate around a network, finding hosts to execute themselves upon. In this way, we can think of deploying smart skin applications as akin to infecting the intelligent surface with a smart virus, injected into the network and deliberately spreading and instantiating itself on the appropriate elements in the surface. In this way, a cell might become host to several mobile agents, as well as its default behaviour, and thus be part of several different applications.

We will first discuss the iSurface structure and the problems caused by load on the processor and communication system (iCell). We will then discuss a video system application for the iSurface, the necessary capabilities necessary to implement it, and why load and iSurface damage make it unfeasible. We will suggest that although a video system, involving rapid updates of information, may never be feasible, the same techniques can be applied to applications that are not time-critical, such as a wallpaper system. This paper focuses on the use

of an iSurface for aesthetic purposes, that is, the display of images across a surface.



[Figure 1] - Screenshot of the iSurface simulator. The three black boxes are buttons, the grey panel detects gestures, and the picture of the dog is a video display.

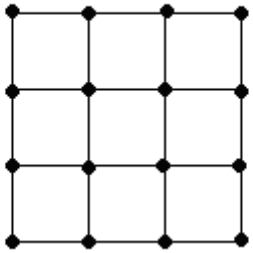
1.2 Related Work

Butera's "Paintable" project **[4]** demonstrates the feasibility of a mobile code paradigm which has formed a basis for the mobile agents used as part of the system described in this paper. However, Paintable never becomes data-heavy or time-critical; example applications are mainly distributed storage systems.

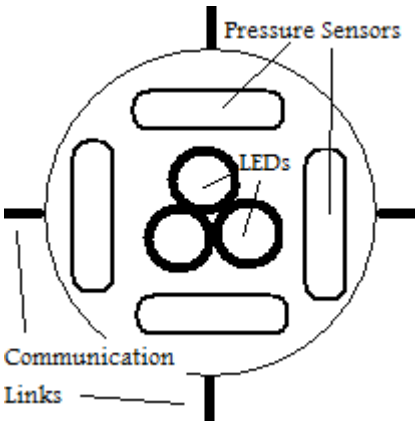
2 The iSurface structure and iCell load

The foundation for the iCell's predicted capabilities is Smart Dust [**5, 6, 7,** and **8**]. At the time of writing, a Smart Dust mote is very primitive compared to the average desktop machine; an 8MHz processor and 10Kbps communication speed. Applying Moore's Law to a Smart Dust mote allows us to calculate that, by 2020, an equivalent device would be operating at 8GHz with a communications speed of approximately 10Mbps.
Modern processors can effectively perform one operation/instruction per processor cycle and thus we can assume that 8 MHz gives 8 MIPS (Millions of Instructions per Second) and 8 GHz gives 8000 MIPS.

A simple way to picture the structure of the iSurface is to consider it as a gridwork of connections, with the iCells at the intersections. These connections can be considered to be perfect; full duplex communications between an iCell and its neighbours, all without the possibility of overlapping and colliding signals.
Communication can be directed to a specific neighbour because of the dedicated connections and there is also a sense of direction with cells having the ability to determine relative up, down, left, and right, in terms of where their neighbours are.



[Figure 2] – The iSurface paradigm; iCells arranged in a gridwork.



[Figure 3] – Diagram of a conceptual iCell.

We are able to state the minimum number of hops necessary for a message to propagate from any given node to any other. Assuming we have no damage to the surface;
$hops = |dx| + |dy|$

Where dx is the difference in x position between the two nodes, and dy is the difference in y position.

It is easy to see that the iSurface is susceptible to queues of messages building up in the iCells and causing massive problems with communication. We have a network of devices, all running asynchronously, and due to the usage of mobile agents to give functionality to the iCells, the iSurface may not be homogenous in terms of what code they need to execute.
Each iCell has finite processing power, and this is identical across the iSurface, therefore whatever processing time is available will be split between all code parts running on an iCell. This means that iCells with more code to execute will actually run slower than iCells with less code.
We term this problem "iCell load", and split this into "message load", referring to the backlog of messages to communicate, and "agent load", referring to the amount of code on the iCell.
Message Load is calculated as the time taken to transmit the chosen message, plus the time taken to transmit all of the other messages before it in the output queue.

$$t(msg, cell) = \left(\frac{size_{msg}}{comms\_speed} \right) + \sum_{q=1}^{length\_of\_queue\_in\_cell} \left(\frac{size_q}{comms\_speed} \right)$$

Agent Load is calculated as the time taken for the iCell core functionality to execute, as well as all of the agents. It can be seen as the time between a given agent being able to execute any given piece of code resident within it.

$$p(cell) = \left(\frac{size_{cell\_core}}{instructions\_per\_second} \right) + \sum_{a=1}^{number\_of\_agents\_in\_cell} \left(\frac{size_a}{instructions\_per\_second} \right)$$

So how does iCell load affect the iSurface? In a simple device like the iCell, higher levels of message and agent load will cause delays in processing and sending data from the communications system. Whichever causes the largest delay is the dominant factor in the iCell load bottleneck. This means that the iSurface as a whole will be directly affected by its slowest components if an application makes use of them.

We can think of this as a form of resistance to signals. Messages that try to go through the areas of high concentration end up being slowed down, and compounding the situation. Messages that go around the obstruction may avoid the load and travel further in the same time it takes a delayed message to traverse the high concentration.

So the lesson learned is that to reduce iCell load, and thus maintain performance, we need to keep communications to a minimum and to try to minimise code on the iCells. For a system that is supposed to be hyperactive in terms of surface activity (code and communications) this is far from ideal.

3 Implementing the iSurface video application

One of the main issues the iSurface is likely to face with applications is the transmission or propagation of large amounts of data across the network, possibly to specific destinations. One such application would be to use the iSurface as a video display. This would require the propagation of image data across the surface to the right location at sufficient speed in order to update the picture 24 frames a second.
This is the speed at which the eye perceives smooth animation.
A notion of positioning is at the heart of a video/picture application. Every iCell intended to display part of the image needs to know its position relative to some origin. This is the same origin that the image data refers to. This is accomplished by causing the mobile agents to replicate themselves across the iSurface until a certain area has been covered. The user would then send a message to one of the iCells that it was the new origin for the coordinate system. The coordinates would propagate across the surface and be updated depending on the direction they are transmitted in.

The image used for development and testing is 256 pixels by 256 pixels and uses a greyscale palette (8-bit). This means there is a total data size of 65536 bytes (or 64k). If we choose to transmit the entire picture from iCell to iCell, we automatically take a crippling performance hit. On a Smart Dust based surface with a 10kbps transfer rate, we would have transmission times of about 53 seconds between each pair of iCells. Flood filling this data across the 256 x 256 surface would take between 13420 seconds (about 224 minutes) and 26840 seconds (about 447 minutes). The only advantage of this system is the ease with which an iCell can obtain its correct colour by simply indexing the image data at the correct point. We would argue that the only feasible alternative is to release a stream of pixel data into the system. Each message takes the form of a coordinate ('x' & 'y'), a pixel value (from 0 to 255), and takes up a total of 7 bytes. This results in a transmission time of about 0.007 seconds per message on a Smart Dust based iCell. Although the amount of data entering the system has increased by a factor of 7, we have broken the data up into more manageable chunks. Flood filling a single pixel across the surface is relatively trivial, between 1.8 seconds and 3.6 seconds, but attempting this for all 65,536 pixels would literally take hours.
The ideal solution would be to route the pixel data from the data entry point(s) to the target iCell using the coordinate system and the coordinate values stored in the message. As each iCell can transmit messages in specific directions an agent only need to decide the proper direction from itself to the target and retransmit accordingly.
We have established experimentally that a system divided into quadrants, with strips of 128 iCells in each, and using a data encoding to take advantage of this structure, is the best approach, with a "score" of 128 cycles to form the image. As a cycle is the time taken to transmit one pixel message, we can calculate how long this would be in "real-time". On a Smart Dust based iCell we have a transmission time per cycle of 0.007 seconds. This gives us a time of 0.896 seconds to render a single frame; way over the 0.04 seconds required for convincing animation. Obviously, an iSurface based on Smart Dust couldn't cope with this application. However, we can calculate the communication speed necessary to run the application as it currently stands which would be 224,000 kbps, 22 times faster than Smart Dust and using Moore's Law we can guess at a target date of about 2012.

4 iSurface "vista" wallpaper

The experimentation described in the previous section has demonstrated that the iSurface paradigm is unsuitable for data-heavy and time-critical applications. Responsiveness of applications has been shown to suffer due to iCell load and the distances that data needs to travel across the iSurface. The original aim of the research was for a user-interface for an Amorphous Computer, and this kind of interactivity demands responsiveness.
If we shifted the intended usage of the iSurface away from user interaction into a less interactive visual oriented domain we would reduce this need for responsiveness. We can now consider the iSurface as aesthetically focused, and this opens up many possibilities for applications for a slow iSurface.

One possibility is we could use an iSurface as modifiable wallpaper. The iSurface can already display pictures, as evidenced in the video display experimentation, but problems with transmitting large amounts of data within restrictive timescales mean that the video aspect of the system is not feasible.
A video is essentially a slideshow where 24 slides are displayed every second in order to have smooth animation. In this slow iSurface scenario, a wallpaper display could be a slideshow, without the requirements for updating the screen so often. Instead, an update could take place over minutes, or even hours.

Initially we'll concentrate on describing the "vista" style wallpaper demonstrated in films such as Total Recall and Back to the Future. In this form of electronic wallpaper the entire wall will display an image such as a photograph of the Grand Canyon, or some other natural world scene. It needn't be just walls covered in this way; billboards would be a perfect application area for this, with images easily cycling through a variety of adverts.
This display would operate in exactly the same way as the video display system; a coordinate system would be propagated across the iSurface, with the origin in the top left-hand corner of the room. Image data would be streamed into the iSurface by whatever system the user wishes. For the sake of argument we'll assume the use of the optimal system described under the experimentation for the video display; input strips at the top and bottom of the wall. The pixel data for the image would then be routed across the iSurface to the destination iCells. However, this time it does not matter if data is delayed, or arrives out of order, just as long as the image fully forms.
This wallpaper system will suffer from the same problems as the regular video system if the iSurface is damaged. As messages are directly routed to the destination iCell with no checks between iCells to see if a message has successfully transmitted, a message can be lost if an intermediate iCell is damaged. With the removal of the time-critical factor it would be possible to create a better routing system. There would now be the luxury of adding checks to see if a message was successfully transmitted, or it would be possible to simply propagate the pixel data message across the entire iSurface, guaranteeing that it would reach its destination if any path exists.

One interesting feature of the video system is the way in which images form. Essentially images start to form in the vertical centre of the iSurface and "grow" outwards towards the top and bottom of the screen and this is an interesting effect to watch. These transitions between images have aesthetic merit of their own, and so might be desirable features to have in a wallpaper application. Transitions would be dependent on when pixel data arrives at the destination, and this in turn would be dependent on where and when the data enters the system.
The outwards growing image example is a result of the optimal data entry we used for the video, but if one used another approach the transition would be completely different.

For instance, if data were entered into the system in a random manner, the image would also form randomly. In this case, the transition would look like a "dissolve", where random pixels swap between images.
To accomplish this we could to "shuffle" the pixel data before it enters the system. This effectively takes the pixel stream, which is an array of bytes (assuming a greyscale or 256 colour image) or an array of 3 byte structures (Red, Green, and Blue), and making random swaps between elements of the array such in the following pseudo-code algorithm.

```
PixelStream as Array(length)
For i = 0 to length
        Index1 = random(length)
        Index2 = random(length)
        TemporaryVariable = PixelStream[Index1]
        PixelStream[Index1] = PixelStream[Index2]
        PixelStream[Index2] = PixelStream[Index1]
Next i
```

When this enters the iSurface in the same streaming manner as in the existing video system we will see the image start to form randomly as data now arrives in the wrong order and in a non-optimal position to reach their destination iCell.

By entering data in specific ways and places we could create effects such as "wipes" in various directions and shapes.
With extra coding it would be easily possible to interpolate between the pixel colour values of the old image and that of the new image in order to create a "fade" effect. Synchronisation would allow the entire image to fade at once, or combine it with the various wipe or dissolve transitions in order to create even more interesting visual effects.

5 Animating the iSurface wallpaper

So far this system concentrates on static images. Based on the evidence of the video system, it is fair to say that full motion video is impossible using the current iSurface paradigm. However, video games of the 8-bit and 16-bit era during the 1980s and 1990s were also incapable of

displaying full motion video, but were capable of displaying animated scenes through use of small bitmap images called sprites. A sprite generally had a small number of frames that were bitmaps of limited size and animation was achieved by "blitting", or rasterising, these frames to the screen. Updating an entire screen was extremely costly for the processor and memory, and so sprites were kept as small as possible.
A similar system could work for the iSurface. As an example, say that we were displaying a countryside scene on the iSurface. In the distance, over the rolling green hills, is a windmill, and cows are grazing nearby. The windmill's sails could be a sprite, with a small bitmap of them turning stored in the iCells that make up the windmill image. With a bit of synchronisation the iCells could flip through the six or so frames that might make up this animation. The cows could also be sprites, with an animation that sees them moving slightly. As time is no longer really an issue for the iSurface, it might be possible to extend the use of sprites to small areas of grass blowing in the wind.

For this system to be successful, we would need to define a significant number of coordinate systems similar to the iCell video system, each independent of each other and able to be identified separately. These could be inserted by hand into the scene, and their frames injected directly into the area. This would work on the same basis as defining a button; the sprite's dimensions would be pre-determined and then once injected into the scene the agents would spread until the specified dimensions are met. A coordinate system specific to that sprite would be then grown and a stream of pixel data, like that of the video, would be passed into the sprite. Each pixel element would have a frame number. Using a synchronization method such as that described by **[9]** for use with Amorphous Computers would allow these frames to cycle correctly.

Another simple method of animation on old video games was colour cycling. For example a waterfall image might have the cascading water drawn with a range of blue shades. The colours stored for individual pixels would begin to loop through the range of blue shades creating the illusion of the water moving. Once an iSurface synchronises the iCells, the pixels showing certain values could begin to cycle through the pre-specified range of colours creating the effect of animation.
Taking the use of changing colour one step further, it would be possible to have the iCells brighten or darken a shade of colour based on the time of day. A scene could therefore darken into night, or brighten into daylight.

6 Other methods of wallpaper generation

Real wallpaper isn't typically a photograph style scene; it is usually a repeated pattern of a limited size image. The existing iSurface display system was not intended to display such repeated images and so it would be necessary to make some minor changes to the coding of the agents. Firstly, directed routing would no longer be possible, and secondly, the iCells would need a new way of identifying which pixel data to display.
Simply propagating pixel data across the iSurface would solve the problem of routing data to the intended iCells. In order for a cell to know if it should display a given pixel would require changes to the way the coordinate system is handled. This could be handled easily by allowing the coordinate system to form as before, but when the iCell processes a bit of pixel data it would perform a "modulo" operation on its stored coordinates with the dimensions of the pattern. This would give coordinates within the dimensions of the pattern and thus the correct pixel location. The result would be a repeated pattern across the iSurface.

Excitable media also provides a rich source of material for dynamically created wallpaper. Conway's Game of Life in particular, would be an excellent application for a slow iSurface and would also provide scope for limited user interaction. If iCells know the state of their neighbours, they can use the rules of Life to set their own state and thus the Game of Life will play itself out across the iSurface. If iCells randomly switch their state occasionally, the system would continue to be perturbed and not settle into a stable state. iCells could also react to the touch of users, switching to an on state and thus allowing the user some measure of interaction with the Life environment.

7 iSurfaces as fashion accessories

These slow iSurface applications would not be just limited to use as a coating for walls. They would be usable on a patch of iCells of any shape or size. LCD picture frames have already been released, and these display digital photographs that can be easily switched, possibly even showing a slideshow of digital pictures. A patch of iSurface could easily replicate this functionality, coating picture frames, book covers, or providing decoration for larger artefacts such as tables, or doors.
Where we believe the slow iSurface applications would really excel is in the fields of customisable clothing or body art. As an example of customisable body art, we again take inspiration from the movie Total Recall. There is a scene where a secretary is digitally painting her fingernails. She selects a colour from a computer screen based palette using a stylus and taps it to her nails. The nails instantly change to the selected colour. This would be an extremely easy application for iCells, simply propagating a colour message to their neighbours by means of a flood-fill algorithm. iCells would be mixed into nail polish and could be painted onto the nails, occasionally being touched up when the nail grows or is damaged.
It would be possible to integrate an image based system into this, but it would be necessary to find a way to place and maintain an origin for a coordinate system on a growing substrate (the nail) that will also suffer damage when it is cut.

We can extend this vision of fashion iSurfaces to a whole family of accessories. In this case we require a form of "coordination" between the different accessories that our hypothetical model is wearing in order to ensure that everything matches and doesn't clash.
A possible approach would be to slave everything to the clothes the model is wearing. The clothes would need to transmit a signal that would inform the accessories to change to a specific colour or pattern. This could be achieved by embedding the iCells in the clothing, and including a small RF transmitter somewhere in the outfit. iCells in the accessories would detect the signal and switch to images, patterns, or colours that correspond to this signal.
Suggested accessories in the family include the aforementioned nail polish, as well as hair bands, wristbands, belts, shoes and bags.

Another potential use for iCells in fashion accessories are as "stones" in bracelets, rings, necklaces and earrings. These iCells would be embedded in the place of the stones within the precious metal jewellery and would react to the signals sent from the clothing. Using the synchronisation system for Amorphous Computing described by [9] they could shift colour in an aesthetically pleasing way.

The final possible use for iCells as a fashion accessory would be as body art or Tattoos. Here the iCells would be actually placed on, or even inserted into or under, the skin. A patch of iCells with their own coordinate system could display and shift images to the wearer's taste.
Friera's 'Nanomedicine' designs **[10]** propose a similar application for his nano-robots. This would be accomplished by inserting an array of nano-scale light emitting devices under the skin. This would also operate as an input system with the nano-devices able to detect variations in their positioning under the skin caused by pressure. However, there is no proposal for programming the behaviour of these devices, either individual behaviours or global behaviour.

8 Conclusions

iCell load and communication contention have made aspects of the iSurface's user interface functionality unachievable. In particular, iCell load causes the lack of responsiveness that the majority of time-critical iSurface applications, such as video replay, can exhibit, and our work has shown that this is an inherent problem for an amorphous comuter based system.

This paper introduced the possibility of developing non-time critical applications for the iSurface. These are variations on the time-critical application.

We began by describing the "wallpaper" agent. This is derived from the video system, but takes advantage of the problems of delays in transmitting data to provide interesting aesthetic effects. We described several methods of these graphical effects for a largely static image such as sprite animation or colour cycling.
Moving away from large surfaces such as walls or pictures, We proposed the use of iSurfaces in fashion accessories such as jewellery, clothing and fake extensions like fingernails and that these could display colours or patterns based on user input or dictated by interaction between accessories to properly colour coordinate. These applications are where we believe interesting use and R&D opportunities for this technology might lie.

References

[1] The Foresight Institute, http://www.foresight.org/
[2] MEMS Research at PARC, http://www.parc.com/smart-matter
[3] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. Knight, R. Nagpal, E. Rauch, G. Sussman, and R. Weiss, "Amorphous computing", Communications of the ACM, 43(5), May 2000.
[4] W. Butera, "Programming a Paintable Computer", PhD Thesis, MIT Media Lab, 2001
[5] S. Hollar, "COTS Dust", Master's Thesis, 2000
[6] V. Hsu, J. Kahn, K. Pister, "Wireless communication for Smart Dust", January 1998
[7] J. Kahn, R. Katz, K. Pister, "Next Century Challenges: Mobile Networking for Smart Dust", date and location of publication unknown
[8] K. Pister, J. Kahn, and B. Boser, "Smart Dust: Wireless Networks of Millimeter-Scale Sensor Nodes. Highlight Article in 1999 Electronics Research Laboratory Research Summary", 1999, Available at http://robotics.eecs.berkeley.edu/~pister/SmartDust/
[9] E. D'Hondt, "Exploring the Amorphous Computing Paradigm", Masters Thesis, Vrije Universiteit Brussel, August 2000
[10] R A. Freitas Jr., "Nanomedicine", Volume I: Basic Capabilities, Landes Bioscience, Georgetown, TX, 1999

