TACTILE GRAPHICS WITH MATHEMATICA

Cristian Bernareggi[1], Hooman Tahayori[1], Masoomeh Moharrer[2]
[1] Universita degli Studi di Milano, Dipartimento di Scienze dell'Informazione
Via Comelico 39/41, 20135, Milano, MI, Italy
E-mail: bernareggi@dsi.unimi.it , hooman.tahayori@unimi.it

[2] Lulea University of Technology, Department of Industrial Marketing and E-Commerce, Sweden
E-mail: masmoh-3@student.ltu.se

Keywords: Mathematica, Tactile, Density

Abstract. *Tactile diagrams require considering specific features like resolution, size, density and even fonts to be perceived properly. Mathematica, as a powerful tool, provides integrated environment for technical computing, and has introduced a new generation of mathematical and algebraic capabilities. By the way, by default it does not respect the features necessary for graphs to be drawn to be fully tactile perceivable. In this paper, we have studied different aspects of graphing with the package and have investigated a criterion regarding the density feature for the drawn graphs to be judged how far are tactile perceivable.*

1 INTRODUCTION

Images and mathematical expressions are indispensable in scientific studies. Visually impaired people use to access images through tactile representations. Whilst different tools to print tactile images are available (e.g. graphic embossers and swell paper printers), some problems are met in tactile image generation. Primarily, not that various accessible tools are available to enable blind persons, especially students, to produce tactile graphics without sighted assistance. Therefore a tool which combines both symbolic, numeric manipulation and graphical features is needed.

Access to graphics through tactile and auditive perception is a research issue mainly studied in human computer interaction and in the design of assistive technologies for blind people. Graphical descriptions are related to a large variety of representations, such as diagrams, flowcharts, drawings, and so on. Regardless of the specific graphical representation, two aspects are important: how to enable blind persons to explore graphics and how to design tools to produce graphics through non-visual techniques. Although there exist various non-visual representations of graphics (e.g. through audio descriptions or through haptic tools), this work is focused on tactile graphs and shows how blind people can construct technical drawings by using a symbolic language and obtain a tactile representation suitable to be perceived by touch.

At present, there are two main groups of techniques to enable blind people to produce tactile drawings, techniques based on tactile feedback and techniques exploiting a language to describe images. However, this paper is based on a language to describe images. A specific language is used to declare the parts of the image, such as shapes, captions, labels, etc. The image is generated by a symbolic manipulation software. The resulting image file can be embossed through a tactile embosser after a process that makes it suitable to be perceived tactually. There is no contextual feedback during the preparation of the image except for the symbolic one. Whether the commands to generate the image are carefully grouped, symbolic feedback may be helpful to recognize and search parts of the image being drawn. In the study conducted, Mathematica [8] was employed for symbolic manipulation and Tiger Graphical Embosser was used to print tactile images.

2 TACTILE IMAGES

In order to compensate for the loss of sight, the parts of a drawing are raised on paper to a different height than the background so that they can be perceived by touch. Tactile graphics can be made in many forms which mainly differ in the construction techniques and in the materials used [1], [2]. Valuable discussions and techniques are mentioned in [9].

To obtain a good tactile representation, the three following construction techniques are the most frequently used. They show specific pros and drawbacks:

-*Thermoform Graphics*. A sheet of plastic is heated and vacuumed on top of a model which represents the shape to be perceived by touch. That allows the production of high quality tactile drawings, but it is necessary to mould a totally new model when a new drawing has to be produced.

-*Swell-Paper Graphics*. Paper with a special coating of heat reactive microcapsules enables special printers to shape raised areas. Variable height raised lines and areas can be obtained, but it is a very expensive process.
-*Embossed Graphics*. Some Braille embossers are able to produce tactile images by punching dots into paper in such a way as to form graphics. This is the best cost-effective technique. Nonetheless only few embossers (e.g. Tiger embossers) are able to produce high quality-variable height tactile drawings [5].
Whatever technique is used, tactile representations of drawings have in common a variety of pros and drawbacks.
Pros:
-Qualitative and quantitative information about the relations among the components of a tactile drawing can be detected precisely and quickly by moving all the fingertips over the tactile representation. For example, tactile exploration of a square conveys information synchronously both about the spatial position and the measure of the sides.
-After constructing a simplified mental image of a tactile drawing, namely after a first overall exploration, it is possible to access directly by touch specific parts in order to retrieve details.
-Qualitative drawings (e.g. qualitative diagrams of functions, figures from Euclidean geometry, etc.) can be understood easily and quickly without looking for details.
-Rather complex drawings, which do not mix a great amount of textual and graphical information (e.g. some particular graphs from graph theory, certain complex automata from automata theory, etc.) can be understood quickly by touch.
Drawbacks:
-Color, gradation, shading and other similar visual effects can be hardly represented. At present the best effective way is through Tiger embossers by Viewplus technologies.
-Complex drawings can be explored slowly and many difficulties arise if they are not represented on large surfaces, but unfortunately representations over too large surfaces can be hardly explored by touch;
-Textual labels and captions are often intertwined with lines and shapes (e.g. in cognitive graphs, flowcharts, UML diagrams, etc.). It is extremely difficult or sometimes impossible to combine properly textual Braille descriptions and tactile shapes.
Techniques to overcome some of these drawbacks rely on the possibility to simplify images for tactile understanding by filtering details in order to obtain semantically equivalent tactile descriptions [6], [7].

3 EXPLORATION

In order to comprehend the reasons which complicate the understanding of graphics in a non-visual mode, first of all let us analyze how visual understanding works and which advantages come from a visual exploration of graphical representations. Literature about how sighted people explore and understand graphics, suggests some basic features which should be reproduced by whatever tool for the exploration of non-visual representations of graphics. A relevant contribution is given by Larkin and Simon [3], who compared the mental computation required in solving problems expounded by diagrams and problems represented as series of textual elements (e.g. characters, words, sentences, etc.). They found that the mental workload involved in the solution by diagrams is lower than the one spent to solve the problem presented through text. Two features of diagram understanding were regarded as the main reasons of the different mental workload: easiness to search and immediacy to recognize. Localization of related parts in diagrammatic representations reduces the need for searching, and consequently it facilitates computation, since symbolic descriptions need not be generated or matched. It means that information represented over a two-dimensional plane can be more efficiently grouped and searched for meaningful items than text along a line. As for recognition, diagrammatic representations allow one to immediately understand meaningful shapes, namely relevant parts can be easily isolated and connected to related diagram components. For example, given the parabola with equation $y = -x^2 + 1$, it is straightforward to recognize by sight which is the part in the first and second quadrant, when it is displayed, whereas it takes a longer time to get it from a textual description (e.g. through some points in a table).

Further contribution is provided by a model which accounts for how visual images are perceived [4]. This model shows that the visual image is analyzed hierarchically, from the overall structure down to the fundamental features or elements. It is observed that the clustering of elements or aggregations of basic elements occurs selectively, maximizing the number of connections between units which have important relationships. The importance of each relationship is quickly defined through a comparison with the other possible relationships (e.g. according to closeness as remarked by Palmer[4]). What is natural in the exploration by sight, often becomes difficult in the exploration through tactile devices. It is mainly due to the possibility of exploring only small areas at any one time by touch. Nonetheless, it may be supposed that a tool which has to improve usability of non-visual descriptions of graphics, should allow users to exploit the same cognitive processes previously analyzed for visual understanding of graphics. Therefore, the following exploration features were taken into account in working with tactile images:

-The possibility to easily recognize basic components or clusters in the diagram (e.g. Braille labels and figures).
-The possibility to identify relationships between basic components or clusters.
-The possibility to easily search for components, either clusters or the basic ones.
-Techniques to hierarchically explore the graphical representation.

4 OBSERVATIONS AND RESULTS

Mathematica provides a rich set of functions and options to draw graphs. However, the graphs, to be completely perceivable while being printed on Braille embossers like Tiger Max that we used, must be treated specially to become appropriate for further processing. Consider the following commands.

```
s1=Plot[Sin[x], {x, 0, 2Pi} ,PlotLabel->"Sin - 1"]
s2=Plot[Sin[x],{x,0,2Pi}, ImageSize→{500,500}, PlotLabel->"Sin - 2"]
s3=Plot[Sin[x], {x, 0, 2Pi}, ImageSize→{500,500}, PlotPoints→5000, PlotLabel->"Sin - 3"]
Export["sin1.bmp",s1]
Export["sin2.bmp",s2]
Export["sin3.bmp",s3]
```

The *s1* is plotted with automatic settings whilst in *s2* the graph is enlarged and the image quality is enhanced in *s3* by increasing the sampling rate. We have exported the graphics to bmp files where the size of resulting images are 288*177 , 500*500 and 500*500, respectively. Of course, we can export any plot to many different formats. Among our tests, we got to the point that jpg and bmp are better choices however, according to the nature of the jpg, some noises would be added to the exported files.
Respecting the Plot[options that enable controlling font of the text appearing in a graph, we used *BrailleITA8* font to set all texts in a graph to be appeared in Braille.

```
s4=Plot[Sin[x], {x, 0, 2Pi}, PlotPoints→5000, ImageSize→ {500,500} ,PlotLabel→ "Sin - 4",
      TextStyle→{FontFamily→ "BrailleITA8", FontSize→12}]
```

Controlling the size of resulting graph is also possible via *ImageSize* option of the *Export* function. Increasing the resolution of the exported image, would led to better quality of the image. Be noticed that if for example, we export *s2* with the resolution 300, the image quality will be enhanced noticeable but the time of exporting and the size of resulting image would be increased considerably, e.g. 18.517 sec,2084*2083.

4.1 Investigating Braille Font

While s4 is exported to a .bmp file with default options, the characters in the text, which are in Braille are somehow distorted. This distortion is not that important for sighted people but while being printed on Braille embossers will cause confusion. To reduce this distortion, increasing the resolution of the exported image is a good remedy. But as the requested image size and image resolution increase, the process takes more time and the corresponding file would be bigger and bigger. For example, exporting *s4* with the resolution 300 will result in an image of the size 2084*2083 stored in a 4,240KB file. But the shape of the dots that each Braille character is composed of some, has less distortion and as the resolution increases this distortion reduces.

Here there are contradictions. On one hand according to the part 2.10.19 of Mathematica help browser, a reasonable resolution for printing purposes is 300 or above, but on the other hand exporting images to files with resolutions 300 or more takes longer time and space. Moreover the image would be too large to be fitted in a 14-inch Braille embosser. So we have to process and produce images with reasonable speed and in reasonable space.

4.2 Image Enhancement

We mentioned that, to get an image with a tolerable quality, we'd better increase the image resolution while exporting, but this increase will cause the image size to be increased too and hence cause the future processing very time consuming. We got to this result that it's better to export images as jpg , with image resolution equal to 150. Then process the produced image and finally export the result as .bmp file.

To fulfill the aim, firstly, we devised a function to reduce a .jpg image to a two gray-level image. To do so, we devised *uaverage* function, which takes in a list and calculate the average of all numbers within the list. Then the function *uimportjpg2graylevel* was developed that takes two arguments, input file name, and a threshold in percentage that determines the darkness ratio of each pixel.

```
Clear[uaverage, uimportjpg2graylevel]

uaverage[inp_]:=
    Apply[Plus, Flatten[inp]] / Length[Flatten[inp]]   /; ListQ[inp] && Length[Flatten[inp]]>0

uimportjpg2graylevel[filename_, threshold_]:=
```

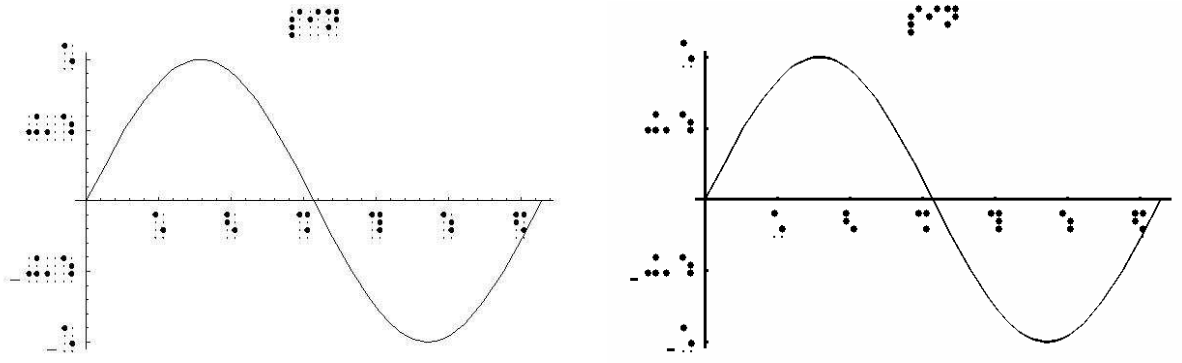
```
(*input:a jpg File*)
(*threshold: color percentage*)
Module[{s,li,lj,i,j, t,thresh},
  s=Import[filename];
  li=Length[s[[1,1]]];
  lj=Length[s[[1,1,1]]];
  thresh=threshold*256;
  For[i=1,i<=li,i=i+1,For[j=1,j<=lj,j=j+1,t:=s[[1,1,i,j]];
      If[uaverage[t]<thresh,s[[1,1, i,j]]=0,s[[1,1,i,j]]=255 ] ] ];
  s[[1,4]]=ColorFunction->Automatic;
  s
]
```

We devised several filters and compared their output on several graphs – further discussion would be find in [9]. The empirical results obtained showed that the best one is the *ufilter* which removes noises, especially those that were related to the nature of the font and also increases the thickness of lines and curves. In this function, we can determine the width of the window and the threshold – which must be mentioned in percent- by which we are to determine the blackness.

```
Clear[ufilter]
ufilter[gs_, window_, threshold_] := Module[{whiteness, halfwin, , thresh, sm, gs2,li, lj, i, j, k},
  whiteness=window*window*256;
  thresh=N[Floor[whiteness*threshold]];
  Print[thresh];
  halfwin=Floor[window/2];
  li=Length[gs[[1,1]]];
  lj=Length[gs[[1,1,1]]];
  gs2=gs;
  For[i=halfwin+1,i<li-halfwin,i++,For[j=halfwin+1,j<lj-halfwin, j++ ,
      sm=Apply[Plus,Flatten[Take[gs[[1,1]],{i-halfwin,i+halfwin},{j-halfwin,j+halfwin}]]];
  gs2[[1,1,i,j]]=If[(whiteness-sm)>thresh,0,255]]];
  gs2[[1,4]]=ColorFunction->Automatic;
  gs2
]
```

At last we designed *uprintplot* which plots the requested function in the desired range and with the requested options; remove the noises and saves it in the file named as its first argument. Moreover it increases the thickness of lines and reforms dots in Brille fonts to be easily perceivable, An example is shown in Fig. 1.

```
Clear[uprintplot]
uprintplot[fn_, f_, {x_, min_,max_}, opt___]:=
  Module[{s, s1},
  Export["Temp.jpg",Plot[f, {x, min, max}, opt], ImageResolution→150];
  s=uimportjpg2graylevel["Temp.jpg", .33];
  s1=ufilter[s,3,.25];
  Export[fn, s1]
]
```



(a) (b)

Figure 1. (a) The *Plot* output (b) The *uprintplot* output

4.3 Image estimation

An important problem about images to be printed on the Braille embosser is how crowd the plot is. If the plot is crowd, when printed, the shapes are not easily perceivable and even detectable. For example in the Fig. 2, the head of the arrows are mixed together and causes the graph not be properly perceivable.

```
<<DiscreteMath`combinatorica`;
gr=MakeGraph[Range[8],(Mod[#1,#2]□0)&];
ShowGraph[gr];
```

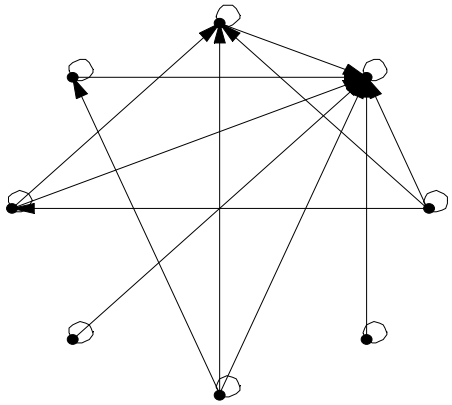


Figure 2. A crowd graph

We have drived a criterion to judge when an image is crowd i.e. is not fine to be printed on the Braille embosser. To do so, we conducted some statistics on the images. We devised a function to determine, what we called it the *density*. The function *udensitypercentage*, was developed to calculat the overall density (darkness percentage), image samples darkness percentage, standard deviation between the samples and deviations between samples and overall darkness percentage. By comparing the outcomes, the fact that density is the best factor to base judgments on was revealed.

We conducted empirical tests on different graphs –some are depicted in the following- and we got to this fact that if *density* is around 3 (mainly less than 3.5) the image would be considered not too crowd and would be expected to be sensed and perceived completely. Existing mechanism to overcome crowd graphs is to enlarge them. Enlarging graphs will couse the density to be reduced, however, it may cause the graphs to be too large to be fitted in a 14-inche page. Hence it is necessary to find an equilibrium in between.

```
Clear[udensitypercentage];
udensitypercentage[fn_]:= Module[{s,win,li,lj,tot,Den, ls, samples, x, y, av, smpav, stddv, mydiv, i},
s=Import[fn];
 s11=s[[1,1]];
 fs11=Flatten[s11];
 sm=Count[fs11,0];
 tot=Length[fs11];
 Den=100*(sm)/tot //N;
 li=Length[s[[1,1]]];
 lj=Length[s[[1,1,1]]];
 win=7;
 ls={};
 samples=Floor[.3*(li*lj)/(win^2)];
 For[i=1,i<=samples,i++,
    x=Random[Integer,{1,li-win}];
    y=Random[Integer,{1,lj-win}];
    av=(win*win-Apply[Plus,Flatten[Take[s[[1,1]],{x,x+win-1},{y,y+win-1}]]])/(win*win)//N;
    ls=Append[ls,av]];
 smpav=Apply[Plus,ls]/Length[ls]//N;
 stddv=StandardDeviation[ls];
 mydiv=Apply[Plus,(ls-Den)^2]/Length[ls];
 Print["Den:",Den, "   STDDV:",stddv,"   SMPAV:",smpav, "   MYDIV:", mydiv, "   LOOP:", loop];
 ]
```

Test Case 1: As a simple case we tried the following Plot function and obtained the results shown in Table 1. It is clearly shown that the criterion coinsides with the blind people judgments.

*Plot[{Sin[x],Cos[x]*Sin[x],Tan[x]}, {x, 0, 2Pi},PlotLabel->"Sin , Sin*Cos, Tan", TextStyle→{FontSize→20}, AxesLabel→{X, Y},ImageSize→{600,600}]*

Experiments					Judgment
Automatic	*:Den: 5.73799*	*STDDV: 0.119015*	*SMPAV: 0.0672423*	*MYDIV: 32.1715*	*Bad*
*300*300*	*:Den: 3.56222*	*STDDV: 0.0866039*	*SMPAV: 0.0363347*	*MYDIV: 12.4394*	*Bad*
*400*400*	*:Den: 2.39438*	*STDDV: 0.0715643*	*SMPAV: 0.0246816*	*MYDIV: 5.62056*	*Good*
*500*500*	*:Den: 1.7716*	*STDDV: 0.0619138*	*SMPAV: 0.0190876*	*MYDIV: 3.07513*	*Good*
*600*600*	*: Den: 1.39917*	*STDDV: 0.0534632*	*SMPAV: 0.0152783*	*MYDIV: 1.918*	*Good*

Table 1: The Results of Test Case 1

Test Case 2: Complete binary trees were studied. A complete binary tree would be generated as follows:

```
<<DiscreteMath`combinatorica`;
numberofnodes=2^3-1;
completebinarytree1=CompleteBinaryTree[numberofnodes];
ShowGraph[completebinarytree1];
```

Seven complete binary trees were embossed varying the number of nodes: 2^2-1, 2^3-1, 2^4-1, 2^5-1, 2^6-1, 2^7-1 and 2^8-1. The sixth and seventh complete binary trees were not perceivable clearly by touch. Therefore can be stated that without changing other parameters for graphical representation, a complete binary tree with depth at most equal to 5 can be perceived properly by touch on a single page. Furthermore, it can be stated that a complete or not complete binary tree can be embossed properly on a single page only whether its depth is at most 5. This constraint is not so restrictive for many educational purposes. Table 2 shows the results.

Experiments					Judgment
Depth 2	*Den: 0.700473*	*STDDV: 0.0344662*	*SMPAV: 0.00599767*	*MYDIV: 0.483481*	*Good*
Depth 3	*Den: 1.114*	*STDDV: 0.0430436*	*SMPAV: 0.0114318*	*MYDIV: 1.21752*	*Good*
Depth 4	*Den: 1.8627*	*STDDV: 0.0830124*	*SMPAV: 0.0259228*	*MYDIV: 3.38064*	*Good*
Depth 5	*Den: 3.20337*	*STDDV: 0.0965271*	*SMPAV: 0.0387634*	*MYDIV: 10.024*	*Good*
Depth 6	*Den: 5.57846*	*STDDV: 0.110166*	*SMPAV: 0.05096*	*MYDIV: 30.5654*	*NotBad-NotGood*
Depth 7	*Den: 8.93374*	*STDDV: 0.166073*	*SMPAV: 0.0853359*	*MYDIV: 78.3218*	*Bad*
Depth 8	*Den: 12.8701*	*STDDV: 0.218218*	*SMPAV: 0.129493*	*MYDIV: 162.371*	*Bad*

Table 2: The Results of Test Case 2

Test Case 3: We also tried the example shown in Figure 2. Again the results coinsides with the observation- Table 3 shows the details.

Experiments					Judgments
Mode =1, Ran 8:	*Den: 4.00873*	*STDDV: 0.108325*	*SMPAV: 0.0452441*	*MYDIV: 15.7209*	*Bad*
Mode =1, Ran 7:	*Den: 3.79292*	*STDDV: 0.115868*	*SMPAV: 0.0394075*	*MYDIV: 14.1023*	*Bad*
Mode =1, Ran 6:	*Den: 2.88749*	*STDDV: 0.0960015*	*SMPAV: 0.0361873*	*MYDIV: 8.13913*	*Good*
Mode =1, Ran 5:	*Den: 2.43056*	*STDDV: 0.079858*	*SMPAV: 0.0249567*	*MYDIV: 5.79327*	*Good*
Mode =2, Ran 5:	*Den: 1.51066*	*STDDV: 0.0670414*	*SMPAV: 0.0192811*	*MYDIV: 2.22869*	*Good*
Mode =2, Ran 6:	*Den: 2.07972*	*STDDV: 0.0604801*	*SMPAV: 0.0166647*	*MYDIV: 4.25983*	*Good*
Mode =2, Ran 7:	*Den: 2.57161*	*STDDV: 0.0691792*	*SMPAV: 0.0225818*	*MYDIV: 6.50234*	*Good*
Mode =2, Ran 8:	*Den: 3.25641*	*STDDV: 0.0927486*	*SMPAV: 0.036469*	*MYDIV: 10.3766*	*Good*
Mode =2, Ran 9:	*Den: 3.81101*	*STDDV: 0.0906073*	*SMPAV: 0.0361068*	*MYDIV: 14.2581*	*NotBad-NotGood*
Mode =2, Ran 10 :	*Den: 4.65254*	*STDDV: 0.115183*	*SMPAV: 0.0505172*	*MYDIV: 21.1918*	*Bad*

Table 3: The Results of Test Case 3

Test Case 4: A clique is generated as follows:

```
<<DiscreteMath`combinatorica`;
numberofnodes=3;
clique=CompleteGraph[numberofnodes];
ShowGraph[clique];
```

It was observed that a circular embedding was used by default to represent complete graphs. It means that the nodes are distributed equally spaced along a circumference. It is a very effective way to represent cliques also for tactile reading. A certain number of cliques were embossed and it was observed that a clique made up of 10 nodes –shown in Fig. 3 - is the largest one which can be properly perceived by touch. It is somehow in contrast with the above criterion and the reason is due to the way the nodes are arranged.

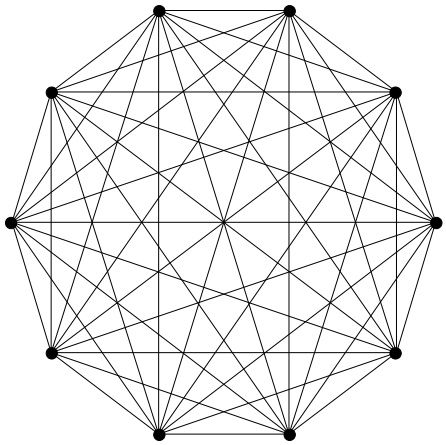


Figure 3: 10-Clique

Experiments	Judgments
5-Clique: Den:2.48119 STDDV:0.0672311 SMPAV:0.0230648 MYDIV:6.0469	*Good*
6-Clique: Den:3.58796 STDDV:0.0689447 SMPAV:0.0352212 MYDIV:12.6267	*Good*
7-Clique: Den:4.62722 STDDV:0.0755377 SMPAV:0.0460089 MYDIV:20.9932	*Good*
8-Clique: Den:5.6942 STDDV:0.0852956 SMPAV:0.0628346 MYDIV:31.7196	*Good*
9-Clique: Den:7.38812 STDDV:0.107824 SMPAV:0.0835245 MYDIV:53.3687	*Good*
10-Clique: Den:9.06154 STDDV:0.102836 SMPAV:0.0938292 MYDIV:80.4303	*Good*
11-Clique: Den:10.659 STDDV:0.124066 SMPAV:0.109608 MYDIV:111.305	*Good*
12-Clique: Den:12.3831 STDDV:0.123657 SMPAV:0.130137 MYDIV:150.149	*Good*
13-Clique: Den:14.4375 STDDV:0.160951 SMPAV:0.151673 MYDIV:204.109	*Bad*
14-Clique: Den:16.5582 STDDV:0.158455 SMPAV:0.167814 MYDIV:268.668	*Bad*
15-Clique: Den:18.527 STDDV:0.165952 SMPAV:0.202955 MYDIV:335.797	*Bad*

Table 4: The Results of Test Case 4

Test Case 5: It was studied how directed graphs with labels on vertices and on edges can be produced. By means of *Graph* function is possible to specify further options (*VertexLabelPosition, EdgeLabelPosition*) in order to put labels in specific positions with respect to the node or to the edge. This allows to better control how the image is generated and then it allows to obtain tactile graphs without overlapping labels. Details are shown in Fig. 4 and Table 5.

```
<<DiscreteMath`combinatorica`;
v={{{20,5},VertexLabel→"A"},{{10,10},VertexLabel→"B"},{{30,15},VertexLabel→"C"},{{25,10},VertexLabel
→"D"}};
e={{{1,2},EdgeLabel→"AB"},{{1,4},EdgeLabel→"AD"},{{2,3},EdgeLabel→"BC"}};
directedgr=Graph[e,v,EdgeDirection→True,TextStyle->{FontFamily->"BrailleITA8",FontSize->20}];
sg=ShowGraph[directedgr,VertexLabel→True];
```

5 CONCLUSION

Scientific literature usually contains graphs and diagrams that are sometimes main keys to understand related concepts. Comparing some factors like bandwidth, resolution and adaptation between visual and tactual

perception reveals the important fact that images to be perceived tactually must be dealt specifically, respecting the nature of tactile sense, otherwise the important information in the image would be missed and consequently the related concept will not be understood as it should.

This work aimed to integrate procedures to produce tactile graphics with symbolic manipulation features already present in Mathematica. This is one more step towards a full work environment for blind students in scientific studies. We showed how it is possible to obtain high quality technical tactile images through Mathematica. Encouraging results are obtained as for function, diagrams and graphs (e.g. those used in graph theory). Further development will focus on a package containing procedures to automatically generate sets of images in a specific knowledge domain (e.g. function diagrams, automata, etc.).

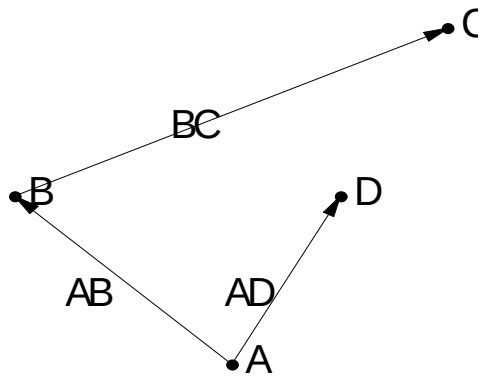


Figure 4: Test case 5 (Image is not enhanced by the aforementioned filter)

Experiment	Judgment
Den:1.64569 STDDV:0.0623519 SMPAV:0.0158999 MYDIV:2.66009	*Good*

Table 5: The Results of Test Case 5

REFERENCES

[1] Edman, P.K., (1991), *Tactile graphics,* American Foundation for the Blind.

[2] John A. Gardner, (1996), "Tactile graphics, an overview and resource guide", Information Technology and Disabilities, vol. 3, n. 4,

[3] Larkin, J H. and Simon, H A., (1987), "Why a diagram is (sometimes) worth ten thousand words", Cognitive Science, vol. 11, pages 65-99, Cognitive Science Society,

[4] Palmer, S., (1977), "Hierarchical structure in perceptual representation", Cognitive psychology, vol. 9, n. 441, Elsevier,

[5] Patricia Walsh and John A. Gardner, (2001), "TIGER, a new age of tactile text and graphics", Proceedings of the 2001 CSUN International Conference on Technology and Persons with Disabilities, Los Angeles,

[6] Sergio E. Hernandez and Kenneth E. Barner, (2000), "Tactile imaging using watershed-based image segmentation", Assets '00: Proceedings of the fourth international ACM conference on Assistive technologies, ISBN 1-58113-314-8, pages 26-33, ACM Press,

[7] Shinohara, M., et al., (1996), "3d-2d-images, Experimental study of 3D tactile display: a Step towards the improvement", Proceedings of ICCHP 96 (Linz), pages 749-754,

[8] Wolfram, S., (2003), *The Mathematica book*, 5th edition, Wolfram Media, Inc., ISBN 1579550223

[9]Way, Thomas P. (1996), *Automatic Generation of Tactile Graphics*, PhD Thesis.