

SVG Rendering for Internet Imaging

S. Battiato, G. Di Blasi, G. Gallo, G. Messina, S. Nicotra

Abstract - The SVG (Scalable Vector Graphics) standard allows representing complex graphical scenes by a collection of graphic vectorial-based primitives, offering several advantages with respect to classical raster images such as: scalability, resolution independence, etc. In this paper we present a full comparison between some advanced raster to SVG algorithms: SWaterG, SVGenie, SVGWave and some commercial tools. SWaterG works by a watershed decomposition coupled with some ad-hoc heuristics, SVGenie and SVGWave use a polygonalization based respectively on Data Dependent and Wavelet triangulation. The results obtained by SWaterG, SVGenie and SVGWave are satisfactory both in terms of perceptual measured quality and compression ratio.

Index Terms—SVG, Triangulation, Watershed, Wavelet

I. INTRODUCTION

SVG is a language for describing two-dimensional graphics and graphical applications in XML ([6], [15]). In this work we are interested in reviewing some heuristic techniques to cover the gap between the graphical vectorial world and the raster real world typical of digital photography in a specific application. Major details can be found directly at the W3C site [18]. An exhaustive overview of the recent SVG development and related applications can be found in the proceedings of SVG Open Conference [19].

Two different techniques SVGenie [3] and SVGWave [5] have been applied to approximate local pixel neighbourhood by triangles: the Data Dependent Triangulation (DDT) ([7]), the Wavelet Based Triangulation (WBT) ([12]). The DDT replaces the input image with a set of triangles according to a specific cost function able to implicitly detect the edge details. The overall perceptual error is then minimized choosing a suitable cost function able to simplify triangulation. Recently further optimization of such function has been introduced for Colour Filtering Array demosaicing ([17]) and for image interpolation ([22]).

The WBT uses the wavelet transformation to properly extract the details from the input images; a reverse process of triangulation, starting from the lowest level, is applied to

derive the final WBT. Therefore a triangulation is, by first, achieved at the lowest level, introducing large triangles; then the process is iterated introducing new smaller triangles for each level, according to the wavelet transformation.

Both these decomposition could be directly managed by SVG primitives. Although the quality achieved in this way is rather good the size of the resulting files may be very large. The triangulation of both DDT and WBT are then processed by the polygonalization. The polygonalization minimizes the dimensions of the resulting files by merging triangles together, reducing the amount of redundancies.

Another recent approach uses a raster-to-vector technique SWaterG ([4]) by advanced watershed decomposition ([8], [21]) coupled with some ad-hoc heuristics devoted to obtain high quality rendering of digital photography. The system is composed by two main steps: first, the image is partitioned into homogeneous and contiguous regions using watershed decomposition. Secondly, an SVG representation of such areas is achieved by ad-hoc chain code building.

In [1] a conversion tool able to obtain SVG representation of *Geomatics* data has been presented but the overall results are limited by the small set of SVG primitives used for the final rendering (e.g. only <rect>). The final .svgz (i.e. compressed) files are approximately 2.5 times larger than the original raster images.

Recently, some commercial and free software have been developed using some ad-hoc solution to the “raster to SVG” problem. Among other: Autotrace ([2]), Kvec ([10]), Potrace ([14]), Ras2Vec ([16]), Vector Eye ([20]). Almost all software are devoted to SVG rendering of graphic images (e.g. clip art) starting from different image formats, showing in such case good performances but also several perceptual drawbacks when applied to digital pictures.

The paper is structured as follows. Sections II and III introduce some details of triangulation by using respectively the DDT and WBT approaches. The next section briefly reviews the heuristics used to drive the final polygonalization. Section V is devoted to the description of the watershed approach. Section VI gives a brief description of some related commercial tools. Section VII reports some details about the experimental results together with some comparisons with other techniques. A final Section closes the paper tracking also directions for future works.

II. SVGENIE

A triangulation is a partition of a two-dimensional plane into a triangles set. The triangulation techniques, named Data Dependent Triangulation (DDT), try to optimize the approximation of a particular function, taking also advantage from the information obtained from the co-domain.

Manuscript received November 26, 2004.

S. Battiato is with the Dipartimento di Matematica e Informatica, University of Catania, viale A. Doria, 6, 95125 Catania, ITALY (corresponding author; phone:+39 0957383020; fax:+39 095330094; e-mail: battiato@dmf.unict.it).

G. Gallo, G. DiBlasi, S. Nicotra are with the Dipartimento di Matematica e Informatica, University of Catania, viale A. Doria, 6, 95125 Catania ITALY (e-mail: {gallo, gdblasi, snicotra}@dmf.unict.it).

G. Messina is with the STMicroelectronics, Imaging Applications & Architectures Team, AST Catania Lab - Imaging Group - STMicroelectronics S.r.l.- FAB. M6, Contrada Blocco Torrazze, Casella Postale 421- 95121 Catania, Italy (e-mail: giuseppe.messina@st.com).

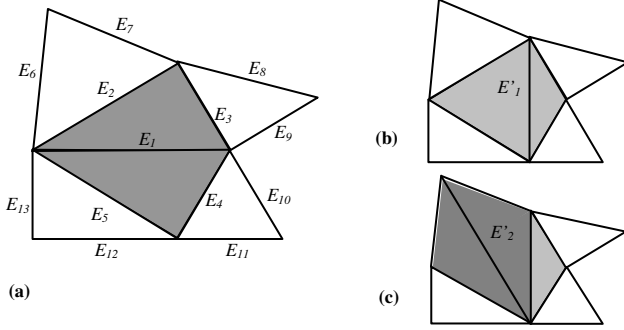


Figure 1 - Example of look-ahead (a) Initial configuration, (b) Exchange of E_1 with the opposite diagonal E'_1 . (c) Simultaneous exchange of E_1 with E'_1 and E_2 with E'_2 .

The algorithm introduced by Lawson ([11]) is an iterative technique able to find a locally optimal triangulation. This approach inspects all the inner edges and exchanges those that reduce the total cost of the triangulation; the algorithm iterates the process until the total cost is still unchanged.

Another approach, based on Lawson's method and developed by Yu, Morse and Sederberg ([22]), has been used as starting point to develop the proposed approach:

1. Every pixel of the image is split into two triangles by using the diagonal with the lowest cost;
2. If the polygon, formed by adjacent triangles, is convex, the following look-ahead step is performed:
 - a. The diagonals are swapped if the swapping introduces a reduction of the edges cost sum;
 - b. Otherwise, for each edge E_i of the polygon, with $i=2, \dots, 5$, if the exchange of E_i with its opposite diagonal E'_i , and the simultaneous swapping of the edge E_i with its opposite diagonal E'_i decrease the costs of the edges of the polygon, then both exchanges are achieved (see Figure 1).

Step 2 is iterated until a locally optimal triangulation is reached. To estimate the triangulation, the optimality criterion is fixed using the following cost function:

$$Cost(E) = |\nabla P_1| \cdot |\nabla P_2| - \nabla P_1 \cdot \nabla P_2 \quad (1)$$

where E is a common edge of two triangles and $\nabla P_i = (a_i, b_i)$ are the gradient of the planes P_i , that are the interpolating linear functions of the triangles. A few iterations are needed to guarantee an effective coupling between original edges distribution and related triangle vertexes map. In our case, to further speed-up the overall process we used a suitable approximation by the cost function described in [7] although alternative approaches could be investigated ([17],[22]).

Therefore the global triangulation cost is summarized in:

$$cost(T) = \sum_{\substack{\forall i, j, 2 \text{ contiguous} \\ \text{in } T}} [(|a_1| + |b_1|) \cdot (|a_2| + |b_2|) - (a_1 \cdot a_2 + b_1 \cdot b_2)] \quad (2)$$

III. SVGWAVE

The Wavelet Based Triangulation algorithm (WBT) generates a triangulation which approximates an image, taking advantage from the characteristics given by the multi-resolution analysis carried out by the discrete wavelet transformation. Due to the wavelet locality property, the

decomposition coefficients provide information about the inner details of restricted regions of the image.

Once the wavelet decomposition coefficients until the level l_i have been obtained (e.g. using a standard bi-orthogonal model) the successive steps to generate the triangulation are:

- Create a initial triangulation using predefined structures and the wavelet coefficient of each level l_i ;
- For each level in the range between the first l_i and the final level l_f iterate n time the following steps:
 - Select the triangles with a higher energy detail;
 - Refine the selected triangles;
 - Adapt all the triangles, finding the good trade-off between the current error and the Delaunay's criterion;
 - Adapt all the vertices with a neighborhood low energy detail;
 - Decimate the shorter edges;
 - Decimate the unnecessary vertices;
 - Repeat the triangle adaptation step.

The image is then subdivided into a regular region grid, taking into account the current resolution and the initial level. Each predefined model represents the better way to achieve a triangulation of a simple region, due to the presence of horizontal, vertical, diagonal or no predominant discontinuities. By using the wavelet coefficients it is possible to discriminate among the aforementioned cases building region by region the initial triangulation of the image. At each level the resolution of the grid is increased, and each triangle belongs to several regions. The refining operation increases the number of triangles into the highly texturized regions and reduces the number of triangles into the less texturized regions. The triangles containing regions with a energy detail higher than a fixed threshold,

$$max_{energy}(T) > \delta_e \quad (3)$$

are subdivided. The subdivision of the triangle T is achieved considering the possible discontinuities directions. If the center of the region belongs to the boundary of the triangle T , adjacent to the triangles T' , the subdivision is accomplished jointly to T' , in function of the common edge to which the center belongs. If both the single triangle subdivision and the coupled triangles subdivision could not be achieved, the simple subdivision using the triangle center of gravity is accomplished.

Furthermore adjustment operators perform different kind of procedures to reduce redundancies and to remove useless vertices or triangles. The first adjustment operator uses the Delaunay's criterion jointly to the Mean Square Error (MSE) to perform an optimal triangulation. The second adjustment operator moves the vertices to the area of the image with larger discontinuities, thus enhancing the characterization of the edges (Figure 2). Other used local operators are the edge removal operator and the vertex removal operator (Figure 3).

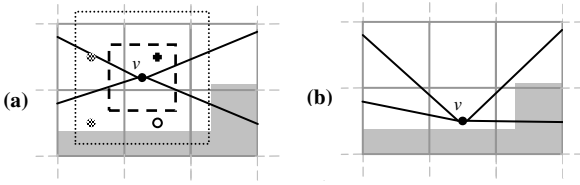


Figure 2 - Displacement example of a vertex v . (a) The only region center (the cross) belonging to the central dashed area has not enough detail energy to satisfy the displacement constraints; also in the larger area the only center with such constraints is the circled dot one. (b) Vertex v after the

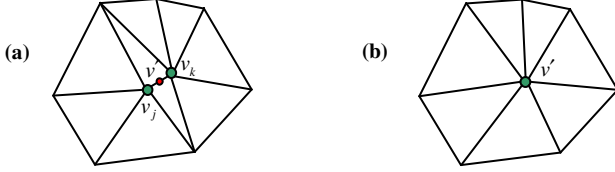


Figure 3 - Example of edge and vertices removal. (a) The edge is shorter than a fixed minimum threshold. (b) The vertex v is fixed as a new vertex.

IV. POLYGONALIZATION

Both the SVGenie and SVGWave approaches require a further polygonalization step. Major details can be found in ([3], [5]). The polygonalization algorithm, starting from the derived triangulation, inspects every triangle looking for similarity among the adjacent polygons or triangles to achieve the fusion.

The measure of similarity is achieved in terms of intensity values into a specific color space (i.e. $YCbCr$ space has been used, because it contains a better liaison to the Human Visual System [13]). The Absolute Difference Separated Ratio (ADSR) is used to estimate the similarity between two elements. The ADSR is true only if

$$\forall j = 1, 2, 3 \quad |I_j(a) - I_j(b)| \leq \tau_j \cdot (\max(I_j) - \min(I_j)) \quad (4)$$

that is for each color intensity I_j the distance between the values a and b must be less or equal to the product of the fixed threshold τ_j and the maximum difference.

It is also possible to reduce the dimension of the final SVG output file by considering further characteristics of the color space and of the polygonalization itself:

1. By giving more relevance to the luminance channel Y rather than to the chrominance;
2. The polygonalization can also be obtained by incremental fusion steps using different increasing thresholds, generating different levels of approximation: starting from zero, the values of τ_j are increased at each fusion step until the threshold value τ_l reaches a fixed maximum approximation target τ ;
3. The boundaries of the single polygons can be simplified to minimize the number of vertices removing all vertices belonging to the same straight line retaining only the first and the last points;
4. The color of each element of the polygonalization is assigned in function of the colors of the respective vertices. It is better to give a different weight to the internal vertices colors. Given n_i and n_e the number of inner and outer vertices, the following values:

$$m_i^j = \frac{\sum_{v_i} c_i^j}{n_i}, \quad m_e^j = \frac{\sum_{v_e} c_e^j}{n_e} \quad (5)$$

are the averages of inner and outer color components. Fixing a threshold $\alpha \in [0, 1]$ the color value of the element j of the polygon will be:

$$c_P^j = \frac{[n_i + \alpha(n_i - n_e)] \cdot m_i^j + [n_e + \alpha(n_e - n_i)] \cdot m_e^j}{n_i + n_e} \quad (6)$$

The proposed algorithm belongs to the polygons-based merging class. Given two distinct and adjacent triangles T_1 and T_2 :

- if T_1 and T_2 do not belong to any polygon then the new polygon P , union of the two triangles, is added to the polygonalization;
- if T_1 do not belong to any polygon but T_2 belongs to the polygon P and T_1 is similar to P then T_1 is combined to P ;
- if T_1 and T_2 belong to two different polygons P_1 and P_2 and those polygons are similar, then P_1 and P_2 are combined together and P_2 is removed from the polygonalization.

V. SWATERG

SWaterG ([4]) is a raster to vector approach based on two major steps: partitioning and contouring. The partitioning step uses watershed decomposition that has been largely used in Computer Vision both for image segmentation and classification ([21]). The algorithm returns a partition of the input image into a number (typically large) of sub-regions called “basins” with the property that each one of them is almost homogenous.

First, minimal values that represent the deepest point in the “catchment basin” are computed. This is obtained evaluating the gradient of the image. For each RGB-component of the image, let G_k be the gradient of the layer k . In order to merge each computed level, we experimented with several metrics choosing L_{MAX} as the most suitable in terms of efficiency and final results. Values of gradient are then normalized into the range $[0, 255]$:

$$G = NORMALIZE(\text{Max}_{k=1 \dots \#layer}(G_k(x, y))) \quad (7)$$

The various level of gradient quantization can significantly reduce the well known over-segmentation problem thus obtaining a lower total number of basins.

Let H be the histogram of the gradient, list of pixels are processed starting from minimum gradient values by assigning to each point in the current list a basin in the following manner:

```

for z = 0 to #number_of_gradient_levels
  for each pixel p in H(z)
    Let m be the number of the basins in
    the 8-neighborhood of the pixel p, then p can be
    marked as
    • a new basin (m = 0)
    • as an edge (m > 1)
    • aggregated to an existing basin (m=1)
  
```

After the segmentation step, a chain code representation ([8]) of each basin is computed following its perimeter with pertinent automata. To build chain code for each basin a deterministic automaton has been implemented. Starting from

the top-left point, the procedure looks for the next pixel of the perimeter in the neighbourhood following a clockwise direction until the path returns to the first pixel.

VI. SVG TOOLS

Some commercial and free tools have been developed for raster-to-vector conversion. Each tool follows a different strategy leading to different results, but in any case when applied to digital images representing real scenes (e.g. picture) the final results show some drawbacks. After several experiments, the best trade-off between quality and size, with respect to our case study, has been obtained using the following software: Vector Eye [20], KVec [10] and Autotrace [2]. In the following some related details for each of them are briefly reported.

A. Vector Eye

Vector Eye generates an SVG file using the following steps: segmentation, simplification, contour extraction, curve approximation. Initially, the algorithm produces a draft segmentation using two threshold values: *low_color_threshold* and *hi_color_threshold*. These values are used to establish the minimum color difference between adjacent regions:

- *low_color_threshold* (an integer value in [0, 440]) is used for regions with low details;
- *hi_color_threshold* (an integer value greater than *low_color_threshold* and lower than 440) is the difference threshold between details.

The segmentation step simplifies and prunes regions with low details having an overall area lower than an integer value *minimum_area*. In the third step regions contours are extracted, using line segments. Finally, the overall contours are approximated through Bezier cubic curves. The parameters used are:

- *rel_curvefit* and *abs_curvefit* establish a maximum error (in pixel) between contours and the approximating curve; *rel_curvefit* depends on the curve length, *abs_curvefit* is curve independent;
- *linear_simplify*, is used to transform curves into lines, limiting the maximum allowed error.

Overall results using VectorEye are good: images appear sharp even if not very detailed, colors maintain a good quality and the PSNR value is high. Also in term of *bpp* (bit per pixel) results are on the average.

B. KVec

The first step performed by KVec is color quantization: the colors number is reduced using the parameter *quantize*. Subsequently the image is segmented using a technique able to find connected pixel regions of similar color (with a difference lower than the value of the parameter *gapfill*). Regions containing a number of pixels lower than a threshold value *grit* are pruned. The next step performs the contour tracking allowing to manage regions with a width lower than the threshold value *lwidth* as lines of suitable thickness. Contours are then approximated using Bezier curves. It is also possible to perform a sub-sampling (parameter *subsampling*) in order to reduce the final file size. Results are not very good: files size are too high, images are blurred, with a bad “blocked” effect and colors are too much different from the original image.

Numerically PSNR value is low.

C. Autotrace

Autotrace performs two preliminary operations: color reduction and *despeckling*. Color reduction is obtained using *color-count* (an integer value in [1, 256]). *Despeckling* step removes tightened areas, using *despeckle-tightness* (a real value in [0,8]). After the contour tracking, an iterative process is executed in order to approximate contours using splines (parameter *error-threshold* establishes the maximum allowed error). Splines are then simplified using two different methods:

- globally, using a single line when the limit imposed by *line-reversion-threshold* is respected;
- locally, a single portion of a curve is substituted by a line, using the *line-threshold* parameter.

This method leads to a “snow” effect caused by the presence of gaps between near elements. This fact penalizes the final quality both in terms of PSNR and perceptual quality. The relative file size, when applied on digital pictures, is very huge.

In the next Section the overall performances of these software will be compared with our recent solutions using a meaningful dataset.

VII. EXPERIMENTAL RESULTS

Each triangle/polygon can be easily remapped into a SVG representation in the following way:

```
<path d="M x1,y1 L x2,y2 L x3,y3 Z" stroke="#FFFFFF"
      fill="#FFFFFF"/>
```

where x_1, y_1 , x_2, y_2 , x_3, y_3 are the vertex coordinates (x_i, y_i for further vertices), and #FFFFFF is the filling colour, obtained using the aforementioned equation. The generalization to RGB images is simply obtained, replacing #FFFFFF with #RRGGBB, where RR, GG, BB are respectively the hex representation of the red, green and blue mean value inside the considered triangle/polygon.

All the proposed methods have been implemented in C programming language. A large set of experiments have been performed in order to verify the capabilities of the proposed techniques comparing the SVG rendering with respect to some commercial tools. The overall parameters involved in SVGenie, SVGWave and SWaterG have been experimentally tuned as described in ([4],[5]). All commercial tool parameters have been manually tuned to obtain the best overall performances in terms of perceived quality. On the other hand parameter tuning is in this case extremely difficult due to lack of documentation.

Figure 6 shows a magnified detail of a digital picture vectorized with our three methods compared with the SVG rendering of VectorEye, AutoTrace and Kvec. The picture shows how our results outperform other tools in terms of perceptual quality.

Similar results have been observed using a large dataset composed by images from Kodak collection ([9]), classical images, text and clip-arts. While SVGenie and SVGWave tend to work granularly, introducing some “zipper” effect, SWaterG shows good performances, mainly for digital photography. In some cases we are not able to distinguish between the original

raster photography and the final SVG file. Further subjective experiments are in progress.

Quantitatively experiments concerning sizes of output files both in SVG and SVGZ together with PSNR values are reported in Figures 5, 6, 7. Note that bpp performances of Autotrace have not been included in the plots due to the different order of magnitude (about ten times greater than the average results of other methods).

Results show how all the advanced methods reach satisfactory level in terms of perceptual quality and good performances with respect to final quality and overall bit rate. Commercial tools show best performances only when applied to image having a limited number of colors and well defined underlying pattern (i.e. clip-arts, cartoon, etc.).

The different resolution size of relative sensor with respect to the viewfinder display in mobile devices is a relevant issue. Figure 8 shows SVG rendering (using SVGenie just for lack of space) of color images acquired by using an Evaluation Kit [23]. The image has been captured in CGA format (320x240 pixels) typically used in PDA devices. The SVG images have been directly rendered doubling the original resolution.

VIII. CONCLUSION

In this paper a comparative study about different raster to vector conversion techniques has been presented. Major details, on line demo and results can be found at the following web address: <http://svg.dmi.unict.it/>. Quality vs. size comparisons for each tool over the whole database will be considered to further validate our approaches.

Future researches will be devoted to study advanced region merging heuristics, together with the use of Bezier curves, region gradient filling and filter enhancement.

REFERENCES

- [1] B. Antoniou, L. Tsoulos, *Converting raster images to XML and SVG*. In Proceedings of SVG Open and Exhibition, 3rd Annual Conference on Scalable Vector Graphics, <http://www.svgopen.org/2004/>, Japan, September 2004
- [2] Autotrace, Convert bitmaps to vector graphics <http://autotrace.sourceforge.net/>, 2005
- [3] S. Battiato, G. Gallo, G. Messina, *SVG Rendering of Real Images using Data Dependent Triangulation*. In Proceedings of ACM/SCCG2004, Slovakia, 2004
- [4] S. Battiato, A. Costanzo, G. Di Blasi, G. Gallo, S. Nicotra, *SVG Rendering by Watershed Decomposition*. In Proceedings of SPIE Electronic Imaging 2005 - Internet Imaging VI - Vol.5670.3, USA, January 2005
- [5] S. Battiato, G. Barbera, G. Di Blasi, G. Gallo, G. Messina, *Advanced SVG Triangulation Polygonalization of Digital Images*. In Proceedings of SPIE Electronic Imaging 2005 - Internet Imaging VI - Vol. 5670.1, USA, January 2005
- [6] D. Duce, I. Herman, B. Hopgood, *Web 2D Graphics File Format*. Computer Graphics forum - Vol.21(1) pages 43-64, 2002
- [7] N. Dyn, D. Levin, S. Rippa, *Data Dependent Triangulation for Piecewise Linear Interpolation*. IMAJ. Numerical Analysis, Vol.10, pages 137-154, 1990
- [8] R.C. Gonzales, R.E.Woods, *Digital Image Processing – Second Edition*. Prentice Hall, 2002
- [9] Kodak's Photo CD system – Public Dataset: <ftp://www.cipr.rpi.edu/pub/image/still/KodakImages/> - 2005
- [10] KVec, Raster Vector Conversion: <http://www.kvec.de>, 2005
- [11] C.L. Lawson, *Software for C1 Surface Interpolation*. Mathematical Software III, J.R. Rice, Academic Press, pages 161-194, New York, 1977
- [12] S. Lee, *Wavelet-Based Multiresolution Surface Approximation from Height Fields*. Ph.D. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, February 2002
- [13] M. J. Nadenau, S. Winkler, D. Alleysson, M. Kunt, *Human vision models for perceptually optimized image processing - a review*, submitted to Proceeding of the IEEE, (<http://dewwww.epfl.ch/~nadenau/>), 2004;
- [14] Potrace, Transforming bitmaps into vector graphics <http://potrace.sourceforge.net/>, 2005
- [15] A. Quint, *Scalable Vector Graphics*. IEEE Multimedia - Vol.3, pages 99-101, 2003
- [16] Ras2Vec, Raster to vector conversion program <http://xmailserver.org/davide.html>, 2005
- [17] D. Su, P. Willis, *Demosaicing of Color Images Using Pixel Level Data-Dependent Triangulation*. In Proceedings of IEEE Theory and Practice of Computer Graphics, pages 16-23, 2003
- [18] Scalable Vector Graphics (SVG), XML Graphics for the Web <http://www.w3c.org/Graphics/SVG>, 2004
- [19] SVG Open Conference and Exhibition, Annual Conference on Scalable Vector Graphics, <http://www.svgopen.com>, 2005
- [20] VectorEye, Raster to Vector Converter, <http://www.siame.com/index.html>, 2005
- [21] L. Vincent, O. Soille, *Watersheds in Digital Spaces: an Efficient Algorithm based on Immersion Simulations*. IEEE Trans. on PAMI - Vol.13(6), pages 583-598, 1991
- [22] X. Yu, B.S. Morse, T.W. Sederberg, *Image Reconstruction Using Data Dependent Triangulation*. IEEE CG&A, Vol.21 (3), pages 62-68, 2001
- [23] Colour Sensor Evaluation Kit VV6501, STMicroelectronics, <http://ccd.sgp.st.com/stonline/books/ascii/docs/10923.htm>, 2004

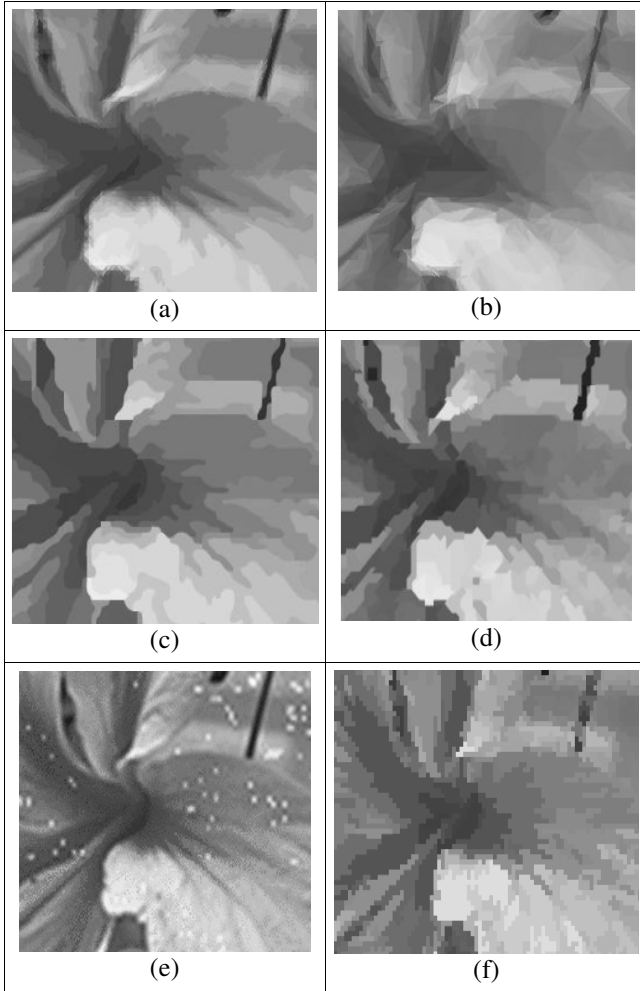


Figure 4 - A comparison between the proposed method and other methods (scale 4:1) (a) SVGenie, (b) SVGWave, (c) Vector Eye (d) SWaterG (e) Autotracer, (f) KVec.

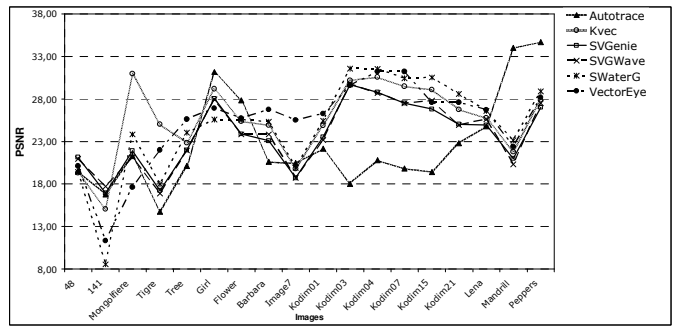


Figure 5 - PSNR comparisons of dataset images outputs.

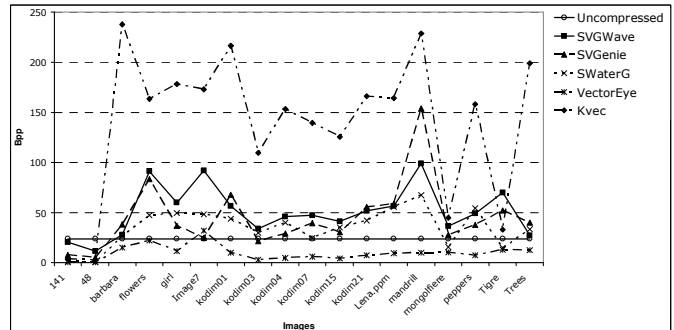


Figure 6 - Bit per pixels comparisons of dataset images outputs.

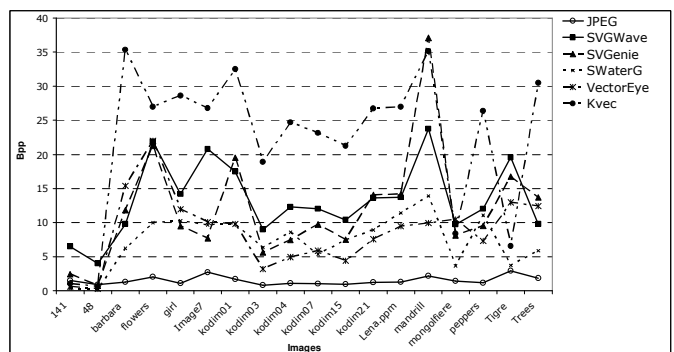


Figure 7: Bit per pixels comparisons of dataset images compressed outputs.

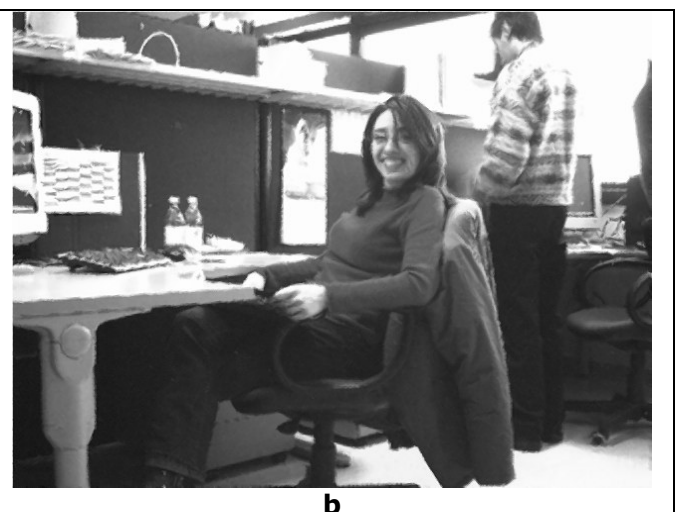
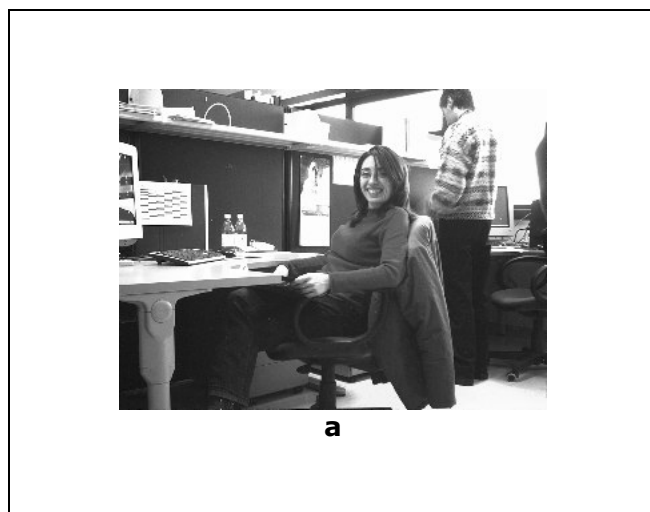


Figure 8. SVG rendering of an image acquired by STV6500 CMOS sensor. The original raster image is in CGA format (320x240). The image has been rendered in SVG by the proposed SVGenie technique doubling original resolution.