Enhances Color Image Segmentation

Rajesh Kumar[1], Malkit Singh[2]
[1]*Student, Golden College of Engineering & Technology Gurdaspur, Punjab*
[2]*Assistant Professor, Golden College of Engineering & Technology Gurdaspur, Punjab*

Abstract **- The key concern in Color Image Segmentation is more accurately and fast to do image segmentation by using algorithm. By using Algorithm, we highlight the color images in two parts that is background and Foreground. This paper focus on segmentation with flow of algorithm using n link nodes, T link nodes. This reduces the time and increase the perfection of segmentation. In which we use the grabcut Algorithm with some type of models.**

Index Terms **- Grabcut Algorithm, Ford-Fulkerson algorithm, Tree Algorithm, Graph Algorithm, Maximum Flow of Graph.**

I.INTRODUCTION

In Which we using Grabcut Algorithm with following are steps. A rectangle is drawn around the object a segment. The pixels outside the rectangle are mark as trimap background, and pixels inside the rectangle as trimap unknown.

The pixels that belong to trimap background are (initially) matte background, while that pixels with unknown trimap value will be assigns the value of matte foreground.

Two GMMs are created: one for the foreground and another for the background. The GMM foreground is initialize with matte foreground value pixels and the GMM background with the value pixels matte background.

Each pixel in the GMM foreground is assigned the component to which you are most likely to have and, each pixel of the GMM background is assigns the component to which it is most likely that belongs.

GMMs are eliminated, and new ones are created from the information previously obtained. Each pixel is assigned to its respective GMM (foreground or background) and the component that GMM that was assigned in step 4.

The graph is constructed, and the cut is achieved minimum running the maximum flow algorithm by Ford-Fulkerson [11]. Based on the result obtained, the matte value of some pixels. Those who stay connected to the source are as matte foreground, and the connected to the destination as matte background.

Steps 4 through 6 are repeated until no change the matte of any pixel. 8. Finally, it is possible to select certain edges and run a matting algorithm, which is an edge enhancement algorithm. This improves the edges of some parts of the image hard to detect as they are very fine edges.

II. RELATED WORK

A flow graph is one in which a node can send flow through an arc between the two nodes connected to it, where the flow passed through the arc it cannot exceed its capacity. Additionally, the amount of flow that a node can receive is equal to the amount of flow coming out of it, unless it is a source or destination node (only send or receive flow respectively). When the amount of flow sent.

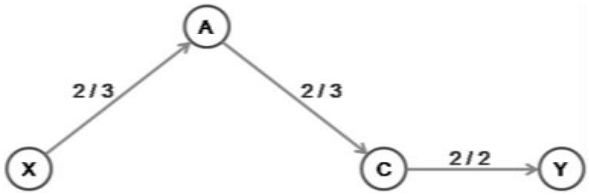


Figure 1: Sending flow from X to Y.

Sending a flow from X to Y means traversing arcs XA, AC and CY, with capacities 3, 3 and 2 respectively. Since the smallest capacity is 2, this value is subtracted from all the arcs that belong to flow, leaving the arcs as 1, 1 and 0 and leaving the arc CY saturated. A particular problem for flow graphs is the called maximum flow (max-flow) where there is only a source node, a target node, and a set of nodes and intermediate arches that connect them. The problem is get the max-flow between source and destination, where the amount of flow exiting the source node equals the amount of flow entering the destination. One of the best-known algorithms for calculating of the max-flow is the Ford-Fulkerson algorithm [11].

The algorithm starts with a flow of 0, and then applies an iterative process where flow increases, until no can apply more. At that time, it is considered that maximum flow was achieved. For the application of this method requires knowledge of two concepts additional: residual graph and augmenting path. A residual graph R is a graph with the same nodes than the original graph G and one or two arcs for each arc for an arc is equivalent to its weight then the arc is saturated. A flow path is a path from a node source to a destination node through a set of unsaturated arches. When flow is sent down a path, the arch with less weight is obtained, and with that value subtract the weight of all the arcs that run through the flow, leaving the arches with the least weight saturated. An example of the flow can be seen in Fig. 1. jo of a graph. Each bow has a pair of numbers <flow> / <capacity>, where the first represents the flow sent by the arc and the second the total capacity of the same. The source node is represented by X and the destination node Y. existing G. If there is an XY arc where flow is sent, and the flow amount is less than the capacity then there is a reverse YX arc with a capacity equal to the arc original minus the amount of flow sent. If the bow is saturated, then the YX arc capacity is equal to the capacity of XY. On the other hand, an augmenting path is a flow path from source to destination R. Figure 6 presents an example of the algorithm of max-flow. The flow graph in the figure shows a source node X and to a destination node Y through the XBCY arcs, sending a flow of 1 down a path of capacities 1, 5 and 2 for each arch respectively.

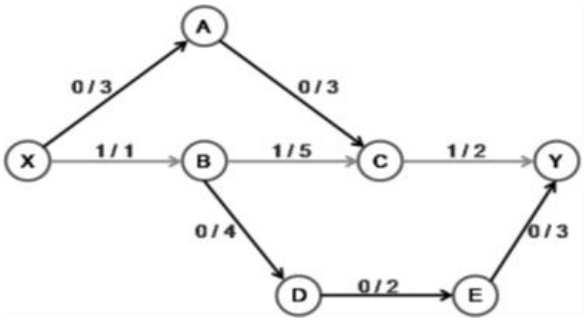


Figure 2: Example of the max-flow algorithm from X to Y.

In Fig. 3 the residual graph generated from the graph of Fig. 2, where the arc XB is saturated it is eliminated and the arch BX with equal weight to the original is created. Similarly, the residual graph contains the arc CB and the YC arc with capacity equal to the flow sent.

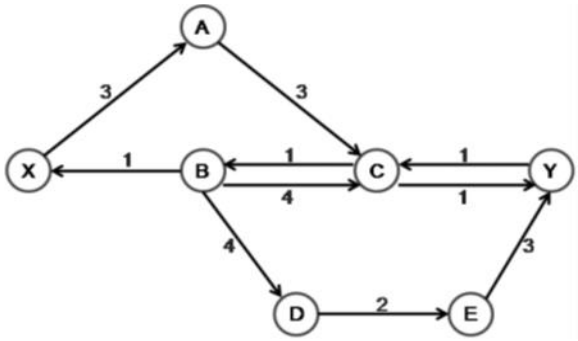
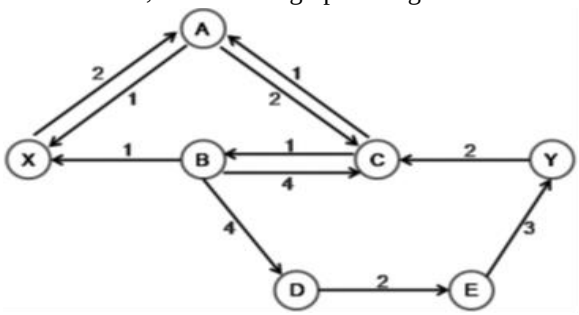


Figure 3: Residual graph of the one shown in Fig. 6.

The process described above is considered an iteration of the Ford-Fulkerson algorithm. In each iteration a new augmenting path is achieved until no path from X to Y is possible. In the graph of Fig. 3, you can still send a flow of 1 through the XACY path, saturating arc CY. Thus, the residual graph of Fig. 8 is obtained.



Continuing with the next iteration, it is possible increase the flow of the graph of Fig. 3 by 1 through the path XACBDEY, thus leaving the residual graph of Fig. 4.

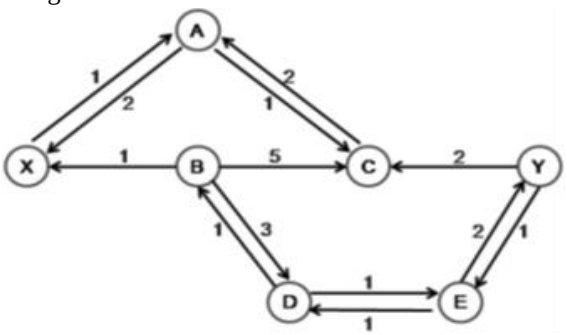


Figure 4: Graph when sending flow through XACBDE.

At this point, there is no augmenting path in the residual graph, so the algorithm has ended. The final graph is shown in Fig. 5 where the minimum cut of the graph is achieved by eliminating the saturated arcs XB and CY (red color), and the cost of the max-flow (cut minimum) is the sum of the weight of the saturated arcs (weight = 3). The generated disjoint graphs are: {X, A, C} and {B, D, E, Y}.

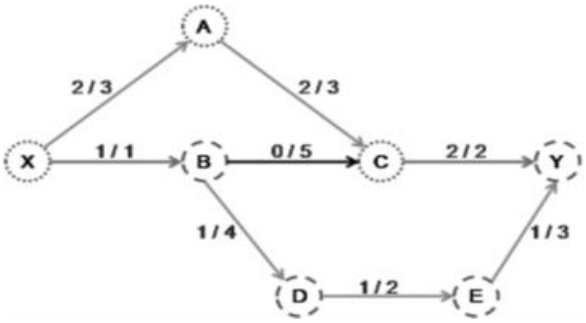


Figure 5: State of the final graph when applying the algorithm of Ford-Fulkerson.

The way in which the path of the source is achieved to the destination, the speed with which the

Figure 4: Residual graph after sending flow through XACY.

Algorithm. A good technique to get on the way is using the width search algorithm in graphs (Breadth First Search - BFS). With the algorithm of BFS, the shortest path is obtained (in number of arches traveled) from source to destination. In this work, a slight variant of the technique for calculating max-flow proposed by Boykov and Kolmogorov [9]. Below is the implemented algorithm.

Max-flow in GrabCut Usually, in each iteration of the algorithm of Ford-Fulkerson a new search is applied from the source to the destination which is very costly for the case of the pictures. As there is a node for each pixel of the image of size W x H, the resulting graph is of size W x H nodes. In this work, instead of perform a new search, two search trees are created search: one from the source and one from the destination. Both trees are created in the first iteration and are used during the execution of the algorithm. Fig. 6 shows the structures used in this algorithm. These are two search trees, S (color red) with root at the source and T- tree (blue color) with root in the destiny. It is important to note that trees do not they must have nodes in common.

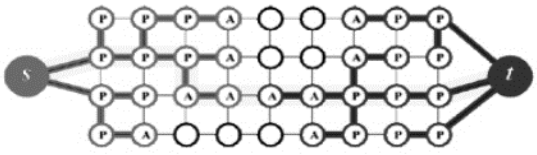


Figure 6: Search trees for the execution of the max-flow.

In the tree S all the arcs of a parent node towards its children are not saturated, while in the T tree all the arches of the nodes children of their father are not saturated. Nodes that do not belong to any of the trees are called free nodes. The nodes per Ownership of trees can be active or passive. Active nodes have arcs connected with nodes free or with nodes from the other tree. Passive nodes are those that all their neighbors already belong to same tree as him. Active nodes allow growth to the tree by acquiring new free nodes, and if they enter contact with a node in the other tree, then there is an augmenting path to increase the flow through it. Each iteration of the algorithm goes through 3 stages: create foundation, increased flow, and adoption. In the stage of growth, active nodes traverse their neighboring nodes us, as long as the arc connecting it to that node does not added as a free node or is connected to the other tree, this node goes from active to passive state. In the flow increase stage, flow is increased by the way achieved. With it, one or more bows will be saturated leaving some nodes disconnected from your tree; these nodes are called nodes orphans. Finally, in the adoption stage, the orphaned nodes and trying to get new parents for these nodes. If no parent is found to the node, assigns it as a free node. The algorithm ends when in the growth stage it is not possible to increase neither tree and the two trees meet separated by saturated nodes. For the main algorithm, a list will be kept active nodes A and a list of orphaned nodes H.

III. PROPOSED TECHNIQUE

SEGMENTATION WITH GRABCUT

GrabCut is a technique for image segmentation generates where little user intervention is required. Initially the user must select a box around the object of interest and then segmentation is performed automatically. Subsequently, the user can select certain areas of the image manually to improve the result obtained. The process explained BELOW in Fig. 7.

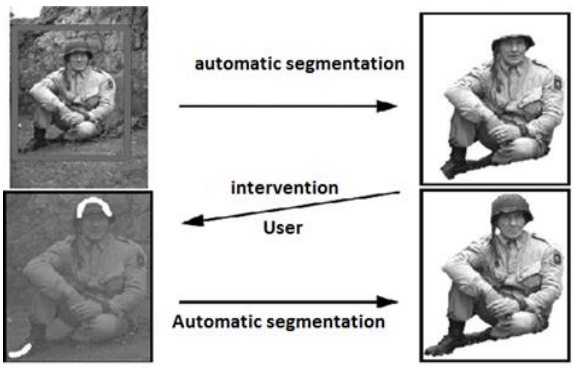


Figure 7: Steps in segmentation using the GrabCut algorithm.

The technique consists of creating a network flow graph [11] from the image to be segmented, where for each pixel a node is generated that represents it in the graph. Then, each node connects to its 8 close neighbors through of undirected arcs which are called N-Link.

Additionally, 2 special nodes are required in the flow graph: source and destination. The source node represents the object to segment in the image (foreground), and the distance represents the background of the image (background). Each one of the nodes of the graph is connected through an arc with the source and with the destination, these arches are called T-Link. The weight of the arches is calculated using a potential energy function based on mixed models Gaussians (Gaussian Mixture Models - GMM) [12], one for the foreground and another for the background. Each GMM is formed by 5 Gaussian components.

In Fig. 8 an example is shown where an image as a flow graph constructed for the GrabCut algorithm, where each pixel is connected with 4 neighbors out of 8 possible neighbors.

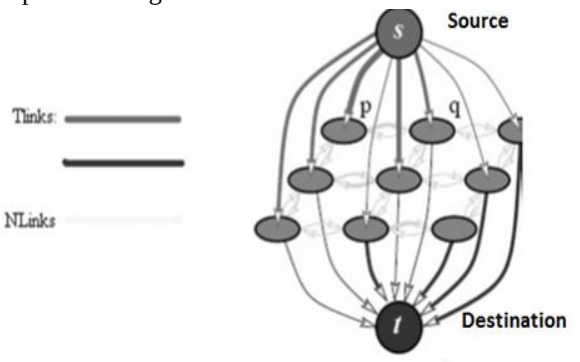


Figure 8: Flow graph constructed to run the GrabCut algorithm

Color image segmentation based on the GRABCUT algorithm engineering magazine from the flow graph it is possible to calculate the algorithm minimum cutoff rhythm [13], which divides the graph into two disjoint graphs as shown in Fig. 9. Thus, a graph will have the source node with a group of pixels, and the other to the destination node with the group of remaining pixels. So, the nodes connected to the source represent the foreground, and the nodes connected to the destination represent they felt the background.

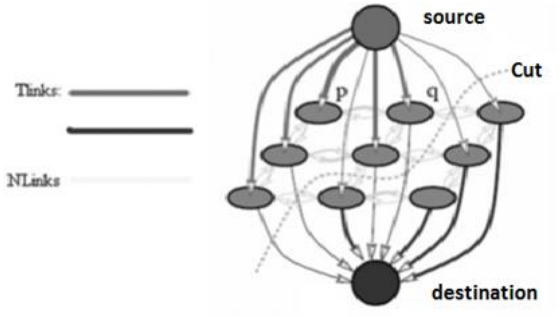


Figure 9: Section in a flow graph.

3.1 Algorithm

The algorithm initially creates an array, f = (f 1, f 2, ..., f k, ..., f N), where f k represents a pixel within the image in RGB color space and N represents the number of pixels.

Likewise, create an array A = (α 1, α 2, ..., α k, ..., α N) where each α k represents the matte value of each pixel f k. The matte indicates an intensity value that can be:

- Foreground: If the pixel belongs to the foreground object interest.
- Background: If the pixel belongs to the background of the image.

The matte values are modified during execution of the algorithm and are used to construct the result final. It is important to note that this value indicates whether the pixel belongs to the GMM component of the foreground or to the GMM component of the background.

Now, for each pixel the number of the Gaussian component to which it belongs. This value together with the value of matte allows to know which component a pixel belongs.

Finally, three timestamps are used to each pixel called trimap. These brands can be background, foreground or unknown (unknown pixel).

These marks are assigned by the user, and do not vary through in the execution of the algorithm unless explicitly modify.

All pixels marked as trimap foreground they will always be marked as matte foreground. Likewise, the pixels selected as trimap back-ground are always defined as matte background. Under this premise, the pixels that will modify its value of matte in the execution of the algorithm will be those who they were initially selected as trimap unknown.

Fig. 10 shows the flow of the execution of our proposal based on 8 steps, which are explained to continuation:

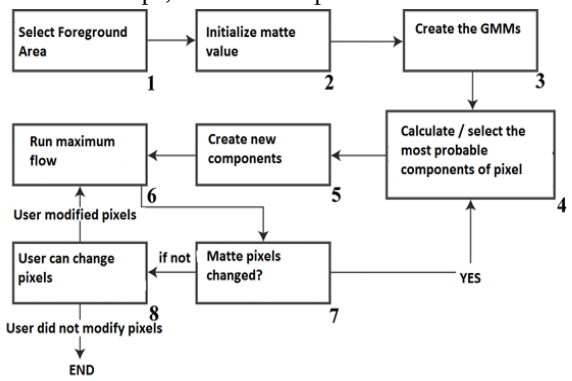


Figure 10: Flow of execution of the proposed algorithm

The following is the pseudo-formal algorithm, where the source node is named f, the destination node d, the tree rooted in source F and the node rooted in the destination D.

1: A ← f, d;
2: H ← empty;
3: F ← f;
4: D ← d;
5: while true do
6: repeat
7: C ← growth (…);
8: until C = some augmenting path;
9: if C ≠ augmenting path proper then
10: break;
11: end if
12: growth (…);
13: adoption (…);
14: end while Likewise,

the pseudo-formal algorithm of the growth stage, increased flow and adoption.

procedure GROWTH
1: while A ≠ empty do
2: Choose an active node p ∈ A;
3: for c / neighbor q of p such that capacity (p, q)> 0 c is saturated. If any of the active node's neighbors is a free node, so it is added to the tree as child of the active node. Now if any of the neighbors belongs to the other tree, then the stage of increased flow. If none of the neighboring nodes is
4: if (Parent (q) = L) then
5: Tree (q) ← Tree (p);
6: Father (q) ← p;
7: Add q to list A;
8: end if
9: if (Tree (q) ≠ L) and (Tree (q) ≠ Tree (p)) then
10: return C;
11: end if
12: end for
13: Remove p from list A;
14: end while fifteen: return C; // empty path

In the algorithm we observe the function Á TREE (Q) that indicates whether a node q belongs to tree F, tree D, or if It is a free node L. The P ADRE (Q) function indicates the parent of node q. The flow increase algorithm takes as input the path C found in the growth stage.

Procedure INCREASE _ FLOW
1: Find the amount of flow X to send;
2: Increase the flow in X through C;
3: for c / arc (p, q) that is saturated do
4: if (Tree (p) = F) and (Tree (q) = F)) then
5: Parent (q) ← empty;
6: Add q to the orphan list H;
7: end if
8: if (Tree (p) = D) and (Tree (q) = D)) then
9: Parent (p) ← empty;
10: Add p to the orphan list H;
11: end if
12: end for

Finally, in the adoption algorithm the CAPACITY function (P, Q), which represents the capacity of the arc that goes from p to q when P ADRE (P) = F, or returns the capacity of the arc from q to p when P ADRE (P) = D. For node p to be a valid parent of node q it must be fulfilled that CAPACITY (P, Q)> 0.

Procedure ADOPTION
1: while H ≠ empty do
2: Pick an orphan node p from H;
3: Remove p;
4: for c / neighbor q of p such that Tree (p) = Tree (q) do
5: if (Capacity (q, p)> 0) then
6: if Path from the root of the tree to q no has saturated node then
7: Parent (p) ← q;
8: end if
9: end if
10: end for
11: if there is no new parent for p then
12: for c / neighbor q de p do
13: if (Capacity (q, p)> 0) then
14: Add qa A;
15: end if

16: if (Parent (q) = p) then
17: Add q to H;
18: end if
19: end for
20: Tree (p) ← L;
21: Remove p from A;
22: end if
23: end while

Once the minimum cut of the graph has been found, proceed to update the matte value of the pixels (step # 7 shown in Fig. 10) and if there were any change returns to step # 4 of the execution flow of Fig. 10, otherwise the algorithm ends.

IV. RESULTS AND DISCUSSION

For the experimental tests of our work, the following applications and tools were used listed below:

Enhanced GrabCut (GM): The application developed called on this job. Implementation presented in his work An Interactive Foreground Extraction Tool [20]. Magic Wand (MW): "magic wand" tool Photoshop [6], with a tolerance level equal to 32. Smart Scissors (TI): Magnetic Tool Photoshop Lasso [6], which uses the technique of Smart Scissors. The images used in the tests can be see Table 1 and Fig. 11. All images They are in the JPEG image format.

Image	Name	Resolution in pixels
1	Plane	481 x 321
2	Downy	600 x 800
3	Lavendra-1	800 x 600
4	Lavendra-2	1024 x 768
5	Lavendra-3	3264 x 2448

Table 1: Description of the images used in the tests

(a) Plane (b) Lavendra (c) Downy

Figure 11: Images used in the tests

4.1 Time measurements the different techniques (i.e., GM, GP, MW and TI) were performed on a conventional PC with the following Features: 3.00 GHz Pentium 4, 512 Mb RAM, under Windows XP. For time measurements, ran each of the applications shown 30 times you give. For the case of MW and TI, the time is measured from the moment the segmentation tool starts, until you get the result. Table 2 shows the time average obtained. Because the time to perform manual operations depends on the expertise of the user river, 3 different users were used with knowledge of tools.

Technique	I1	I2	I3	I4	I5
GM	12 s	22 s.	34 s.	50 s.	285 s.
GP	10 s	80 s	163 s	180s	--
MW	5 s.	7 s.	30 s.	40 s.	50 s.
TI	53 s.	25 s.	43 s.	55 s.	130 s.

Table 2: Test execution times made.

The GP algorithm could not be applied to segment image 5. It was executed using other images of similar size, but the same problem. Possibly the application does not support images greater than 5 Megapixels. On the other hand, the MW tool failed to finish in 35% of the cases for images 8, 9 and 10. Therefore, it was considered only the occasions where it was possible to execute. It can be seen that the presented algorithm in this work it performs the segmentation in a time acceptable. Due to the number of operations perform, no real-time results are obtained for the execution platform used. On PCs with higher computing capacity, this value will decrease. Now well, the visual result obtained with the segmentation GrabCut-based proposal is good and detailed below continuation.

4.2 Visual results for each of the four test applications, segmentation was performed on all images shown in Table 1. The visual results of the Image 1 can be seen in Fig. 12, where shows only a portion of the segmentation with the objective of analyzing the results. It can be seen than for GM and GP techniques, the edge pixels present aliasing compared to the results of MW and IT techniques. The reason for this is the smoothing done by Photoshop software after applying your tools.

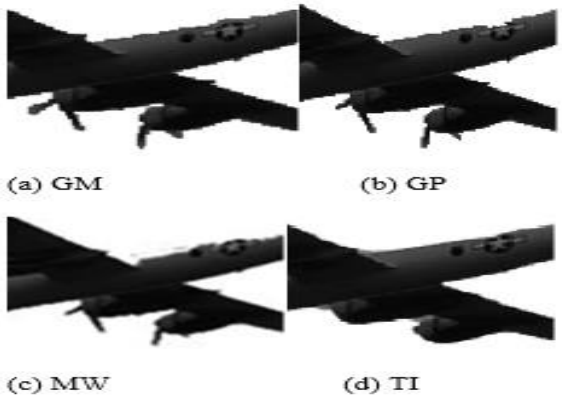


Figure 12: Visual results of the Airplane image

Segmentation with the GM technique is carried out in an iteration and no user intervention for the image 1. With the GP technique, the intervention of the user to assign some pixels as foreground. The visual result is very similar; however, you can observe in Fig. 12b how certain parts were removed of the original image (propeller and logo) which does not happen in Fig. 12a. Fig. 12c shows the result of applying the technique MW and Fig. 12d from TI technique. Note that with the MW technique, a part of the image is eliminated which it is undesirable for segmentation. In the case of TI, a lot of user intervention was required (adding multiple control points manually). The segment the obtained image lacks certain areas of the image like propellers and add certain parts belonging to the background. In the segmentation of image 2, good visual results with all applications, segments correctly filling the colored plastic bottle blue off white unicolor background. So, the result is an image in RGBA format where the channels RGB represent the foreground pixels and channel A represents the background. The scheme of storing the background on the alpha channel, it is applied to all treated images.

The results obtained with images 8, 9 and 10 are very similar, so it was considered as an algorithm of the different applications. This process was carried out using the tools of the operating system only case to study. The difference lies in the time execution time shown in Table 2.

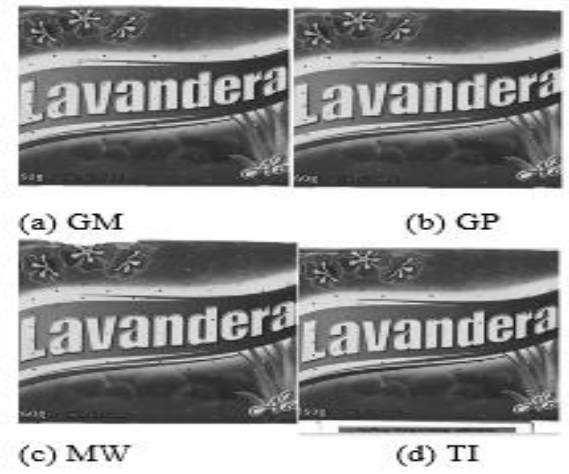


Figure 13: Visual results of the Wagtail image.

Again, for Fig. 13 only a section of the image for comparison. In Fig. 13a and Fig. 13b, you can see results very similar. The foreground/background separation is satisfactory in both cases, using only one cycle of the algorithm. However, the results obtained in Fig. 13c and 13d were not suitable. Furthermore, both techniques required a lot of user intervention when moment of segmentation. It should be noted that part of the object that is initially selected as foreground, it is part of the background (i.e., box white with brown). Another point of comparison between the different applications cautions and their application of segmentation on test images, is the amount of memory occupied during the process detailed below.

4.3 Memory consumption for this test, the amount of memory was obtained consumed by the application once loaded and executed native Windows XP. For this test, we considered only images 3, 4 and 5 because they are the largest resolution compared to images 7 and 8. The result is shown in Table 3, where it is observed in megabytes occupied.

Technique	I3	I4	I5
GM	75 Mb.	112 Mb.	970 Mb.
GP	50 Mb	97 Mb.	--
MW	20 Mb.	93 Mb.	100 Mb
TI	28 Mb	90 Mb	112 Mb

Table 3: Amount of memory occupied by the different techniques

Table 3 shows the memory increment with engulfed by our algorithm as the image is larger. This is due to the amount of information that is stored for each pixel of the image, as a node in the graph. Regarding the structures data, double precision reals (64-bit) are used for the N-Link and T-Link values at each node. At the same time, the GMMs obtained at each node are stored to be used in a next iteration (if required). At the end of algorithm execution, used memory is completely freed.

V. CONCLUSION

This paper presents a modification of the GrabCut technique for image segmentation in order to get better results. The best can be seen in the modification of the calculation of the N-Links and T-Links, which is based on the calculation of the GMMs for the foreground and background, with 5 components each. Our algorithm provides many advantages compared to the original version presented by Rother et al. [3]. First, let's get a better association of the value of the N-Links in the graph due to the function used. Similarly, the calculation of the T-Links with 5 Gaussian components is on proper setting for good grouping of

the pixels. Finally, the minimum cut algorithm was efficiently implemented in the language of chosen schedule. Those in this work are similar. However, the implementation required greater interaction by part of the user to achieve the desired results. On the other hand, the algorithms of Magic Wand and Scissors Smart requires clear, high-contrast images color difference between the background and the object of interest. Our algorithm has a couple of disadvantages with respect to other implementations and techniques: computation time and memory consumption. In some applications the computation time is acceptable, but it is greater than the other applications. Consumption of memory increases considerably when working with images greater than 8 Megapixels (approx. 1 Gb). In the future, it is proposed to use data structures that reduce memory consumption. For example, a Quadtree structure will allow you to create a node in the graph containing multiple pixels. As mentioned above, our algorithm considers each pixel as a node in the graph. On the other hand, the implementation could be carried out under a parallel environment of high performance as the Graphic Processing Unit (GPU) using an architecture like CUDA, Open CL or Direct Compute.

ACKNOWLEDGMENT

The above contents and survey we mentioned is true to my knowledge.

REFERENCES

[1] Mortensen, EN and Barrett, WA “Intelligent scissors for image composition ". In Proceedings of the22nd annual conference on Computer Graphics and Interactive Techniques SIGGRAPH '95. New York, NY, USA, ACM Press, 1995, pp. 191-198.

[2] Boykov, Y. and Jolly, MP. "Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in ND images”. In Proceedings of the Eight IEEE International Conference on Computer Vision ICCV. Vancouver, BC, Canada, IEEE Computer Society, 2001, pp. 105-112.

[3] Rother, C.; Kolmogorov, V. and Blake, A. “GrabCut: Interactive Foreground Extraction using Iterated Graph Cuts”. ACM Transactions on Graphics. New York, NY, USA. Vol. 23, 3, pp. 309-314, August 2004.

[4] Santle C., K. and Govindan, VK “A Review on Graph Based Segmentation”. International Journal of Image, Graphics and Signal Processing. ECS Publisher, Vol. 4, 5, pp. 1-13, June 2012.

[5] Pajares, G. and De La Cruz, J. Computer vision: Digital images and applications. Madrid Spain, Ra-Ma, 2001, pp. 179-198.

[6] Photoshop CS5. Adobe Systems Incorporated. Accessed June 22, 2012, available at http: // www.photoshop.com

[7] Mortensen, EN and Barrett, WA “Toboggan-based intelligent scissors with a four-parameter edge mo- of the". In Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR'99). ACM Press, 1999, pp. 452-458.

[8] Kass, M.; Witkin, A. and Terzopoulos, D. “Snakes: Active contour models”. International Journal of Computer Vision. New York, NY, USA. Vol. 23, 3, pp. 309-314, August 2004.

[9] Boykov, Y. and Kolmogorov, V. “An Experimental Comparison of Min-Cut / Max-Flow Algorithms for Energy Minimization in Vision”. IEEE Transactions on Pattern Analysis and Machine Intelligence. Washington, DC, USA. Vol. 26, 9, pp. 1124-1137, September 2004.

[10] Kolmogorov, V. and Zahib, R. “What Energy Functions Can Be Minimized via Graph Cuts? " IEEE Transactions on Pattern Analysis and Machine Intelligence. Washington, DC, USA. Vol. 26, 2, pp. 147-159, February 2004.

[11] Cormen, TH; Leiserson, CE; Rivest, RL and Stein,C. Introduction to Algorithms . New York, USA, MIT Press, 2001, 2nd edition, pp. 651-664.

[12] Chuang, YY.; Curless, B.; Salesin, DH and Szeliski, R. "A Bayesian Approach to Digital Matting." In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR. IEEE Computer Society, 2001, pp. 264-271.

[13] Sedgewick, R and Wayne, K. Algorithms. Boston, USA, Addison Wesley, 2011, pp. 570-587.

[14] Implementing GrabCut. Talbot, J. and Xu, X. Brigham Young University. Accessed June 22, 2012, available at http://research.justintalbot.org /papers / Grabcut.pdf

[15] Ross, SM Introductory Statistics. Canada, Academic Press, 3rd edition, 2010, pp. 290-293.
[16] Timm, NH Applied Multivariate Analysis. USES, Springer, 1st edition, 2002, pp. 79-90.
[17] McLachlan, G. and Peel, D. Finite Mixture Models. Canada, Wiley-Interscience, 2002, pp. 24-33.