Computing and Informatics, Vol. 37, 2018, 213–228, doi: 10.4149/cai_2018_1_213

ELLIPSE DETECTION IN FORENSIC BLOOD STAIN IMAGES ANALYSIS

Tomasz Wójtowicz, Dariusz Bułka

CYBID Ltd.
Kuźnicy Kołłatajowskiej 15c/L2
31-234 Kraków, Poland
e-mail: tomasz.wojtowicz@ii.uj.edu.pl

Abstract. This paper presents an algorithm for ellipse detection on stains of blood, which is directly suited for the needs of forensic analysis. The algorithm is of the edge-analyzing type. It performs convexity detection and is able to split contours of overlapping ellipses by concave regions analysis. A filtering is applied to the fit ellipses to reduce the results only to those results necessary for blood drop trajectory tracing. Results for running time and fitting quality tests performed on real-life and artificial data are presented. The solution answers to both quality and running time expectations of the field of application.

Keywords: Ellipse detection, ellipse fitting, edge analysis, contour segmentation, blood stain, forensic analysis

1 INTRODUCTION

The modern forensic analysis employs computer to recreate the site of an accident or a criminal action with the use of dedicated modeling software, basically 3D, currently also 4D (with time). The on-site measurements and taken pictures allow to create a polygon mesh and orthophotomap textures for this mesh. One particular feature found in pictures that meets with high attention are the blood stains. It is possible to recreate the trajectory of a blood drop that created a given stain, producing a valuable insight into the analysed event, if the geometric characteristics of such stain are obtained. The trajectory tracing is a rather simple Newtonian/Bernoulli mechanics, the really difficult part is to produce a proper geometric description of a stain. This process is currently done manually by encircling a blood stain image

with a vector ellipse on screen. It is an exhausting and error-prone proces with scenes that sometimes may contain tens or hundreds of blood stains. For this reason, automation of this process is desired. This paper presents a working automatic blood stain detection algorithm, which uses ellipse detection, and is implemented with the aim to augment the scene modelling software.

There are substantial differences between the scientific interest in ellipse detection and the requirements of this particular field of application. The scientific papers of the topic usually boast the fitness of their algorithms in the terms of a number of detected ellipses and how complicated the input may be. On the other hand, the aspect of running time is often skipped in those papers. For the forensic analysis it is not necessary to detect every single ellipse there may happen. For instance it would be more productive to produce 20 representative ellipses rather than flood the user with 200 of them. By representative we mean such ones which will not be found as a questionable evidence in the court – only highly regular stains really matter, so the algorithm should not bother with disfigured stains. Therefore, we have introduced a filtering mechanism to reduce the output to those ellipses which would be the most relevant.

However, the running time becomes an important matter. The presented solution is supposed to work in an interactive mode, aside or within the modelling software. On the average computer found in the forensic laboratory the detection time should take some 10 seconds. Be it 10 minutes, the solution is useless, as the technician would do the job manually in that time. This restriction practically eliminates more complicated approaches that are known for having a longer running time.

Figure 1 presents a fragment of a taken picture of a real blood stain. This is the input on which the algorithm is typically working and such is the average quality. As seen in the picture, the edge of a stain is blurred, what in binarisation produces a certain level of jagginess. The stain has a tail and occasional extruding or intruding artifacts, all of which must be removed before the ellipse fitting takes place. Moreover, occasionally two or more separate splatters intersect, resulting in a cojoined stain just like the one in this picture. Cojoined stains are considered a lesser quality evidence, nevertheless the algorithm must correctly detect them just to know that status. The algorithm presented in this paper succesfully copes with the problems mentioned above.

1.1 State of the Art

Ellipse detection is an important topic in computer image understanding with numerous different applications, many of which in natural sciences, medicine and related fields. There are several different approaches to ellipse detection, employing various techniques, what is reflected in running times and output qualities of particular solutions. One such approach, probably the eldest, are voting based algorithms, such as Hough transform and its variants [3, 9]. Another approach is based on genetic algorithms, such as [10] which executes the detection as a multi-objective

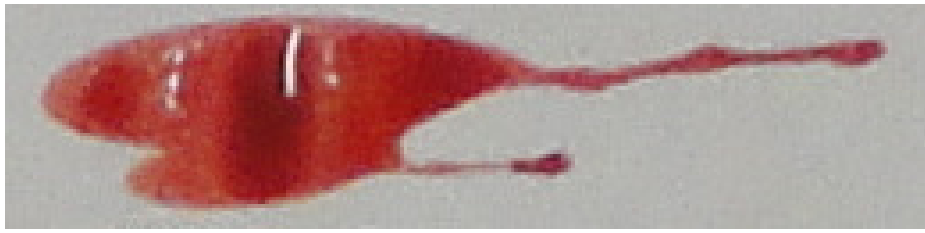

Figure 1. A stain of real blood consisting of two joined splatters, both with comet tails. Also visible camera flash reflection

optimization. Third approach is edge analysis [2, 5, 6, 7, 8], to which the algorithm presented herein belongs.

The edge-analysing designs typically implement following steps: region of interest (ROI) extraction, arc detection and joining, ellipse fitting, result improvement/filtering. For instance, the paper [5] proposes such algorithm, which extracts the ROI with the use of gradient. Then with the use of Gaussian Mixture Model the arcs are detected, then the Bayesian Ying-Yang Harmony learning algorithm supervises arc joining, ellipse fitting and result improvement. Unfortunately, the paper skips a running time discussion. It appears upon the description that it is at least $O(n \log n)$, with a large linear coefficient too. Another paper [1] proposes an ellipse fitting algorithm with great capability of detecting intersecting ellipses. The paper however skips the ROI extraction aspect, presenting only the edge analysis topic. Aside the common elements, this algorithm performs its specific Ellipse Refinement operations, which repeat certain steps until the best result is found. While offering a very high quality of the result, the running time of this algorithm is not mention but again appears too high.

The algorithm presented in this paper is similar to the algorithms such as [1, 5], but we managed to create a solution which performs the above mentioned steps in much simpler way, which results in a smaller period of running time, practically around $\Theta(n)$ with just one exception (the surface filter, Sections 4, item 4), and still maintaining the quality on the acceptable level.

2 BINARISATION

Binarisation is the process of highlighting the regions of interest, which in case of this edge-analysing algorithm would be the outlines of blood stains. There are two well known approaches: a gradient-based and a direct classification. The gradient-based method is very common in edge detection. It exploits the fact that an edge is a boundary between different colors or shades, therefore difference in pixel values will be the greatest at the edge. We have tested this method for blood stains and found it working, yet not satisfactorily. The differences among the obtained gradient values resulted in incomplete, torn outline, which would complicate the further processing.

Some of the effort of the algorithms [1, 5] is to combat the problem of gradient-based binarisation.

Because the blood stains are all in a narrow and very well specified range of color, we have discovered that direct classification works very well in this application. The direct classification method works by inspecting each pixel whether it depicts blood. In order to execute this method, the input image is converted into HSV (hue, saturation, value) color space, in which the possible colors of blood can be represented as one continous subset of each component. The decision process for each pixel is simply to check whether its HSV components are within the expected ranges.

Obviously there will be situations where the algorithm should look for stains of color outside the predefined normal blood range. For instance, with time, a blood stain may fade even to invisibility. The forensic technicians reveal such ones with luminol, after which the blood stain will be of blue color. Also liquids other than blood may be of interest. To answer these needs, our application provides a "color picker" tool, with which a user should point at least two possibly different pixels within the ROI in order to establish custom ranges for classification (an appropriate margin will be added automatically). This color picker can also be used to narrow down the ranges for normal blood stains. Note that whether the liquid would be arterial blood, venous blood, luminol treated, or perhaps coffee, all the stains in one picture would be of the same color, therefore color picking of one representative stain should result in all of them being properly detected.

The result of this process are white stains on the black background. The outline of a stain may now be easily extracted with any well known algorithm. Note that in this approach the result is always a closed loop outline without any forementioned problems related to the gradient-based method. The extracted outline is stored not as a 2D image, but as a one dimensional list of coordinates which preserves the neighborhood of edge pixels.

3 OUTLINE ANALYSIS

The purpose of this stage is to obtain out of the outline a vector ellipse described by five parameters (x, y, a, b, ϕ) where x, y are the center point coordinates, a, b are the axes lenghts and ϕ is the direction angle. The ellipse fitting algoritm will produce a worthy result only if the fitting is perfomed on pixels which belong to an elliptic curve. Figure 3 presents an outline obtained from the blood stain image in Figure 1 and it is full of unwanted features. The processing discussed in this section filters the outline to obtain only elliptic arcs and arranges them in sets representing separate ellipses if so necessary.

3.1 Convexity and Concavity Detection

In this step it is decided whether a given pixel belongs to the convex or concave fragment of the outline. First in order to do so, each pixel is described by the parameter

value of turn which marks how much the curve turns in the proximity of this pixel and in which direction. The value of turn for a given pixel B is calculated based on two auxiliary pixels A and C which are *step* away in the pixel neighbourhood sense. The *step* should be high enough to ignore noise and jagginess occuring due to binarsation of blurred edge, yet it must be low enough not to overlook the concavity we are trying to find. The best value is just above the point where the average noise no longer causes false detection. We have tried different functions calculating the *step* value from the pixel count in the outline, among which constant and linear functions, to establish that square-root-like curve works the best. On our test set, the *step* = 30 works optimally for outlines larger that 200 pixels. For the smaller ones it needs to be firmly reduced, but for the greater ones it does not need to grow equally fast, hence the use of square root curve.

The formula (1) defines the *value of turn* (B_α) as a measurement of how much $\vec{BC}$ bends from $\vec{AB}$ (Figure 2). It is 0.0 when ABC are colinear, and rises proportionally. After the (1) calculation, a plus or minus sign is assigned to specify whether it turns left or right. For this reason, the entire outline must be traversed in one direction, because ABC produces the same value but opposite sign than CBA.

$$B_\alpha = acos\left(\frac{\vec{AB}\cdot\vec{BC}}{|AB||BC|}\right). \tag{1}$$

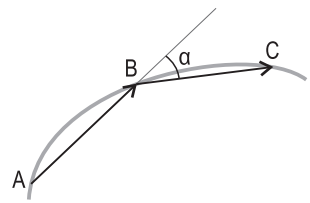

Figure 2. The method of measuring the value of turn

3.2 Outline Segmentation

In a closed convex outline every pixel's *value of turn* is of the same sign, provided the outline was traversed in one direction. Consequently, any concave feature will manifest itself as the opposite sign. In order to determine which sign represents the convexity, one must only check which sign is more frequent. In this step the convex segments are being identified on that basis (Figure 3). It is important to note, that we allow to be classified as convex the pixels whose value of turn is slightly of the opposite sign, like 0.001 radian. This small overlap neutralizes the effect of random classification of straight segments.

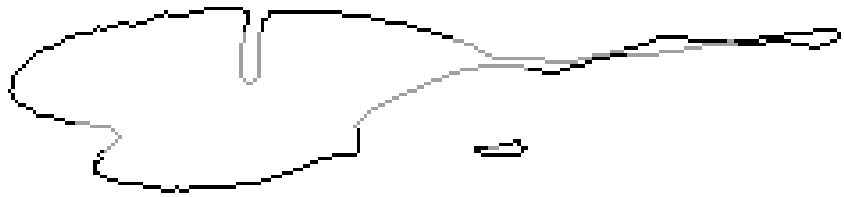

Figure 3. The outline of stain from Figure 1 with convex parts (black) identified

3.3 Segment Joining

Each segment obtained in the previous step is a convex fragment of the investigated stain's edge. If fed into the fitting algorithm right now, it would result in a separate ellipse fit for each such segment (Figure 4), which is a wrong result. The segments must now be organized into sets such that each one contains fragments of only one ellipse.

Figure 4. Useless fitting of unjoined segments

Only the neighbouring segments may be joined by their appropriate ends, thus if there are N segments out of the outline, there are exactly N joining decisions to be made. The general idea of the decision making process devised by the authors is depicted in the Figures 5 and 6. The thick gray lines are the convex segments, the thinner one is the earlier detected concave impurity. If the tangent versors are facing the same direction nearby the corresponding ends of the segments, we assume that the segments are parts of the same curve and therefore should be joined (Figure 5). Otherwise, if the versors are facing essentially different direction, like around V-shaped junction of two splatters, then we assume these are separate curves (Figure 6). The Euclidean distance limit also must be taken into consideration. Note Figure 6: the versors far enough from the V-shaped junction are facing the same direction. Without the distance limit this would be mistaken as an instance of Figure 5 case.

In our implementation, the tangent versor in each pixel is represented as a single value named *azimuth*. It is an angle between the tangent versor and one arbitrarily

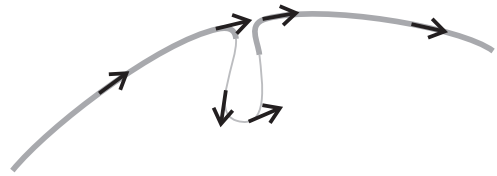

Figure 5. Tangent versors around an impurity in the stain's edge

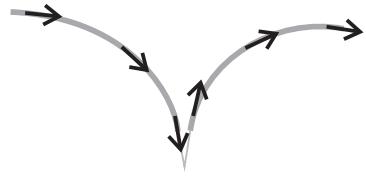

Figure 6. Tangent versors around a two splatters' junction point

chosen straight line. We are using the already obtained *value of turn* information to calculate the *azimuth*. If for the i^{th} pixel in the whole outline we denote i_α for the value of turn and i_β for the azimuth, then we calculate it as:

$$i_\beta = \frac{1}{step} \sum_{0}^{i-1} i_\alpha. \tag{2}$$

This way, the tangent versor of the 0^{th} pixel becomes the arbitrally chosen straight line. This method turns out to be very accurate – the azimuth value calculated again in the 0^{th} pixel after passing the entire outline misses 2π by less than 10^{-8} radians, up to 10^{-13} in the best noted cases.

In the decision process of joining for given two neighouring segments, each pair of pixels, one taken from the first segment, the other from the second, is inspected. This is the only $O(n^2)$ operation in this part of the algorithm, however, with the Euclidean distance limit mentioned earlier, which should be set between 0.25 and 0.5 of *step*, only very limited fraction of pixels is inspected on the corresponding ends of the segments. Considering that the number of segments is also very limited, the resulting running time impact is negligible.

For each inspected pair of pixels the absolute difference of *azimuths* is considered. The segments will be joined if the difference is below the limit which differentiates between cases of Figure 5 and Figure 6. Additionally, to help with more accurate fitting, the pixels from that pair, which has the smallest difference in azimuth, become the new endings of their respective segments.

Segments getting joined means that they are placed in one set as still separate segments. It does not mean that the gap between their respective ends is filled with artificial pixels. The ellipse fitting algorithm does not require that the input constitutes one connected component, and also there might be certain legal objections

on processing the evidence by the use of assumptions. The proposed system, as it is now, operates only on the real data.

3.4 Ellipse Fitting

The sets of segments are now fed into the ellipse fitting algorithm. The results are vector ellipses. Figure 7 shows the result in regard to the input segments, whereas Figure 8 shows the result superimposed on the input picture. More about the ellipse fitting algorithm is presented in Appendix A.

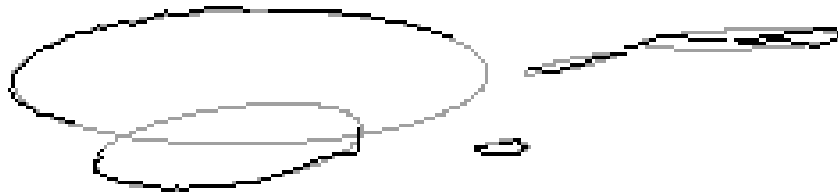

Figure 7. Remaining fragments of outline (black) being the input to the fitting algorithm and vector ellipses out of them (gray). This result is correct.

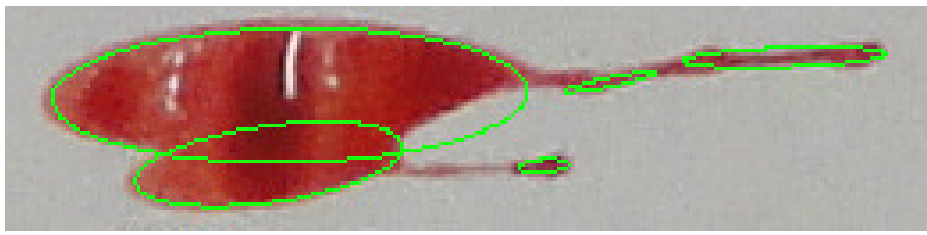

Figure 8. Vector ellipses from Figure 7 imposed on the original picture before filtering. Note the largest ellipse exceeds the back part of the splatter. Just like the "comet tail", this back part is deformed by viscosity phenomenons. What really matters is how well the vectorized ellipse follows the front part of the splatter and in the shown case it is a successfull result.

4 RESULT FILTERING

The results from previous stage are vector ellipses produced from convex sets of pixels. However, not all of them represent a worthy result. One problem, as visible in Figure 8, is that additional fittings happened on convex parts of the tails. Another one are incorrect fittings whenever the input segment was too short to produce a fitting correctly covering the edge of a stain. We employ a system of four consecutive filters:

1. The thickness filter removes all results which smaller axis is below 6 pixels. Aside the tiny ellipses which definitely are not a worthy results, occasionally there happen to be results of the fitting algorithm's instability on near-straight segments: grotesque ellipses of 0-pixel width and height exceeding the picture's dimensions.
2. The bounding-box filter compares the bounding boxes of an ellipse and the outline from which it was produced. If the ellipse protrudes over a specified margin, it is deleted, because ideally the ellipse should be exactly over the outline.
3. The area filter calculates from the bounding box the area covered by the ellipse. For all ellipses out of a single outline, removed are the ones whose area is less than 25 % of the largest one of them.
4. The surface filter inspects the surface encircled by an ellipse on the binarized image and calculates the percentage of the "white" area, effectively answering to what extent the ellipse encircles a bloody surface. If it is below 90 %, the ellipse is removed.

The filters 1 and 3 deal with the ellipses in the tail. The filters 2 and 4 deal with the fittings which do not encircle a stain correctly. In case of the filter 4 also correct fittings on damaged stains are removed, because for a court evidence only these matter which are being an unquestionable evidence and it has been assumed that 90 % of undamaged surface be the threshold.

As for running time, the filters 1, 2 and 3 require only one conditional check each, also bounding box and area calculation is not exhausting, therefore their time impact is negligible. On the other hand, the filter 4 needs to inspect every single pixel in the encircled area, therefore its running time is like $O(n^2)$. In the aim of speeding it up we had been testing solutions where only a fraction of all pixels is visited, chosen by either pattern or Monte Carlo method. However, as the running time tests have shown (Section 5.2), the solution meets the assumed time requirements with the filter visiting all the pixels. Note that, if there were no filter 2, the filter 4 would very well cover its functionality, therefore the filter 2 works as a preprocessor which speeds up the filter 4.

5 PERFORMANCE TESTING

The presented algorithm has been tested in two separate experiments, one using real, the other artificial images. The first one used 45 pictures of real blood stains provided by a forensic laboratory of the police. The purpose of this experiment is to test the quality of produced results on this kind of pictures for which it has been designed. Because of the limitations of this experiment, which are mentioned below, another experiment with artificial images has been performed to capture the algorithm performance in more quantitative terms.

5.1 Real Images

The basic problem with images depicting real blood stains is that there is no *a priori* information of what stains are there on the picture and where, therefore there is no straighforward way to tell which are the correct results. The algorithm has been designed to replace a human job, therefore we worked out the problem by performing manual encirclings, just like forensic technicians do, and then compared the results with the output of the algorithm. Whenever a detected ellipse matches the manual one within visually acceptable tollerance, we record a success event (S). If a detected ellipse intersects a manual one, yet its properties exceed the tollerance, then we record a mistake (M). If there is a manual ellipse and no detected ellipse, we call it underfit (U) and the opposite case we call overfit (O). By treating manual ellipses as one population and the detected ones as the other, we are able to calculate Dice coefficient out of the aforementioned sets (# denotes set's cardinality):

$$Dice = \frac{2 \cdot \#S}{\#(S \cup M \cup U) + \#(S \cup M \cup O)}. \tag{3}$$

One important matter discovered in this experiment is that not all provided pictures were worthy as an input. Good pictures are these which depict the stains in a proper resolution – stain's diameter of 50 pixels would be a desirable value, with no upper limit. In the bad pictures the blood stains are tiny, usually around the established lower threshold of 6 pixel diameter. With such low resolution the signal-to-noise ratio is low and so is the numerical stability, in effect giving poor results. Apparently the bad pictures have been taken only to provide a photographic overview of the scene, whereas the good one were taken with the blood tracing in mind. For the bad ones even manual encircling is not a sound option as there are uncertainties in human understanding of the picture.

On the good pictures, 154 ellipses were detected manually, the algorithm detected 222, getting Dice coefficient of 69.4 %. Expressing the classes as a fraction of the manual result we get: success $S = 83.9\,\%$, mistakes $M = 5.8\,\%$, underfit $U = 10.3\,\%$ (sums up to 100 % of manual fittings), and overfit $O = 51.83\,\%$. The best case for one file was: Dice 78 %, $S = 95.5\%$, $M = 3.0\,\%$, $U = 1.5\,\%$ and $O = 46.3\,\%$. The numbers of success, mistakes and underfit show that the algorithm has a desirable tendency to fit correctly or does not fit at all. On the other hand the relatively high number of overfits show that it also finds irrelevant objects. However, for a human operator it is much easier job to select and delete such overfits rather than perform the encirclings. An auxiliary experiment has shown that raising the size threshold from 6 to 12 pixel can reduce the number of overfits by the factor of 3, however also affecting other classes by a lesser degree.

In the case of bad pictures, there results are even incomparable between the files, as there are controversies in human interpretation of the picture. In general, the results are worse, as for instance in one file it was Dice 32.8 %, $S = 40.7\,\%$, mistakes not counted – too small stains, underfit $U = 59.3\,\%$ and overfit $O = 107.4\,\%$. In

another one it was: Dice 46.8 %, $S = 66.7\,\%$, $M = 23.3\,\%$, $U = 10\,\%$, $O = 95\,\%$. The increase in mistakes and overfit in comparison to good pictures is directly linked to the problems with numerical stability. Note that the bad file is only, if the picture contains only the too small stains. If a picture is mixture of too small and appropriate stains, then we can skip the tiny ones and treat the larger ones as the good input.

5.2 Running Time

The algorithm has been designed with the running time in concern, because it is supposed to work as a tool in an interactive process, therefore it should not overuse the operator's patience. Note that the parts of the algorithm dealing with the largest input are all $\Theta(n)$, and whenever there is a part of $O(n^2)$ (Sections 3.3, 4 item 4), the "n" is significantly smaller. It has been assumed for practical reason, that 10 seconds is target running time for an average file.

Our time testing setup consists of a laptop of the year 2013, Intel Core i5-3340M 2.70 GHz, 16 GB DDR3-1600 RAM, OS Windows 7 Professional. The application was compiled under Visual Studio 2010 in the Release mode, with the use of OpenCV library's functions (*findContours, fitEllipse*) and structures (*Mat*) as well as STL containers (*vector, list, deque, set*). The code was executed in one thread – no pararell processing.

The test consisted of the mentioned 45 pictures of real blood stains, each having between 12 and 18 million pixels. The average running time was 2.464 seconds with standard deviation of 0.431 sec., giving an average throughput of 7.16 million pixels per second (standard deviation 0.82 Mpix/sec.). The running time is mostly affected by the number of detected ellipses: correlation coefficient of 76 %. The correlation between running time and amount of bloody surface is 57 % and between running time and the number of pixels in the picture is the least of them of 54 %. Moreover, the correlation between number of ellipses and amount of bloody surface is only 44 %. As expected, the number of ellipses affects the throughput with correlation coefficient of $-80\,\%$. The target time of 10 seconds should be attainable even on older computers.

5.3 Artificial Images

The artificial image consists of a painted ellipse. Because we know exactly what were the parameters for the painting routine, we can precisely measure how well the detection results match. In this test we skip the binarisation step (the painted ellipse would either binarize fully or not at all), as well as we skip the filters, to measure just the performance of the edge-analysis, joining and fitting described in Section 3.

The most interesting question is how the algorithm performs on the damaged outline, just as it may occur with real stains. For this reason we introduced a damage mechanism – the surface of the ellipse is punctured randomly with artifacts. We

use two types of artifacts: the fine one is a 1-pixel square which imitates "pepper and salt" noise such as from high ISO sensitivity, bad pixels on camera sensor and physical impurites of a size of a grain of sand. The coarse one is 3x3 square to imitate impurities like hairs, fibers, tangles of dust, air bubbles, but also some camera flash reflections. The parameter damage factor D represents the percentage of the ellipse's surface which is lost due to the applied damage. Obviously, the isolated artifacts inside the ellipse have no contribution to the test, as the algorithm works solely on the outline, but those which coincide with the edge evidently produce an effect. Because of the stochastic nature of the damage process, each test was repeated 100 times and the average was taken to eliminate coincidence.

In each test an ellipse of sizes 180×300 (Large), 100×150 (Medium) and 40×70 (Small) was painted as if it came out of binarisation, then damaged in either Fine or Coarse process (producing 6 cases: LF, MF, SF, LC, MC, SC) to the assumed level of $D = [0.0; 0.55]$, step 0.05. Such setup resulted in 7 200 individual test instances. The parameters of the painted ellipse (x, y, a, b, ϕ) and the detected one (x', y', a', b', ϕ') were used to calculate the following scores of accuracy A:

$$A_\phi = 100\,\% - \frac{|\phi - \phi'|}{\pi}, \tag{4}$$

$$A_{ab} = 100\,\% - \frac{|\frac{a}{b} - \frac{a'}{b'}|}{\frac{a}{b}}, \tag{5}$$

$$A_{xy} = 100\,\% - \frac{\sqrt{(x - x')^2 + (y - y')^2}}{\sqrt{w^2 + h^2}}. \tag{6}$$

In the angular accuracy A_ϕ formula (4), π is the range of the returned value by the fitting algorithm. In the center point accuracy A_{xy} formula (6), w, h represent the dimensions of a bounding rectangle of the original ellipse.

As seen in Figures 9, 10, 11, the algorithm works reliably up to $D = 30\,\%$. Keeping in mind, that the stains with surface damage above 10 % would be discarded by the surface filter anyway, this leads to the conclusion that the devised algorithm is well capable of performing the entrusted task.

6 CONCLUSION

The blood drop trajectory tracing functionality of the forensic modelling software requires the blood stains to be vectorised in the form of ellipses. The solution presented herein provides automation of this process, which otherwise is done manually. The authors' experience in manual encircling for the need of the Section 5.1 test has shown that manual encircling is a highly exhausting task. It may take from 5 to 20 minutes to perform manual fitting on one picture file, however the effort required to establish carefully all the ellipse's parameters with 1 pixel precision, and so for each one ellipse individually, makes the operators wearier with each following

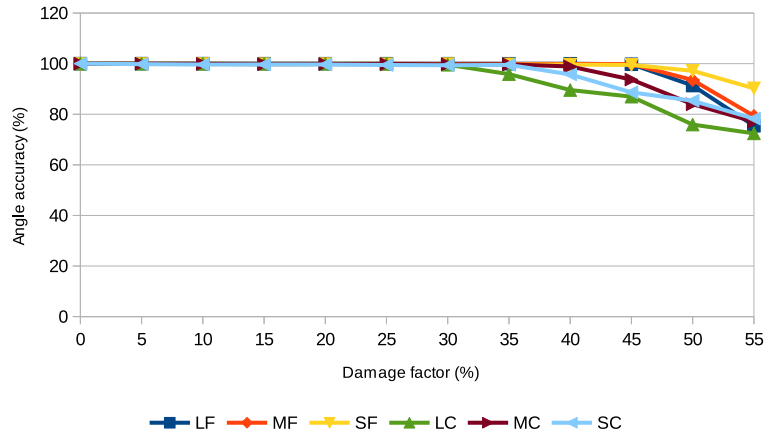

Figure 9. Angular accuracy

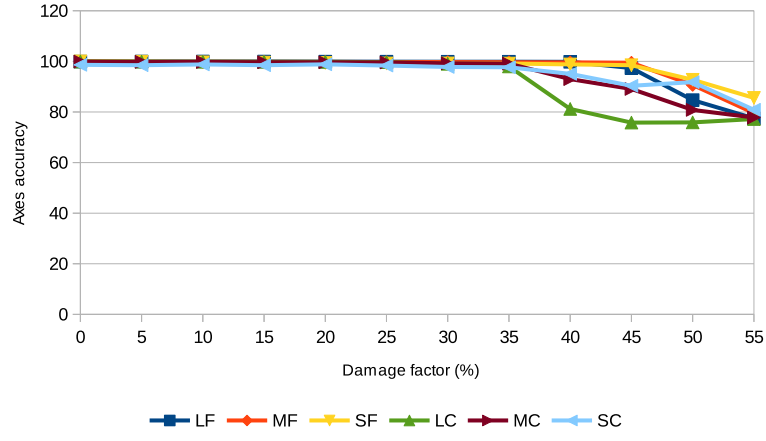

Figure 10. Axes accuracy

file, tempting them to start skipping certain stains for their comfort. Considering that for forensic technicians the manual fitting is not the essence of their job, rather a small but laborious task in the chain of many other, the automatic detection is going to spare technicians' energy for tasks where human input is more worthy.

The design of this solution took into account the qualitative requirements of what possibly may be a court evidence, so the algorithm tries to emphasize the results on the most trustworthy appearing input. The obtained test results show that the quality of the produced result meets the requirements of the field of application so does the running time.

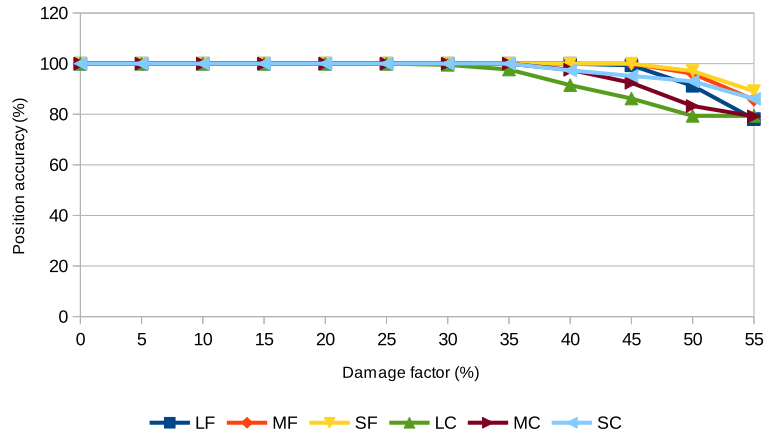

Figure 11. Center point accuracy

Acknowledgements

This work has been financially supported by the National Centre for Research and Development (`http://www.ncbr.gov.pl/`) as a part of the project "Reconstruction of the Course of Events Based on Bloodstain Pattern Analysis" (DOBR/0006/R/ID1/2012/03) realized within Defence and Security Programme (2012–2015).

Thanks to professor Zbisław Tabor, Ph.D. of Cracow University of Technology for critical feedback during the development process.

Thanks to CLKP, the Central Forensic Laboratory of the Police (`http://clk.policja.pl/`) for supplying the pictures of real blood stains.

A ELLIPSE DETECTION BY CONIC FITTING

The paper [4] investigates six approaches to the ellipse detection, four of which are of the algebraic distance type with the running time of $26n$, whereas the two other have $50n$ and $1\,300n$. The algebraic distance is the fastest approach and therefore it is widely used for the task, as in the OpenCV's function *fitEllipse* used in our implementation.

With the help of the conic section equation:

$$v_5x^2 + v_4xy + v_3y^2 + v_2x + v_1y + v_0 = 0 \tag{7}$$

the algebraic distance seeks such $v = [v_5, v_4, v_3, v_2, v_1, v_0]^T$ which satisfies the above equation with the smallest sum of errors for the given input set $S = \{(x, y)\}$. The four algebraic distance algorithms described in [4] differ in the way how it is achieved.

One way is to create the *design matrix* D of $n \times 6$ (n is the count of S) such that $D \cdot v$ implements the Equation (7). Now, the sought result can be found by solving an eigensystem

$$D^T Dv - \lambda v = 0. \tag{8}$$

The sought value v will be the eigenvector corresponding to the smallest eigenvalue λ. In the final step, the obtained v is transformed into the commonly used (x, y, a, b, ϕ).

REFERENCES

[1] Bai, X.—Sun, C.—Zhou, F.: Splitting Touching Cells Based on Concave Points and Ellipse Fitting. Pattern Recognition, Vol. 42, 2009, No. 11, pp. 2434–2446, doi: 10.1016/j.patcog.2009.04.003.

[2] Chia, A. Y.-S.—Rahardja, S.—Rajan, D.—Leung, M. K.: A Split and Merge Based Ellipse Detector with Self-Correcting Capability. IEEE Transactions on Image Processing, Vol. 20, 2011, No. 7, pp. 1991–2006.

[3] Duda, R. O.—Hart, P. E.: Use of the Hough Transformation to Detect Lines and Curves in Pictures. Communications of the ACM, Vol. 15, 1972, No. 1, pp. 11–15, doi: 10.1145/361237.361242.

[4] Fitzgibbon, A. W.—Fisher, R. B.: A Buyer's Guide to Conic Fitting. Proceedings 6$^{\text{th}}$ British Machine Vision Conference (BMVC '95), Birmingham, 1995, Part 2, pp. 513–522, doi: 10.5244/C.9.51.

[5] Huang, L.—Ma, J.: A Probabilistic Mixture Approach to Automatic Ellipse Detection. Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV 2013), 2013, pp. 573–580.

[6] Kim, E.—Haseyama, M.—Kitajima, H.: Fast and Robust Ellipse Extraction from Complicated Images. Proceedings of IEEE International Conference on Information Thechnology and Applications (ICITA 2002), 2002, pp. 357–362.

[7] Mai, F.—Hung, Y. S.—Zhong, H.—Sze, W. F.: A Hierarchical Approach for Fast and Robust Ellipse Extraction. Pattern Recognition, Vol. 41, 2008, No. 8, pp. 2512–2524, doi: 10.1016/j.patcog.2008.01.027.

[8] Prasad, D. K.—Leung, M. K. H.—Quek, C.: ElliFit: An Unconstrained, Non-Iterative, Least Squares Based Geometric Ellipse Fitting Method. Pattern Recognition, Vol. 46, 2013, No. 5, pp. 1449–1465, doi: 10.1016/j.patcog.2012.11.007.

[9] Xu, L.—Oja, E.—Kultanen, P.: A New Curve Detection Method: Randomized Hough Transform (RHT). Pattern Recognition Letters, Vol. 11, 1990, No. 5, pp. 331–338, doi: 10.1016/0167-8655(90)90042-Z.

[10] Yao, J.—Kharma, N.—Grogono, P.: A Multi-Population Generic Algorithm for Robust and Fast Ellipse Detection. Pattern Analysis and Application, Vol. 8, 2005, No. 1-2, pp. 149–162.

Tomasz Wójtowicz received his M.Sc. in computer science from Jagiellonian University and is a Ph.D. candidate therein. His research interests include pattern recognition, computer vision, artificial inteligence and also biocybernetics and evolutionary biology.

Dariusz Bułka received his M.Sc. in computer science from AGH University of Science and Technology. He is the Chief Engineer in CYBID Ltd., a company developing specialized software and systems for forensic analysis, simulations, data gathering and crime/accident event reconstruction.