Basic Concept of Cuckoo Search Algorithm for 2D Images Processing with Some Research Results
An Idea to Apply Cuckoo Search Algorithm in 2D Images Key-points Search

Marcin Woźniak and Dawid Połap
Institute of Mathematics, Silesian University of Technology, Kaszubska 23, 44-100 Gliwice, Gliwice, Poland

Keywords: Key-points Search, 2D Image Processing, Cuckoo Search Algorithm, Evolutionary Computation.

Abstract: In this paper, the idea of applying cuckoo search algorithm to search for key-points in 2D images is formulated. For a set of test images we present and verify simplified version, it's efficiency and precision. Research results are presented and discussed in comparison to classic methods like simplified SURF and SIFT algorithms to show potential efficiency of applied computational intelligence.

1 INTRODUCTION

Evolutionary computation (EC) methods find their application in various modern sciences, IT projects, economy and industry. Computational intelligence (CI) has many mechanisms that can be used efficiently to calculate even sophisticated mathematical models. They are efficient, easy to implement and precise. Let us mention only some of them.

Application of EC methods to dynamic systems positioning is discussed in (Nowak and Woźniak, 2008). In (Gabryel et al., 2012) CI was applied to create learning sets for artificial intelligence (AI) control systems. CI positioning of queueing systems applied in computing network models is presented in (Gabryel et al., 2013) and (Woźniak et al., 2014b) or (Woźniak et al., 2014a). Finally one may present EC efficiency in 2D image preprocessing (please see (Woźniak and Marszałek, 2014)). In this paper we would like to present simplified Cuckoo Search Algorithm (CSA) for 2D image processing. Cuckoo Search Algorithm is one of very promising EC methods. Even though it was formulated a few years ago there have been many successful applications of it's dedicated versions. CSA helps to optimize semantic web composition, see (Chifu et al., 2012). In (Wang et al., 2012) is presented it's efficiency in Markov models analysis. While training spiking neural models is discussed in (Vazquez, 2011). Some aspects of CSA design optimization are presented in (Yang and Deb, 2013). Allocation in distribution network by the use of CSA is presented in (Moravej and Akhlaghi, 2013). One may also find CSA used in solving scheduling problem (see (Chandrasekaran and Simon, 2012)). CSA is efficient in stochastic and reliability optimization, see (Valian et al., 2013). In (Bhargava et al., 2013) is discussed CSA application in phase equilibrium calculations. While in (Bulatović et al., 2013) optimum synthesis of six-bar double dwell linkage. Finally in (Woźniak, 2013) is discussed application of CSA to positioning queueing systems. Summing up, EC methods are applied, where CI may help to improve data processing. There are many different fields of efficient applications of EC, as presented above. All this convinced us to implement one of them in 2D image processing.

In the following sections we try to present and discuss potential application of EC method, in particular Cuckoo Search Algorithm, in the process of 2D image processing. Research results show potential efficiency and high precision of CSA in search for key-points in 2D images. For the research were taken sample images from open test images databases[1][2]. When compared to other conventional methods of key-points search, applied CSA method can efficiently search for key-areas in various pictures. The method discussed in section 2.3 is less complicated. Therefore performed calculations have lower computational complexity. This makes presented solution efficient, faster and easier to implement in comparison to classic methods.

[1] *www.imageprocessingplace.com*
[2] *www.ece.utk.edu/gonzalez/ipweb2e/downloads/*

WoÅžniak M. and Połap D..
Basic Concept of Cuckoo Search Algorithm for 2D Images Processing with Some Research Results - An Idea to Apply Cuckoo Search Algorithm in 2D Images Key-points Search.
DOI: 10.5220/0005015801570164
In *Proceedings of the 11th International Conference on Signal Processing and Multimedia Applications* (SIGMAP-2014), pages 157-164
ISBN: 978-989-758-046-8
Copyright © 2014 SCITEPRESS (Science and Technology Publications, Lda.)
157

2 KEY-POINTS SEARCH

When we look at image, specific way of recognition is processed in our brain. From points in the picture, our eyes select areas where points of special properties compose objects. These areas are processed in our brain to recognize what they present. However to classify objects properly, the eyes choose some special points that help to verify each object. Similar process of recognition can be implemented in computational mechanism. Digital images are composed of points. Each of these pixels have measurable position $X = (x,y) = (x_i,x_j)$ and some special properties. We can name saturation, sharpness, brightness and many more. All these features compose areas in image that can be classified by computer algorithms (see (Woźniak et al., 2014c) and (Woźniak and Marszałek, 2014)). However correct classification depends on right recognition from key-points. Therefore one may say, that position of each pixel and it's properties are crucial aspects. These pixels compose objects, which can be classified. Therefore essential part in classification process is key-points search. CI brings many interesting methods that may help.

2.1 Classic Attempt–SURF

One of classic methods used for key-points recognition is SURF (Speeded-Up Robust Features) algorithm. This method is description of the image by selecting characteristic key-points (for more details see (Abeles, 2013), (Bay et al., 2008) or (Mehrotra et al., 2009)). It combines selection of key-points with calculating 64 element descriptor. In SURF is applied integrated image and filter approximation of block Hessian determinant (see (Decker and Paulus, 2011) or (Gossow et al., 2011)). In our SURF we approximated these points with block filter of Hessian determinant, which for point X in image I is calculated for Hessian matrix $H(X,\sigma)$ in X at scale σ as

$$H(X,\sigma) = \begin{bmatrix} L_{xx}(X,\sigma) & L_{xy}(X,\sigma) \\ L_{xy}(X,\sigma) & L_{yy}(X,\sigma) \end{bmatrix}, \tag{1}$$

where $L_{xx}(X,\sigma)$ is convolution of Gaussian second order derivatives D_{xx}, D_{yy} or D_{xy}. To detect interesting points is used particular Hessian matrix approximation. We define it using formula

$$det(H_{ap}) = D_{xx}D_{yy} - \left(\frac{|L_{xy}(\sigma)|_F\,|D_{xx}(\sigma)|_F}{|L_{xx}(\sigma)|_F\,|D_{xy}(\sigma)|_F} D_{xy}\right)^2 . \tag{2}$$

Then, image is blurred to get DoG (Difference of Gaussian) images, what helps to find edges (for details please see (Brown and Lowe, 2002)). To localize interesting points is used non maximum suppression in $3 \times 3 \times 3$ neighborhood. Maximum determinant of Hessian matrix is interpolated at scale σ to differ between first level and each octave. We fix reproducible orientation based on information from circular region around pixel. Then, is constructed square region aligned to selected orientation and SURF descriptor is extracted. Please see (Abeles, 2013) and (Bay et al., 2008) for more details.

Algorithm 1: Simplified SURF for RGB (Red-Green-Blue) color values applied to search for key-points in 2D images.

```
 1: Calculate number of pixels and rows in image,
 2: while $j < rows$ do
 3:     while $i < pixels$ do
 4:         $rowsum_j = (R_{ij})^2 + (G_{ij})^2 + (B_{ij})^2$,
 5:     end while
 6: end while
 7: Integral image is sum of rowsum,
 8: Calculate approximated Hessian value using (2),
 9: Build response layers for the image,
10: while $i < pixels$ do
11:     Calculate descriptor vector for each point,
12:     Determine orientation,
13: end while
14: Construct descriptor vector for each image point.
```

2.2 Classic Attempt–SIFT

SIFT (Scale-Invariant Feature Transform) transforms image into scale-invariant coordinates relative to local aspects. This generates features that may densely cover image for full range of scales and locations, please see (Pope and Lowe, 1998) and (Se et al., 2002). SIFT idea is based on (Zhang et al., 1995), where possibility of matching Harris corners over large image by using correlation window around each corner was discussed (see also (Azad et al., 2009) and (Sun et al., 2013) for details). This idea was developed in (Schmid and Mohr, 1997) to general image recognition, where Harris corners were applied to select key-points by rotationally invariant descriptor of local image regions. However Harris corner detector can be sensitive to changes in image scale, what makes it inefficient in processing images of different sizes. Therefore (Nelson and Selinger, 1998), (Pope and Lowe, 1998) and (Shokoufandeh et al., 1999) extended local feature approach to achieve scale invariance. Then some special features like multidimensional histograms summarizing distribution of measurements useful for recognition of textured objects with deformable shapes were discussed in (Schiele and Crowley, 2000). Final version of this scale descriptor, less sensitive to local image distortions, was given in (Lowe, 2004).

In our SIFT, features are first extracted from set of images and stored in memory. Key-point is matched by individually compared examined feature to these previously stored using Euclidean distance of feature vectors. Correct key-points are filtered from set of matches by identifying subsets of interest points that agree on object, it's location, scale and orientation. To determine these clusters we perform hash table implementation of generalized Hough transform. Each cluster of features is then subject to further verification. SIFT uses special detector to find scale-space extrema where continuous function is Gaussian. According to (Lowe, 2004) we have used $L(X,\gamma_D)$ as scale space of image defined in

$$L(X,\gamma_D) = G(X,\gamma_D) * I(X), \tag{3}$$

where $L(X,\gamma_D)$ is produced from convolution of variable-scale Gaussian $G(X,\gamma_D)$ with input image point. To scale selection is used approximated DoG filter. Then key-point is localized by taking Taylor series expansion of scale-space function

$$D(\vec{X}) = D + D_x^T\vec{X} + 0.5\vec{X}^T D_{xx}^T \vec{X}, \tag{4}$$

where $D(\vec{X})$ and it's derivatives are evaluated at image points and T is offset from these points (for details see (Brown and Lowe, 2002)). Finally after filtering, descriptor operations are performed. Descriptor is local statistic of orientations of the gradient of the Gaussian scale space.

Algorithm 2: Simplified SIFT applied to search for key-points in 2D images.

```
1: Calculate maximum number of Level,
2: Build DoG pyramid,
3: Find maximum and minimum of Level,
4: Compare vector of pixel with it's rescaled neigh-
   bors,
5: for Each Level of DoG pyramid do
6:     Match to sub pixel maximum location,
7:     Eliminate edge points,
8: end for
9: Construct keys using interpolated value.
```

2.3 Basic Concept of Cuckoo Search Algorithm

Cuckoo Search Algorithm is very efficient gradient free optimization technique, where some Gauss distribution versions are applied to optimization (for more details please see (Walton et al., 2011) and (Woźniak, 2013) respectively). In the research we have applied CSA to search for key-points. This decision is based on research results in other fields (see section 1) what suggest potential efficiency in 2D image processing.

CSA is mapping behavior of birds that everyone has heard of, cuckoos. These birds, accept for famous sounds, have special nature of breeding. It is very interesting how they do it. A cuckoo is flying and looking for nest to lay an egg. They try to choose host in peculiar way. Cuckoos are choosing nest, where are already eggs. Moreover these eggs must look familiar to cuckoos. They lay an egg and fly off. When hosts come home they either get rid of intruder egg or just simply accept new situation. This process is modeled and applied as EC algorithm, where we assume:

1. Cuckoo, egg or host nest is similar for CSA algorithm (it will be explained in this section).
2. Points in 2D image (all pixels) are potential host nests, that are of interest to flying cuckoos.
3. Each cuckoo has only one egg to lay.
4. Total amount of flying cuckoos is constant.
5. Best nests containing egg (pixels of highest quality) will be transferred to next generation.
6. Rest of cuckoo population will be taken at random within all given 2D image pixels.
7. Hosts may find that intruder egg is hosted with $1 - p_\alpha \in \langle 0,1\rangle$ probability and get rid of it. In this case a new cuckoo is placed randomly in image.

In dedicated CSA cuckoo, egg or host nest is similar. All these are the same, because the algorithm is placing points in 2D image. First we call these pixels (potential key-points) flying cuckoos. Then these cuckoos are lying eggs, therefore wa call these points eggs. Then we simulate decision taken by hosts to drop the egg or not. That is why we can use these tree names simultaneously. Moreover, for simplified numerical calculations, we assume that number of placed points is constant. We check fitness function for each of them. The best points are transferred to next round. The rest of population is taken at random to maintain constant level of cuckoos. Since these all operations are completed we start next round in the CSA algorithm. New generation of cuckoos is placed in the image and we start procedure from the beginning.

Presented method seems to be complicated, however it is not. In every generation we only model the choice of place to lay an egg with particular equations. This movement has some statistic background. It uses a concept of random walks, what helps to perform non local search in different type solution spaces. Virtual cuckoo movement is modeled with formula

$$X^{t+1} = X^t + \mu \cdot L(\beta,\gamma,\delta), \tag{5}$$

where the symbols are: $X^{t+1} = (x_i^{t+1}, x_j^{t+1})$ – next solution in CSA (potential key-point), μ – length of step in random walk based on normal distribution $N\left(\frac{\gamma}{cuckoos}; 0, 1\right)$, L(β,γ,δ) – Lévy flight for a given step length β, δ – length of minimum step for random walk and γ – scaling parameter for Lévy flight.

Lévy flight is also called random walk, in which length of particular step has value determined with special probability distribution (please see (Yang and Deb, 2009) or (Yang and Deb, 2013) respectively). Lévy flights are made isotropic in random directions, according to formula

$$L(\beta,\gamma,\delta) = \begin{cases} \sqrt{\frac{\gamma}{2\pi}} \frac{exp[-\frac{\gamma}{2(\beta-\delta)}]}{(\beta-\delta)^{\frac{3}{2}}}, & 0 < \beta < \delta < \infty \\ 0, & \text{other} \end{cases} . \quad (6)$$

For more details on Lévy flights and non local optimization theory please see (Pavlyukevich, 2007). These simple equations map flying cuckoos while searching for best nest. Finally, for each 2D image point we only have to decide if it is ”found” by hosts. This decision is modeled with equation

$$H(X^{t+1}) = \begin{cases} 1 - p_\alpha & \text{drop the egg} \\ p_\alpha & \text{the egg stays} \end{cases} , \quad (7)$$

where the symbols are: $H(X^{t+1})$ – decision taken by hosts about intruder egg X^{t+1}, $p_\alpha \in \langle 0, 1 \rangle$ – chance of each cuckoo egg to stay.

Algorithm 3: Basic CSA applied to search for key-points in 2D images.

```
 1: Define all coefficients: $p_\alpha \in \langle 0,1 \rangle$, β, γ, δ, $bestratio$, number of $cuckoos$ and number of $generations$,
 2: Dedicated criterion function: brightness of pixels according to (8),
 3: Create at random initial population in the image,
 4: t:=0,
 5: while $t \leq generations$ do
 6:     Move $cuckoos$ according to (5) and (6),
 7:     Hosts decide if the eggs stay or no – decision according to (7),
 8:     Sort points (cuckoos) according to the value of criterion function,
 9:     Evaluate population and take $bestratio$ of them to next $generation$,
10:     Rest of $cuckoos$ take at random,
11:     Next $generation$ $t{+}{+}$,
12: end while
13: Best $cuckoos$ from the last $generation$ are potential key-points in 2D image.
```

2.4 An Idea to Apply CSA to 2D Images Key-points Search

In the research we try to find simple and efficient method for 2D image processing. Algorithm presented in section 2.3 was applied to search for 2D images key-points. Each cuckoo is representing single pixel (point in 2D image). We put population of cuckoos to move from point to point and search for specific areas. Searching operation is based on fitness function. In the research we have used simplified fitness function. This function reflects brightness of each image point X as

$$Brightness(X) = b, \quad (8)$$

where symbol b denotes brightness of evaluated pixel. This measure reflects value in scale from 0.0 to 1.0, where the colors change from black to white. When cuckoos fly in each iteration (round), they pick points with best fitness within the range of their flight. Then from all cuckoos we take *bestratio* of them, where fitness function is highest or lowest depending on experiment. These points (cuckoos) are taken to next round and the rest of population is taken at random from all image points. This is made to have constant number of cuckoos, as defined in the algorithm. Taking at random some points in each iteration help to search entire image for points of interest. Finally, the last generation of cuckoos cover areas of our interest, dark or bright depending on the experiment (see algorithm 3). In the research, simulations were performed for 120 cuckoos in only 40 generations with set coefficients: $\beta = 0.5$, $\gamma = 0.3$, $\mu = 0.25$, $\delta = 0.2$ and $bestratio = 30\%$. As we can see in the following sections, even so little cuckoos could find dark or bright objects correctly.

3 RESEARCH RESULTS

As objects for examinations were taken standard test images (see section 1). We have performed experiments on different types of pictures: sharp, blurred, landscapes or human postures and faces. In the research we were looking for objects of similar type, here we have concentrated on brightness and saturation of pixels defined in standard way using (8). Each of resulting key-points (pixels) is marked in red. We have provided close-up areas showing enlarged portion of each image.

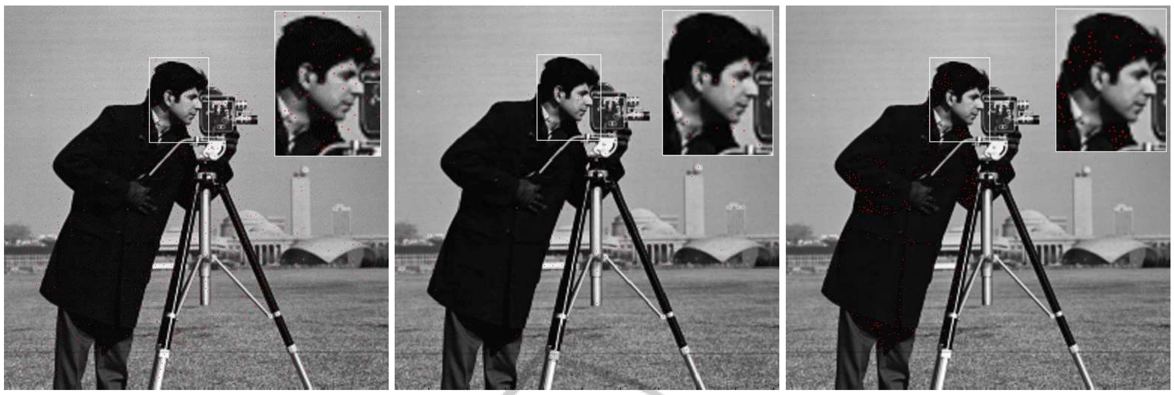
Figure 1: Dark key-points search in sharp and human posture images: from the left SURF result, SIFT result and CSA result.

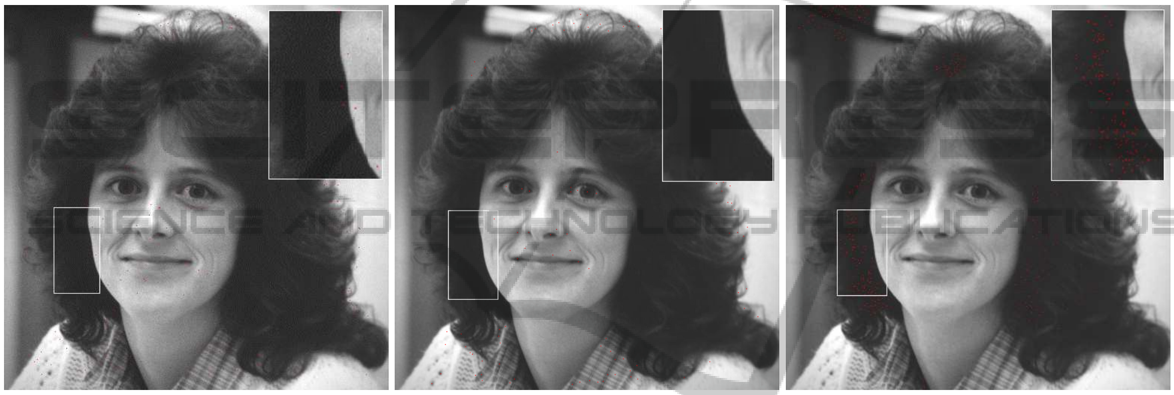
Figure 2: Dark key-points search in sharp and human face images: from the left SURF result, SIFT result and CSA result.

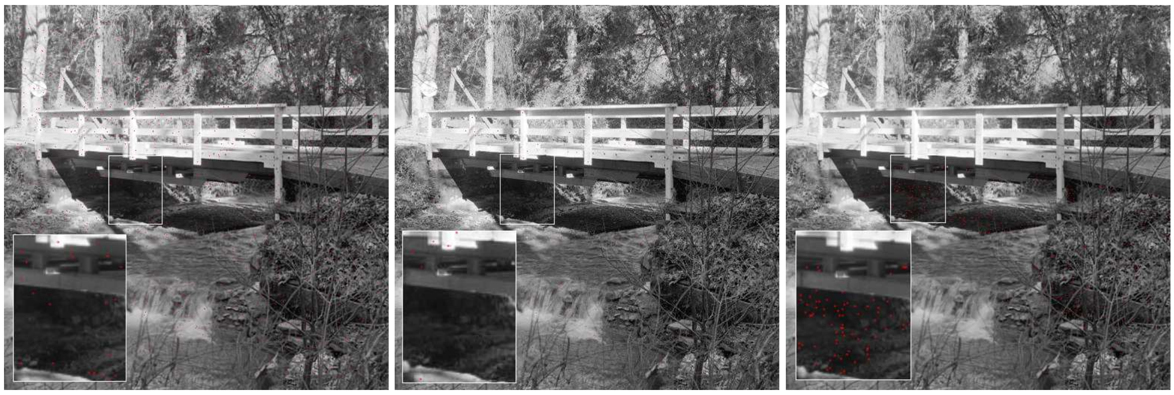
Figure 3: Dark key-points search in landscape images: from the left SURF result, SIFT result and CSA result.

3.1 Dark Areas in Images

Let us first present research results for dark objects localization, see figures 1 – 3. Similar dark objects are present in many different images. They can represent elements of landscape (trees, blocks, different constructions, etc.), natural phenomena (tornadoes, shadows, etc.), human figures or human appearance (face features, hair, eyes, etc.). We can see that dedicated CSA can easily find dark objects of different shapes. In figure 3 are presented research results of searching for buildings like bridges or natural phenomena like shade under constructions. In figure 1 we can see that CSA localized dark areas like man posture correctly. It also localized some parts of machinery (camera tripod). CSA is also efficient in localizing dark elements of human appearance (see figure 2). All these areas were found by dedicated CSA even using very small number of individuals and iterations. In comparison to SIFT, CSA results are slightly better. In compari-

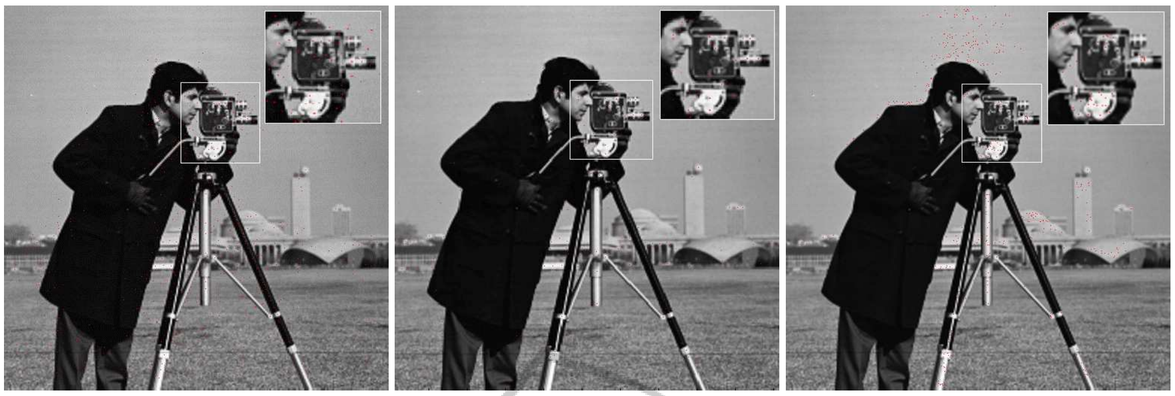
Figure 4: Bright key-points search in sharp and human posture images: from the left SURF result, SIFT result and CSA result.

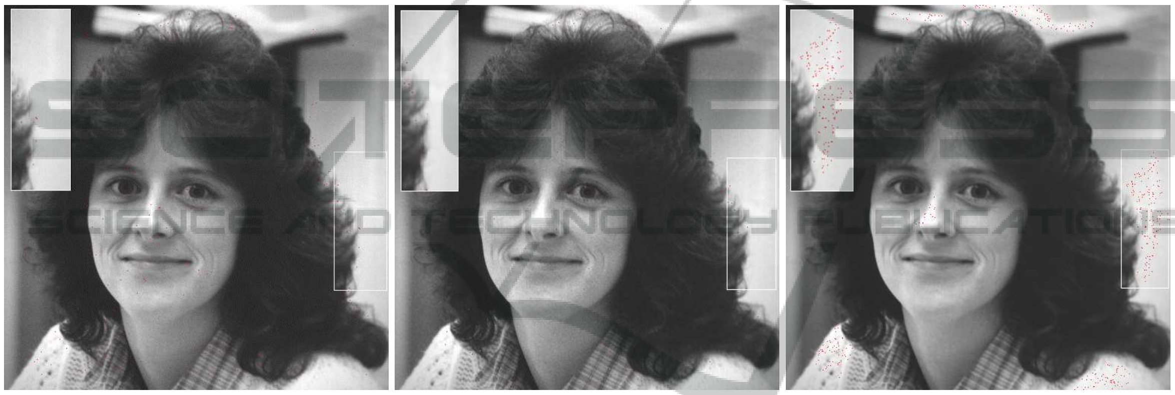
Figure 5: Bright key-points search in sharp and human face images: from the left SURF result, SIFT result and CSA result.

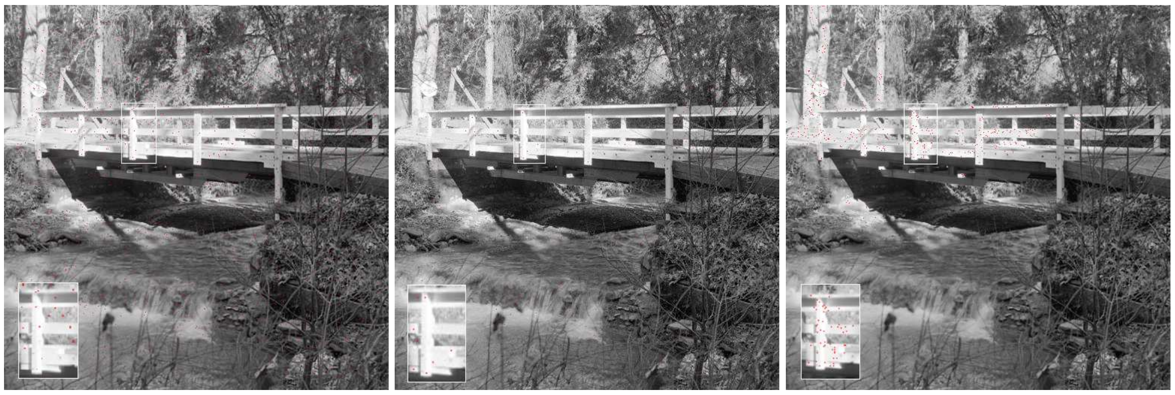
Figure 6: Bright key-points search in landscape images: from the left SURF result, SIFT result and CSA result.

son to SURF one can notice many similarly detected points, what proves CSA efficiency.

3.2 Bright Areas in Images

Let us now present research results for bright objects localization, see figures 4 – 6. Bright objects are present in many different images. They can represent objects in landscape (bright or lightened constructions), natural phenomena (clouds, sun, stars, etc.), human figures or human appearance (gray hair, eyes, make-up, bright clothing, etc.).

We can see that dedicated CSA can easily find bright objects of different shapes. CSA pointed human faces or bright clothes. Our method pointed bright parts of camera (see figure 4). It correctly localized brightest areas of human faces (see figure 5). In figure 4 one may see bright parts - details of mechanisms pointed by presented CSA. Dedicated CSA is also efficient in locating bright or lightened constructions like buildings or bridges present in figures 4 or 6.

3.3 Conclusions

Application of CSA helps to find key-points in examined 2D images. Human postures, face appearance, detailed objects like mechanisms or some nature elements (trees or shades) can be efficiently searched for. High contrast of each pixel in relation to surroundings increases CSA efficiency. If the algorithm must find points among many pixels of similar kind it may be complicated. For example if photos were taken during night, all objects of dark properties are darkened, therefore these areas may be not so easy to find. All these made process more complicated. However similar to dark areas all bright areas were found by CSA even using very small number of individuals.

4 FINAL REMARKS

Dedicated CSA allows to easily and reliably select areas of interest. At the same time, CSA allows to efficiently explore entire image in search for objects without complicated mathematical operations. This feature is it's main advantage. As presented in section 3.1 and 3.2, CSA found areas of interest covering them with key-points (red pixels), while SURF or SIFT concentrate mainly on borders. This makes CSA efficient tool for AI classifiers. Moreover performed operations are simple and have low complexity. We just use (5) – (6) to search entire examined 2D images. Research presented in this paper show that EC methods, in particular CSA, can be valuable tools for key-points search in 2D images of any kind. The algorithm gives good results if one is looking for patterns representing human shapes or architecture. Results of research show that low complexity (we have used only 120 points in 40 iterations) does not influence efficiency. We have presented only basic concept of CSA application in key-points search process. It is necessary to continue research. We will try to examine impact of bigger populations on precision and complexity. It seems that (5) can be improved to even better choose potential key-points. Moreover it can be efficient to examine if other Gauss distributions can improve this process. Another idea is to perform similar research using other EC methods.

REFERENCES

Abeles, P. (2013). Speeding up surf. In *ADC'2013 - Lecture Notes in Computer Science*, number 8034, pages 454–464. Springer.

Azad, P., Asfour, T., and Dillmann, R. (2009). Combining harris interest points and the sift descriptor for fast scale-invariant object recognition. In *IROS'2009 Proceedings*, pages 4275 – 4280. IEEE.

Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Surf: Speeded up robust features. *Computer Vision and Image Understanding*, 110(3):346–359.

Bhargava, V., Fateen, S., and Bonilla-Petriciolet, A. (2013). Cuckoo search: a new nature-inspired optimization method for phase equilibrium calculations. *Fluid Phase Equilibria*, 337:191–200.

Brown, M. and Lowe, D. (2002). Invariant features from interest point groups. In *BMVC'2002 Proceedings*, pages 656–665.

Bulatović, R., Bordević, S., and Dordević, V. (2013). Cuckoo search algorithm: a metaheuristic approach to solving the problem of optimum synthesis of a six-bar double dwell linkage. *Mechanism and Machine Theory*, 61:1–13.

Chandrasekaran, K. and Simon, S. (2012). Multi-objective scheduling problem: hybrid appraoch using fuzzy assisted cuckoo search algorithm. *Swarm and Evolutionary Computation*, 5(1):1–16.

Chifu, V., Pop, C., Salomie, I., Suia, D., and Niculici, A. (2012). Optimizing the semantic web service composition process using cuckoo search. In *IDC'2012 - Studies in Computational Intelligence*, number 382, pages 93–102. Springer.

Decker, P. and Paulus, D. (2011). Model based pose estimation using surf. In *ACCV'2010 - Lecture Notes in Computer Science*, number 6469, pages 11–20. Springer.

Gabryel, M., Nowicki, R. K., Woźniak, M., and Kempa, W. M. (2013). Genetic cost optimization of the GI/M/1/N finite-buffer queue with a single vacation policy. In *ICAISC'2013 - Lecture Notes in Artificial Intelligence*, number 7895, pages 12–23. Springer.

Gabryel, M., Woźniak, M., and Nowicki, R. K. (2012). Creating learning sets for control systems using an evolutionary method. In *SIDE'2012 (ICAISC'2012) - Lecture Notes in Computer Science*, number 7269, pages 206–213. Springer.

Gossow, D., Decker, P., and Paulus, D. (2011). An evaluation of open source surf implementations. In *RoboCup'2010 - Lecture Notes in Computer Science*, number 6556, pages 169–179. Springer.

Lowe, G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Mehrotra, H., Majhi, B., and Gupta, P. (2009). Annular iris recognition using surf. In *PReML'2009 - Lecture Notes in Computer Science*, number 5909, pages 464–469. Springer.

Moravej, Z. and Akhlaghi, A. (2013). A novel approach based on cuckoo search for DG allocation in distribution network. *International Journal of Electrical Power & Energy Systems*, 44(1):672–679.

Nelson, R. and Selinger, A. (1998). Large-scale tests of a keyed, appearance-based 3-d object recognition system. *Vision Research*, 38(15):2469–2488.

Nowak, A. and Woźniak, M. (2008). Multiresolution derives analysis of module mechatronical systems. *Mechanika*, 6(74):45–51.

Pavlyukevich, I. (2007). Lévy flights, non-local search and simulated annealing. *Journal of Computational Physics*, 266(2):1830–1844.

Pope, A. and Lowe, D. (1998). Probabilistic models of appearance for 3-d object recognition. *International Journal of Computer Vision*, 40(2):149–167.

Schiele, B. and Crowley, J. (2000). Recognition without correspondence using multidimensional receptive field histograms. *International Journal of Computer Vision*, 36(1):31–50.

Schmid, C. and Mohr, R. (1997). Local gray value invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–534.

Se, S., Lowe, D., and Little, J. (2002). Global localization using distinctive visual features. In *ICIROS'2002 Proceedings*, pages 226–231.

Shokoufandeh, A., Marsic, I., and Dickinson, S. (1999). View-based object recognition using saliency maps. *Image and Vision Computing*, 17:445–460.

Sun, C., Wang, J., Liu, Z., and Li, J. (2013). Weighted multi-scale image matching based on harris-sift descriptor. *TELKOMNIKA*, 11(10):5965–5972.

Valian, E., Tavakoli, S., Mohanna, S., and Haghi, A. (2013). Improved cuckoo search for reliability optimization problems. *Computers & Industrial Engineering*, 64(1):459–468.

Vazquez, R. (2011). Training spiking neural models using cuckoo search algorithm. In *CEC'2011 Proceedings*, pages 679–686. IEEE.

Walton, S., Hassan, O., Morgan, K., and Brown, M. (2011). Modified cuckoo search: a new gradient free optimization algorithm. *Chaos, Solitons & Fractals*, 9(44):710–718.

Wang, F., He, X., Wang, Y., and Yang, S. (2012). Markov model and convergence analysis based on cuckoo search algorithm. *Journal Computer Engineering*, 11(38):180–185.

Woźniak, M. (2013). On applying cuckoo search algorithm to positioning GI/M/1/N finite-buffer queue with a single vacation policy. In *MICAI'2013 Proceedings*, pages 59–64. IEEE.

Woźniak, M., Kempa, W. M., Gabryel, M., and Nowicki, R. K. (2014a). A finite-buffer queue with single vacation policy - analytical study with evolutionary positioning. *International Journal of Applied Mathematics and Computer Science*, (accepted–in press).

Woźniak, M., Kempa, W. M., Gabryel, M., Nowicki, R. K., and Shao, Z. (2014b). On applying evolutionary computation methods to optimization of vacation cycle costs in finite-buffer queue. In *ICAISC'2014 - Lecture Notes in Artificial Intelligence*, number 8467, pages 480–491. Springer.

Woźniak, M. and Marszałek, Z. (2014). An idea to apply firefly algorithm in 2d image key-points search. In *ICIST'2014 - Communications in Computer and Information Science*, number (accepted–in press). Springer.

Woźniak, M., Połap, D., and Marszałek, Z. (2014c). On handwriting preprocessing for 2D object recognition systems. In *IPCT'2014 Proceedings*, pages 46–53. The IRED, Digital Seek Library.

Yang, X. and Deb, S. (2009). Cuckoo search via lévy flights. In *NaBIC'2009 Proceedings*, pages 210–214.

Yang, X. and Deb, S. (2013). Multiobjective cuckoo search for design optimization. *Computers & Operations Research*, 6(40):1616–1624.

Zhang, Z., Deriche, R., Faugeras, O., and Luong, Q. (1995). View-based object recognition using saliency maps. *Artificial Intelligence*, (78):87–119.