

Objects Based Change Detection in a Pair of Gray-Level Images

OFER MILLER^{*}, ARIE PIKAZ, AMIR AVERBUCH

School of Computer Science

Tel-Aviv University, Tel-Aviv 69978, Israel

E-mail^{}: millero@post.tau.ac.il Fax number: 972-3-6409357*

Abstract

The goal of the presented change detection algorithm is to extract objects that appear in only one of two input images. A typical application is surveillance, where a scene is captured at different times of the day or even on different days. In this paper we assume that there may be a significant noise or illumination differences between the input images. For example, one image may be captured in daylight while the other was captured during night with an infrared device. By using a connectivity analysis along gray-levels technique, we extract significant blobs from both images. All the extracted blobs are candidates to be classified as changes or part of a change. Then, the candidate blobs from both images are matched. A blob from one image that does not satisfy the matching criteria with its corresponding blob from the other image is considered as an object of change. The algorithm was found to be reliable, fast, accurate, and robust even under extreme changes in illumination and some distortion of the images. The performance of the algorithm is demonstrated using real images. The worst-case time complexity of the algorithm is almost linear in the image size. Therefore, it is suitable for real-time applications.

Key words: Object extraction, Blobs, Matching process, Boundary distribution, Illumination independent.

* Author to whom correspondence should be sent

1. Introduction

Change detection identifies the locations in the field of view where structural differences between two registered input frames occurred. Many of the change detection (CD) algorithms have been used as basis for machine vision applications such as tracking systems [1], [2], [3], intelligent user interfaces [7], segmentation of moving objects [4],[5] and traffic flow analysis [6]. Thus, many CD algorithms have been developed based on "typical" video sequences. Most of the video sequences do not have substantial illumination differences between their consecutive frames and the illumination of the input frames is assumed to be almost identical. Moreover, when the change detection in video sequences is motivated by compression needs [8], the goal is to detect areas of change rather than the objects of change. An area of a change may include only part of a complete object.

This paper addresses different change detection problems, which are associated with surveillance systems such as monitoring or environmental protection. The input is not limited to "typical" video sequences. It is usually captured at different times of the day or on different days, by an aircraft or satellite flying over a site to be monitored (see Figure 1).



Figure 1: A pair of input registered images that were captured at different periods of the day. Each image contains an object (vehicle) that is not contained in the other image.

The input device for these applications may be an infrared camera, which captures one image during night while the other image is captured at daylight. Thus, illumination differences arising from changes such as light conditions, atmospheric conditions, clouds appearances or significant noise of the capturing device, have to be considered. Many surveillance systems are motivated to detect new activities or events in the monitoring site, such as digged tunnels, urban expansion, new asphalt surfaces, passing vehicles, new

constructions, etc. These changes may be captured in a long time lag between input frames. Therefore, CD algorithms for these applications must have the ability to detect the appearances of new objects in a given scene.

Many change detection techniques have been developed over the years in the literature. Some deal with time varying illumination or noisy input, and can be roughly divided into the following categories: *Statistical based approach* of first or higher order statistics [9] - [14], *surfaces modeling* which models the gray-level surface such that the surface of the errors is negligible [15], and techniques which are based on *contrast invariant* representation [16].

Kurt and Jain [9] suggested a *statistical based approached* in which each gray-level is considered as a product of two components: (1) the amount of source light incident on the scene and (2) the amount of light reflected by objects in the scene. They assumed that the amount of source light incident on a small region is approximately uniform, but the reflected light from two adjacent objects may be different. Then, in order to determine if a change occurred in a given region, a variance calculation of the intensity ratio of the region is taking place. However, the size of the region has to be large enough to fit the size of the changed region. Hsu *et al.* [15] suggested a *surface modeling* that models the gray level distribution such that the surface of the errors is negligible. When two corresponding regions are given, they are represented by a set of seven parameters: six are coefficients for the quadratic polynomial patch and one is for the sum of the square differences between the polynomial's patch and the gray-levels. Then, under the assumption that the approximating patch represents the gray-levels surface up to uncorrected noise errors, a likelihood test is used in order to determine the region of change. Monasse and Guichard [16] presented a *contrast invariant* technique that sets a new representation of an image, which is contrast independent. They decomposed the input image into a tree of "shapes", based on connected components of level sets, which provides a full and no redundant representation of the image. This method, which is based on the assumption that there is invariance under change of contrast, can be used to perform change detection between images. Since all these methods were designed for video sequences, they are inadequate to deal with significant differences in illumination that arise from changes in the source of light or from noisy and poor quality images captured by infrared sensors. Moreover, some of these techniques assume that motion between consecutive frames must be detected in the changed regions. Thus, it is not designed to deal with the appearance of a new object.

Other change detection algorithms, e.g. [17], [18], [19], were designed to deal with input that come from site monitoring and large lag of times rather than video sequences. Gee and Newman [18] suggested an algorithm that models a site, which utilizes the information of the structural surface of the site. In [17], [19] it is assumed that the information regarding the potential object of change is given, and thus, they developed an automatic target recognition (ATR) systems. However, these techniques, which are directed for known object detection, cannot handle well the problem of detecting the appearance of new objects in the scene because it is usually not realistic to specify a specific geometrical model of the object beforehand. In addition, [20] and [23] do not assume a-prior information (e.g. size, shape and texture) about the model site or the objects in it. Carlotto [20] suggests an algorithm which is based on detecting patterns of changes among consecutive frames that were obtained when specific types of activities occurred. On the contrary to change detection algorithms, which are able to detect significant change between pairs of images, their approach extracts only the changes that match a given pattern over time. Bruzzone and Fernández [23] present an adaptive semi-parametric technique for an unsupervised estimation of the statistical terms associated with the gray levels of changed and unchanged pixels in a difference image. They utilize the reduced parzen estimate (RPE) and the expectation-maximization (EM) algorithm in the estimation procedures. Then, the resulting estimates and to a Markov Random Field approach that were used to model the spatial-contextual information contained in the multitemporal images considered, they generated a change detection map between the two input images.

The proposed algorithm gets as an input a pair of registered gray-level images of some geographical area. The input images may have substantial illumination differences. A long time interval between captured images is also possible. In addition, each image may contain objects that are not contained in the other. The algorithm is an object based approach. A set of all objects that do not exist in both images is defined as the 'object of change'. If the same object exists in both images in different locations, it is also considered as an 'object of change'. No prior knowledge on the site, its statistics or its objects are given. The advantage of an object-based approach is that the input images are partitioned into non-overlapping independent regions that correspond to the image content. On the other hand, the region based approaches (e.g. [9], [15][16], [16]) usually partition the image into pre-defined squares of blocks and each region (block) is considered as a 'change' or 'unchanged' independently of the other blocks. The main drawback of a block-based approach is that we

have to determine ahead of time the size of the blocks independently from the image content.

In the first step of the proposed algorithm, a blobs extraction algorithm that is based on connectivity analysis technique [21] is applied on the two images. All the conspicuous blobs in both input images (I_1 and I_2) are extracted. The extraction of the blobs in I_1 and I_2 is based on local considerations, and thus, it is insensitive to illumination changes between the images. We denote by $\{C_i^{(1)}\}_{i=1}^{N_1}$ and $\{C_j^{(2)}\}_{j=1}^{N_2}$ the set of N_1 and N_2 blobs (objects) that were detected and extracted by the blob extraction algorithm in I_1 and I_2 , respectively. The blobs that may be considered as 'change' belong to either $\{C_i^{(1)}\}_{i=1}^{N_1}$ or $\{C_j^{(2)}\}_{j=1}^{N_2}$. Then, the problem of change detection is reduced to the problem of detecting blobs in $\{C_j^{(2)}\}_{j=1}^{N_2}$ that are not included in $\{C_i^{(1)}\}_{i=1}^{N_1}$ and vice versa. A perfect match between the corresponding blobs is not expected. Moreover, several blobs from one image may be connected or united into a bigger blob in the other image (thus, even if no change occurred, the values N_1 and N_2 can be different). Therefore, for each blob in $\{C_i^{(1)}\}_{i=1}^{N_1}$ a search for a corresponding blob in $\{C_j^{(2)}\}_{j=1}^{N_2}$ is performed, and vice versa. A weight of change is computed for each blob based on the gradient magnitudes saliency and distribution along its boundary. The weight is computed for the blob contour in both images. The existence of the corresponding object in one or both images is determined according to the contour weights in both images. If the object exists in only one of the two images it is declared as an "object of change". The proposed algorithm is efficient, accurate and robust. The overall time complexity is almost linear in the image size (see details in section 4.2), and thus it is adequate for real-time applications.

The rest of the paper is organized as follows. In section 2 we describe the blob extraction process. Section 3 introduces the core of the change detection algorithm which is based on a match analysis between the extracted blobs. Implementation of the change detection steps followed by complexity analysis is given in section 4. Experimental results are presented in section 5 and we conclude this paper in section 6.

2. Extraction of significant blobs

The first step of the proposed algorithm extracts all the significant blobs in each input image. These blobs are the potential candidates to be classified as objects of change. This step is based on an earlier technique for connectivity analysis along gray levels (CAG) suggested in [21]. In this section we introduce how the CAG technique is used to extract significant blobs.

2.1 Blob extraction: an overview

We denote by $I^{(t)}$ $t = 0, 1, \dots, G-1$, the binary image that emerges after thresholding the gray-levels image I with the threshold t where G is the number of gray levels in I . A connected component in a binary image is defined as a set of black pixels such that there exists a 4-connected path of black pixels between every two pixels in the set. This component is classified as a significant blob (object) if it is conspicuous relative to its local area. It is clear that a binary image can be represented by the set of all connected components it contains. A gray-levels image I with G gray-levels can be represented by the set $\{I^{(t)}\}, t = 0, 1, \dots, G-1$ of G binary images. The goal of this algorithm is to automatically detect all the significant blobs in an input gray-level image I . Figure 2 is an example of an input image with its detected significant blobs.

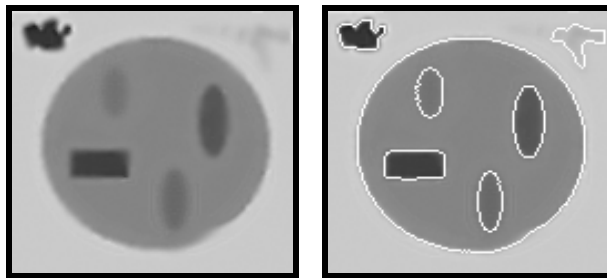


Figure 2: left: the original input image; right: the extracted objects bounded by white curves.

Generally, a blob in a gray-levels image is considered visually conspicuous (significant blob) if one or more of the following exist:

1. The gray-levels inside the blobs are considerably different from the gray-levels of the local background of the blobs.
2. Most of the magnitudes of the gradients that correspond to pixels along blobs boundaries are higher than the magnitudes of the gradients that correspond to pixels in the local background.

3. The texture inside the blobs is considerably different from the texture of the local background of the blob.

The algorithm considers blobs that satisfy the first two conditions. It is assumed that for each blob there exists a unique threshold value t , $t \in 0, 1, \dots, G-1$, such that the blob becomes a connected component in this specific $I^{(t)}$. At the moment, we are interested only in objects which are darker than the background (for bright objects we just have to negate the image). Let $C_i^{(t)}$ be the i^{th} connected component in the binary image $I^{(t)}$ of the image I . For a value $\tilde{t} > t$ there exists a (single) connected component $C_j^{(\tilde{t})}$ in the binary image $I^{(\tilde{t})}$ that contains $C_i^{(t)}$. If we treat t as a time parameter, connected components spread-out as time advances. As t increases, connected components are united into a bigger connected-component until finally, for $t=G-1$, the whole image becomes one connected component. The spread-out of a connected component in $I^{(t)}$ as t increases, is demonstrated in Figure 3, which was taken from [21].

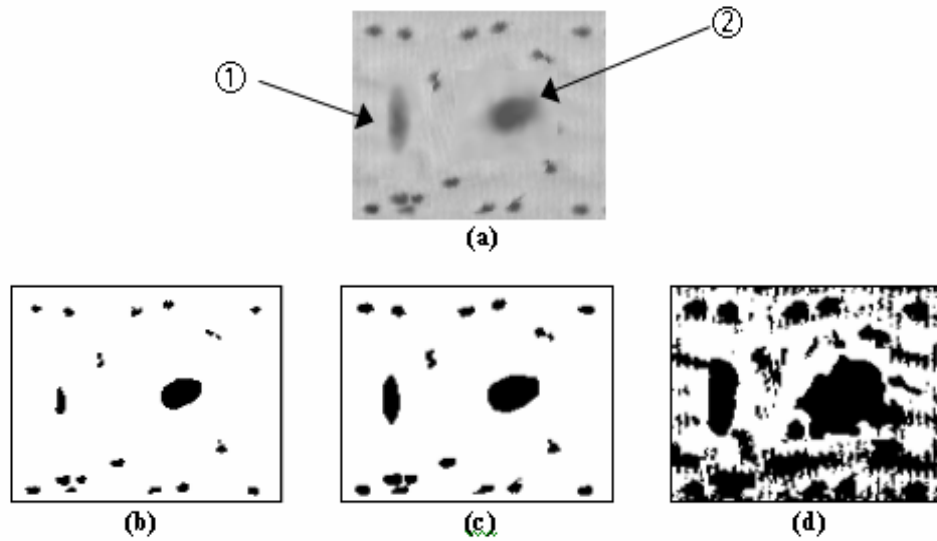


Figure 3: (a) Original image. (b) The binary image $I^{(120)}$. (c) The binary image $I^{(140)}$. (d) The binary image $I^{(190)}$. Notice that blobs ① and ② in (a) spread-out as t increases.

For a given blob O there exists a sequence of candidate blobs $(C^{(t)})_{t=t_0}^{G-1}$, where $C^{(t)}$ is a significant blobs (connected component) in $I^{(t)}$, if the following are satisfied:

1. $t' < t''$ and $C^{(t')} \subseteq C^{(t'')}$ where $t', t'' \in 0, 1, \dots, G-1$
2. There exists a value $\tilde{t} \geq t_0$ such that $O \subseteq C^{(\tilde{t})}$.

There is a value of t (or interval of values) for which $C^{(t)}$ represents best the significant blob among all the candidate blobs. If t is too small, the corresponding component $C^{(t)}$ is an internal part in a significant blob. If t is too big, the significant blob is an internal part in $C^{(t)}$. This is illustrated in Figure 3. Figure 3b presents the binary image $I^{(120)}$ (that is, $t=120$) that contains components that are only part of the “real” blobs. Figure 3d is the binary image $I^{(190)}$ that contains components that are bigger than the “real” significant blobs. The binary image in Figure 3c corresponds to $t=140$, which best represents the two selected blobs in Figure 3a

The threshold t that corresponds to each object has to be automatically computed in the blobs extraction process. Therefore, a weight is assigned to each component. The weight is a function of the threshold parameter t . It is denoted by $w(c^t)$, where C is the relevant component. The function $w(c^t)$ is defined as follows: In the binary image $I^{(t)}$ there exists at most a single segment C' that satisfies $C' \cap C \neq \emptyset$. The value of $w(c^t)$ is defined as the average value of the gradient magnitudes along the boundary of the component C' . The gradient magnitude is chosen as the weight of each pixel. Hence, the weight of a blob is defined to be the average weights of the pixels along the object boundary. The weight is expected to be proportional to the blobs saliency. Let $C^{(\tilde{t})}$ be a component that corresponds to an object. Its weight is expected to be the maximal weight among the weights of all the clusters that are not disjoint to $C^{(\tilde{t})}$. Any connected component $C^{(t)}$ satisfies one of the following: $C^{(t)} \cap C^{(\tilde{t})} = \emptyset$ or $C^{(t)} \subseteq C^{(\tilde{t})}$ and then $t \leq \tilde{t}$, or $C^{(\tilde{t})} \subseteq C^{(t)}$ and then $\tilde{t} \leq t$. The weight function $w(c^t)$ is expected to have local maximum at values of t that correspond to the binary image $I^{(t)}$ that contains the object.

The weight which is assigned to each pixel, in the input image I , is a measure of edge saliency. In particular, pixel that resides on an edge gets higher weight than a non-edged pixel. The magnitudes of the gradients of I are used for the pixel weight calculation. The approximation of the gradient vector in a pixel (u, v) is given for $1 \leq u, v \leq N - 1$ by:

$$I^{MAG}(u, v) = \left(\frac{f(u+1, v) - f(u, v) + f(u+1, v+1) - f(u, v+1)}{2}, \frac{f(u, v+1) - f(u, v) + f(u+1, v+1) - f(u+1, v)}{2} \right) \quad (1)$$

where $f(u, v)$ is the gray-level of the pixel in column u and row v in the image I .

For a given pixel $\underline{p}_i=(u_i,v_i)$ let $w(\underline{p}_i)$ be the intensity value of the pixel $I^{(\text{MAG})}(u_i,v_i)$. The weight of a given blob C is then defined by:

$$w(C^t) \stackrel{\text{def}}{=} \frac{1}{|\partial C|} \sum_{\underline{q} \in \partial C} w(\underline{q}) \quad (2)$$

where ∂C is the set of boundary pixels of the component C and $|\partial C|$ is the size of this set. A pixel \underline{q} is defined as a boundary pixel of the connected component C if it belongs to C and at least one of its four nearest neighbors does not belong to C . Let \underline{p} be a pixel that is united with the connected component C . Let C' be the result of the union between C and $\{\underline{p}\}$. The weight of C' is computed by:

$$w(C') = \frac{w(C) \cdot s(C) + w(\underline{p}) - \sum_{\substack{\underline{q} \in \partial C \\ \text{and } \underline{q} \notin \partial C'}} w(\underline{q})}{s(C)+1} \quad (3)$$

where $s(C)$ is the number of pixels in C . It is clear that the set $\{\underline{q} | \underline{q} \in \partial C \text{ and } \underline{q} \notin \partial C'\}$ is composed only from pixels that are the nearest-neighbors of the pixel \underline{p} . Therefore, only a constant number of operations are required to compute $\sum_{\substack{\underline{q} \in \partial C \\ \text{and } \underline{q} \notin \partial C'}} w(\underline{q})$. The same is true for the computation of $w(C')$.

A graph of the weight function (Eq. 3) that corresponds to object (2) in Figure 3a is presented by Figure 4.

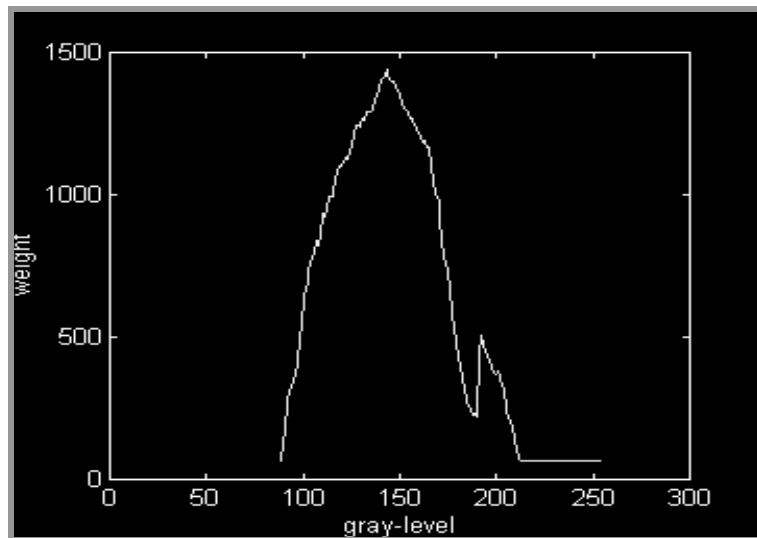


Figure 4: The weight function which corresponds to object 2 in Figure 3. The local maximum is achieved at threshold $t=142$.

Note that the weight of the union of a connected component C with a component that is composed of a single pixel is also computed in a constant number of operations. If an examined blob O corresponds to a connected component C in the binary image $I^{(\tilde{t})}$ then its gray-levels must differ from the gray-levels of its local background. From the definition of the function $w(c^t)$ at threshold t , if $w(c^t)$ reaches a local maximum at \tilde{t} then the blob O is salient related to its local background. Thus, the combination of connectivity and gradients along the boundary of a connected component is a reliable indication for the significance of the given blob.

However, an implementation of the above with an exhaustive search through all the connected components in all $\{I^{(t)}\}$, $t=0,1,\dots,G-1$ is not practical in term of time computational. A more efficient search is needed. For this task a *Union-Find* data structure that handles *Set Disjoint* structure ([22], chapter 22) is used. The construction and analysis of all the connected components through G gray-levels is accomplished in worst case time complexity of $O(\alpha(n,n)\cdot n)$ where $\alpha(n,n)$ is the inverse of the Ackerman function [22], and n is the number of pixels in image I (see complexity analysis in section 4.2).

2.2 Notation and Implementation

The implementation (See appendix A) of the blob extraction with its associated data structures that are needed for the match analysis step is described next. We apply four times the blobs extraction process on each of the two input images I_1 and I_2 and their negatives $\overline{I_1}$ and $\overline{I_2}$. Its application on the original images extracts the dark blobs relative to its surroundings while its application on the negative extracts the bright object. The outputs from the blob extraction are represented by four lists of objects: SOL_1 , $\overline{SOL_1}$, SOL_2 and $\overline{SOL_2}$ where $SOL_1 = \{C_i^{(1)}\}_{i=1}^{n_1}$, $\overline{SOL_1} = \{\overline{C_i^{(1)}}\}_{i=1}^{m_1}$, $SOL_2 = \{C_i^{(2)}\}_{i=1}^{n_2}$, $\overline{SOL_2} = \{\overline{C_i^{(2)}}\}_{i=1}^{m_2}$. Denote by $SOL^{(1)}$ the list $SOL_1 \cup \overline{SOL_1}$, and similarly by $SOL^{(2)}$ the list $SOL_2 \cup \overline{SOL_2}$. We denote by $O_i^{(1)}$ an object from $SOL^{(1)}$ where i , $i=1,\dots,m_1+n_1$ is its index in $SOL^{(1)}$. $O_i^{(2)}$ is an object from $SOL^{(2)}$ where $i=1,\dots,m_2+n_2$ is its index in $SOL^{(2)}$. The lists $SOL^{(1)}$ and $SOL^{(2)}$ contain all the candidates to be classified as objects of change that exist in image I_1 and I_2 , respectively.

Our assumption is that each object of change exists in one of the four lists of objects SOL_1 , $\overline{SOL_1}$, SOL_2 or $\overline{SOL_2}$. The objects in each of the four lists are disjoint, but there might be a pair of connected components from SOL_1 and $\overline{SOL_1}$ (and similarly from SOL_2 and $\overline{SOL_2}$) with non-empty intersection.. The idea is to find for each object in $SOL^{(1)}$ a matched object in I_2 , and then for each object in $SOL^{(2)}$ a matched object in I_1 . An example for the outputs from the blobs extraction is given in Figure 5 where each image (a,b,c and d) represents one list of objects.

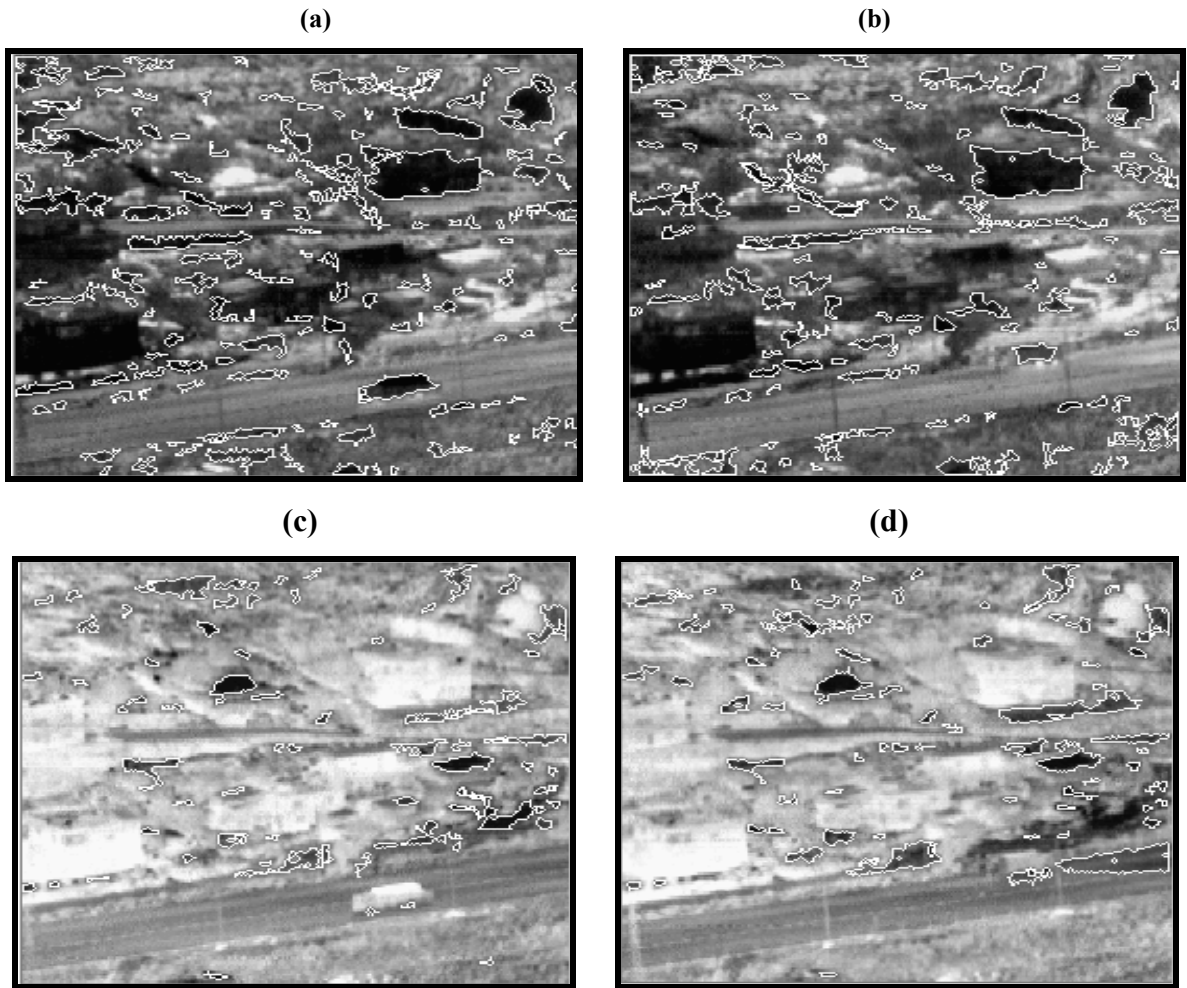


Figure 5: The outputs from the application of the blob extraction algorithm on Figure 1. The extracted blobs are bounded by white curves. The blobs that were extracted correspond to a certain set of parameters in order to restrict the number of potential objects of change (e.g. the maximal object's size in pixels was chosen to be 200 and the minimum object saliency was chosen to be 50 from the range 1-255). (a) the objects from the list $\overline{SOL_1}$ of $\overline{I_1}$ (the negative of I_1). (b) the objects from the list $\overline{SOL_2}$ of $\overline{I_2}$ (the negative of I_2). (c) the objects from the list SOL_1 of I_1 . (d) the objects from the list SOL_2 of I_2 .

3. Matching analysis

The presented algorithm is composed of two main steps. First, we extract all the objects from both input images and their negatives, and then a matching analysis is performed. In an ideal situation where noise or luminance differences are not present in the images, each pixel of $O_i^{(1)}$ $i=1,\dots,m_1+n_1$ has a corresponding pixel in $O_j^{(2)}$ $j=1,\dots,m_2+n_2$ with identical coordinates. However, as mentioned, the inputs for change detection applications may have significant level of noise, and so, its may be interpreted as structural changes between the images. Since, the blobs extraction process is based on connectivity between gray levels, it is not realistic to anticipate that objects, which exist in both images, will have the same exact boundaries (see Figure 6).

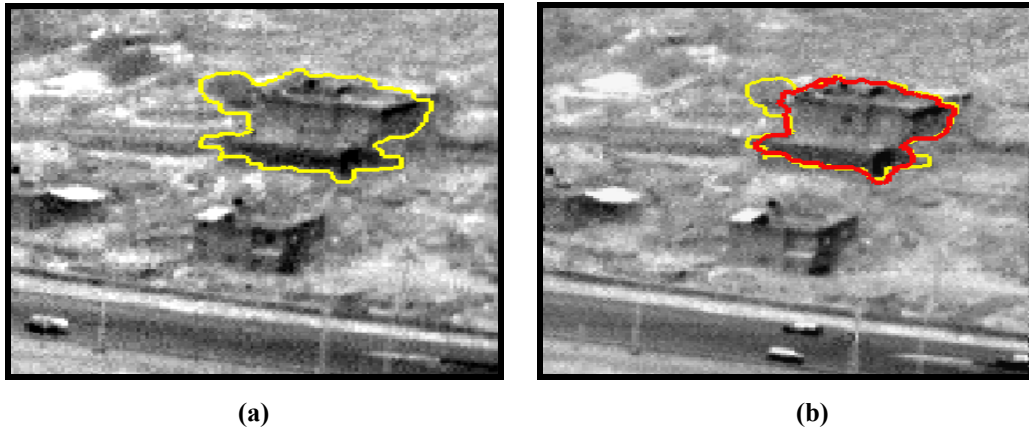


Figure 6: Noisy registered images that were captured at different periods of the day. The yellow curve in image (a) represents a single object taken from SOL_1 . The red curve in image (b) represents a single object taken from SOL_2 , and the yellow curve behind it is the object that was detected in (a). Noise and local differences in illumination between the images caused that the same objects from both images do not extracted with the identical boundaries.

Therefore, the matching analysis step is performed between each object and its corresponding location in the compared image rather than with its corresponding object in the compared object list. This ensures that a change in the gray level surface due to noise from the capturing device, which may affect the quality and performance of the blobs extraction process, will not affect the matching process between objects.

3.1 Matching Analysis Based Objects' Saliency

The process that measures the match among objects in $SOL^{(1)}$ and objects in I_2 is described next. The same procedure is applicable for measuring the match among objects in $SOL^{(2)}$ and objects in I_1 . As was mentioned, match that is based on the location of the objects in both lists is insufficient to determine if a given object is a change. However, *only if* objects from both images were extracted with the same exact coordinates (they are overlapped) then we can consider them to be the same objects. We use this argument before the match analysis is invoked in order to decrease the number of objects that participate in the matching process. Finding the objects that entirely overlapped each other is less expensive than calling the matching analysis process (a comprehensive complexity analysis will be discussed in section 4.2). Therefore, the following matching process deals with all the objects in the lists, which do not have corresponding objects with the same overlapped coordinates in the objects list from the other image.

According to the blob definition (section 2.1), a given object in the $SOLs$ lists have be conspicuous relative to its surroundings. In order to measure how much the object is conspicuous we define its saliency measure (SM). The saliency measure is based on gradient magnitudes of the gray-levels, which are less sensitive to luminance changes than the gray-levels. It is calculated as a function of the magnitudes of the gradients of its boundary pixels as follows. The saliency of an object $O_i^{(1)}$, $i \in 1, \dots, m_1 + n_1$, with boundary $\partial O_i^{(1)}$ in I_1 is defined to be:

$$sm(O_i^{(1)}, I_1) \stackrel{Def}{=} \frac{1}{|\partial O_i^{(1)}|} \sum_{(x,y) \in \partial O_i^{(1)}} |\nabla I_1(x,y)| \quad (4)$$

where $|\partial O_i^{(1)}|$ is the number of boundary pixels, $\nabla I_1(x,y)$ is the gradient vector of the image I_1 at pixel (x,y) and $|\nabla I_1(x,y)|$ is its magnitude. In addition, the saliency measure of an object $O_i^{(1)}$, $i \in 1, \dots, m_1 + n_1$ in I_2 , denoted by $sm(O_i^{(1)}, I_2)$, is similarly computed, but the gradient values of the pixels $(x,y) \in \partial O_i^{(1)}$ are taken from I_2 .

However, it is clear that we cannot use a predefined threshold to determine how much the object has to be conspicuous in both images in order to be considered as a 'no change' object. Objects that exist only in one image may be located in strong textured regions, and thus, it produces high level of saliency where the object is located in both images (see Figure 7). Moreover, such a pre-defined threshold requires having prior information on the

captured images. Instead, we define a new saliency relation (SR) measure of a given object $O_i^{(1)}$. Assume, $O_i^{(1)}$ is contained in $SOL^{(1)}$. The saliency ratio of $O_i^{(1)}$ is defined to be a ratio between $sm(O_i^{(1)}, I_1)$ and $sm(O_i^{(1)}, I_2)$ as follows:

$$SR_{O_i^{(1)}}^{(I_1, I_2)} = \frac{\sum_{(x,y) \in \partial O_i^{(1)}} |\nabla I_1(x, y)|}{\sum_{(x,y) \in \partial O_i^{(1)}} |\nabla I_2(x, y)|} \quad i \in 1, \dots, m_1 + n_1. \quad (5)$$

Then, we argue that *only if* the SR of a given object $O_i^{(1)}$ in $SOL^{(1)}$ yields high value we can determine that $O_i^{(1)}$ is an 'object of change', otherwise no reliable decision concerning the object's type ('change' or 'static') can be made (see Figure 7). A high value of SR means that the object's boundary in the compared image does not exist in its original location or does not exist at all. Moreover, if SR is high due to a low value of SM in the compared frame then it is clear that no texture or object edges is visible in this location. In these cases the decision whether an object is considered an 'object of change' is obvious. Therefore, the next section deals with a case where $SR \approx 1$. Such a case may occurs when the corresponding location of the blob, in the compared image, is characterized by a non-homogeneity or noisy background (see Figure 7b). Thus, both $sm(O_i^{(1)}, I_1)$ and $sm(O_i^{(1)}, I_2)$ are high and their ratio (SR) decreases. Therefore, an analysis of the gradient distributions is required

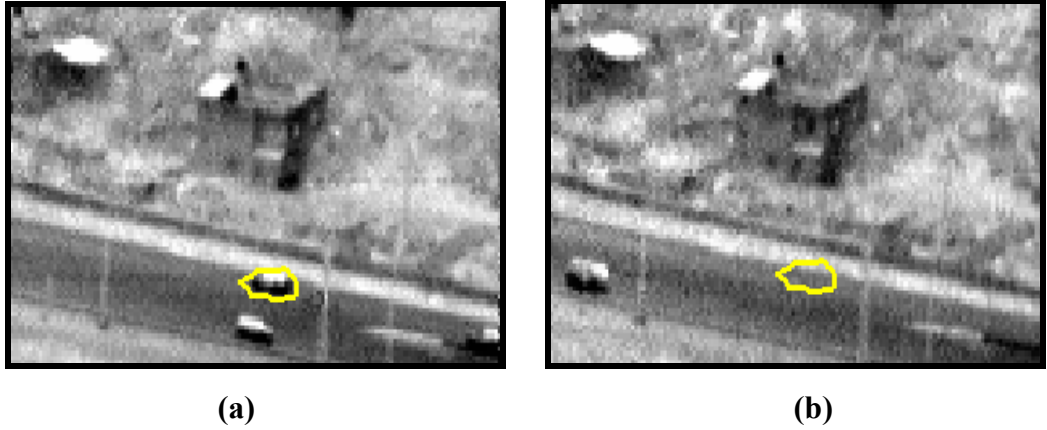


Figure 7: Illustration of a specific case where relying on a low SR value cannot lead to a certain decision about the status of the object. The object outline by yellow $O_i^{(a)}$ is existed only in 7a, and thus, it declared as an 'object of change'. However, due to the fact that $O_i^{(a)}$ lies close to the edge of the road in 7b, which has high magnitudes of its gradients, the values of SM of $(O_i^{(a)}, I_b)$ is similar to the value of $(O_i^{(a)}, I_a)$. Thus, $SR_{O_i^{(a)}}^{(I_a, I_b)}$ yields small value.

3.2 Gradients Distribution Analysis

Due to the fact that SM is based on the derived images, it is considered to be insensitive to noise and luminance changes. In addition, when the SR of a given object is sufficiently high, we consider the object as 'object of change'. Otherwise, we have to differentiate between two different situations as this section describes.

The examples I_1 and I_2 in Figure 8 demonstrate a syntactic situation, which roughly simulate "real" occurrences taken from Figure 7. I_1 in Figure 8 contains an 'object of change' $O_i^{(1)}$, whose boundary coordinates share more than 70% of the pixels of $O_j^{(2)}$ and $O_k^{(2)}$ ($O_k^{(2)}$, $k=1, \dots, 9$ is a set of nine objects in I_2). Let examine the comparison between I_1 relative to I_2 ("relative" means that we search for new objects in I_1 that do not appear in I_2). The saliency $sm(O_i^{(1)}, I_1)$ is expected to be high because it was extracted as a significant blob. However, the saliency $sm(O_i^{(1)}, I_2)$ is also expected to produce high value since $|\partial O_i^{(1)} \cap (\partial O_j^{(2)} \cup \partial O_k^{(2)})| \approx 0.7 \cdot |\partial O_i^{(1)}|$.

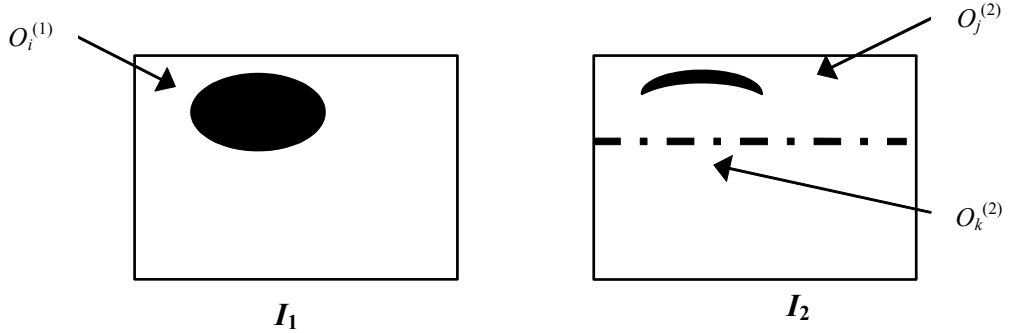


Figure 8: I_1 contains an ellipse (pointed by arrow) which is the 'object of change'. It is represented in the list $SOL^{(1)}$ as a single object $O_i^{(1)}$. I_2 contains several objects. $O_j^{(2)}$ is a single object that is overlapped with the upper area of $O_i^{(1)}$. $O_k^{(2)}$ represents nine objects, which some of them are overlapped with the lower area of $O_i^{(1)}$.

In this case (Figure 8), the SR of $O_i^{(1)}$ is *low* even though $O_i^{(1)}$ is an 'object of change'. A matching criterion that is based only on saliency measure may lead to a wrong classification of $O_i^{(1)}$ as a 'change' when the SMs are similar. In other words, the SR provides a reliable indication ('change' or 'static') only when it produces high value.

Therefore, when the SR value is low, analysis of the gradients vector distribution has to be considered to handle more complex background environments. A gradient distribution of

a given object reflects its internal gray level structure relative to its surrounding. Thus, it is expected to have different distribution of gradient magnitudes along the boundary of different objects (see $O_j^{(a)}$ in Figure 7). On the other hand, boundaries of similar objects that were taken from similar backgrounds are expected to have similar gradient magnitudes distribution.

The saliency of an object O_i is determined according to the magnitude of this boundary pixel. Denote boundary pixel number j of ∂O_i by $f(O_i, j)$, $j = 1, \dots, |\partial O_i|$ where f is the gradient magnitude. We claim that $f(O_i, j)$ contains the required information that is needed to reach a decision whether the given object exists in the other image. The saliency measure (Eq. (4)) of an object $O_i^{(1)}$ is defined as the average of $f(O_i, j)$, $j = 1, \dots, |\partial O_i|$. However, this average is considered to be a first-order statistics of $f(O_i, j)$. A measure that is based on first-order statistics cannot discriminate between different distribution that represents the gradient vector $f(O_i, j)$. An example for that is shown in Figure 9, which describes $f(O_i^{(1)}, j)$ and $f(O_i^{(2)}, j)$ (Figure 9a and Figure 9b, respectively) of a given 'object of change' as a function of j , $j = 1, \dots, |\partial O_i|$. The saliency measures of $f(O_i^{(1)}, j)$ and $f(O_i^{(2)}, j)$ gradients vector are similar since they both have the same average.

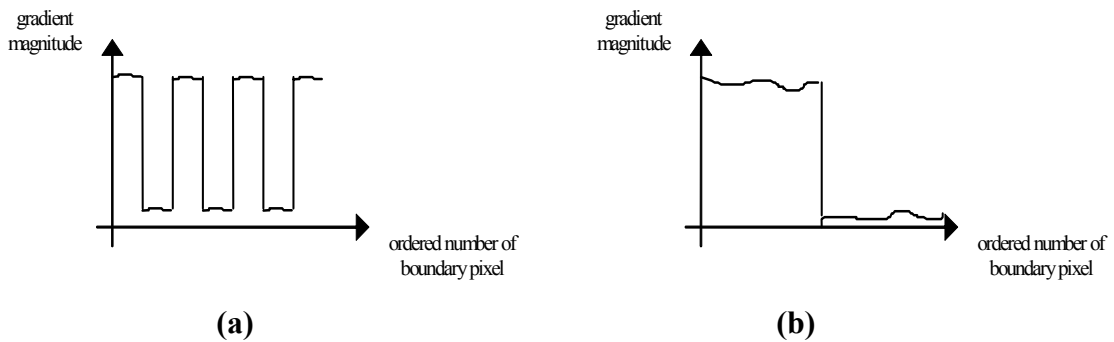


Figure 9: Two different examples of the function $f(O_i^{(1)}, j)$ and $f(O_i^{(2)}, j)$, which are defined as the gradient magnitudes of the boundary pixels $j = 1, \dots, |\partial O_i|$. (a) is the $f(O_i^{(2)}, j)$ of the original object. (b) is the $f(O_i^{(1)}, j)$ of the compared frame where it was detected.

Hence, we define a gradient distribution measure as an indication whether the gradients along the boundary are uniformly distributed.

Assume $|\partial O_i^{(1)}| = n$. Let $\{j_1, \dots, j_m\}$ be the indices of the pixels in $\partial O_i^{(1)}$ with gradient magnitudes greater than the MIN_WEIGHT (See appendix A) threshold used by the Blob Extraction algorithm. MIN_WEIGHT is an experimental value of gradient magnitude of a pixel to be considered as salient. Then, the indices of the m pixels, $\{j_1, \dots, j_m\}$ should be spread along the whole contour. If the number of “salient pixels”, m , is less than half the boundary pixels n , then the object $O^{(1)}$ is not considered as a potential candidate for an object of change.

Let
$$P_k \stackrel{\text{def}}{=} \frac{j_{k+1} - j_k}{n}, \quad k=1, \dots, m-1$$

and

$$P_m \stackrel{\text{def}}{=} \frac{n - P_m + P_1}{n},$$

It is clear that for each k , $P_k \in [\frac{1}{n}, 1]$ and

$$\sum_{k=1}^m P_k = 1.$$

Let

$$E_{O_i^{(1)}} = -\sum_{k=1}^m P_k \cdot \log P_k$$

be the entropy of P_1, \dots, P_m . The maximal computed entropy that is reached when $P_k, P_k \in [\frac{1}{n}, 1]$ are uniformly distributed.

Let
$$E_m = -\sum_{k=1}^m \frac{1}{m} \cdot \log \frac{1}{m}$$

be the maximal entropy. When there are more P_k , which are uniformly distributed, we get a higher gradients distribution values and it is denoted by:

$$dst(O_i^{(1)}, I_1) \stackrel{\text{Def}}{=} \frac{E_{O_i^{(1)}}}{E_m} = \frac{E_{O_i^{(1)}}(p_1, \dots, p_m)}{-\sum_{k=1}^m \frac{1}{m} \cdot \log \frac{1}{m}} \quad (6)$$

where $\frac{m}{n} \geq 0.5$.

Usually it is impossible to determine how the gradient vector should be distributed without a prior information, but we argue that this information is not required since we are motivated to determine whether the objects' distributions are similar in both images instead of characterizing the objects. Therefore, we calculate the relation distribution (DSR) of $dst(O_i^{(1)}, I_1)$ and $dst(O_i^{(1)}, I_2)$ that will indicate their boundaries dissimilarity:

$$DSR_{O_i^{(1)}}^{(I_1, I_2)} = \frac{dst(O_i^{(1)}, I_1)}{dst(O_i^{(1)}, I_2)}. \quad (7)$$

As mentioned, this calculation is required *only if* the saliency relation $SR_{O_i^{(1)}}^{(I_1, I_2)}$ is low. To estimate how "low" the saliency relation still enables to perform a DSR means that we have to establish a predefined threshold. If $O_i^{(1)}$ exists in both I_1 and I_2 then from $dst(O_i^{(1)}, I_1)$ and $dst(O_i^{(1)}, I_2)$ we expect that $DSR_{O_i^{(1)}}^{(I_1, I_2)} \approx 1$. Therefore, multiplying the DSR of a given object O_i with its SR , offsets low values of $SR_{O_i^{(1)}}^{(I_1, I_2)}$ only if the saliency measured was computed on two different boundary distributions. The indication whether a 'change detection' of a given object in I_1 occurs also in I_2 is given by:

$$CD_{O_i^{(1)}} = \begin{cases} 0 & m_1 / n \leq 0.5 \text{ or } m_2 / n \leq 0.5 \\ SR_{O_i^{(1)}}^{(I_1, I_2)} \cdot DSR_{O_i^{(1)}}^{(I_1, I_2)} & \text{else} \end{cases} \quad (8)$$

Where m_1 and m_2 are the number of salient boundary pixels in $O_i^{(1)}$ and $O_i^{(2)}$ respectively. We apply Eq. (8) for each object i in SOL_1 to determine whether objects exist in I_1 and not in I_2 . To obtain the objects of change that exist in I_2 and not in I_1 we apply:

$$CD_{O_i^{(2)}} = \begin{cases} 0 & m_1 / n \leq 0.5 \text{ or } m_2 / n \leq 0.5 \\ SR_{O_i^{(2)}}^{(I_2, I_1)} \cdot DSR_{O_i^{(2)}}^{(I_2, I_1)} & \text{else} \end{cases} \quad (9)$$

for each object i in SOL_2 .

4. Implementation and Complexity

In this section we present a detailed description of the algorithm, including the step implementations of the main process follows with time complexity analysis.

4.1 Implementation

Input:

I_1 and I_2 are the input pair of registered gray-levels images. \bar{I}_1 and \bar{I}_2 are their negative images. β is set to be 2.5 (See section 5.4)

Output:

$SOL^{(out)}$ final list of ‘objects of change’. It is initialized to be an empty list.

Process:

1. Apply the blobs extraction algorithm (section 2) on the images I_1 and \bar{I}_1 in order to get the output lists of objects, SOL_1 and $\overline{SOL_1}$, respectively. Denote the union $SOL_1 \cup \overline{SOL_1}$ by $SOL^{(1)}$. Similarly, construct the list $SOL^{(2)}$.
2. **For** each object $O_i^{(1)}$, $i = 1, \dots, m_1 + n_1$ in $SOL^{(1)}$ **do**:
 - 2.1. Let $\underline{p}=(x_i, y_i)$ be a representative pixel of $O_i^{(1)}$. Assume, without loss of generality, that $O_i^{(1)}$ was extracted from I_1 (and not from \bar{I}_1). Let $O_j^{(2)}$ be the object in I_2 that contains the pixel \underline{p} .
 - 2.2. **If** the object $O_i^{(1)}$ is entirely overlapped by an object $O_j^{(2)}$ we consider $O_i^{(1)}$ to be in I_2 .
 - 2.3. **Else** compute $sm(O_i^{(1)}, I_1)$ and $sm(O_i^{(1)}, I_2)$ and also $dst(O_i^{(1)}, I_1)$ and $dst(O_i^{(1)}, I_2)$, using the *contour following* algorithm ([24], chapter 9.5).
If $CD_{O_i^{(1)}} < \beta$ then the object $O_i^{(1)}$ is not an object of change.
Else ($CD_{O_i^{(1)}} \geq \beta$) $O_i^{(1)}$ is an object of change, insert $O_i^{(1)}$ into $SOL^{(out)}$.
3. **Repeat** step 2 for each object $O_i^{(2)}$ in $SOL^{(2)}$, while replacing the roles of image I_1 with I_2 .

4.2 Complexity analysis

The overall time complexity of the algorithm is almost linear in the image size, n . It is $O(n \cdot \alpha(n, n))$ where $\alpha(n, n)$ is the inverse of the Ackermann function ([22] chapter 22), which is almost a constant.

Following is the time complexity analysis of each of the three steps in the algorithm. The numbers correspond to the step numbers that appeared in the pseudo-code description.

1. The construction of each of the four *SOL* lists takes $O(n \cdot \alpha(n, n))$ operations in the worst-case, where $\alpha(n, n)$ is the inverse of the Ackermann function. Therefore, the worst-case time complexity for the creation of $SOL^{(1)}$ and $SOL^{(2)}$ is $O(n \cdot \alpha(n, n))$. Discussion of the complexity for connectivity analysis along gray-levels based technique is given in [21].
2. As shown in step 1 of the complexity analysis, the computations related to all the objects in $SOL^{(1)}$ and $SOL^{(2)}$, are linear in the total number of pixels in all the objects of both lists. Since the union of all the objects in one image is not bigger than the image size, the time complexity of all the iterations of this step is $O(n)$. Here is a detailed description of the time complexity for all stages in this part:
 - 2.1. Given a pixel p , the object that contains it can be found in $O(1)$ operations by keeping an array of n entries such that entry i points to the pixel who is the head of the class.
 - 2.2. In order to find the percentage of matched pixels between two objects, a single pass on both of them is required. This pass is linear in the number of pixels of the object. Each object is represented by a pixel that functions as a “head of class”, which is part of the Disjoint-Sets data structure. The list of all the pixels that compose an object can be found by applying the BFS ([22], chapter 23) on the image that contains the object. Hence, the worst-case complexity of this pass is also linear in the number of pixels of the object.
 - 2.3. The boundary of each object is extracted by a single pass on the boundary pixels, using the *contour following* algorithm ([24], chapter 9.5). It requires a pixel that is known to reside on the boundary. Such a pixel is attached in advance to each object, as part of the output of the blob extraction algorithm (section 2). Then, the worst-case time complexity for computing the saliency/distribution measure of the object $O_i^{(1)}$ in the image I_2 is linear in the number of boundary pixels.
3. As in part 2, the worst-case time complexity is $O(n)$.

5. Experimental Results

In this section we present three examples that illustrate different steps of the algorithm. The first example (section 5.1) presents the output of the blobs extraction algorithm and the final result from the matching analysis. In section 5.2 we demonstrate the matching analysis results of the potential object followed by the algorithm output. Section 5.3 demonstrates how the algorithm operates on two input images that were captured from a long distance, and suffer from atmospheric turbulences. A statistic tests for β parameter as a function of miss detections and false alarms are given in section 5.4.

5.1 Example I

Figure 10 presents the input of two registered gray levels images that were captured in different periods of the day by an infrared device. Image I_1 contains two objects (bus in the bottom and another vehicle in the top) that are not contained in I_2 . Image I_2 contains also two objects (vehicles in the top and in the middle) that are not contained in I_1 . The four potential ‘objects of change’ are surrounded by red circles.



Figure 10: Pair of registered gray levels input images that were captured with a lag of two hours between them. Each image contains two different objects which do not exist in the other.

The absolute differences between I_2 and I_1 , $I^{(abs)}(x, y) = |I_1(x, y) - I_2(x, y)|$ (see Figure 11), demonstrates the noise and the differences in illumination between the images in Figure 10. As seen there are many similar regions in Figure 11 that have high absolute difference values that are caused by changes in illumination or noise.

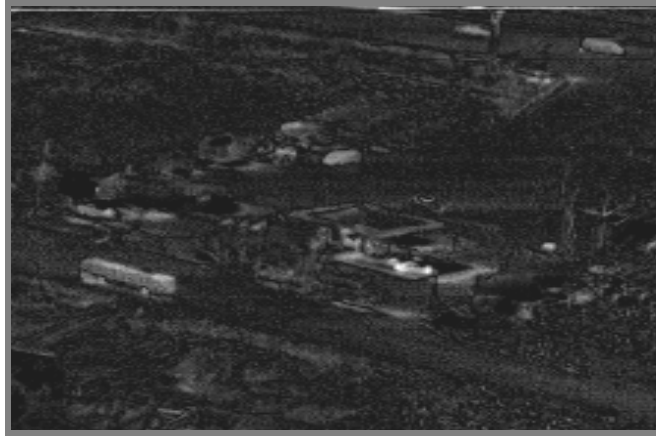


Figure 11: The absolute differences between the images I_1 and I_2 in Figure 10

The output from the application of the blob extraction algorithm is presented in Figure 12. It was applied only on the original source images I_1 and I_2 . The blobs extraction outputs from the negative images \bar{I}_1 and \bar{I}_2 are irrelevant in this example since the four 'objects of change' have already been extracted from the source images. Figure 12 displays the two lists SOL_1 and SOL_2 of significant objects and they are surrounded by red contours.

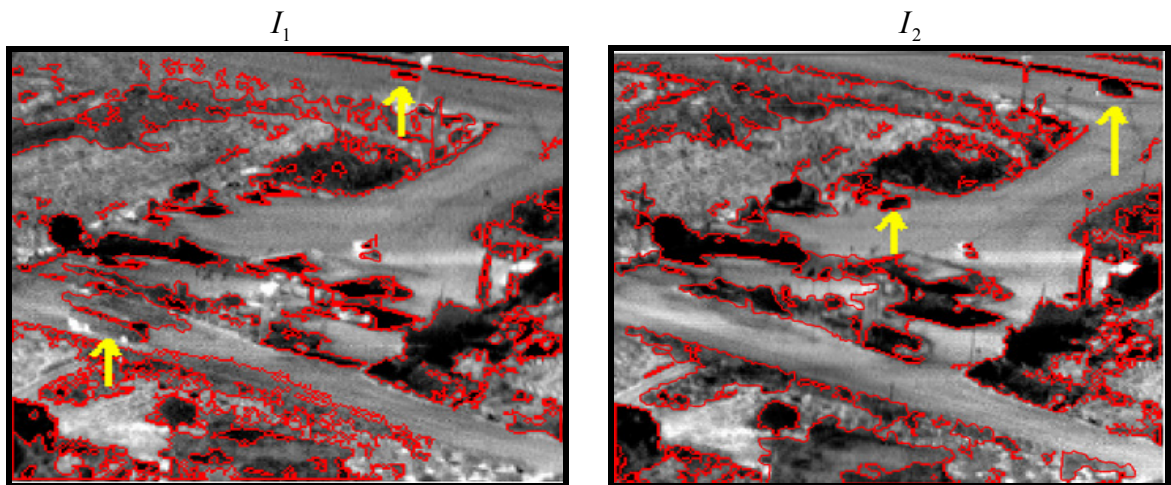


Figure 12: The blobs extraction results of the $SOLs$ list. Each blob in the $SOLs$ list is marked by red contour.

Despite the fact that several boundaries of objects (extracted by the blobs extracted step) in I_1 and I_2 are not similar (see Figure 12), the matching analysis produces similar values of $CD_{O_i^{(1)}}$ and $CD_{O_i^{(2)}}$ for each of them. Thus, they are not considered as a 'change'.

The four objects of change yields significant higher values of $CD_{O_i^{(1)}}$ and $CD_{O_i^{(2)}}$ than the rest of the objects. Thus, they were extracted as the 'objects of change'.



Figure 13: The outcome from the change detection algorithm after comparing I_1 (image a) relative to I_2 (image b) and I_2 relative to I_1 from the images in Figure 10. Each 'object of change' is surrounded by a yellow contour.

5.2 Example II

Figure 14 presents two noisy inputs of registered images that were captured in different periods of the day. Both images are poor quality, with significant differences in illumination.



Figure 14: Image I_1 contains three objects that do not exist in I_2 . Image I_2 contains one object that does not exist in I_1 . The objects of change are surrounded by red circles.

The blob extraction output for this example is presented in Figure 15. It was applied on the four images I_1 and I_2 (taken from Figure 14), \bar{I}_1 and \bar{I}_2 (their negatives) and produced the lists $SOL^{(1)}$ and $SOL^{(2)}$. The lists $SOL^{(1)}$ and $SOL^{(2)}$ are shown in Figure 15a and Figure 15b, respectively, where each object is outlined by a red curve. As shown in Figure 15a, the three 'objects of change' that were outlined in I_1 (Figure 14) were extracted by the application of the blobs extraction step and are part in the object list $SOL^{(1)}$. Similarly, the single object of change that was outlined in Figure 14 I_2 is now a part of $SOL^{(2)}$.

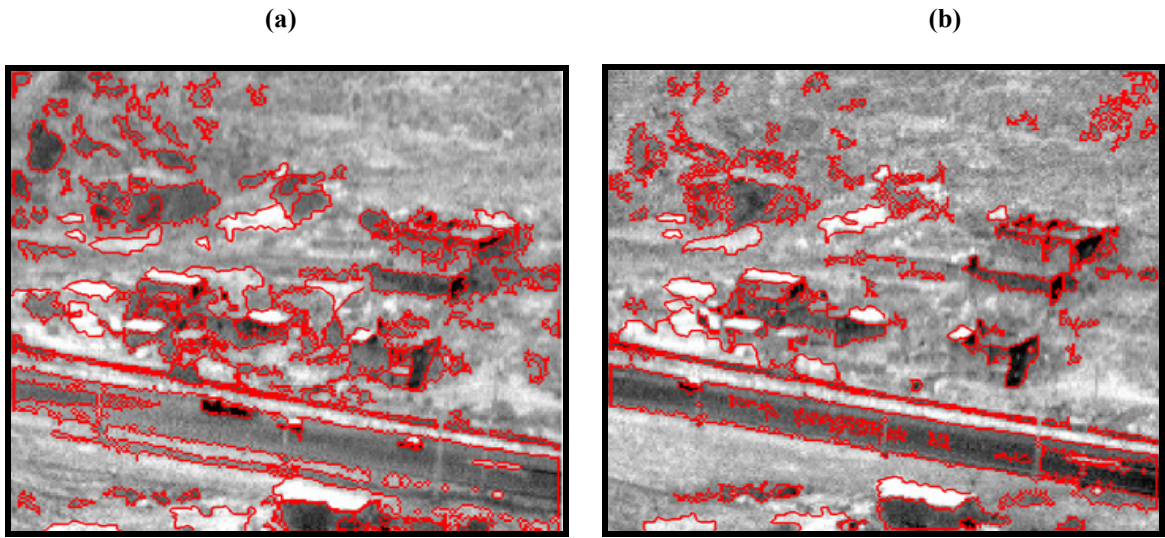


Figure 15: The $SOLs$ list of the blob extraction output. Each boundary of the extracted blobs is marked by a red contour.

The matching analysis (section 3) of the $SOLs$ above (see Figure 15) is performed. After matching the locations of the extracted blobs only few objects were found to have overlapped coordinates to declare them as matched objects. These objects were removed from the $SOLs$ list without performing the matching analysis. However, those that remained in the $SOLs$ list were analyzed in the matching phase. The matching results are shown in Figure 16, where the x -axis is the object indices remains in $SOL^{(1)}$ and in $SOL^{(2)}$. The y -axis represent the CD outcome of each object in $SOL^{(1)}$ and $SOL^{(2)}$.

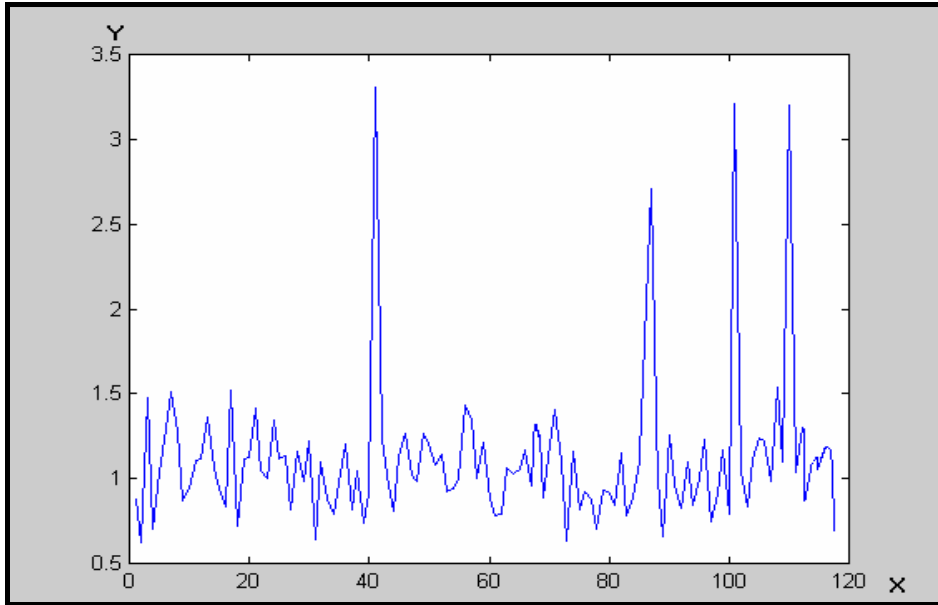


Figure 16: Illustration of the the four peaks that correspond to the four vehicles appear only in one of the two input images in Figure 14.

The output after the application of the matching phase is presented in Figure 17. The four 'objects of change', which remain in the *SOLs* list, are marked by the yellow curves. The matching analysis of the four 'objects of change' (see Figure 16) yields a significant higher CD_{o_i} value (around factor of three) than the rest of the objects.



Figure 17: (a) The final output after the comparison of I_1 relative to I_2 taken from Figure 14. (b) The output after the comparison between I_2 relative to I_1 .

5.3 Example III

Figure 18 presents two input frames, where the camera is very distant from the objects. These images suffer from turbulence effects, which reflect as local distortions of the image surfaces. Image I_1 in Figure 18 contains a single object, which is the vehicle (seen as a white blob) that is shown in I_1 and does not exist in I_2 . It is marked by a red circle. In this example, there are no 'objects of change' in I_2 that do not exist in I_1 .



Figure 18: Pair of registered gray levels input images that were captured from a very long distance in different periods of the day. Image I_1 contains one object which does not exist in I_2 .

The output from the blob extraction in this example is presented in Figure 19 by the two object lists $SOL^{(1)}$ and $SOL^{(2)}$. Each boundary of object is marked by a red curve. The 'object of change' is pointed by the yellow arrow.

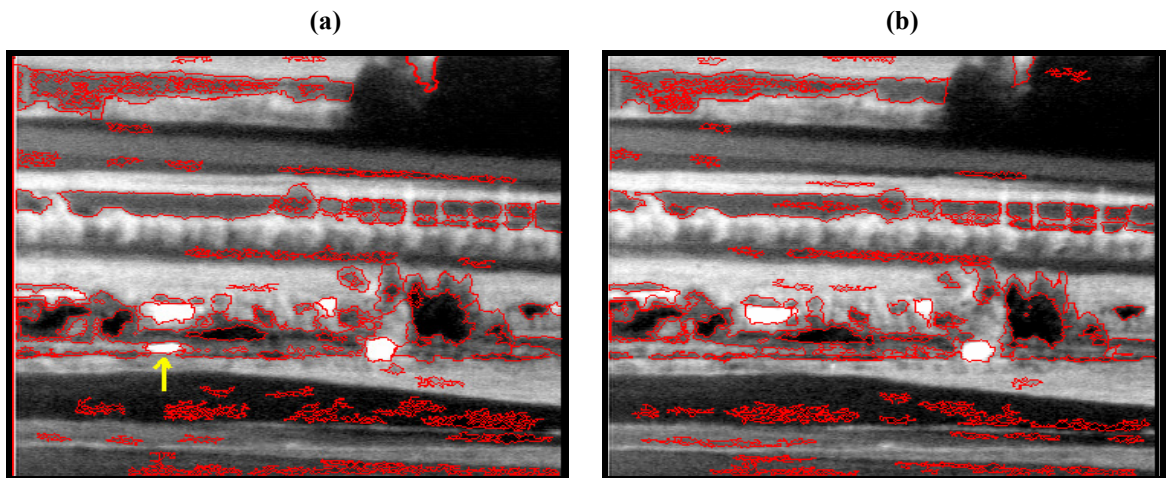


Figure 19: The red contours mark the objects boundaries of $SOL^{(1)}$ and $SOL^{(2)}$.

Figure 20 shows sixty objects that remain after removing the overlapped objects. The x -axis represents i^{th} object in $SOL^{(1)}$. The y -axis represents the $CD_{o_j^{(1)}}$ outcome for $i = 1, \dots, m_1 + n_1$.

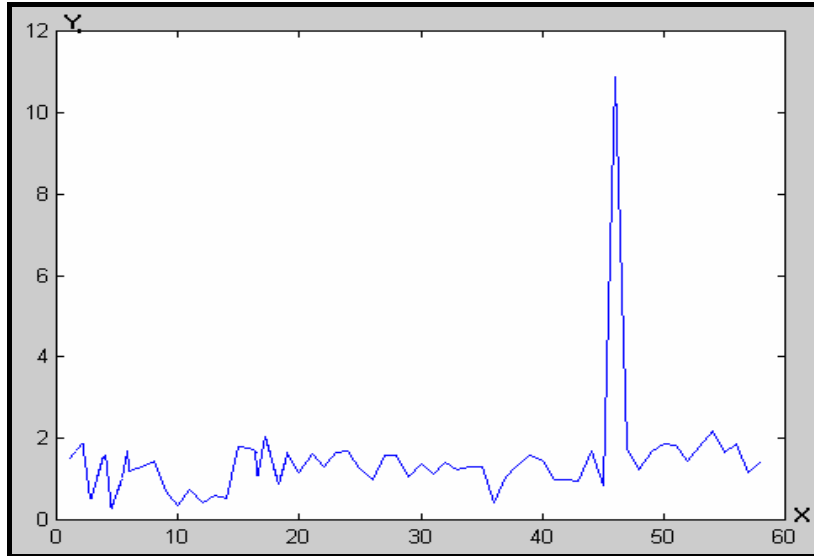


Figure 20: The significant peak corresponds to the vehicle that appears only in one of the two input images in Figure 18.

The algorithm output is presented at Figure 21. The 'object of change' that remains in the SOL list after the matching phase completed (which is the object that obtained the highest CD value, see Figure 20), are outlined by the red curve on its boundary.



Figure 21: Final output of the algorithm for the comparison of image (a) relative to (b) taken from Figure 18. No 'object of change' was found of image (b) relative to (a).

5.4 Summary

We tested the application on a data base of 112 image pairs (that is, 224 images). The total number of objects of change was 287. Minimal object size was 30 pixels. We summarized the number of detections and false alarms as a function of the β parameter in the following table:

β	Miss-detections	False Alarms
1	49	0
1.7	17	0
2	11	0
2.3 to 2.7 (optimal)	0	5
3	0	10
3.5	0	27
4 to 4.2	0	57

Table 1: A description of the miss detection and false alarms amount as a function of β parameter.

6. Summary and Discussion

In this paper we presented an efficient and robust algorithm to perform change detection between a pair of registered gray-level images under severe noise and differences in the illumination between them. The output of the algorithm is a set of connected components, where each component is an ‘object of change’, which is a significant blob that exists in only one of the two images. Our algorithm contains two main steps. The first is constituted of the blob extraction algorithm, based on connectivity analysis along gray level that extracts significant blobs in both images. In the second step, each blob from one image is searched for a corresponding blob in the other image. Since the two images are assumed to be registered, the corresponding blob is expected to reside in the same location, and to have similar size and shape as the one we try to match. Practically, a blob may have no corresponding blob in the other image, even if the blob is not an ‘object of change’. Therefore, a matching analysis was needed. For each object, which remained unmatched, a saliency and distribution of its boundary were measured in both input images. If the product of these measures is sufficiently high, than the object is output as an ‘object of change’. The time complexity of the change detection algorithm is almost linear in the image size.

Therefore, it is suitable for real-time applications. The examples in this paper demonstrate its robustness even under extreme noise and changes in the illumination.

The following are the main advantages of the proposed method, which achieved as a result of the object-oriented concept.

1. The bound detection of the change object is accurate. Therefore, our method works also for noisy input with very small ‘objects of change’ (less than 30 pixels).
2. The input images can contain several ‘objects of change’ with a considerable difference in their sizes. It stems from the fact that our method does not use a window with a pre-defined size, but directly works on the extracted blobs.
3. The detection of change is robust and insensitive to noise as long as the change is a connected component.
4. The worst-case time complexity of the algorithm is almost linear in the image size.

Appendix A: The Blob Extraction pseudo-code

Input:

I - gray-levels image.

$I^{(\text{MAG})}$ - image of weights.

t_{start} - the minimal gray-level value (default is 0). The gray-level of a pixel with gray-level $< t_{\text{start}}$ is regarded as t_{start} .

t_{stop} - the maximal gray-level value (default is $G-1$). The gray-level of a pixel with gray-level $> t_{\text{stop}}$ is regarded as t_{stop} .

Output:

SOL - a list of clusters. Each cluster is contains size and weight properties.

Local variables:

Q, Q_b - array of size G where each entry in the array contains a queue

UF - a Union-Find structure (see [22]).

Notation:

1. $I(p)$ denotes the gray-level of that pixel in the image I , for a given pixel p .
2. $w(p)$ denotes the weight of pixel p , for a given pixel p .
3. $\text{Idx}(p)$ denotes the index (ordinal number) of the pixel, for a given pixel p .
4. $\text{Weight}(\text{UF}, \text{Idx}(p))$ denotes the weight of the set that contains pixel p , for a given pixel p ,
5. For a given pixel p , $\text{Perimeter}(\text{UF}, \text{Idx}(p))$ denotes the perimeter of the set that contains pixel p .
6. We assume that a data-structure of queue is supplied, with the following operations:
QueueInsert(Q, p) - inserts the pixel p into the queue Q .
QueueDelete(Q) - returns a pixel p from the head of the queue Q .
QueueIsEmpty(Q) - returns “true” if and only if the queue Q is empty.
7. Unite(UF, i, j) uses the Union-Find data-structure. Any operation of the type $X(\text{UF}, i, j)$ implies the use of the Find(i) and Find(j) for getting the head of the sets to which items i and j belongs.

The algorithm:

1. Construct the array of G queues such that the queue $Q[i]$ contains the pixels of the image I whose gray-levels is i . The order inside the queue is not important. The size of queue number i is the number of pixels with gray-level i .

The construction of the array of queues:

For each pixel of the image I do QueueInsert($Q[I(p)],p$)

1. Initialize the array of G queues, Q_b .
2. Initialize the Union-Find structure, UF.
3. Initialize the list of significant object (SOL) to an empty list.
4. For $t \leftarrow t_{start}$ to t_{stop} do
 - 4.1. while not QueueIsEmpty($Q[t]$) do
 - 4.1.1. $p \leftarrow$ QueueDelete($Q[t]$)
 - 4.1.2. if Border[Idx(p)] $>$ t then
 - QueueInsert($Q_b[Idx(p)],$ Border[Idx(p)])
 - $Perimeter(UF,Idx(p)) \leftarrow Perimeter(UF,Idx(p))+1$
 - $Weight(UF,Idx(p)) \leftarrow Weight(UF,Idx(p))+w(Idx(p))$
 - 4.1.3. for each one of the four nearest neighbor, p_{nn} , do
 - if $I(p_{nn}) \leq t$ then Unite(UF,Idx(p_{nn}),Idx(p))
 - 4.2. While not QueueIsEmpty($Q_b[t]$) do
 - 4.2.1. $p \leftarrow$ QueueDelete($Q_b[t]$)
 - 4.2.2. $Perimeter(UF,Idx(p)) \leftarrow Perimeter(UF,Idx(p))-1$
 - 4.2.3. $Weight(UF,Idx(p_{nn})) \leftarrow Weight(UF,Idx(p))-w(Idx(p))$
 - 4.3. UpdateSOL as follows. ForEach new component C_{ufl} that was updated in this iteration is inserted to the SOL if:
 - 4.3.1. $w(C_{ufl}) > MIN_WEIGHT$ (a pre-defined value) and $size(C_{ufl}) > MIN_SIZE$ (a pre-defined value), and
 - 4.3.2. for each son (component contained in C_{ufl}) C_{sol} of C_{ufl} from the SOL the following is satisfied:
 - 4.3.3.1 $w(C_{ufl}) > w(C_{sol})$ or $size(C_{ufl}) > \alpha \cdot size(C_{sol})$

We experimentally set the parameters to the values: $MIN_WEIGHT=60$, $MIN_SIZE=30$ (pixels), and $\alpha=3$. The results are not sensitive to the exact selection of parameters.

The complexity of the algorithm is as the complexity of the Union-Find data-structure, that is, $O(n \cdot \alpha(n,n))$ where $\alpha(n,n)$ is the inverse of the Ackermann function ([22], chapter 22).

References

- [1] C. W. Fu and S. Chang, "A motion estimation algorithm under time varying illumination case," *Pattern Recognit. Lett.*, vol. 10, pp. 195–199, 1989.
- [2] L. H. Chen and S. Chang, "A video tracking system with adaptive predictors," *Pattern Recognit.*, vol. 25, pp. 1171–1180, 1992.
- [3] S. H. Lai and S. Chang, "Estimation of 3-D translational motion parameters via Hadamard transform," *Pattern Recognit. Lett.*, vol. 8, pp. 341–345, 1988.
- [4] Jianbo Shi and Jitendra Malik "Motion Segmentation and Tracking Using Normalized Cuts" *University of California, Berkeley Report No. UCB/CSD-97-962* June 1997.
- [5] Thomas Meier and King N. Ngan. "Video Segmentation for Content-Based Coding". *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 9, No. 8, December 1999.
- [6] H. H. Nagel, "Recent advances in motion interpretation based on image sequences," *ICASSP*, pp. 1179–1186, 1982.
- [7] C. Wren, A. Azarbaygani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 780–785, July 1997.
- [8] D. Le Gall, "MPEG: A video compression standard for multimedia applications," *Commun. ACM*, vol. 34, no. 4, pp. 47–58, 1991.
- [9] Skifstad Kurt and Jain Ramesh, "Illumination Independent Change Detection for Real World Image Sequences", *Computer Vision, Graphics, and Image Processing*, Vol. 46, pp. 387-399, 1989.
- [10] Til Aach, Andre Kaup, Rudolf Mester, "Statistical model-based change detection in moving video", *Signal Processing*, Vol. 31, pp. 165-180, 1993.
- [11] Sze-Chu Liu, Chang-Wu Fu, and Shyang Chang, "Statistical Change Detection with Moments under Time-Varying Illumination", *IEEE Transactions On Image Processing*, Vol. 7, No. 9, September 1998.
- [12] H.J. Zhang, A. Kankanhalli, and S.W. Smoliar, "Automatic Partitioning of Full-Motion Video", *ACM/Springer Multimedia Systems*, Vol. 1, No. 1, pp. 10-28, 1993.
- [13] N.H. Nagel, "Formulation of an Object Concept by Analysis of Systematic Time Variation in the Optically Perceptible Environment", *Computer Graphics and Image Processing*, Vol 7. pp. 149-194, 1978.

- [14] R. Jain, D. Militzer, and H. H. Nagel, "Separation non-stationary from stationary scene component in a sequence of real world TV images, in *proceeding, IJCAI 1977*, pp. 612-618.
- [15] Hsu Y.Z., Nagel H. H, Rekers G., "New likelihood test methods for change detection in image sequences", *computer Vision Graphics Image Processing*, vol. 26, pp. 73-106, 1984.
- [16] P. Monasse, F. Guichard, "Fast Computation of a Contrast-Invariant Image Representation", *IEEE Trans. on Image Processing*, Vol. 9, No. 5, 860-872, 2000.
- [17] N. Nandhakumar, Jonathan D. Michel, D. Gregory Arnold, George A. Tsihrintzis, and Vince elten, "Robust Thermophysics-Based Interpretation of Radiometrically Uncalibrated IR Images for ATR and Site Change Detection". *IEEE Transactions on image processing*, Vol 6, No. 1, janouary 1997.
- [18] S. J. Gee and A. M. Newman, "RADIUS: Automating image analysis through model-supported exploitation," in *Proc. DARPA ImageUnderstanding Workshop*, Apr. 1993, pp. 185–196.
- [19] M. J. Gauder, V. J. Velten, L. A. Westerkamp, J. Mundy, and D. Forsyth, "Thermal invariants for infrared target recognition," *ATR Syst. Technol. Conf.*, 1993.
- [20] Mark J. Carlotto, "Detection and Analysis of Change in Remotely Sensed Imagery with Application to Wide Area Surveillance", *IEEE Transactions on image processing*, Vol 6, No. 1, janouary 1997.
- [21] Pikaz Arie, "Connectivity Analysis in digital gray levels images and its applications", *Ph.D thesis, Tel-Aviv University 1999*, Chapter 10.
- [22] Cormen Thomas, Leiserson Charles, Rivest Ronald, *Introduction to ALGORITHMS*, The MIT Press, 1990, chapter 22.
- [23] Lorenzo Bruzzone, and Diego Fernández Prieto "An Adaptive Semiparametric and Context-Based Approach to Unsupervised Change Detection in Multitemporal Remote-Sensing Images", *IEEE Transactions on image processing* Vol. 11, No.4, April 2002.
- [24] Jain Anil K., *Fundamentals of Digital Image Processing*, Prentice-Hall, 1989.