Locating IED in SAR Imagery Using Change Detection Methods

Alireza Nazari

New Mexico State University

Abstract

In this project, I implemented a paper which presents several methods for change detection in a pair of multi-look synthetic aperture radar (SAR). This paper applied those methods to detect mines while we use them to locate any change and eventually to detect any potential Improvised Explosive Device. Two groups of methods are compared. First, simple approaches with little computational complexity, differencing, Euclidean distance, and image ratioing are implemented although we do not expect them to show a good performance in presence of high speckle noise. Then Wiener prediction-based method, Subspace Method and Mahalanobis distance are implemented which incorporate second order statistic calculations in making a change decision in efforts to mitigate false alarms arising from the speckle noise, misregistration errors, and nonlinear variations in SAR images. We apply these methods on a pair of images which are simulated output of our Ground Penetrating Radar system in order to find any change and then calculate Receiver Operation Characteristic (ROC) to compare them. More bright changes are more important for human user since we do not need automatic alarm. The purpose of this project is applying this detector parallel to our SAR system and help solders to have safer sweep.

Introduction

Multitemporal change detection aims at discerning areas of change on digital images between two or more times of taking picture from the same scene. Changes detection in Synthetic Aperture Radar (SAR) images can have applications in land cover monitoring [1], which principally consists in detecting the seasonal vegetation changes; land use monitoring [2], which is the characterization of changes mostly due to human activities; mine detection [3]; and damage detection. The main obstacle of robust change detection of SAR images is speckle noise. Speckle noise is a granular noise that inherently exists in and degrades the quality of the active radar and SAR images. It results from random fluctuations in the return signal from an object that is no bigger than a single image-processing element. It increases the mean grey level of a local area [4].

There are several methods of change detection in SAR images, classification based methods [5], kullback-leiber divergence methods [6] and computer vision feature based methods which are mostly used for satellite SAR images, land monitoring and land use monitoring but to detect small spatial changes such as mine, simple statistical approaches typically are used [3]. Despite the diversity of applications, change detection researchers employ many common processing steps and core algorithms. Apparent intensity changes at a pixel resulting from camera motion alone are virtually never desired to be detected as real changes. Hence, a necessary preprocessing step for all change detection algorithms are accurate image registration, the alignment of several images into the same coordinate frame [7]. When the scenes of interest are mostly rigid in nature and the camera motion is small, registration can often be performed using low-dimensional spatial transformations such as similarity, affine, or projective transformations. This estimation problem has been well studied, and several excellent surveys [8,9], and software implementations (e.g., the Insight toolkit [10]) are available, so we do not detail registration algorithms here. Rather, we note some of the issues that are important from a change detection standpoint.

As a part of a project by Army Advanced Computer Architecture High Performance Computing Research Center (ACHPRC), we want to add a fast IED detector to a forward-looking ground penetrating radar. An improvised explosive device (IED), also known as a roadside bomb, is a homemade bomb constructed and deployed in ways other than in conventional military action. It may be constructed of conventional military explosives, such as an artillery round, attached to a detonating mechanism. IED recognition faces with a big problem; there is no specific feature for IED SAR image because every buried object can be an IED. So we have to use change detection methods to find any change between two times of road sweep by solders and every change is potentially an IED.

We apply change detection methods which are typically used for mine detection, to detect Improvised Explosive Devices. These methods have two characteristic; they are suitable for small change area in comparison of image area and speckle noise and since we need real time output (implementation on FPGA) to use the device in battlefield they are computationally light. Since we still don’t have the output of real system, we have to simulated version. We simulated the output of our ground penetrating radar (SAR radar) with high amount of speckle noise. Our test images have some added object and are correlated with pretty high speckle noise. (Fig 1)

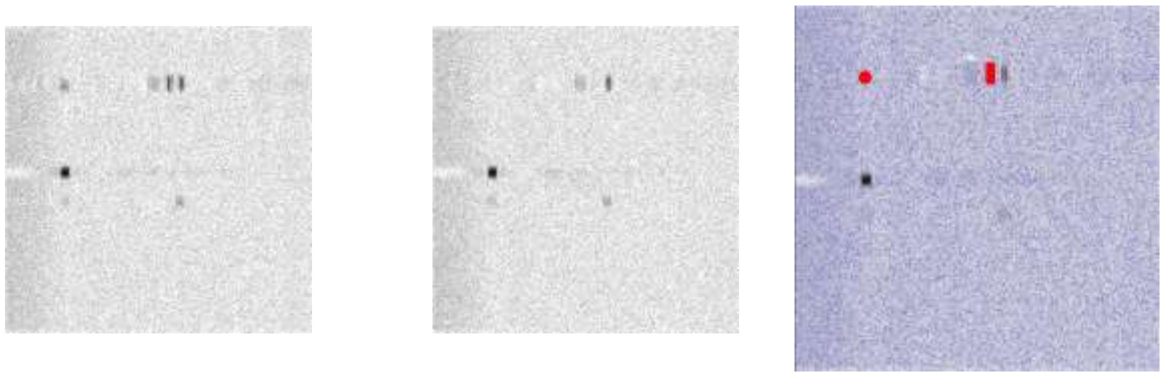


Fig 1-Input Images

Project Description

We implemented and compared two sorts of approaches, first, simple pixel based approaches such as Difference, Euclidean distance, and image ratioing change detection methods which are not expected to have good performance in presence of high correlated speckle noise. Second, we will implement second order statistical approaches, Mahalanobis distance-based change detection, subspace projection and Wiener prediction-based change detection.

Difference, Euclidean distance, and image ratioing change detection methods

The Simplest method is subtracting pixel by pixel and using a threshold, but since the image has speckle noise, it doesn’t show good performance. One possible improvement is using bigger windows to reduce the effect of noise by taking L2 norm of window and comparing it with the reference image.

$$e_E(i,j) = \left\| \mathbf{y}_{ij} - \mathbf{x}_{ij} \right\|$$

Where x and y represent image and its reference image respectively. With this Euclidian distance of two windows we are taking neighborhood difference to the account and this might be more helpful in noise rejection. Closely related to the Euclidean distance metric for change detection is image ratioing. In many change detection applications ratioing proved more robust to illumination effects than simple differencing. It is implemented as follows:

$$e_R(i,j) = \frac{y_{ij}}{x_{ij}}$$

Mahalanobis distance-based change detection

To reduce noise sensitivity we can use statistical features of texture to comparison. Therefore, the Mahalanobis distance measure is used to detect changes in SAR imagery. In this change detection application, we obtain an error (change) image by computing the Mahalanobis distance between x and y as follows.

$$e_{MD}(i,j) = \sqrt{(\mathbf{x}_{ij} - \mathbf{y}_{ij})^T \mathbf{C}_X^{-1} (\mathbf{x}_{ij} - \mathbf{y}_{ij})}$$

Where C is covariance matrix. By considering the effects of other pixels in making a change decision with the inclusion of second order statistics, the Mahalanobis distance method is expected to reduce false alarms.

Subspace Projection-based change detection

In order to apply subspace projection to change detection a subspace must be defined for either the reference or test image. We computed the covariance of a sample from fx and its eigenvectors and eigenvalues as follows:

$$\mathbf{C}_X = (\mathbf{X} - \boldsymbol{\mu}_X)(\mathbf{X} - \boldsymbol{\mu}_X)^T$$

where X is a matrix of pixels whose columns represent a block of pixels from fx arranged as described in Section 6, having mean $\mu_{x=}1X_{NxN}.1_{NxN}$ is a square matrix of size NxN whose elements are 1/N, where N is the number of columns of X. We define the subspace of the reference data, which we can express using eigen-decomposition in terms of its eigenvectors, V, and eigenvalues, Λ

$$\mathbf{C}_X = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T$$

We truncate the number of eigenvectors in V, denoted by $\check{V}$, to develop a subspace projection operator, P_x

$$\mathbf{P}_X = \tilde{\mathbf{V}}\tilde{\mathbf{V}}^T$$

The projection of the test image onto the subspace of the reference image will provide a measure of how much of the test sample is represented by the reference image. Therefore, by computing the squared difference between the test image and its projection onto the subspace of the reference image we obtain an estimate of the difference between the two images,

$$e_{SP}(i,j) = \sqrt{[\mathbf{y}_{ij}^T (\mathbf{I} - \mathbf{P}_X)\mathbf{y}_{ij}]}$$

We evaluated the effects of various levels of truncation and display the best results achieved. In our implementations we include the CX-1 term in the subspace projection error term as follows:

$$e'_{SP}(i,j) = \sqrt{\mathbf{y}_{ij}^T(\mathbf{I}-\mathbf{P}_X)^T\mathbf{C}_X^{-1}(\mathbf{I}-\mathbf{P}_X)\mathbf{y}_{ij}}$$

We expect it will play a similar role as it does in the Mahalanobis prediction and further diminish false alarms by suppressing the background clutter.

Wiener prediction-based change detection

The final method which we will implement is a Wiener prediction-based approach to reduce the false alarm caused by speckle noise. The Wiener filter is the linear minimum mean-squared error estimator for second-order stationary data. Consider the signal $\hat{\mathbf{y}}_{ij} = \mathbf{W}\mathbf{x}_{ij}$, the goal of Wiener filtering is to find W which minimize the e_W.

$$e_W(i,j) = \left\|\mathbf{y}_{ij} - \hat{\mathbf{y}}_{ij}\right\|$$

where y represents a desired signal. In the case of change detection y is taken from the test image, f_Y , which contains changes as compared to the reference image, $f_{X.}$. If the linear minimum mean-squared error estimator of Y satisfies the orthogonality condition, the following expressions hold:

$$E\{(\mathbf{Y}-\mathbf{W}\mathbf{X})\mathbf{X}^T\} = 0$$
$$E\{\mathbf{Y}\mathbf{X}^T\} - E\{\mathbf{W}\mathbf{X}\mathbf{X}^T\} = 0$$
$$\mathbf{R}_{YX} - \mathbf{W}\mathbf{R}_{XX} = 0$$

whererepresents the expectation operator, X is a matrix of pixels whose columns represent a block of pixels from f_X, Y is a matrix of pixels from f_Y whose columns represent a block of pixels at the same locations as those in X, R_{YX} and is the cross correlation matrix of X and Y, and R_{XX}^{-1} is the auto correlation matrix of X.

$$\mathbf{W} = \mathbf{R}_{YX}\mathbf{R}_{XX}^{-1}$$

Therefore, we have:

$$\hat{\mathbf{y}}_{ij} = \mathbf{R}_{YX}\mathbf{R}_{XX}^{-1}\mathbf{x}_{ij}$$

Where $R_{XY=}YX^T$ and $R^{-1}{}_{XX=}(XX^T)^{-1}$. So the error can be computed as follows

$$e_W(i,j) = \left\| \mathbf{y}_{ij} - \hat{\mathbf{y}}_{ij} \right\|$$

In a modified implementation of the Wiener prediction-based change detection method, we insert a normalization (or whitening) term, C_X^{-1}, into the error equation as follows:

$$e'_W(i,j) = \sqrt{(\mathbf{y}_{ij} - \hat{\mathbf{y}}_{ij})^T \mathbf{C}_X^{-1} (\mathbf{y}_{ij} - \hat{\mathbf{y}}_{ij})}$$

We expect it will play a similar role as it does in the Mahalanobis prediction and further diminish false alarms by suppressing the background clutter.

Where

$$\hat{\mathbf{y}}_{ij} = \mathbf{R}_{YX}\mathbf{R}_{XX}^{-1}\mathbf{x}_{ij}$$

And R is cross correlation.

Implementation consideration

There are two methods for dealing with blocks, as illustrated in Fig. 2. One option is to apply the decision reached at to all the pixels in the block [Fig. 2(a)], in which case the blocks do not overlap, the change mask is coarse, and block artifacts are likely. The other option is to apply the decision only to pixel [Fig. 3(b)], in which case the blocks can overlap and there are fewer artifacts; however, this option is computationally more expensive.

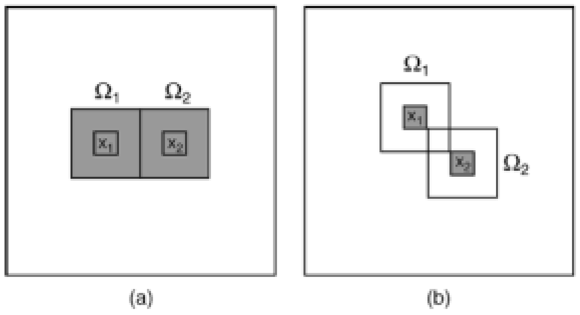


Fig 2-Overlapping methods

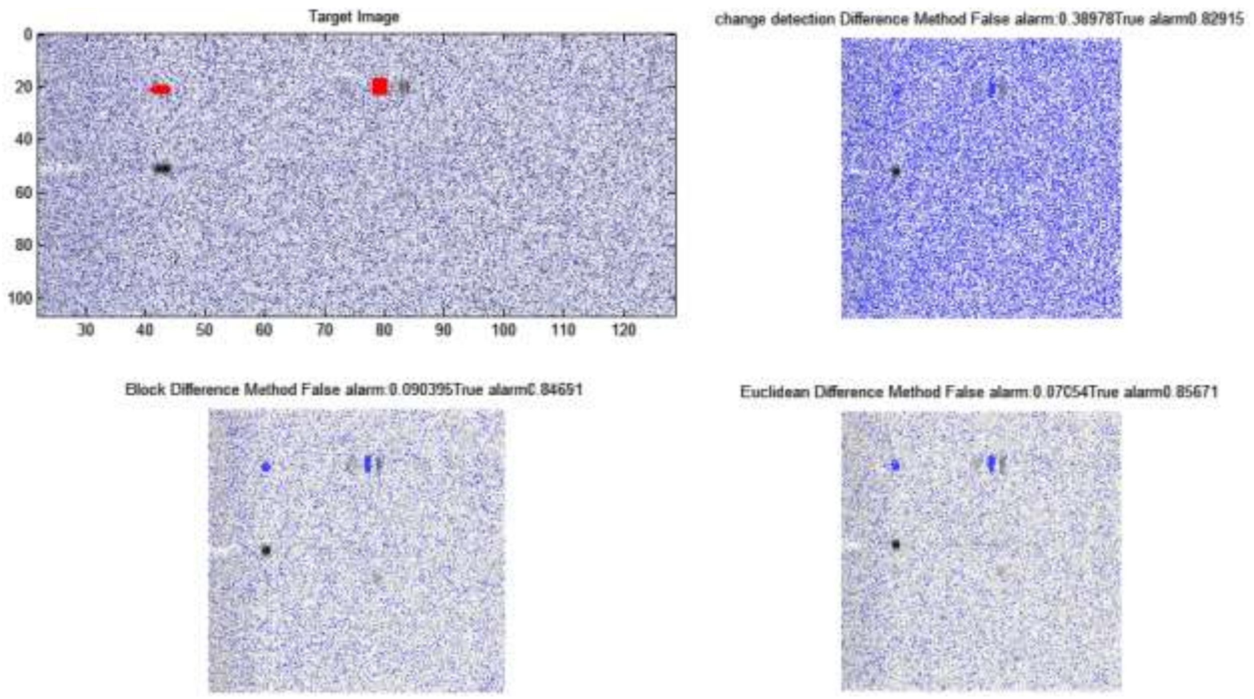
In difference-based methods we used sliding window around one pixel with 50 percent overlap but in implementation of Mahalanobis distance, subspace projection-based, and the Wiener filter-based change detection methods, we constructed our data matrices and vectors locally in order to compute the local statistics and generate a change image. We generate X, Y, x, and y using dual concentric sliding windows. X and Y are matrices of pixels contained within an outer moving window (size mxm), whose columns are obtained from overlapping smaller blocks (x1, x2, …,x_M each of size nxn) within the outer windows of the reference image and test image, respectively, as demonstrated in Fig. 3 for X where X=[$x_{1,..,}x_n$] . Accordingly, the dimensions of X and Y will be NxM where N=n^2 and M=m^2. Note, for clarity the windows that generate the columns of X and Y are not shown as overlapping in Fig. 3. However, in actual implementation the X and Y matrices are constructed from overlapping windows within X and

Y. x and y, as shown in Fig. 1, are Nx1 vectors composed of pixels within the inner window (size n x n) of the reference image,, and test image, , respectively. In this local implementation we must compute the necessary statistics at each iteration as we move pixel by pixel through the image, thereby generating new X and Y matrices and x and y vectors.

Results

All the methods presented in the paper are implemented and tested with 40 different threshold points. The output is scaled between zero and one but since there are different critical bands (the point that detection rate changes fast), 30 points are collected in that band with an equal steps. IEDs have more 1 square meter area (20 pixels). Both of test and reference images are 900*900 pixels. Ground truth was provided which indicates the approximate center location of the known mines; we defined the mine area as an area slightly bigger than the average mine, centered at the locations indicated by ground truth information. To determine the performance, a hit was tabulated if at least a single pixel within the mine area surpassed the threshold. Any pixel outside of a mine area which surpassed the threshold was counted as a false alarm (Fig 3).

As it’s explained in implementation consideration part, there are two different window sizes are used. The bigger is 13X13 (where it is applicable) and the small window is 3X3. The result of window is only considered as the result for detection of center (not the whole block).



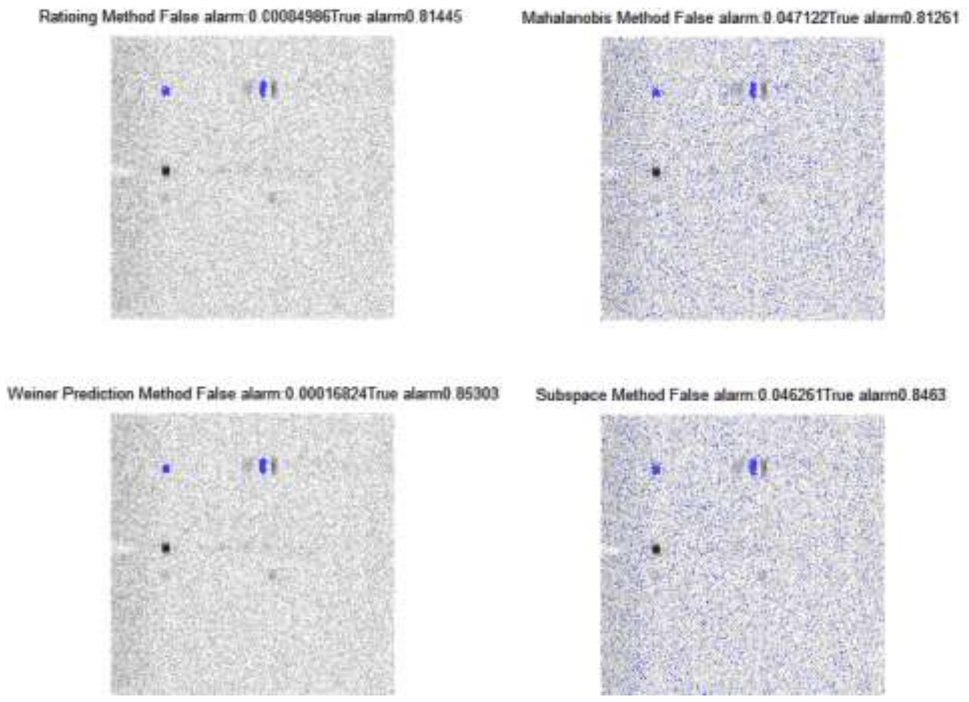


Fig 3- Results of implementation of proposed detection methods

This point should be noticed that we are aware of this fact that in real world the image pair needs image registration. Here we assume that we are applying the algorithm after registration because all the images have perfect spatial and temporal stamp which helps to the registration algorithm just before this block. Since the input images are generated data, the noise is purely speckle and there is no device or beam error in the image, the change detection methods performance was not different in different places of image. So the biggest problem was speckle noise which reduced more by methods which use C_x (Fig 4).

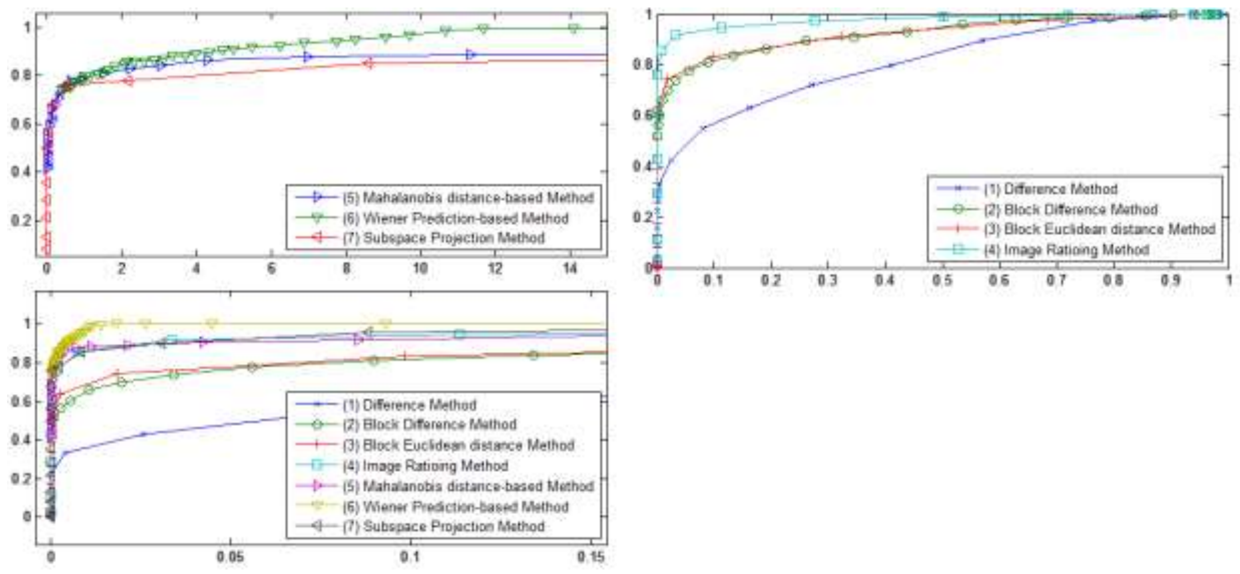


Fig 4- ROC plots of 40 different threshold

Conclusion

We tested the algorithms on a limited set of data so it is not prudent to make broad conclusions. However same as the paper, we showed that in general the more complex algorithms exhibited superior performance as compared to the simple methods; the exception seems to be the ratio method. The ratio method, although a simple implementation, has performance which is competitive with the more complex change detection methods. Implementation shows us that Wiener prediction method is the absolute winner. Results indicate that taking into account the local spatial and statistical properties of the image improves the change detection performance. Local characteristics are used in computing the inverse covariance matrix, resulting in fewer false alarms as compared to the simple change detection methods.

References

1) "*Mapping and monitoring land cover in Corumbiara area, Brazilian Amazônia, using JERS-1 SAR multitemporal data", *Almeida-Filho, R. ; Kuplich, T.M. ; de Freitas, R.M. Geoscience and Remote Sensing Symposium, 2007. IGARSS 2007. IEEE International, pp. 3370 - 3373,23-28 July 2007.

2) *"Urban Land Cover in Rome, Italy, monitored by single-parameter multi-temporal SAR images"*, Del Frate, F. ;Pacifici, F. ; Solimini, D. Urban Remote Sensing Joint Event, pp. 1-5,11-13 April 2007.

3) *"Wiener prediction-based change detection for locating mines in multilook SAR imagery"*, Nasrabadi, N.M. Geoscience and Remote Sensing Symposium,2009 IEEE International IGARSS 2009, pp. II-113 - II-116, 12-17 July 2009

4) *"Speckle Reduction of Synthetic Aperture Radar Images Based on Fuzzy Logic"*, Cheng Hua and Tian Jinwen, First International Workshop on Education Technology and Computer Science, Wuhan, Hubei, China, March 07–08 2009. **1**. pp. 933–937. July 2009.

5) *"Unsupervised Change Detection in SAR Image using Graph Cuts", Keming Chen Chunlei Huo"*, Zhixin Zhou, Hanqing Lu, Geoscience and Remote Sensing Symposium, IGARSS 2008. IEEE International, pp. III - 1162 - III – 1165, 7-11 July 2008.

6) *"Change detection on SAR images by using a parametric estimation of the Kullback-Leibler divergence"*, Inglada, J.,Geoscience and Remote Sensing Symposium, 2003. IGARSS '03. Proceedings, 2003 IEEE International, pp. 4104 - 4106, 21-25 July 2003.

7) *"Image change detection algorithms: a systematic survey",* R. J. Radke, S. Andra, O. Al-Kofahi and B. Roysam, IEEE Trans. Neural Networks, vol. 14, no. 3, 2005, pp. 294-307.

8) *"Asurvey of image registration techniques"*, L. G. BrownACMComput. Surv., vol. 24, no. 4, 1992.

9) *"Image registration methods: A survey"*, B. Zitová and J. Flusser, Image Vis. Comput., vol. 21, pp. 977–1000, 2003.

10) *"The ITK Software Guide:The Insight Segmentation and Registration Toolkit"*, L. Ibáñez, W. Schroeder, L. Ng, and J. Cates, 1.4 ed: Kitware, Inc.,2003.

Appendix A: Matlab Codes

```
function [target_Difference_mask target_Difference] =
change_detection_Difference(picture1, picture2, threshold)
%==========================================================================
% function [target_Difference_mask target] =
% change_detection_Difference(picture1, picture2, threshold)
%
% Description: This function computes Difference Method
%
%
% Input Arguments:
% Name: picture1, picture2
% Type: 2D matrix image
% Description: test image and refrence image
%
% Name: threshold
% Type: float
% Description: threshold
%
%
% Output Arguments:
% Name: target
% Type: 3D matrix image
% Description: Output with blue targets
%
% Name: target_mask
% Type: 2D boolean matrix image
% Description: 2D boolean matrix of targets
% References:
%
%--------------------------------------------------------------------------
% Notes:
%
%--------------------------------------------------------------------------
% Author: Alireza Nazari
%
% Creation Date: Oct 2012
%
%--------------------------------------------------------------------------
% Revision History:
%
%==========================================================================
D = abs(picture1-picture2);
h = fspecial('gaussian');
e = imfilter(D,h);
e  =  1- e;
[m n] = size(e);
target_Difference = zeros(m, n,3);
target_Difference_mask = zeros(m, n);
for ii = 1:m
    for jj = 2:n
        if e(ii,jj) > threshold
            target_Difference(ii,jj,1) = picture2(ii,jj);
            target_Difference(ii,jj,2) = picture2(ii,jj);
            target_Difference(ii,jj,3) = picture2(ii,jj);
        else
            target_Difference_mask(ii,jj) = 1;
            target_Difference(ii,jj,1) = .25;
            target_Difference(ii,jj,2) = .25;
            target_Difference(ii,jj,3) = 1;
        end

    end
end

function [target_Difference_mask target] =
change_detection_Block_Difference(picture1, picture2, threshold)
%==========================================================================
% function [target_Difference_mask target] =
% change_detection_Block_Difference(picture1, picture2, threshold)
%
% Description: This function computes Block Difference Method
%
%
% Input Arguments:
% Name: picture1, picture2
% Type: 2D matrix image
% Description: test image and refrence image
%
% Name: threshold
% Type: float
% Description: threshold
%
%
% Output Arguments:
% Name: target
% Type: 3D matrix image
% Description: Output with blue targets
%
% Name: target_mask
% Type: 2D boolean matrix image
% Description: 2D boolean matrix of targets
% References:
%
%--------------------------------------------------------------------------
% Notes:
%
%--------------------------------------------------------------------------
% Author: Alireza Nazari
%
% Creation Date: Oct 2012
%
%--------------------------------------------------------------------------
% Revision History:
%
%==========================================================================

block_size = [3 3];     % Must be odd!
m_block = block_size(1);
n_block = block_size(2);
[m n] = size(picture1);
```

```
target = zeros(m, n,3);
target_Difference_mask = zeros(m, n);
for ii = 1+(m_block-1)/2 : m-(m_block-1)/2
    for jj = 1+(n_block-1)/2 : n-(n_block-1)/2
        Block_1 = picture1(ii-(m_block-1)/2 : ii+(m_block-1)/2, jj-(n_block-
1)/2 : jj+(n_block-1)/2);
        Block_2 = picture2(ii-(m_block-1)/2 : ii+(m_block-1)/2, jj-(n_block-
1)/2 : jj+(n_block-1)/2);
        D = abs(Block_1 - Block_2);
        D =  mean(D(:));
        if D < threshold
            target(ii,jj,1) = picture2(ii,jj);
            target(ii,jj,2) = picture2(ii,jj);
            target(ii,jj,3) = picture2(ii,jj);
        else
            target_Difference_mask(ii,jj) = 1;
            target(ii,jj,1) = .25;
            target(ii,jj,2) = .25;
            target(ii,jj,3) = 1;
        end
    end
end

function [target_mask target] = change_detection_Mahalanobis(picture1,
picture2, threshold,Cx)

%==========================================================================
% function [target_mask target] = change_detection_Mahalanobis(picture1,
% picture2, threshold,Cx)
%
% Description: This function computes Mahalanobis distance-based Method
%
%
% Input Arguments:
% Name: picture1, picture2
% Type: 2D matrix image
% Description: test image and refrence image
%
% Name: threshold
% Type: float
% Description: threshold
%
% Name: Cx
% Type: 2D matrix
% Description: Covariance of 3*3 windows
%
% Output Arguments:
% Name: target
% Type: 3D matrix image
% Description: Output with blue targets
%
% Name: target_mask
% Type: 2D boolean matrix image
% Description: 2D boolean matrix of targets
% References:
%
```

```
%-------------------------------------------------------------------------
% Notes:
%
%-------------------------------------------------------------------------
% Author: Alireza Nazari
%
% Creation Date: Oct 2012
%
%-------------------------------------------------------------------------
% Revision History:
%
%=========================================================================

block_size = [3 3];     % Must be odd!
m_block = block_size(1);
n_block = block_size(2);
[m n] = size(picture1);
target = zeros(m, n,3);
target_mask = zeros(m, n);
for ii = 1+(m_block-1)/2 : m-(m_block-1)/2
    for jj = 1+(n_block-1)/2 : n-(n_block-1)/2
        Block_1 = picture1(ii-(m_block-1)/2 : ii+(m_block-1)/2, jj-(n_block-
1)/2 : jj+(n_block-1)/2);
        Block_2 = picture2(ii-(m_block-1)/2 : ii+(m_block-1)/2, jj-(n_block-
1)/2 : jj+(n_block-1)/2);
        D1 = Block_1(:);
        D2 = Block_2(:);
        D = D1 - D2;
        C = cov(D1);
        MD = ((D' * inv(C)) * D)/633.91;
        if MD < threshold
            target(ii,jj,1) = picture2(ii,jj);
            target(ii,jj,2) = picture2(ii,jj);
            target(ii,jj,3) = picture2(ii,jj);
        else
            target_mask(ii,jj) = 1;
            target(ii,jj,1) = .25;
            target(ii,jj,2) = .25;
            target(ii,jj,3) = 1;
        end
    end
end

function [target_mask target] = change_detection_Ratioing_2(picture1,
picture2, threshold)
% Image Ratioing Method
%=========================================================================
% function [target_mask target] = change_detection_Ratioing(picture1,
% picture2, threshold)
%
% Description: This function computes Ratioing Method
%
%
% Input Arguments:
% Name: picture1, picture2
% Type: 2D matrix image
% Description: test image and refrence image
%
% Name: threshold
% Type: float
% Description: threshold
%
%
% Output Arguments:
% Name: target
% Type: 3D matrix image
% Description: Output with blue targets
%
% Name: target_mask
% Type: 2D boolean matrix image
% Description: 2D boolean matrix of targets
% References:
%
%--------------------------------------------------------------------------
% Notes:
%%
%--------------------------------------------------------------------------
% Author: Alireza Nazari
%
% Creation Date: Oct 2012
%
%--------------------------------------------------------------------------
% Revision History:
%
%==========================================================================
block_size = [3 3];     % Must be odd!
m_block = block_size(1);
n_block = block_size(2);

[m n] = size(picture1);
target(:,:,1) = picture1;
target(:,:,2) = picture1;
target(:,:,3) = picture1;
target_mask = zeros(m, n);
for ii = 1+(m_block-1)/2 : m-(m_block-1)/2
    for jj = 1+(n_block-1)/2 : n-(n_block-1)/2
        Block_1 = picture1(ii-(m_block-1)/2 : ii+(m_block-1)/2, jj-(n_block-
1)/2 : jj+(n_block-1)/2);
        Block_2 = picture2(ii-(m_block-1)/2 : ii+(m_block-1)/2, jj-(n_block-
1)/2 : jj+(n_block-1)/2);
        D = abs(sum(Block_1) / sum(Block_2));
        if D > threshold
            target(ii,jj,1) = picture2(ii,jj);
            target(ii,jj,2) = picture2(ii,jj);
            target(ii,jj,3) = picture2(ii,jj);
        else
            target_mask(ii,jj) = 1;

            target(ii,jj,1) = .25;
            target(ii,jj,2) = .25;
            target(ii,jj,3) = 1;
        end
    end
end


function [target_mask target] = change_detection_Subspace_1(picture1,
picture2, threshold)
% Subspace Projection distance-based Method
%==========================================================================
%function [target_mask target] = change_detection_Subspace_1(picture1,
%picture2, threshold)
%
% Description: This function computes Subspace Projection Method
%
%
% Input Arguments:
% Name: picture1, picture2
% Type: 2D matrix image
% Description: test image and refrence image
%
% Name: threshold
% Type: float
% Description: threshold
%
%
% Output Arguments:
% Name: target
% Type: 3D matrix image
% Description: Output with blue targets
%
% Name: target_mask
% Type: 2D boolean matrix image
% Description: 2D boolean matrix of targets
% References:
%
%--------------------------------------------------------------------------
% Notes:
%%
%--------------------------------------------------------------------------
% Author: Alireza Nazari
%
% Creation Date: Oct 2012
%
%--------------------------------------------------------------------------
% Revision History:
%
%==========================================================================

block_size = [3 3];    % Must be odd!
m_block = block_size(1);
n_block = block_size(2);
[m n] = size(picture1);
X = zeros( m_block * n_block,(m - (m_block-1)) * (n-(n_block-1)) );
target_mask = zeros(m, n);

index = 1;
for ii = 1+(m_block-1)/2 : m-(m_block-1)/2
```

```
    for jj = 1+(n_block-1)/2 : n-(n_block-1)/2
        Block_1 = picture1(ii-(m_block-1)/2 : ii+(m_block-1)/2, jj-(n_block-
1)/2 : jj+(n_block-1)/2);
        Block_1 = Block_1(:);
        X(:, index) = Block_1;
        index = index + 1;
    end
end
Cx = cov(X');
[V,D] = eig(Cx);
[S ind] = sort(diag(D), 'descend');
D_sort = diag(S);
V_sort = V(:, ind);
N_trunc = 7;
V_trunc = V_sort(:, 1:N_trunc);
D_trunc = D_sort(1:N_trunc, 1:N_trunc);
Px = V_trunc  * V_trunc';
target = zeros(m, n,3);
target_mask = zeros(m, n);
I = eye(m_block * n_block);

for ii = 1+(m_block-1)/2 : m-(m_block-1)/2
    for jj = 1+(n_block-1)/2 : n-(n_block-1)/2
        Block_1 = picture1(ii-(m_block-1)/2 : ii+(m_block-1)/2, jj-(n_block-
1)/2 : jj+(n_block-1)/2);
        Block_2 = picture2(ii-(m_block-1)/2 : ii+(m_block-1)/2, jj-(n_block-
1)/2 : jj+(n_block-1)/2);

        Block_1 = Block_1(:);
        Block_2 = Block_2(:);

        D1 = V_trunc' * Block_1;
        D2 = V_trunc' * Block_2;

        D = D1-D2;
        MD = sqrt(D' * D);

        C = cov(Block_1);

        if MD < threshold

            target(ii,jj,1) = picture2(ii,jj);
            target(ii,jj,2) = picture2(ii,jj);
            target(ii,jj,3) = picture2(ii,jj);
        else
            target_mask(ii,jj) = 1;

            target(ii,jj,1) = .25;
            target(ii,jj,2) = .25;
            target(ii,jj,3) = 1;
        end
    end
end
function [target_mask target] = change_detection_Weiner(picture1, picture2,
threshold, Cx)
```

```
% Weiner Prediction distance-based Method
%========================================================================
%function [target_mask target] = change_detection_Weiner(picture1,
%picture2, threshold, Cx)
%
%
% Description: This function computes Subspace Projection Method
%
%
% Input Arguments:
% Name: picture1, picture2
% Type: 2D matrix image
% Description: test image and refrence image
%
% Name: threshold
% Type: float
% Description: threshold
%
% Name: Cx
% Type: 2D matrix
% Description: Covariance of 3*3 windows
%
% Output Arguments:
% Name: target
% Type: 3D matrix image
% Description: Output with blue targets
%
% Name: target_mask
% Type: 2D boolean matrix image
% Description: 2D boolean matrix of targets
% References:
%
%------------------------------------------------------------------------
% Notes:
%%
%------------------------------------------------------------------------
% Author: Alireza Nazari
%
% Creation Date: Oct 2012
%
%------------------------------------------------------------------------
% Revision History:
%
%========================================================================
big_block_size=[13 13];
block_size = [3 3];    % Must be odd!
m_block = block_size(1);
n_block = block_size(2);
b_m_block = big_block_size(1);
b_n_block = big_block_size(2);

[m n] = size(picture1);
target = zeros(m, n,3);
target_mask = zeros(m, n);
target(:,:,1)=picture1;
target(:,:,2)=picture1;
target(:,:,3)=picture1;
```

```
for ii = 1+(b_m_block-1)/2 : m-(b_m_block-1)/2
    for jj= 1+(b_n_block-1)/2 : n-(b_n_block-1)/2
        B_Block_1 = picture1(ii-(b_m_block-1)/2 : ii+(b_m_block-1)/2, jj-
(b_m_block-1)/2 : jj+(b_n_block-1)/2);
        B_Block_2 = picture2(ii-(b_m_block-1)/2 : ii+(b_m_block-1)/2, jj-
(b_n_block-1)/2 : jj+(b_n_block-1)/2);
        Block_1 = picture1(ii-(m_block-1)/2: ii+(m_block-1)/2, jj-(n_block-
1)/2 : jj+(n_block-1)/2);
        Block_2 = picture2(ii-(m_block-1)/2 : ii+(m_block-1)/2, jj-(n_block-
1)/2 : jj+(n_block-1)/2);
        B_Block_1 = B_Block_1(:)';
        B_Block_2 = B_Block_2(:)';
        rxx = B_Block_1 * B_Block_1';
        ryx = B_Block_2 * B_Block_2';
        W = ryx * pinv(rxx);
                Block_1 = Block_1(:);
                Block_2 = Block_2(:);
                y = W * Block_1;
                D1 = y(:);
                D2 = Block_1(:);
                D = D1 - D2;
                C = cov(D1);
                MD = ((D' * inv(Cx)) * D)/203.6817;

                if MD < threshold
                    target(ii,jj,1) = picture2(ii,jj);
                    target(ii,jj,2) = picture2(ii,jj);
                    target(ii,jj,3) = picture2(ii,jj);
                else
                    target_mask(ii,jj) = 1;
                    target(ii,jj,1) = .25;
                    target(ii,jj,2) = .25;
                    target(ii,jj,3) = 1;
                end
            end
end
% figure
% imshow(target)
% figure
% imshow(target_mask)

    end

function [target_Euclidean_mask target] =
change_detection_Euclidean_Difference(picture1, picture2, threshold)
%==========================================================================
% function [target_Difference_mask target] =
% change_detection_Euclidean_Difference(picture1, picture2, threshold)
%
% Description: This function computes Euclidean distance Difference Method
%
%
% Input Arguments:
% Name: picture1, picture2
% Type: 2D matrix image
% Description: test image and refrence image
%
% Name: threshold
% Type: float
% Description: threshold
%
%
% Output Arguments:
% Name: target
% Type: 3D matrix image
% Description: Output with blue targets
%
% Name: target_mask
% Type: 2D boolean matrix image
% Description: 2D boolean matrix of targets
% References:
%
%-----------------------------------------------------------------------------
% Notes:
%
%-----------------------------------------------------------------------------
% Author: Alireza Nazari
%
% Creation Date: Oct 2012
%
%-----------------------------------------------------------------------------
% Revision History:
%
%=============================================================================

block_size = [3 3];     % Must be odd!
m_block = block_size(1);
n_block = block_size(2);

[m n] = size(picture1);
target = zeros(m, n,3);
target_Euclidean_mask = zeros(m, n);

for ii = 1+(m_block-1)/2 : m-(m_block-1)/2
    for jj = 1+(n_block-1)/2 : n-(n_block-1)/2
        Block_1 = picture1(ii-(m_block-1)/2 : ii+(m_block-1)/2, jj-(n_block-
1)/2 : jj+(n_block-1)/2);
        Block_2 = picture2(ii-(m_block-1)/2 : ii+(m_block-1)/2, jj-(n_block-
1)/2 : jj+(n_block-1)/2);

        D = (Block_1 - Block_2) .^ 2;
        D =  mean(D(:));

        if D < threshold
            target(ii,jj,1) = picture2(ii,jj);
            target(ii,jj,2) = picture2(ii,jj);
            target(ii,jj,3) = picture2(ii,jj);
        else
            target_Euclidean_mask(ii,jj) = 1;
            target(ii,jj,1) = .25;
            target(ii,jj,2) = .25;
            target(ii,jj,3) = 1;
        end
    end
end
function [target_Euclidean_mask target] =
change_detection_Euclidean_Difference(picture1, picture2, threshold)
%==========================================================================
% A Domo of using functions on an input image
%
%--------------------------------------------------------------------------
% Notes:
%
%--------------------------------------------------------------------------
% Author: Alireza Nazari
%
% Creation Date: Oct 2012
%
%--------------------------------------------------------------------------
% Revision History:
%
%==========================================================================

close all
clear all
clc

%***************************************************************************

load ('pic.mat')
%*****************************
Target_org = imread('Target_org.jpg');

% imshow(picture1)
figure
image(x_final_image2, y_final_image2,Target_org);
title('Target Image')


Target_org_1 = (Target_org(:,:,1) > 240) & (Target_org(:,:,2) < 20) &
(Target_org(:,:,3) < 20);
Target_org_1 = uint8(double(Target_org_1) * 255);
Target_org_mask = double(Target_org_1 == 255);

block_size = [3 3];    % Must be odd!
m_block = block_size(1);
n_block = block_size(2);
[m n] = size(picture1);

X = zeros( m_block * n_block,(m - (m_block-1)) * (n-(n_block-1)) );
target_mask = zeros(m, n);

index = 1;
for ii = 1+(m_block-1)/2 : m-(m_block-1)/2
```

```
    for jj = 1+(n_block-1)/2 : n-(n_block-1)/2
        Block_1 = picture1(ii-(m_block-1)/2 : ii+(m_block-1)/2, jj-(n_block-
1)/2 : jj+(n_block-1)/2);
        Block_1 = Block_1(:);

        X(:, index) = Block_1;
        index = index + 1;
    end
end
Cx = cov(X');

set(0,'DefaultFigureWindowStyle','docked')

tic
[target_mask target] = change_detection_Difference(picture1, picture2, .86);
display('elapsed time of change_detection_Difference');
toc
[detection false_alarm]=find_Error(target_mask, Target_org_mask)
figure
imshow(target)
title (['change detection Difference Method False alarm:'
num2str(false_alarm) 'True alarm' num2str(detection)])



tic
[target_mask target] = change_detection_Block_Difference(picture1, picture2,
.17);
display('elapsed time of change_detection_Block_Difference');
toc
[detection false_alarm]=find_Error(target_mask, Target_org_mask)
figure
imshow(target)
title (['Block Difference Method False alarm:' num2str(false_alarm) 'True
alarm' num2str(detection)])

tic
[target_mask target] = change_detection_Euclidean_Difference(picture1,
picture2, .04);
display('elapsed time of change_detection_Euclidean_Difference');
toc
[detection false_alarm]=find_Error(target_mask, Target_org_mask)
figure
imshow(target)
title (['Euclidean Difference Method False alarm:' num2str(false_alarm) 'True
alarm' num2str(detection)])

tic
[target_mask target] = change_detection_Ratioing_2(picture1, picture2, .817);
display('elapsed time of change_detection_Ratioing');
toc
[detection false_alarm]=find_Error(target_mask, Target_org_mask)
figure
imshow(target)
title (['Ratioing Method False alarm:' num2str(false_alarm) 'True alarm'
num2str(detection)])
```

```
tic
[target_mask target] = change_detection_Mahalanobis(picture1, picture2, .06,
Cx);
display('elapsed time of change_detection_Mahalanobis');
toc
[detection false_alarm]=find_Error(target_mask, Target_org_mask)
figure
imshow(target)
title (['Mahalanobis Method False alarm:' num2str(false_alarm) 'True alarm'
num2str(detection)])

tic
[target_mask target] = change_detection_Weiner(picture1, picture2, .187,Cx);
display('elapsed time of change_detection_Weiner');
toc
[detection false_alarm]=find_Error(target_mask, Target_org_mask)
figure
imshow(target)
title ([' Weiner Prediction Method False alarm:' num2str(false_alarm) 'True
alarm' num2str(detection)])

tic
[target_mask target] = change_detection_Subspace_1(picture1, picture2, .57);
display('elapsed time of change_detection_Subspace_1');
toc
[detection false_alarm]=find_Error(target_mask, Target_org_mask)
figure
imshow(target)
title (['Subspace Method False alarm:' num2str(false_alarm) 'True alarm'
num2str(detection)])
```