Computer Science
Technical Report

Colorado State University

A Tutorial on a Sliding Window Target Detection Algorithm Implemented in the DARPA Image Understanding Environment [*]

J. Ross Beveridge and Jim Steinborn
Colorado State University
ross/steinbor@cs.colostate.edu

November 14, 1997

Technical Report CS-97-121

Computer Science Department
Colorado State University
Fort Collins, CO 80523-1873

Phone: (970) 491-5792 Fax: (970) 491-2466
WWW: http://www.cs.colostate.edu

[*]This work was sponsored by the Defense Advanced Research Projects Agency (ARPA) through the Topographic Engineering Center under the auspices of the U.S. Army Research Office Scientific Services Program and monitored by Battelle. (Grant number DAAL03-91-C-0034 and TCN 96188 D.O. #1958)

A Tutorial on a Sliding Window Target Detection Algorithm Implemented in the DARPA Image Understanding Environment [*]

J. Ross Beveridge and Jim Steinborn
Colorado State University
ross/steinbor@cs.colostate.edu

November 14, 1997

1 Introduction

The Image Understanding Environment (IUE) is an object-oriented system implemented in C++ with the aim of aiding research in image understanding (IU). The exchange of results amongst research, industry and government groups will be facilitated by use of this common platform. This system includes standard implementations of common IU algorithms as well as standard IU data structures. By this means, progress in algorithm development may be tracked, and new concepts in image understanding may be tested.

This technical report provides a tutorial level explanation of a Sliding Target Box Detector (STaBD) [1]. This algorithm was selected as an archetypical Automatic Target Recognition algorithm and it is based loosely on the concepts of a sliding window detector set out by Nguyen [D. 90]. This algorithm is also of practical interest because it is used as the first phase of a two phase target detection algorithm on the Unmanned Ground Vehicle Program's Semi-Autonomous Scout Vehicles. That detection algorithm was demonstrated on live imagery in July 1995 at the Lockheed Martin test site outside Denver and in the Summer of 1996 at Fort Hood, Texas.

The IUE STaBD task presented here accepts Forward Looking Infrared (FLIR) images in the TIFF format, and returns two other TIFF images: one containing the t-values indicating to the possibility of there being a detection at that point, and the other containing the original image with target boxes drawn around all the detections that are greater than a user-specified threshold.

2 Overview

STaBD determines whether or not a detection exists at a particular point by comparing the pixel values in a target detection box centered at that point and the pixel values in a frame that is con-

[*]This work was sponsored by the Defense Advanced Research Projects Agency (ARPA) through the Topographic Engineering Center under the auspices of the U.S. Army Research Office Scientific Services Program and monitored by Battelle. (Grant number DAAL03-91-C-0034 and TCN 96188 D.O. #1958)

[1]This report will use exact code examples when finished. These examples are not be included now because the code must be changed to reflect the new IUE/KHOROS task protocol released in September 1996

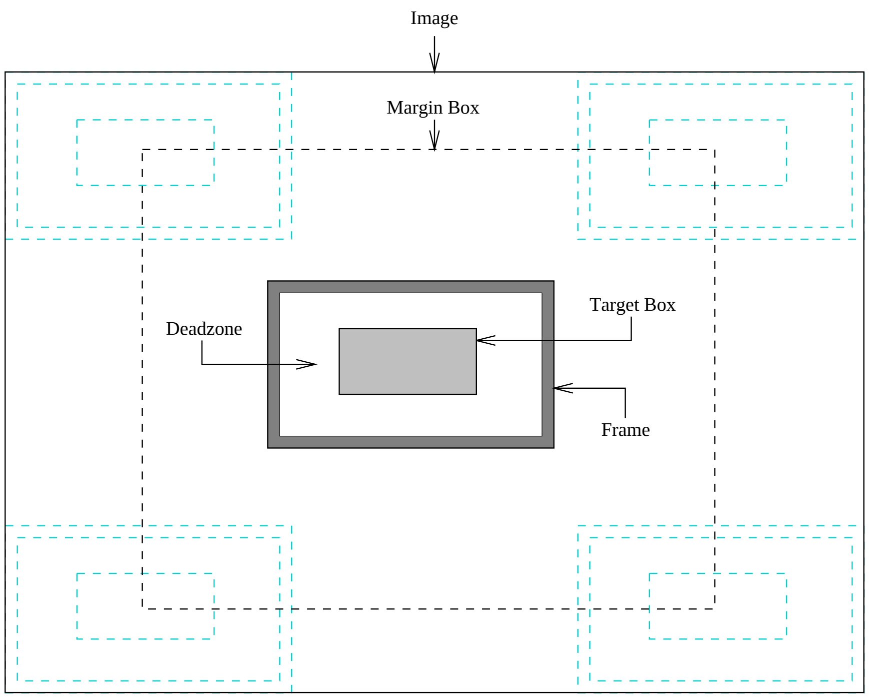


Figure 1: Layout of the components of the sliding detection box.

centric with the detection box. Referring to figure 1, you can see that the target box is surrounded by the deadzone, an area of unsampled pixels. Outside of this zone is the frame. The dimensions of the target box and the width of the deadzone are specified by the user on the command line.

Means and variances are computed separately for both the target box and the frame. The t statistic is then computed, and from this can be calculated the probability p that the pixel values in a target detection box are from the same normal distribution as the pixel values in the frame. The target box and associated frame are 'slid' over the entire image, and the pixel distributions of the target box and frame are examined. These pixel areas are compared by means of the t statistic - a measure of how likely a value belongs to a normal distribution - along with the associated probability p that the pixels comprising the two areas are from the same distribution. The t values and the p values are stored in separate arrays that are the same dimensions as the image. It should be noted that the t statistic is more useful in this case, because we are more interested in the likelihood of a detection, rather than proving that the target box is 'different enough' than the frame region.

3 Example

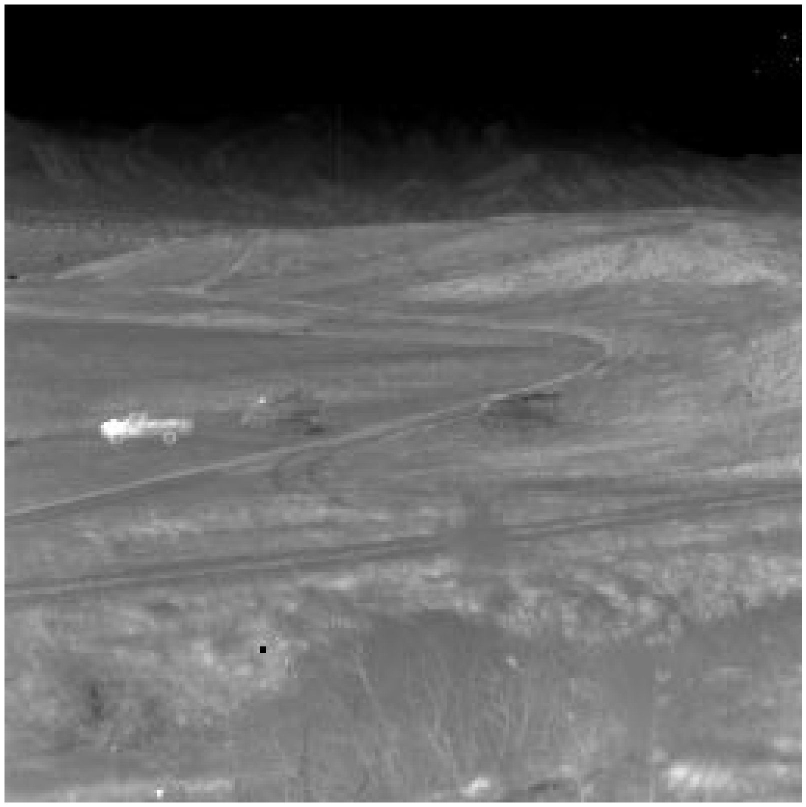


Figure 2: Sample FLIR image from the Fort Carson Dataset.

The command line for STaBD is as follows:

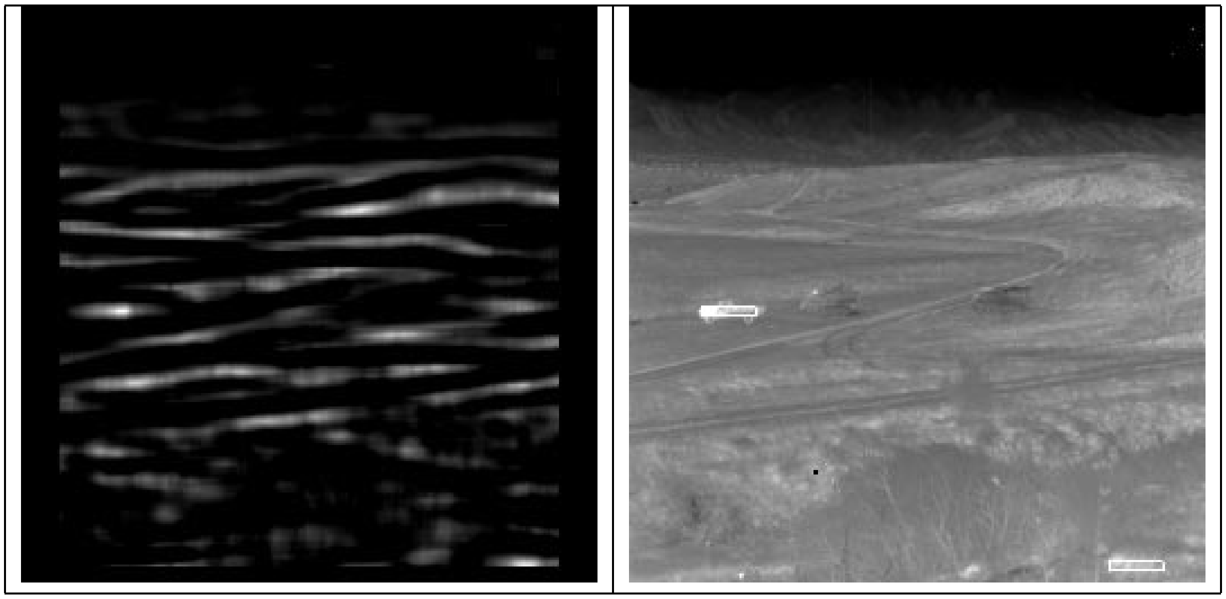
```
tbox imagefile width height deadZone hot/cold upThresh
```

Imagefile is the input TIFF image; width and height are the pixel dimensions of the target box; deadZone is the width (in pixels) of the unsampled space around the target box; hot/cold is a binary variable representing the type of detection desired: hot = hot targets, cold = cold targets; upThresh represents the top x percent of the detections that are to be considered as detections: for example, 0.1 = the top 10% of detections.

For this example, we applied StaBD to a FLIR image, nov110000.tiff (see figure 2), from the Fort Carson Data Collection [BPY94]. The following command line was issued:

```
tbox nov11000.tiff 25 5 3 hot 0.1
```

This indicates that the target box is 25 pixels wide and 5 pixels high, with an unsampled space of 3 pixels around it. We are looking for the top 10% of hot detections. Figure 3 shows the results of the run: **a.** shows the t values, the low values are dark; **b.** shows the detection boxes overlaid on the original FLIR image.



a. t value image **b.** FLIR image with detection boxes overlaid

Figure 3: Results of running STaBD on the sample image.

4 The Class

We define a new class of object for efficiently computing the statistics over axis-aligned boxes of pixels within images. This specialized mechanism uses the accumulated sum and sum of squared image values of all pixels down and to the left of the current pixel. See section 5.3 for more details.

The class is called WinTStat (for Window T Statistic). Table 1 lists the private members (variables) of this class, and table 2 lists its public members.

	IUE_array_2d<IUE_INT>
sum	array in which to store sums
sumSq	array in which to store squared sums
	IUE_scalar_image_2d_of<IUE_UINT8>
img	pointer to the image array
imgDetPixels	array (same dimensions as img) in which to store detection results
imgDetBoxes	array (same dimensions as img) in which to store detection boxes
	IUE_scalar_image_2d_of<IUE_DOUBLE>
imgTValue	array (same dimensions as img) in which to store t values
imgPValue	array (same dimensions as img) in which to store p values
imgTScratch	array (same dimensions as img) in which to store intermediate results

Table 1: The class WinTStat private members.

5 Initialization

Section 4 described the class WinTStat; this section describes the operation of the class constructor during initialization of a new WinTStat object. The dimensions of the frame around the target box and the margin box are computed. The aforementioned accumulation arrays containing sum and sum of squared image pixel values are also calculated.

5.1 Widths and Offsets

For efficiency, the various target boxes are frames that are stored as integer offsets to their corners from the current pixel. The widths and offsets of the target box and frame (see figure 1) are computed from the user-specified values on the command line. In order to compare the contents in the target box to the contents in the frame, both of these areas need to contain a similar number of pixels. The width of the frame is computed via the quadratic formula using the dimensions of the target box and the width of the dead zone. Since all of these dimensions are integer values, there may be an inequality between the number of pixels in the frame and the target box. If this occurs, the frame area will be made larger than the target box area.

5.2 Margin

The margin box (see figure 1) allows us to easily test whether the current pixel is at the center of a target box wholly contained within the image itself. One of the methods of the class IUE_discrete_axis_aligned_box_2d is called `is_in`. This enables an easy call to determine if a pixel location exists within this margin box. The margin is computed from the dimensions of the outer

	IUE_discrete_axis_aligned_box_2d
margin	box used to determine if frame falls off the edge of the image
	IUE_DOUBLE
upThresh	threshold of the top percentage of t values which will be considered
	IUE_BOOL
hotCold	a flag signaling whether the user is interested in hot or cold detections
	IUE_INT
widthTarget	width of the sliding target box
heightTarget	height of the sliding target box
widthFrame	width of the outside of the frame
heightFrame	height of the outside of the frame
deadZone	width of the area of unsampled pixels between the target box and the frame
frameZone	width of the frame
nTarget	number of pixels contained within the target box
nFrame	number of pixels contained within the frame
xMax	x dimension of image array
yMax	y dimension of image array
xMinA	A is the target box
xMaxA	these are the offsets from the current
yMinA	pixel to its corners
yMaxA	
xMinB	B is the outside of the frame
xMaxB	these are the offsets from the current
yMinB	pixel to its corners
yMaxB	
xMinC	C is the inside of the frame
xMaxC	these are the offsets from the current
yMinC	pixel to its corners
yMaxC	

Table 2: The class WinTStat public members.

border of the frame surrounding the target box. As a result of this, you will note in the t value image (see figure **3a.**) has a dark border of pixels where it was impossible to perform any statistical evaluation.

3	3	1	2	1
2	1	2	3	2
1	1	2	3	2
3	1	3	2	2
1	3	2	1	2

0	10	17	27	38	47
0	7	13	22	31	39
0	5	10	17	23	29
0	4	8	13	16	20
0	1	4	6	7	9
0	0	0	0	0	0

Figure 4: Comparison of the lower left portion of an image (left) and its sum array (right).

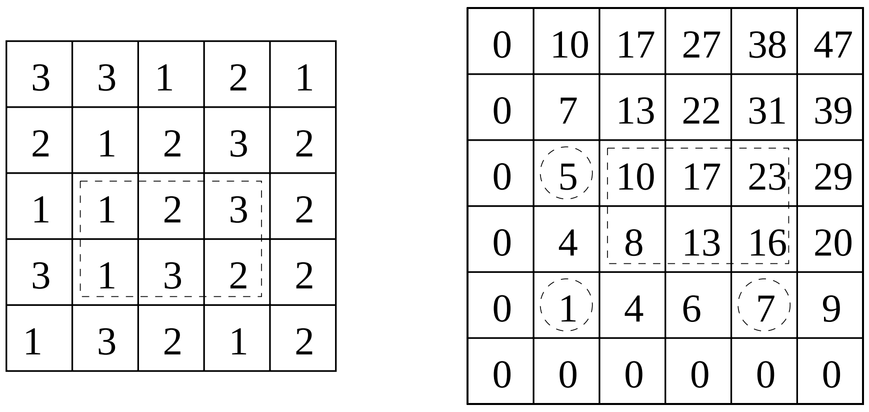
5.3 Sum and SumSq arrays

These arrays accumulate the sum and squared sum image values for all of the pixels below and to the left of the current pixel. By this means, the mean and variance over rectangular windows within the image may be computed efficiently, needing to access only the values at the four corners of the rectangle. These two sum arrays have dimensions similar to the image array, except that they have an extra row added to the bottom and an extra column appended to the left. This '-1' row and column contain zeros: these values are useful when computing the statistics by means of this accumulated-sum technique (see figure 4).

Using this type of array, the sum of all the image values in the dashed rectangle (see figure 6) is the value stored in the upper-right corner (23) minus the value to the left of the upper-left corner (5) and the value below the lower-right corner (7) plus the value below and to the left of the lower-left corner (1). This works because each cell in this matrix contains the sum of the values in all the cells below and to the left of the current cell. Therefore, the upper-right cell of the rectangle (23) contains the sum of all of the cells from (-1, -1) to (3,2).

Referring to figure 6, the cell in the upper right of **A** contains the sum of all pixels down and to the left of that cell. If we subtract all of the pixels to the left of and below the unhatched box labeled by **A**, we would be left with just the sum of pixels in **A**. To do this, take the value in the upper right cell of **A**, subtract the values in the upper right cells of **C** and **D**, and add the value in the upper right cell of **B**. Since **B** is included in both **C** and **D**, it has been subtracted twice, making this last addition necessary.

Referring to figure 5, note that for this computation you have:



3	3	1	2	1
2	1	2	3	2
1	1	2	3	2
3	1	3	2	2
1	3	2	1	2

0	10	17	27	38	47
0	7	13	22	31	39
0	5	10	17	23	29
0	4	8	13	16	20
0	1	4	6	7	9
0	0	0	0	0	0

Figure 5: Image array vs. sum array - computing the sum of the values in the box.

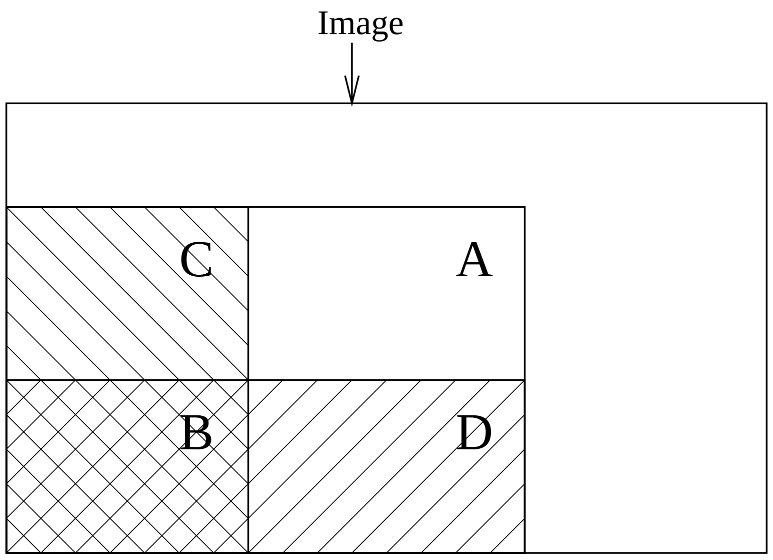


Figure 6: Sum of cells in inner unhatched box = **A** - **C** - **D** + **B**

- from image array: 1 + 3 + 2 + 1 + 2 + 3 = 12
- from sum array: 23 - 7 - 5 + 1 = 12

As you can see, summing the values in a rectangle in the image array is O(# cells in rectangle), whereas doing the same using the sum array only requires accessing 4 values.

6 Computation of Statistics

Having obtained the sum of all the pixels within a box in the image, one can now use statistical analysis to determine whether or not the pixel at the center of the target box represents a detection. The t and p values are computed for every pixel location in the image. If the user is looking for cold detections, the t value is negated. These values are then stored in the imgTValue and imgPValue arrays (these arrays have the same dimensions as the image array).

If the current pixel is not within the margin box, 0.0 is assigned to the t value and p is set to 0.5, which is a reasonable assumption. If the pixel is inside the margin box, the means and variances of the values contained within the target box and the frame are computed separately. From these, the pooled variance is determined, and the t value is computed. If the two means are equal or if the the pooled variance is equal to zero, the t statistic is assigned the value 0.0. The p value is then computed, which is the probability that the pixel values in the target box and the frame are drawn from the same normal distribution.

7 Detection strength image

An image which represents the detection strength of every pixel is generated. This is done by converting the values in the imgTValue array to the range 0..255.

8 Detection boxes image

To see where the detections are in the image, detection boxes are drawn into a copy of the t value image (imgTScratch). The maximum value is extracted from the t value array, and a detection threshold is created from it by multiplying by (1.0 - upThresh). This makes the threshold equal to the top upThresh percent of the maximum value. A detection box is drawn in the boxes image (imgDetBoxes) at this location, and an area equal to the outside frame of the sliding detection box is cleared around the current pixel location. Clearing this area ensures that there is only one detection box drawn around the pixel with the maximum value. This procedure is repeated until there are no more t values greater than the threshold.

References

[BPY94] J. Ross Beveridge, Durga P. Panda, and Theodore Yachik. November 1993 Fort Carson RSTA Data Collection Final Report. Technical Report CSS-94-118, Colorado State

University, Fort Collins, CO, January 1994.

[D. 90] D. M. Nguyen. An iterative Technique for Target Detection and Segmentation in IR Imaging Systems. Technical Report November, (CECOM) Center for Night Vision and Electro-Optics, 1990.