Robust motion detection in live outdoor video streams using triplines

Master thesis of G.J. Kamstra
Supervisor: J.A.G. Nijhuis

Table of Contents

Abstract

Motion detection is an interesting research subject in computer vision. Multiple approaches are possible. In this thesis wel will give a full overview of using triplines in motion detection applications. First we will discuss preprocessing techniques like illumination change compensation. Then we will focus our attention on the actual triplines. Topics here are the placing, the shape, the length and so on. We will discuss several algorithms to use on this triplines (Thresholded Difference, Positive Thresholding, Kolmogorov Smirnov test statistic, Smirnov test statistic, Modified Smirnov test statistic, Approximate Entropy and FFT detect). We will make a comparison between these algorithms on triplines and other ways of motion detection (including non computer vision based). After the discussion on the algorithms, we will define some performance measurements. We will argue the downfall of the count error, which has been used in several papers, and introduce our new and improved count error. The last step in our motion detection is the filtering step. Sequence filtering and average filtering are discussed. Finally we will discuss the impact of frame rate reduction on the performance of our algorithms.

1 Introduction

Nowadays many locations are overseen by surveillance cameras. The applications for these camera's are many and range from traffic surveillance, through monitor revel public, to security surveillance to guard company premises. Camera's can be used to check who uses a toll road, and bill him accordingly. In Groningen camera's are used to guard the public during their time on the spree. This resulted in a significant drop in the number of fights/riots during these times. Allthough the applications of these systems are quite interesting, we will focus on the technical backbone of these systems: what happens with the realtime video stream that is captured by the camera. Two cases can be distinguished: The video data is inspected by a human guard or by a computer algorithm. The algorithms most widely used in these sytems are called motion detection algorithms. These algorithms try to detect (relevant) motion in the received video stream. Several algorithms are available.One of the problems with using standard motion detection algorithms is, that they generate many false alarms when used in outdoor situations, because of several difficulties with outdoor behaviour. Things that can generate false alarms in outdoor situations are:

- A bird flying by. When a bird flies through the view, an alarm shouldn't be generated. We're not interested in birds, allthough this is motion, we don't want to detect it.
- The camera shaking in the wind. When the camera shakes, the whole view gets translated to another position. This should of course be ignored.
- Altered lightning conditions (evening). Due to our day and night cycle, the recorded image will change. A picture taken at dawn looks a lot different then one taken at noon, even though this isn't motion. This should be ignored by the motion detection algorithm as well. Another thing that should be taken into account is the adaptive powers of the used camera. When the recorded image gets dark, the camera will adapt its sensitivity accordingly. This can results in undesired side effects, like changes in illumination which aren't really present in the observed area.
- Trees, leafs moving in the wind. The wind will cause a lot of motion is bushes, trees, flags and so on. Although this is motion, this should for all practical usages be ignored.
- Shadows. Shadows can be caused by several things: clouds, interesting objects (intruders, cars) and uninteresting objects (birds, trees). Ideally shadows should be ignored, even when caused by interesting objects. The objects themselves should be detected, not their shadows.
- Fata Morgana. When the air is hot, reshaping of object observation can take place. This shouldn't generate a detection.
- Changing weather conditions (rain, hail, snow) When it rains, lots of movement is present in the scene. This should be ignored ass well.

In this thesis we will investigate the possibility of using triplines in surveillance operations. Triplines are known from military and security operations (real wire, or an electric eye). In image processing this technique can also be used. You can create a line in the image, and in the algorithm you only examine the pixels on that line. This can really increase the performance of such a system: only a few pixels need to be examined instead of the whole picture.

Important aspects which will be dealt with is this thesis are:

- Which algorithm to use on the tripline.
- Changing illumination.
- Where to place triplines.

- How many triplines.
- Size of triplines.
- Comparison against other commerically available systems (including a short explanation about them).
- Impact of frame rate on performance.

Performance tests will be conducted on two test sequences of images.

In the next chapter we will give a short overview of the technology behing motion detection sytems. In the following chapters we will give the full description of all the steps in our detection system. Our system uses a classical approach from litterature:

1 Preprocessing: manipulating the input data so it can be easier handled in the next step.

2 Processing: in this step, the real work is done, this is were the motion detection algorithms are used.

3 Postprocessing: now the output of the motion detection algorithms is improved to give better results.

After the description of the technical aspects of our detection system, we will define our performance measurements. We will then focus on our test procedure, and will give the results of the performance tests we conducted.

2 Technology of Video Motion Detection-systems (VMD)

In this chapter we will give a brief overview of the state of technology of VMD-systems (VMD is the digitization and analysis of a video picture generated, usually by a CCTV camera). Movement is detected as a change in the video signal in relation to a reference image of a specific location, video scene or part of a video scene. It is actually "Computer Vision" and is used in many scientific and industrial applications as well as security.

2.1 Analog vs. Digital

In the old days most VMD were analog systems. They provided a very limited ability to distinguish between leaves blowing in the wind, other uninteresting movement and real intruder detection. Either the nuisance alarm rates are high or the sensitivity must be set so low that detecting a true event is questionable. Camera vibration can't be compensated, and illumination compensation isn't available either. Digital VMD's are generally classified as units that use A/D (Analog to Digital) converters to sample the incoming video signal and electronically convert it to a digital value. The internal processor of the VMD then determines video motion by processing the image in the digital domain. Some of the advantages of a digital sytem when compared to an analog system are:

- Number and placement of sensors sometimes cost prohibitive by other methods
- Target tracking features
- Direction Sensing
- Change (mask) detection area dimensions (full or partial screen... multi-points)
- Multiple detection zones for one camera image
- Illumination compensation
- Vibration detection and alarm suppression
- Ability to discriminate between wind, rain, snow, blow leaves, small animals or birds
- In many cases timed alarm responses and computer communication ports to proprietary systems or large scale matrix switchers.
- Multiple setup modes for day/night or by times of day
- Alarm Inputs to AND/OR with other motion detection technology to further reduce false alarms

Although these advantages can be realized, they aren't all realized as of yet. Especially in outdoor environments, these problems haven't been solved at all. Another remark is, although is seems like an advantage to be able to adjust several settings, it would be handier, if the system could automatically determine the best settings. This reduces the costs of installing such a system dramatically.

2.2 Memory and Processor

Digital VMD's use memory to hold a reference image or reference output of a certain algorithm which is run on the input image. Also the memory is used to buffer the video-stream. If the VMD is designed to track objects, more memory is needed to store this information.

Generally the faster the processor, microprocessor or DSP the more pixels that can be processed per second. The faster this analysis, the better the resolution and accuracy of the VMD. The speed of the processing is not determined only by the processor speed but also by how many images computations per second is actually accomplished. A PC based VMD may have a processor speed of 500 Mhz but actually running at 10 MIPs (Million Instructions per Second) of video computation. A stand alone unit may have dedicated DSP and/or dedicated microprocessor and/or a PLD running at 12Mhz each and have a combined processing speed of 52MIPs or higher.

Another way to setup a VMD-system is to use multiple dumb camera's whose only goal is to record the scene, and transmit it to a standard PC located in the building. This PC processes all incoming data. This PC will probably be cheaper and faster then using logic in each camera. The PC will also be responsible for warning an operator and recording the stream when an intruder is detected.

For the used algorithms the system architecture isn't really important. All discussed algorithms take a video-stream as input, and generate an alarm as output. Whether the logic is built into the camera or in a separate PC, which processes multiple streams isn't relevant.

3 Preprocessing: Illumination compensation

As we have told earlier one big cause of false alarms is altering lighting conditions. That's why we can improve the performance of motion detection algorithms by using illumination compensation techniques. An Illumination compensation algorithm tries to estimate the lighting change, and compensates the image accordingly. In fact illumination compensation is one of the most important aspects in outdoor surveillance. Lighting conditions can vary strongly in outdoor video streams. Not only the usual day/night cycle can change lighting conditions, but also clouds can change lighting conditions within a couple of seconds. Take for instance a shiny day, with only a few clouds in the sky. When one of these clouds gets between the sun, and the monitored area, first only part of the recorded image will be darkened. After a few seconds (depending on wind speed) the whole image will be darkened. This is an important property. It makes several algorithms unusable since they try to compensate global lighting changes. Illumination compensation can be used as a preprocessing step, before using tripline algorithms. Since lighting changes are a form of motion, motion detection algorithms can generate many false alarms when lighting conditions are changing. This undesired side-effect of illumination changes can however be compensated. In the next sections we will discuss several illumination compensation algorithms.

3.1 Global illumination compensation

If we assume an uniform change in illumination [YFH:CDPS], all pixel intensities will be multiplied with a constant factor:

$$x_k(n,m) \equiv a\, x_{(k-1)}(n,m)$$

Here $x_k(n,m)$ is the intensity of the pixel (n,m) in image k.

Now in order to find this lighting factor a , the mean of the intesities of the pixels must be calculated in the two consecutive images. The a can now be found by the following formula:

$$a = \frac{\bar{I}_2}{\bar{I}_1}$$

Here $\bar{I}_2$ and $\bar{I}_1$ are the average intensities of image 2 and 1 respectively. By dividing all the pixels in the new image by this factor, the image has been rescaled to the same lighting conditions. The motion detection algorithm won't know the difference, and won't act upon the lighting change. We will call this illumination technique linear illumination compensation.

Another way of finding the lighting factor is described in [MPMPA] .
They try to calculate the a by looking at the energy difference between frames. This energy will be minimized in the following formula. The corresponding a, for which this formula is minimized will be used for the illumination compensation.

$$\sum n : \sum m : (x_k(n,m) - a\, x_{(k-1)}(n,m))^2$$

By minimizing this formula, a can be calculated:

$$a_{opt} = \frac{(\sum n : \sum m : x_k(n,m) x_{(k-1)}(n,m))}{(\sum n : \sum m : x_k(n,m) x_{(k)}(n,m))}$$

By using this formula , a_{opt} can be found in $O(n\ m)$.
a_{opt} Can then be used to rescale the image to the old lighting conditions. From now on, we will call this method, the quadratic illumination compensation method.

3.2 Pixel/Boxed based illumination compensation

In the previous section we calculated one a for the whole image. This was called global illumination compensation. These global illumination compensation algorithms do not work very well in the aforementioned cloud-case, since the illumination change isn't global, but local. However, by running these algorithms for each pixel in a neighborhood around this pixel, these methods can be used to correct one pixel. Take for example an image of 25 * 25 pixels. We determine a for each pixel individually by evaluating a square around it (say 5 *5 pixels). We then compensate the pixel using the calculated a. The problem with this method however is it relatively slow. It's 25 times slower (based on a window of 5 *5 pixels) then the global compensators (each pixel is part of 25 examined windows). There is however a mid form between the two kinds of algorithms. By segmenting the image in windows of say 5*5 pixels, and determining a by using one of the before mentioned algorithms. When using a to scale the window the time needed for the calculation remains the same as the global scheme. The results will be better then the global techniques, however not as good as the pixel based method. The results however will be blocky. A full comparison of these methods can be found in the next section.

To clarify the previous mentioned algorithms, their Pseudo code equivelant is given:

```
Box Based Compensation:
Divide the image in X boxes;
For each box do:
        Calculate average light change in box;
        Compensate Box accordingly;
End;
```

```
Pixel Based Compensation:
For each pixel do:
        Calculate average light change of box surrounding pixel
        Compensate pixel accordingly;
End;
```

3.3 Comparison of illumination compensation techniques

In order to test the performance of the aforementioned algorithms, a sample set of images was constructed. For our first experimental test, we selected an image of the same location, taken at a

different times.

Illustration 1 Reference Image

The reference image is quite dark. It was teaken by night at a petrol station near Rotterdam, see Chapter Test Sets for more details on this.

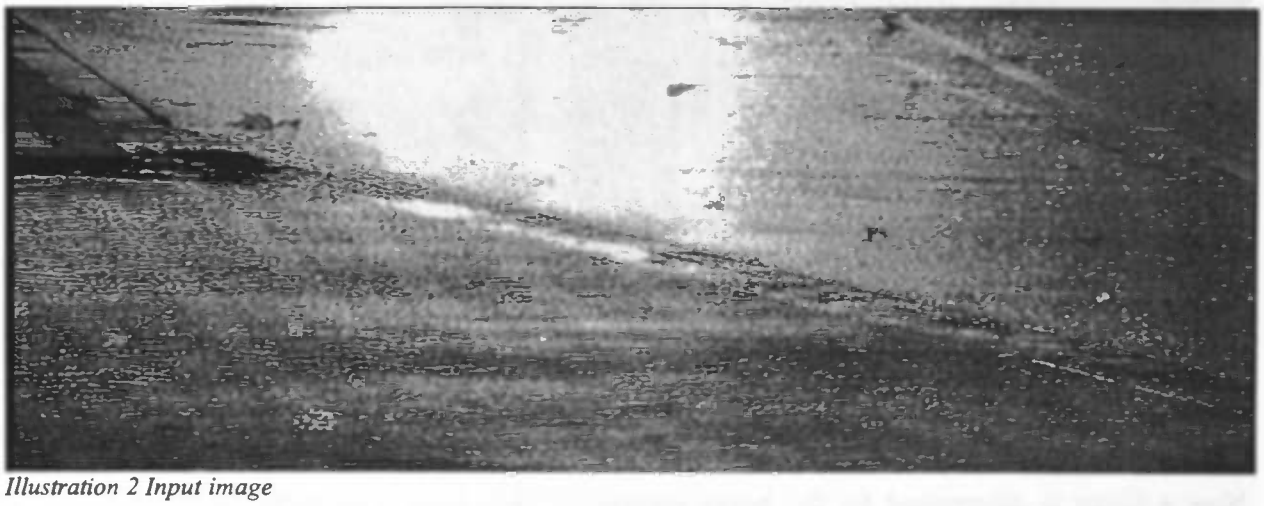


Illustration 2 Input image

The input image is clearly a lot lighter, this needs to be compensated. Observe, that the lighting changes aren't uniform over the whole image, some regions differ more, than others. The biggest change is in the upper middel part of the image. A car is approaching the viewed area, and the light of its headlights is already visible in the image.

Illustration 3 Linear global compensated input image

The global compensation method results in a darker image. Because the lighting change isn't global, the result is too dark at some regions, while being too light on other regions.

Illustration 4Linear, box based compensated image, boxsize=5

Now a factor is determined for the boxes separately. This gives a much better result. Now the regions which need more compensation, are more compensated. A disadvantage however is the blockyness of the result. The borders of the boxes can be clearly observed.

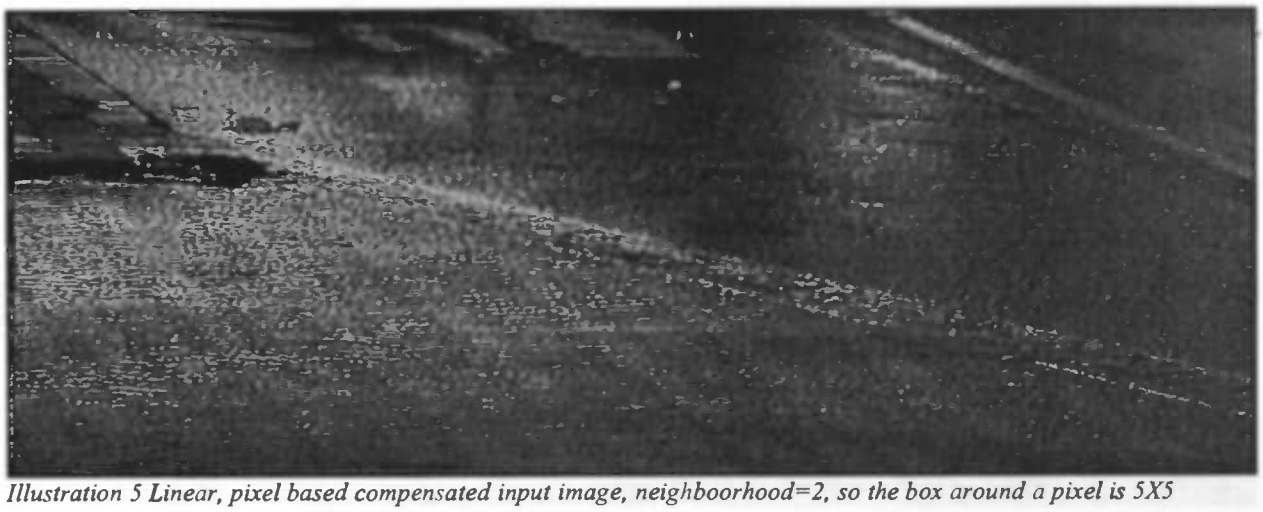


Illustration 5 Linear, pixel based compensated input image, neighboorhood=2, so the box around a pixel is 5X5

The pixel based method obviously gives the best result. The resulting images looks like the reference image, and no box borders are visible. Now we will show the results of the quadratic methods, the results are quite similar to the linear results, so only an observation about the general results of the quadratic method, compared to the linear method is given.

Illustration 6Quadratic, box based compensation

Illustration 7Quadratic, pixel based compensation, Neigbourhood=2

Illustration 8Quadratic, global compensated image

The results of the quadratic method are quite similar, to the ones of the linear method. The global method has problems with the non-global illumination change, the box based method gives a blocky result, and the pixel based compensation gives the best result. It will be when we determine the results of the full detection process, when we can really make some objective statements about the differences in the performance of the both methods.

Now we'll take a look at another reference image:

Illustration 9Reference image 2

This image was taken at the entrance of a parking lot.

Illustration 10 Input image 2

This is the same position, but now a artificially created bright light has been positioned above the scene.

Illustration 11 Linear global compensation

The global compensation doesn't work on this picture, because the lumination change isn't uniform. The boxed version gives much better results:

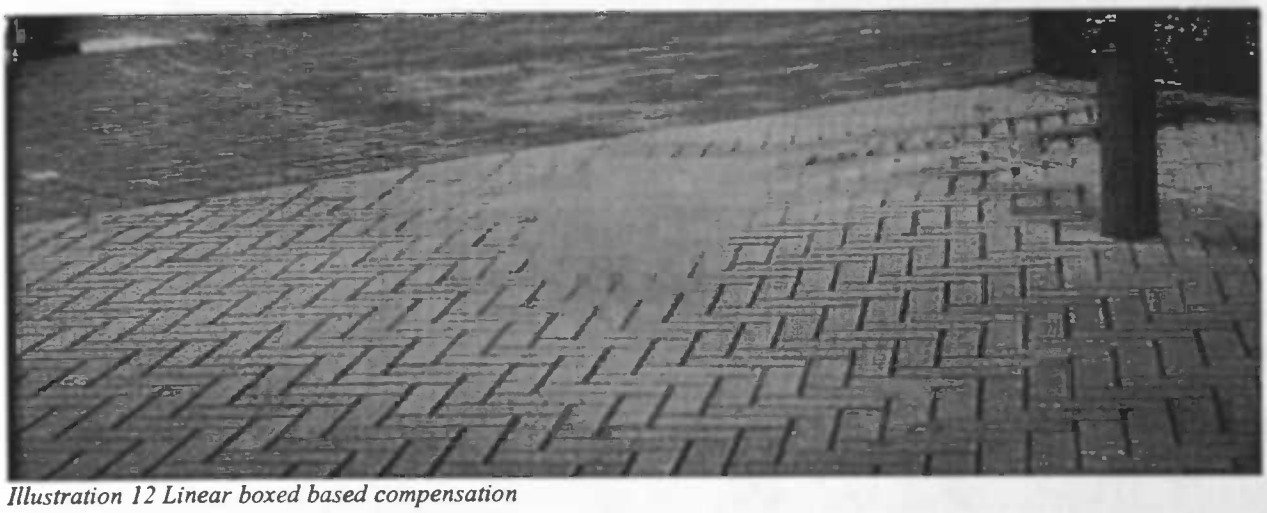


Illustration 12 Linear boxed based compensation

Only the centerpiece of the change (the light spot) can't be fixed properly. The pixel based method gives the following result:

Illustration 13 Pixel based linear compensation

The pixel based method does quite a good job at the centerpiece. It gets a little bit blurry, but the result is much better than the other methods.

3.4 Motion related changes

Interesting effects occur when differences are visible, which aren't illumination related. Take for example the following input image:

Illustration 14input image: A car enters the scene

A car has entered the scene, and generates big differences in the image. When using small boxes, the compensated image will look like this:

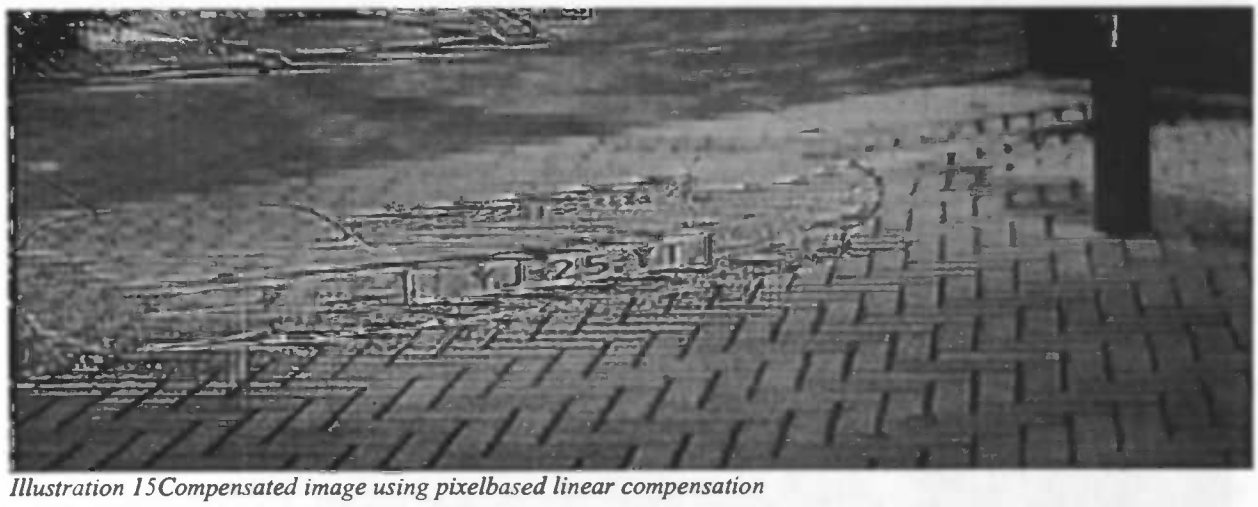


Illustration 15Compensated image using pixelbased linear compensation

The compensation has undesired side effects. The car is also compensated for, en gets partly wiped away. This is an undesired side-effect of illumination compensation. To prevent this undesired

compensation a threshold may be used. If the change exceeds a certain threshold, the pixel isn't compensated. In pseudo-code this looks like this.

```
If a> 1+ α or a<1-α then begin
                out[i,j]:= in[i,j];
end else begin
                out[i,j]:= a * in[i,j];
end ;
```

Thresholding in pseudo code

Here α is a value between 0 and 1. A value of 0.10 means a illumination change of 10 % can be compensated. A larger change in illumination will be ignored. So if α is smaller, less will be compensated. Let's take a look at the results of thresholding.

Illustration 16 boxed, linear, threshold=0.6

The car remains visible, but is still a bit too much compensated. When using a smaller threshold the results get worse, too few is compensated:

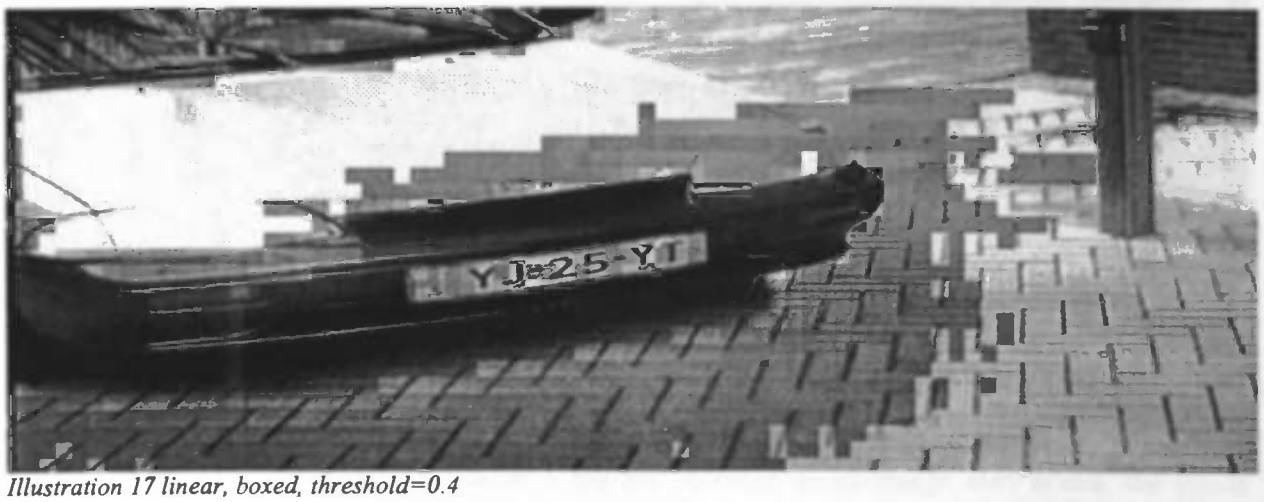


Illustration 17 linear, boxed, threshold=0.4

The car stays fine now, but the street isn't compensated enough. The quadratic method and the pixel based compensation schemes give similar result. The draw back of using a threshold, is that some illumination related changes won't get compensated. If we take back a previous input image, and use

a threshold of 0.6, we get the following result:

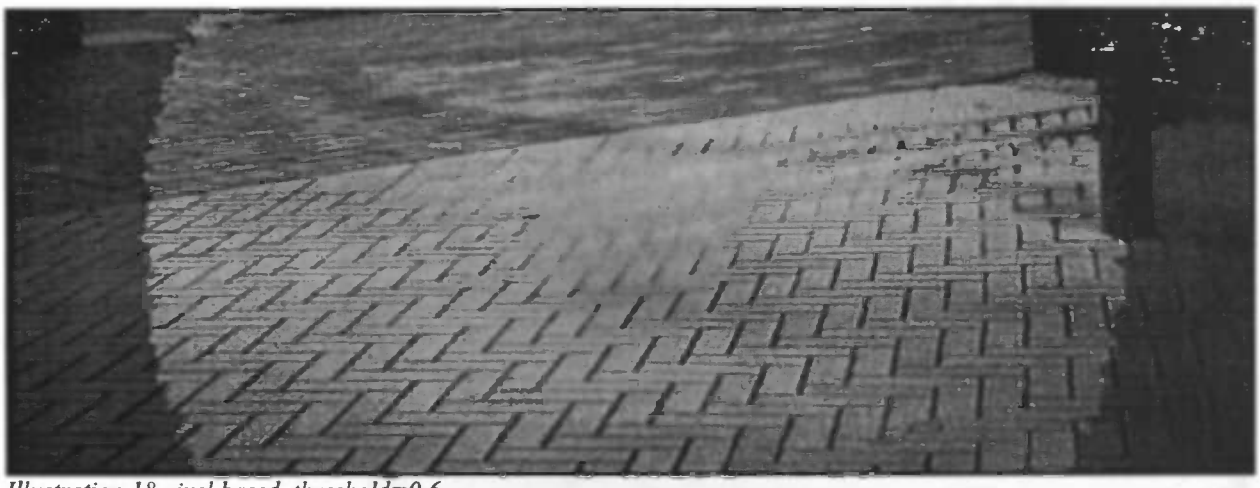


Illustration 18 pixel based, threshold=0.6

The outside isn't compensated anymore. When using a smaller threshold (car gets less compensated/ erased), the results get even worse:

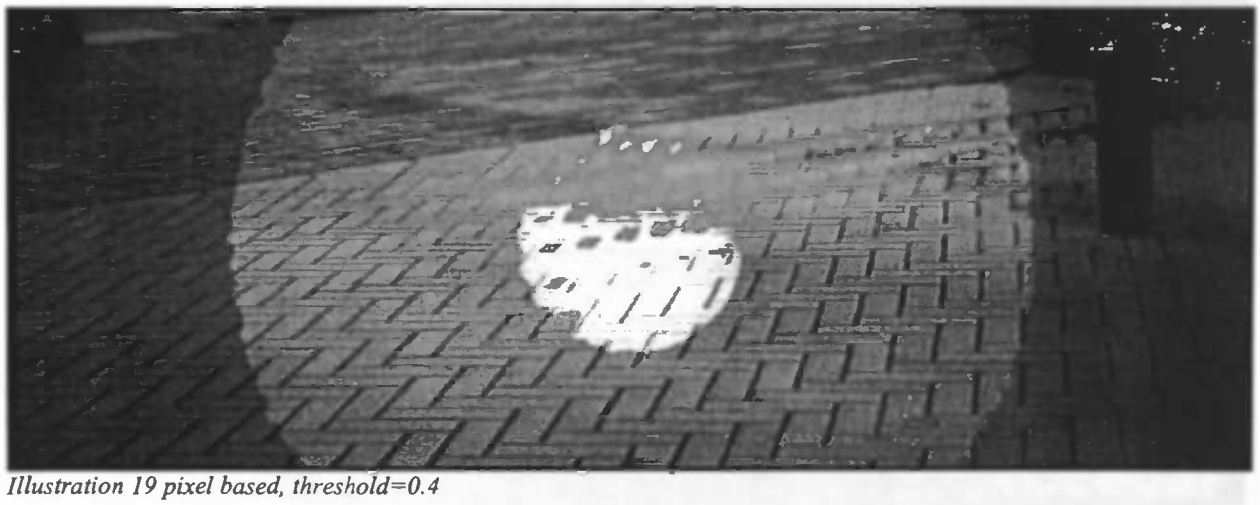


Illustration 19 pixel based, threshold=0.4

Now the center isn't changed either. We believe a threshold of approximately 0.6 to be well suitable, this way a balance between compensation and ignoring.

3.5 Hystheris threshold

Another way of thresholding is using the hystherisis thresholding method [SHB:IPA]. Hystheris thresholding uses a double threshold (t_0, t_1)to evaluate the output. Let's take a look at the pseudo code version of this algorithm:

```
1 Mark all points/boxes with change less then t1 as correct
2 S=all pixels/boxes in range [t1,t0]
3 Mark all points/boxes in S as correct, if it borders a pixel that is correct.
4 Repeat step 3, if pixels/boxes were marked in step 3.
```

Hystherisis thresholding in pseudo code

Note that this thresholding method can only be used with the box and pixel based methods. In the global method only one *a* is calculated, so there are no neighbors. The only thing remaining is to define what bordering means. When defining bordering, we can use 4- or 8-connectivity. These are illustrated in illustration 18.

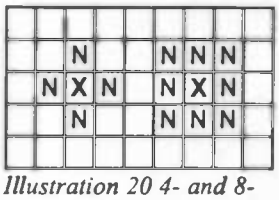


Illustration 20 4- and 8-connectivity

In this illustration **X** is considered point, and **N** denotes the neighbors of **X**.

Illustration 21Pixel based Linear compensation, using hystheresis thresholding, t1=0.2 t2=0.6

Now the cars stays in the frame, but not much is compensated.

Illustration 22Pixel based Linear compensation, using hystheresis thresholding, t1=0.4 t2=0.8

This seems to be a reasonable compromise, between compensation and keeping the car. Some parts that should be compensated aren't (right under part), while part of the car do get compensated.
One thing to note is the more fluent passage when going from compensated to not compensated. One big disadvantage of this scheme is the calculation costs. The process needs to be iterated multiple times, and so performance takes a big hit. Another disadvantage also arises: the speed increase tricks mentioned in the next section can't be used as good as in the aforementioned methods. This algorithm needs neighbor information, so when using box based compensation, multiple extra boxes need to be calculated around the tripline, which wouldn't usually have to be

calculated. When using pixel based compensation, the penalty is even worse, now lots of extra pixels around the tripline need to be calculated, and the speedup trick mentioned before, has almost no influence anymore.

3.6Speeding up the process

Visualizing the results can give a human quite some information on the performance of illumination compensation algorithms. The question however remains: how much do the results of the motion detection algorithms improve when using these algorithms? This question can only be answered after the discussion of the several techniques an can be found in chapter Performance Results. Another thing to consider is, when using a tripline only a few pixels (the ones on the line) need to be calculated, instead of the entire image. In this case it may be better to use a pixel based compensation technique, since it won't be such a big increase in computing time anymore. Take for example the following situation, first the tripline with a boxed compensation method:

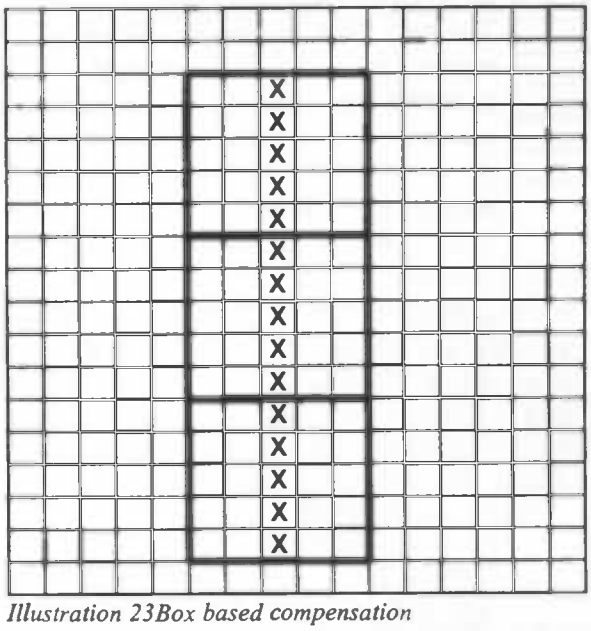


Illustration 23Box based compensation

In this figure a small tripline is used (15 pixels long) The box size is five by five. This results in the compensation in three boxes. 3 Factors need to be calculated based on 25 pixels each.. This calculation costs approximately (24+1)*3=75 operations. The 1 extra is the division to calculate the average. Notice, this measurements are based on the linear compensation method. The quadratic method should give both method the same hit in performance, so it can be safely ignored.

The situation is different when using per pixel compensation. Now 15 factors need to be calculated. Each factor is based on a 5x5 box (same size box, results in a better result). Each box takes 25 operations. Now the compensation takes 24 * 15= 360. This is a 380 % increase in computing time. This method however can be largely sped up, by using an intelligent calculation scheme. This scheme is based on the overlapping of the relevant boxes.

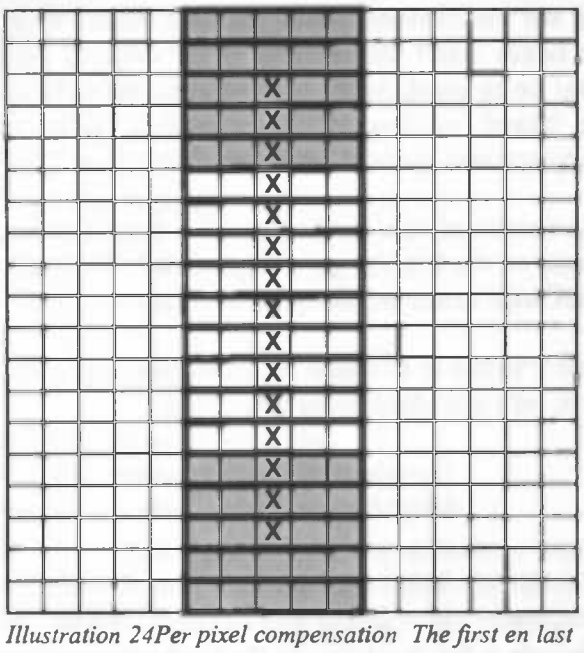


Illustration 24Per pixel compensation The first en last box are colored.

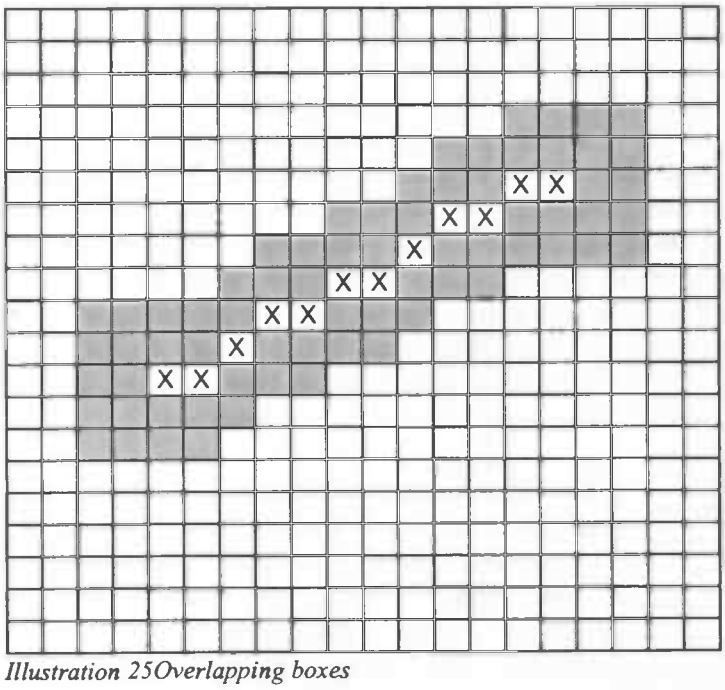


Illustration 25Overlapping boxes

A simple trick that can be applied is the following. The sum of each row is calculated (row in the boxes). The illumination of the first box can be calculated by adding the first 5 row-result. The following box can be calculated by subtracting the first row of the previous box (this one isn't contained in this box) and add the row-result of the next row. This results in a lot less calculations. First the row totals need to be calculated. This takes 15 * 5= 75 operations. Now the first box total needs to be calculated. This takes 4 additions. The next 14 boxes can be calculated in 2 operations each. The averages can be calculated in 15 divisions. So the total number of operations can be reduced to 75+4+14*2+15= 122 operations. This is only an increase of ((122-75)/75) *100% = 63%. So if the results of using this method are beneficial , the complexity increase can be acceptable.
This method can be easily used when a tripline is positioned horizontal or vertical. The method however can be extended to a random tripline. Take a look at illustration 25. In this illustration one can see the relevant rows (this time columns, see tripline points for more details). As one can see, the boxes aren't the same as in the original pixel based algorithm. Logically speaking, the results wont be as good, as the real pixel based method. The results however should still be better than in the boxed version of the algorithm, since the blocks which are clearly visible box based compensation method, won't be visible here.

The results of this method aren't a fully compensated image, as the previous methods, since only points on the tripline get compensated. We won't give a visual representation of the results, since it is difficult to see anything in the image.

Now it's time to examine the exact running times of the algorithms, when used on a tripline.
First, let's introduce some quantities.

L	Length of tripline in number of pixels.
N	Neighbourhood size, note: box size= $2N+1$

First the box based method:
We need $(L+2N)\mathrm{DIV}(2N+1)$ boxes. All these boxes take $(N2+1)^2$ operations to calculate a for that box. So the total running time will be $(N2+1)^2((L+2N)\mathrm{DIV}(2N+1))$. The div is like a division. To simplify the calculations we assume the div is a real division.The previous formula can then be simplified to: $L+2NL+2N+4N^2$.

Next the improved pixel based algorithm.
We need $L+2N$ rows of $2N+1$ pixels. The rows must be internally added, to get the row-total. This takes $2N$ operation pro row. The first box calculation takes $2N$ additions. The following L-1 boxes each take one addition and one subtraction. All the factors can be calculated from the box totals in 1 operation pro box. So the total running time will be $(L+2N)\,2N+2N+(L-1)\,2+L$.
When simplifying this formula, we get $3L+2NL+2N+4N^2-2$.

Now both complexities are known, we focus on the difference. We already augmented that the improved pixel based method is slower, so we subtract the box based solution running time from that one.

$$3L+2NL+2N+4N^2-2-(L+2NL+2N+4N^2)=2L-2.$$

Percentually this is: $\dfrac{(2L-2)}{(L+2NL+2N+4N^2)}\,100\%$
When we plot this difference, we get the following figure:

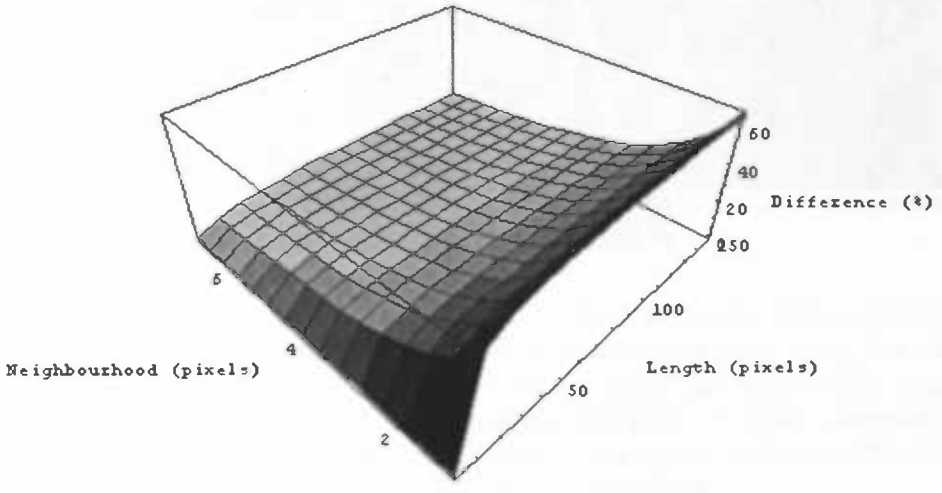


Illustration 26 difference in number of calculations between box based and improved pixel based.

A few things can be observed in this figure. If the neighborhood gets bigger, the difference will get asymptotically smaller. If the length increases, the difference will increase too, This is however also asymptotically. Let's investigate this behavior. If we take the limit of L to infinity, the formula will have the following result:

$$2\frac{L}{((2N+1)L)}\,100\%=\frac{2}{(2N+1)}\,100\%$$

If we take $n=2$, we so get an increase in the number of calculations by 40% (when using a long

tripline). Another interesting observation is, that the time increase gets smaller, when using a larger neighbourhoodsize.

To clarify this performance decrease we will give an example using a tripline which was actually used to test our sytem. It was a tripline of 396 pixels long. The used neighbourhood was a box of 5 by 5, so N=2. The increase in the number of calculations is 39.5%. Since this is only one step in our detection sytem, the performance hit on the whole process will procentually even smaller.

4 Triplines

In this chapter we will give an overview of the concept of the tripline. First we will focus on the placing of triplines, then a closer examination of the tripline is given, and last we will suggest several other shapes for the tripline.

4.1Tripline Positioning

An important question is where to place the tripline in the viewable area. In some situations the answer is quite straight forward. Take for instance a traffic surveillance system, which is located on top of a speed way. A tripline should then off course be positioned perpendicular to the traffic flow, spanning the whole track, see illustration 27.

Illustration 28Tripline on lane

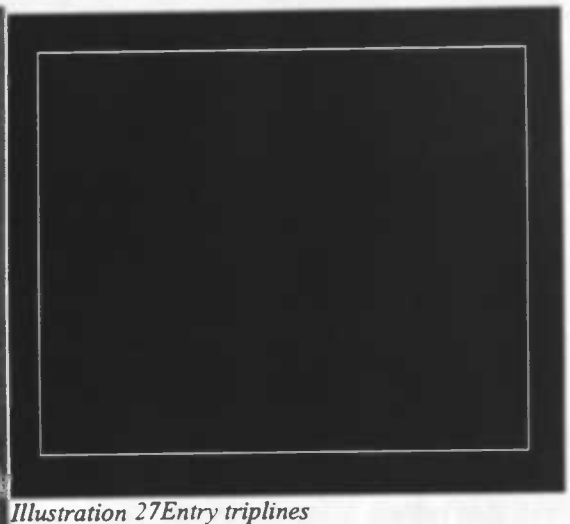


Illustration 27Entry triplines

When using the system for security surveillance purposes, the question gets a little bit more complicated. A simple solution would be to position the lines along the edges of the image. This way, when something enters the scene, it will be detected immediately. A problem however arises, illumination compensation is based on surrounding pixels. It makes therefor more sense to place the tripline a little bit inward the picture, otherwise the points on the tripline won't have neighbours on one side. See illustration 28 for the placing of the triplines.

This way most entries into the observed scene will be reported. Only when something enters the scene, and almost immediately leaves it again at the same side, it won't be detected. This placing seems to work good, but there is one small problem when objects enter the scene from the middle of the view. This can happen when somebody leaves a building through a door, when a car leaves a garage and when occlusion can happen at the side points. To illustrate the previous situations, we have made the following pictures.

Illustration 30 Occlusion causes middle entries

Illustration 29 Middle entry through door

Another problem is the sky, most of the times, the camera will be aimed at the ground, so the sky won't be visible in the picture. The camera however can be placed, so the sky is visible too. In this case, clouds will be detected , since a tripline was placed along the upper edge. This is off course an undesired side effect, and a false detection. Now also object can enter the scene from the horizon (if it isn't blocked by other objects). Another way of placing the triplines, is using a human supervisor to place the lines. This supervisor can recognize objects in the scene, and can place triplines on pavements and roads. He will also refrain from placing illogical triplines. Take for example the next scene in illustration 31.

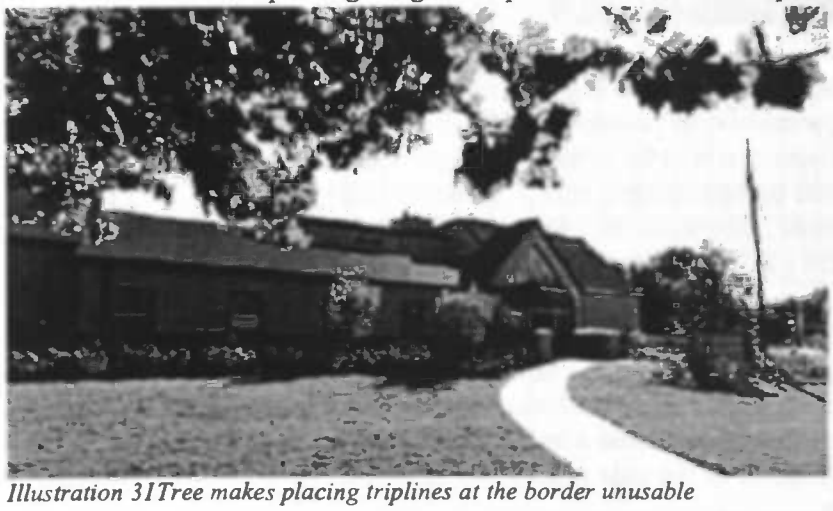


Illustration 31Tree makes placing triplines at the border unusable

If the tripline is placed along the edge, it will probably (depending on the algorithm used), result in many false detections. The leafs of the tree blow in the wind, en generate a lot of noise. Also it can block real objects that enter the scene. The triplines need to be positioned at another place. The human supervisor can place the triplines in more logical positions. Roads, pavements are off course a good position. A problem arises however when using the system for security surveillance. A burglar may enter the scene at an unexpected position. He may climb over a fence, or cut through it, instead of using the door and pavement. The supervisor should take this in account when placing triplines. The lines can then best be places near the buildings. Again at the places where a door marks an entrance to a building , it isn't enough to place a tripline there. The burglar may break into a window. It is impossible to make an objective statement about placing triplines. In some cases (road/ traffic surveillance for example) it is easy to

place the triplines, but in other cases (security surveillance fo example) it's quite difficult.

4.2 Automatic placing

Another way to determine the interesting places for triplines in a view, is by first using a calibration stage. In this calibration stage a global motion detection algorithm, can be used to determine the regions in which there is movement. These regions then need to be covered by triplines.

This method has however some disadvantages:

- Triplines can get to be positioned over uninteresting regions, for example a tree blowing in the wind. Since the tree generates motion, the tripline will be positioned over the three.
- In security applications the common event is no detection, there isn't a trespasser too often. This means the calibration stage can take a lot of time, or will get very inaccurate.
- The global algorithm, will need more computer power, then the tripline-algorithm. This implies a faster then necessary CPU is needed, or a special calibration box, that can be attached to the unit.

Taking all the disadvantages of automatic placing into account, it is clear that placing the triplines by a supervisor is the preferable option.

4.3 Size of triplines

The length of a tripline is very dependent on the actual situation. For traffic surveillance, it is clear the tripline needs to stretch out over a lane. For surveillance applications it is a bit more tricky. Take for example the exit covering triplines in illustration 27 . These triplines are very long. As we will show in the next chapter, most algorithms use the average difference pro pixel. This implies, that when using long triplines, motion might be ignored, since it is located in only a small area of the tripline. The placing in illustration 32 should give better results.

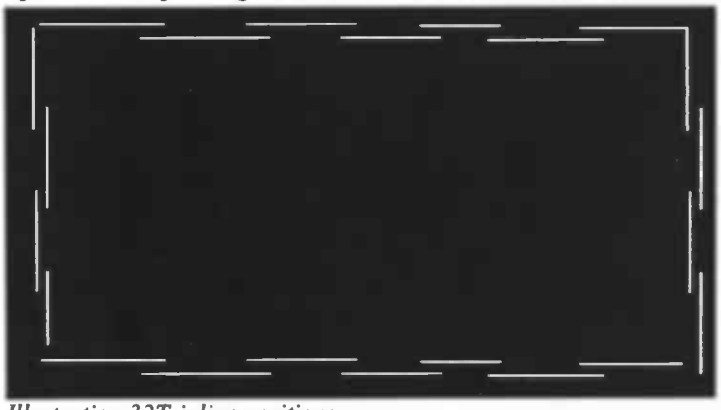


Illustration 32Tripline positions

Here we use several smaller triplines, which slightly overlap. This overlap is done, so that motion near the separation point of two triplines will be detected on both lines. If we would just align the triplines next to each other, these might go undetected, since it is a small change (only half the real change in both lines) . It is of course possible to really let the triplines overlap on several pixels,but that can't be visualized clearly in a black and white image. In practice we use overlapping triplines in this case.

4.4 Tripline Point positions

In this section we will focus our attention to the translation of the concept tripline to the actual usage of triplines. A tripline can be arbitrarily placed upon a picture. The tripline begin- and endpoint should result in a sequence of points to be examined. When using a horizontal or vertical tripline, these points can be easily determined. Difficulties however arise when using sloping triplines. Now we will suggest two methods , which can be used to determine the points. These methods are interpolation and handy-placing. Let us first focus on handy-placing.

4.5 Handy-placing

When drawing a line on a computer screen, similar problems arise when determining the pixels to be colored. The computer screen is a discrete grid of pixels. The simplest way of drawing the line is determining the largest difference in pixels, by comparing Δ y and Δ x (Δ x= | x_1-x_2|, Δ y= | y_1-y_2|). The largest of the two will be the number of considered pixels. These pixels can be found by rounding. Let's use an example to clarify this. Take begin point = (0,0) and endpoint (5,2). It's clear Δx is the largest, so we walk along that dimension. The directional coefficient is 2/5. The next logical point would be (1,2/5) but that pixel doesn't exist. Instead the pixel position gets rounded. So the next considered pixel will be (1,0). The third pixel will be (2,1). The full sequence will be (0,0), (1,0),(2,1),(3,1),(4,2),(5,3). In the following figure, several tripline with different dc's are shown.

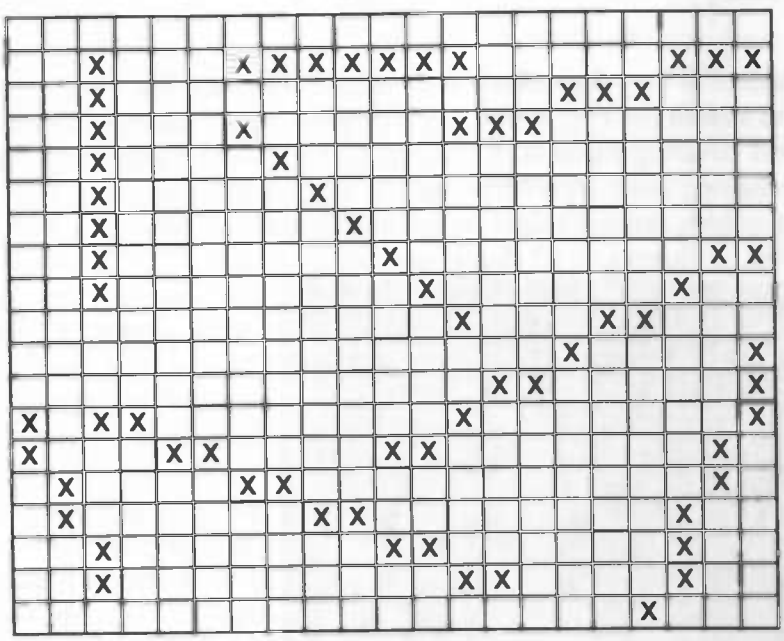


Illustration 33Several triplines

Generally speaking, if $x_1<x_2$ and $\Delta x \geq \Delta y$, the points of the tripline will be:

$$\{(X_1+I, Round(Y_1+I\ \frac{(Y_2-Y_1)}{(X_2-X_1)}))|I\in\mathbb{N}\wedge I\leq(X_2-X_1)$$

The other point sets can be derived from this. If $x_1 > x_2$, the roles of the starting and endpoint are

just switched. If $\Delta x < \Delta y$ the formula will look like this ($y_1<y_2$)

$$\{(Round(X_1 + I\ \frac{(X_2 - X_1)}{(Y_2 - Y_1)}), Y_1 + I) | I \in \mathbb{N} \wedge I \leq (Y_2 - Y_1)$$

These are all standard linear functions.

4.6 Interpolation

The afore mentioned problem can also be solved by interpolation. When a point is needed which doesn't exist, it can be interpolated based on the surrounding pixels. Several interpolation schemes exist. We will use a simple linear interpolation scheme to calculate the value of a not existing point. First a definition of terms: (x0,y0) is the rounded down value of the desired point (x1,y1). (x2,y2) is the rounded up version. So the four considered points (linear 2D interpolation) are (x0,y0),(x0,y2), (x2,y2) and (x2,y2). The value of (x1,y1) can be determined by the following formula:

$$Output = f(x0, y0)*(2 - x1 + x0 - y1 + y0) + f(x0, y2)*(2 - x1 + x0 - y2 + y1)$$
$$+ f(x2, y0)*(2 - x2 + x1 - y1 + y0) + f(x2, y2)*(2 - x2 + x1 - y2 + y1)\ \frac{1}{4}$$

This formula is based on linear interpolation. The 4 surrounding points are weighted by their distance from the desired point. Notice, this weight is based on the average of the distance in x and y direction. We off course take 1 - distance, because we want the weight to be smaller if the distance is greater.
Note: the number of points on the tripline needs to be selected beforehand, since we can determine a arbitrary number of points between any two points.

4.7 Evaluation Tripline point positions

Although all the interpolation algorithm has great beauty, it remains to be seen, whether it will be necessary. We doubt it. The handy placing will probably be preferential. The computing costs are the lowest of the two methods, but the result will probably remain the same.

4.8 Other Shapes

Till now we have dealt with triplines, which are one line of pixels. We can however also use thicker triplines of two or three pixels in height. These thicker triplines imply a higher calculation cost, so we should clearly weigh the performance increase against the calculation cost increase. We can however go even further. Till now we looked at lines as triplines, but since a tripline algorithm just works on a sequence of pixels, a tripline can have an arbitrary shape. In fact it doesn't even have to be connected shape. It can be just about any set of points. It is doubtful these arbitrary point sets will generate good results, but in certain fields other shapes might be quite useful. For traffic

surveillance the line is still the best form, but for security surveillance some other forms are useful. Take for example the the situation in illustration 34.

Illustration 34Burglar detection using a sinoid like tripline

In this case the sinusoid-like shape ensures that only bigger objects are detected. Other shapes which might be useful are the circle, triangle (in fact any polygon could be considered for use) and round, twisty lines.

Now let's focus our attention on some more arbitrary, but still line-like triplines, we come to the sort of triplines shown in illustration 35. These are still, clearly vertical lines, but use extensions, to improve performance. In some cases the number of pixels isn't extended, and in some cases it is. The following table shows the difference in pixels pro "line". Type 0 is the simple straight line, as we have seen in the previous section.

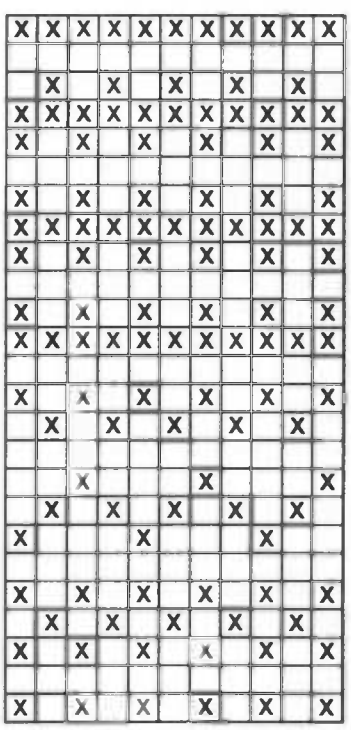


Illustration 35Different triplines

Type	*Number of pixels*	*Percentual increase to standard line*
Type 0	11	0,00%
Type 1	22	100,00%
Type 2	22	100,00%
Type 3	17	50,00%
Type 4	11	0,00%
Type 5	11	0,00%
Type 6	17	50,00%
Type 7	6	-50,0%

These increases in the number of pixels also imply a similar change in calculation costs. This differences between the algorithms, based on their complexity.

5 Algorithms to use on triplines

In this chapter several algorithms to be used on triplines will be introduced and discussed.

5.1Types of algorithms.

Roughly spoken, two types of algorithms can be distinguished.

1. Reference based algorithms. These algorithms use a single reference image. They determine a metric for the encountered differences. If this difference is larger then a certain threshold, it is reported. Two types of reference image can be distinguished: a static pre- selected reference, or an adaptive reference. This adapted reference is changed during the process.

2. Sequence based algorithms. These algorithms work on a sequence of images. These are mostly statistics based. They try to guess the probability, the input image is the next one in the sequence. If it is unlikely to occur, an object has entered the scene. Examples of Sequence based algorithms are Approximate entropy and algorithms using hidden Markov models.

5.2 Thresholded difference

The simplest difference detector, is the one based on differences. By just calculating the differences on the tripline, compared to the reference image, we get a measurement of the changes that occurred. By selecting a suitable threshold, one can separate detections from nothings. The advantages of this scheme is the fast calculation of the measurement. The biggest disadvantage is its applicability. When used with an illumination compensation scheme, this method might bring some good results, but without it, it will get lost very soon. Another drawback situates in the ability of people/cars to disguise themselves in the same color as the background. Since the threshold needs to be rather large, to prevent false detections, this threshold prevents those objects to be detected. To ease the use of this scheme, we calculate the average difference on a tripline. This make a certain threshold usable for all lengths of triplines:

$$Average\ Difference = \frac{(\sum |x_{ref}(i) - x_{input}(i)|)}{n}$$

Now the threshold needs to be chosen. A to small threshold causes false reports, while a high threshold causes several events to be ignored. We will investigate the response of the algorithm to a sequence of images in which a car enters the scene. In illustration 36 the output of this algorithm on a sequence of 45 images is shown. About and around image 16 the car starts to cover the tripline. Based on the response of the algorithm, a threshold of around 40 is usable. Then image 16 isn't causing a trigger, but that's no problem, since the car is only partially covering the tripline.

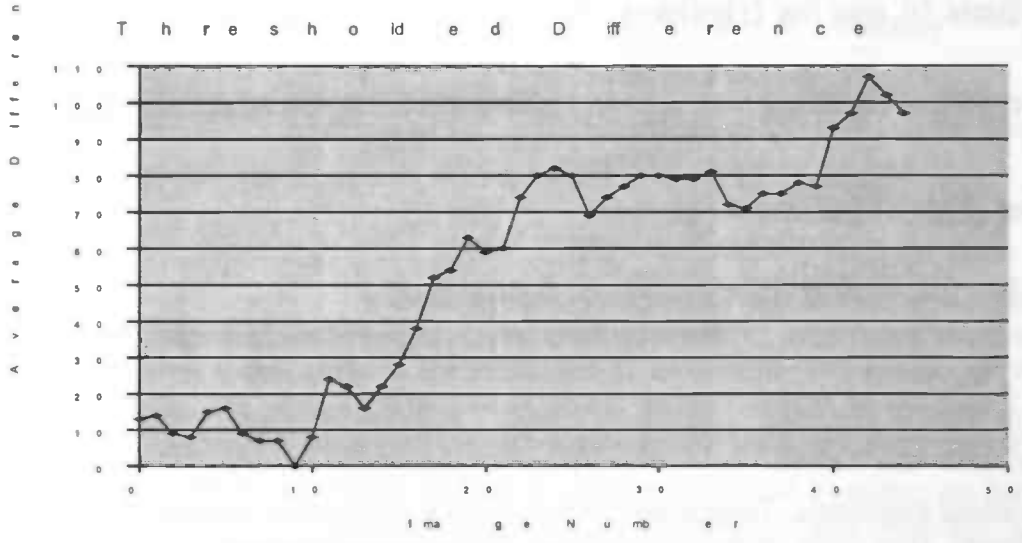


Illustration 36Response of Thresholded Difference algorithm

5.3 Approximate entropy

In [N:MDuAE] a motion detection algorithm using approximate entropy is devised. Approximate entropy (ApEn) is a metric to measure irregularity in a sequence. It is a sequnce based algorithm. The basis of using the ApEn calculating algorithm on the tripline is, that an intruder won't be a common event, i.e. an irregularity. This will result in the detection of the intruder. Approximate Entropy is a statistical algorithm, it tries to measure the chance of the occurrence of an event.
In the aforementioned article mister Ngan suggests a windowed version of ApEn calculation.
At every moment in time, W sequence values are considered. The evaluated time instance is situated at the middle of this window. This measurement will be explained using a tripline considering a single point. In the end we will extend it to the generic case.

$u(i)$ is the value of the only pixel on the tripline at time i.

We define U as the sequence of inputs:
$U = u(0)..u(N-1)$, where N is the number of input images available.

We are using a windowed version of the Approximate Entropy Algorithm, so we will select a window W out of this input sequence, this is in fact a subsequence:

$$W_l = u(l)..u(l+ws-1)$$

Where ws is the window size, and l the start point of the window, note $0 \le l \le n-ws$. Within a window, a number of blocks is considered, which are essentially windows within the windows. We define a block X as:

$$X_{l,i} = w_l(i)..w_l(i+m-1)$$

Where m is the block size, and i the start position of this block. So essentially, when substituting the

definition of a window:

$$X_{l,i}=u(i+l)..u(i+m+l-1)$$

When we take two blocks within a window, we can define the distance between these blocks to be:

$$d(x_{l,i},x_{l,j})=\underset{k=0..m-1}{MAX}|x_{l,i}(k)-x_{l,j}(k)|$$

So the distance between two blocks is the maximum of the differences between all respective entries in the two windows. For the main algorithm a threshold is necessary. This threshold is dependent on the standard deviation (SD) of the inputs.

$$T(l,r)=MAX(SD(W_l),r)$$

Where r is the minimal threshold, which is a parameter of the algorithm, together with block size and window size. The standard deviation is a measurement of the spread inside a dataset, so by relating the threshold to this standard deviation, we can make the algorithm independent of the input. We don't need to select a fixed threshold. We now define the frequency (= how many times) with which the distance exceeds the threshold *T(l,r)* to be:

$$C_{l,i}^{m}(r)=\frac{|\{j|d(x_{l,i},x_{l,j})\le T(l,r)\wedge j\in\mathbb{N}\wedge j<ws-m+1|}{ws-m+1}$$

The <= is used here because we are still working with chances, a chance lower then the selected threshold is less likely.

We normalize this formula over all boxes, this is in fact the average of the output of all boxes:

$$\Phi_{l}^{m}(r,ws)=\frac{1}{ws-m+1}\sum_{i=0}^{ws-m}C_{l,i}^{m}(r)$$

The complete metric is defined as:

$$ApEn(m,r,l,ws)(u)=\begin{cases}\Phi_{l}^{m}(r,W)-\Phi_{l}^{m+1}(r,W) & m\ge 1\\ -\Phi_{l}^{1}(r,W) & m=0\end{cases}$$

The output of this algorithm must be interpreted as followed:

Output	*Meaning*
$ApEn \leq 0.2$	Static Background
$0.2 < ApEn < 0.4$	Moving parts, this is in fact the thing we want to detect.
$ApEn \geq 0.4$	Temporal Clutter, this is moving parts, but it's present during the whole sequence, take for example the leafs of a tree blowing in the wind.

Table 1 Approximate Entropy Output meanings, see illustration 37 and 38.

The above mentioned algorithm works on a tripline consisting of one pixel, we will now extend it to the generic case. We just run the algorithm on all points of the tripline independently. Afterwards, we just need to combine this results. A logical combination would be taking the average over the individual results, but that doens't work, due to the special output of this algorithm. The average of the output on temporal clutter and the output on static background can be equal to the output of moving part (see table1). This is an undesired side effect, which makes averaging impossible. Instead of averaging, we count the number of pixels that result in an output value of "moving parts". This way the algorithm delivers an output like most other algorithms: a number. This number can then be thresholded, and output can be filtererd afterwards.

The major disadvantage of this algorithm is evident: high calculation costs. By using this algorithm on a tripline, a speed boost will be accomplished. The speed however will be suboptimal, so it must perform pretty good, to be of any use.

Illustration 37

(a) Greyscale image at t=120,

(b) Output of the ApEn algorithm,

(c)Static background calculated by selecting ApEn < 0:2,

(d) Moving object mask calcu-lated by selecting 0:2 < ApEn < 0:4,

(e) Temporal clutter mask calculated by selecting ApEn >0:4.

Illustration 38 Example images from the imagesequence used to demonstrate the ApEn algorithm.

5.4 K-S test statistic

The Kolmogorov-Smirnov test statistic is defined as:

$$D_{KS} = \underset{I}{MAX} |P_1(I) - P_2(I)|$$

Where $P_1(I)$ and $P_2(I)$ are the cumulative histograms of the first and second images respectively. The K-S test statistic is relatively straight forward statistic.

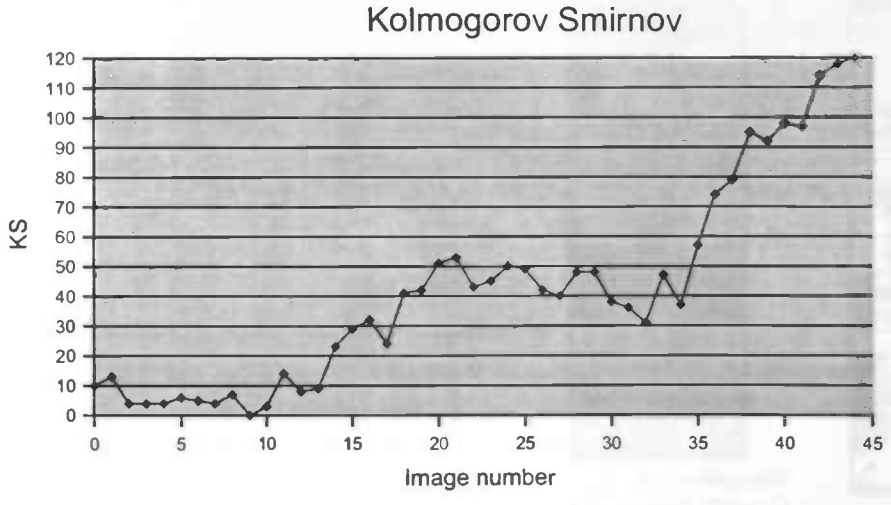


Illustration 39Response of Kolmogorov Smirnov Algorithm

If we take a look at the output of this algorithm (illustration 39), a threshold of around 25 is suitable, to detect the car entering the tripline at image number 16.

5.5 Smirnov's ω^2 statistic

The Smirnov's w^2 statistic is defined as:

$$W\frac{2}{S} = \sum 0 \le x \le N : (g(x)\frac{[R_1(x) - R_2(x)]^2}{n})$$

Where $R_1(x)$ and $R_2(x)$ are the intensities of the x^{th} ranked data point in the first and second distribution respectively. So $R_1(0)$ is the lowest value on the tripline in image 1, and $R_1(N-1)$ is the highest value on the tripline in image 1. This is why the intensities on the tripline need to be sorted on intensity. N is the number of pixels on the tripline, and g(x) is a weighting function. Typically g (x) is a constant.

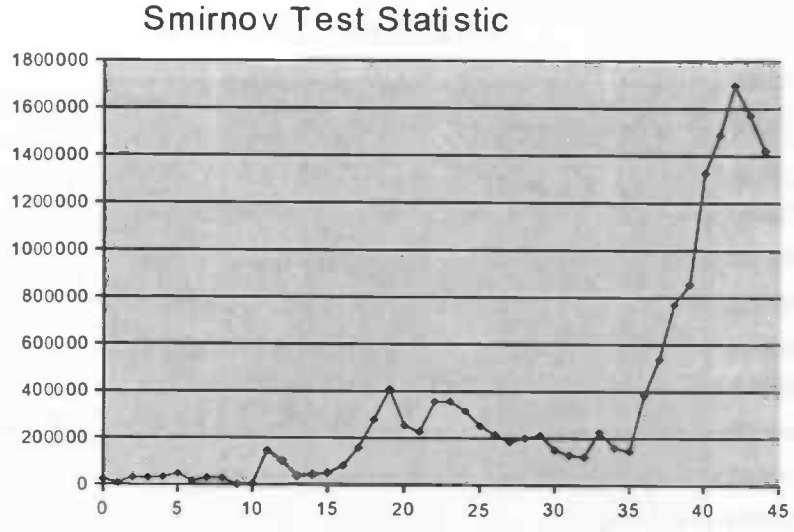


Illustration 40Response of Smirnov test statistic

If we take a look at the output of the algorithm (illustration 40) we can conlcude the following. This response isn't very good, because there is a peek at image number 11, while the car doesn't cover the tripline until image number 16. That's why selecting an appropriate threshold is impossible: sthe peek at image number 11 is higher then at image number 32. At image number 32 the car still covers the tripline. We need to test this on a bigger test set, to see whether or not this is an exception, or a nasty feature of the Smirnov test statistic.

5.6 Modified Smirnov Test Statistic

A modified version of the Smirnov statistic also exists:

$$W_{MS}^{2} = \sum 0 \le X \le N : \frac{(g(x)[I_1(x) - I_2(x)]^2)}{N}$$

This time x is an index of spatial location on the tripline.$I_i(x)$ is the intensity of the pixel at position x, soixel pairs on the same position are considered. This statistic is therefore quite reactive to camera movement. Note that this is a special case of Thresholded difference.

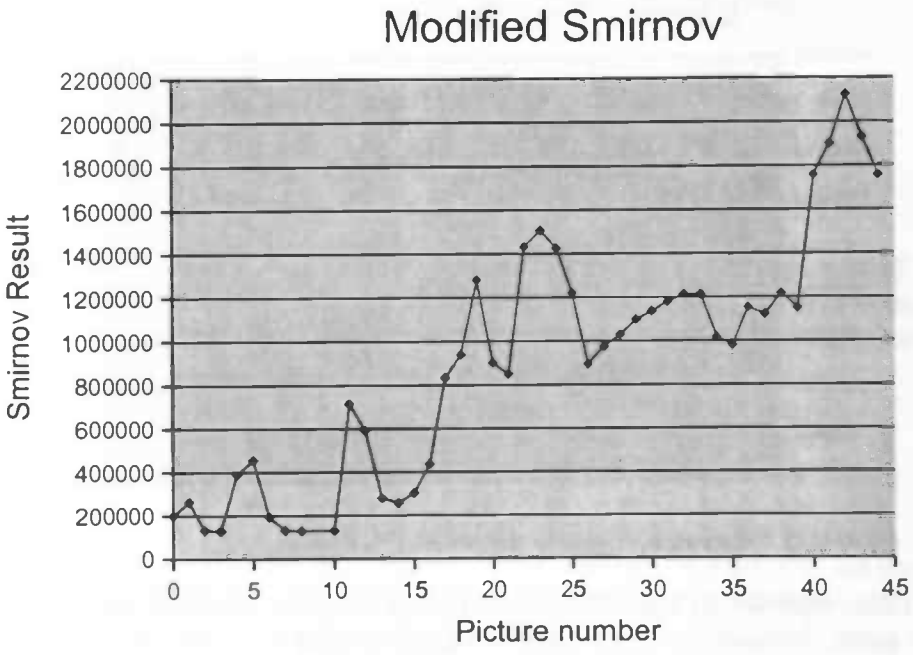


Illustration 41Response ofModified Smirnov Algorithm

The response is still not great, but it is better than with the original Smirnov statistic. All image after 16 result in an output higher then the peek at image number 11. We can consider image 16 to be an intermediate case (see Chapter: Three valued Logic).

6 Post processing

After the algorithm has determined his output (either boolean, or valued). One can post process these data. When using valued output, a method is required to transform these values to boolean output. This will be dealt with in the first sections of this chapter. Next we will focus our attention on boolean output, which must be postprocessed.

6.1 Determining the best Threshold

Almost all algorithms result in a numeric output. To get from this numeric output, to a boolean output, a thresholding scheme needs to be used.

6.2 Simple Threshold

The simple threshold just takes a threshold level. When the output exceeds this threshold, True is delivered, otherwise false. The only problem is determining the right threshold. We have used the brute force method to do this, i.e. we just try all possible thresholds (minimum output < Threshold < maximum output), and determine the best. The problem is this threshold, is determined for a certain data set, so the question remains, whether it will be a good threshold for an other data set. This measurement we will call the generalizing ability of the method used, i.e. does the algorithm give the output, independent of the input data set used. We have devised the following formula to give a measurement for the generalizing capabilities of an algorithm. Here P_1 and P_2 are the performances on data set 1 and 2 respectively, and P_{1+2} is the performance on the glued together output set, i.e. the output values of both methods concatenated.

$$Average\ Performance = \frac{P_1 + P_2}{2}$$

$$GenCap = 100\% \ \frac{2\ P_{1+2}}{P_1 + P_2}$$

So GenCap is the relative performance on the glued data set, compared to the average performances of the separate sets. Note that in order for Average performance to be correct the data sets need to be of approximate the same size.

6.3 Hysteresis Thresholding

As with the illumination compensation, we can also use hysteresis thresholding here. One result of using hysteresis thresholding is that the output will lag behind the input, since the hysteresis thresholding needs to be done on a sequence of frame outputs. That's why we need a maximum number of steps to go back. The plain hysteresis algorithm uses an infinite number of steps to go back, but we need to contrain it. We only have one dimension in this case, as opposed to the hysteresis thresholding in the illumination compensation section. That's why we can't used eight or four connectivity. Instead we use a simple neighbour algorithm, an output value is has 2 neighboors,

on before, and one after. The remainder of the algorithms stays the same.

6.4 Filtering

A very common method to improve results, based on the boolean output of an algorithms, is filtering. We will discuss two methods of filtering: sequence filtering and average filtering. Both algorithms are based on the fact, that a detection won't be singular event. If a car passes by, it will generate more then one detection, several consecutive frames.

6.5 Sequence Filtering

The sequence filter removes all true's from the output, except the ones who are part of a sequence of truths with at least a length L . This can best be clarified with an illustration.

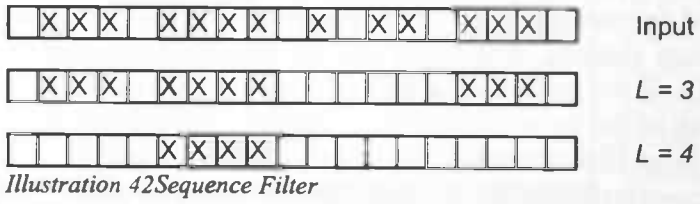


Illustration 42Sequence Filter

As one can see, the sequences with at least length L are kept.

6.6 Average Filtering

Another filter is the average filter. This filter determines the average in the neighborhood W of the output. This neighborhood is centered around the actual output. The output it self is measured too, for the average and neighborhood size. When the average is above a certain threshold, the output becomes true. Another way to look at this, is by just counting the truths in the neighborhood, and thresholding this with an other parameter: N . This N gives the number of truths required for the output to be true. Let's take a look at an example.

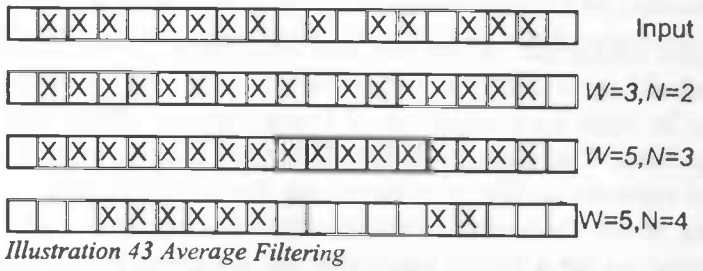


Illustration 43 Average Filtering

Other then the sequence filter, the average filter also adds true's to the result.

6.7 Filtering: The Preliminary conclusion

Performance of these filters can be examined in the chapter Performance Results. One interesting observation about the two of them has to be made now however. The sequence filter only throws away detections, while the average filter also introduces new detections. In other words, the sequence filter can only filter out FD-errors, while the average filter can also filter out FI-errors. Note that the sequence filter can also introduce FI-errors, while the average filter can introduce FI- and FD-errors. See chapter Performance measurements for more details.

6.8 Postprocessing: The prelimenary Conclusion

When combining the postprocessing methods (thresholding and filtering) we can see the parameters of both methods are related to each other. We can distinguish the following cases:

Threshold	*Filtering*
Higher	Average filter that adds extra detection / Sequence filter with low length
Average	Combined average filter (deletes/adds) / Sequence filter with average length.
Lower	Average filter that removes detections / Sequence filter with large length

Table 2 Link between thresholding and filtering

Note that the sequence filter can only delete detections, so it will work poorly when using a higher threshold. It's all about when to remove the false high output of the algorithm, during thresholding or during filtering. The question remains which case to use, in the next chapters we will give an answer to that question.

7 Alternative Commercial Systems

In this chapter we will describe some other systems which are commercially available.

7.1 3M Microloop

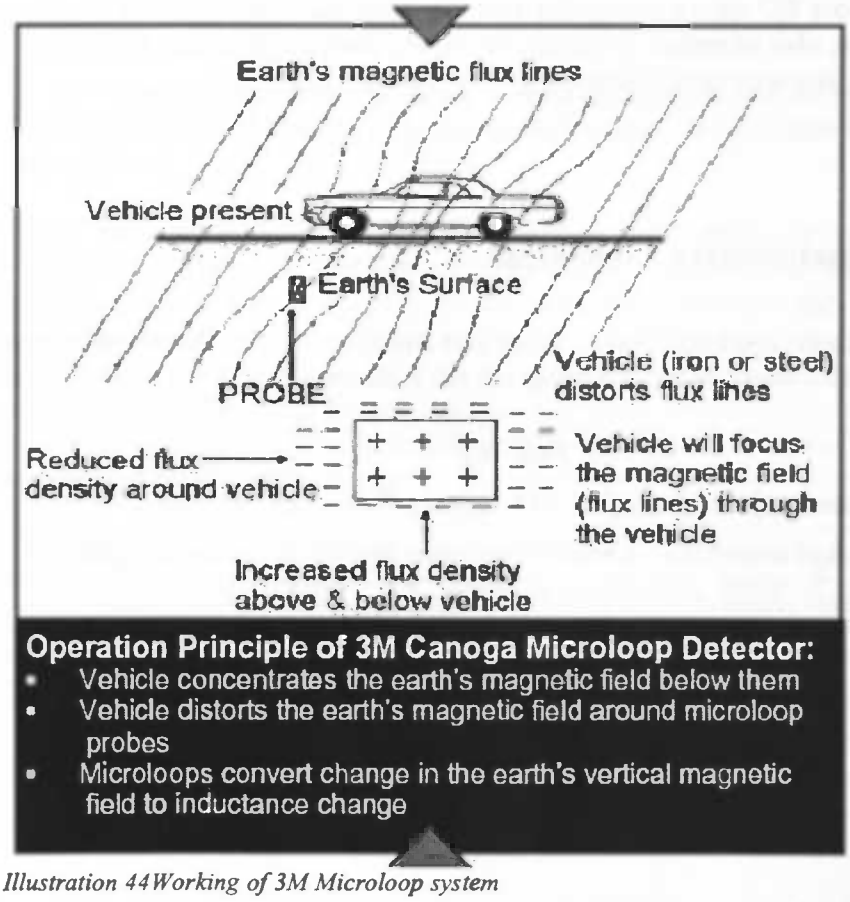


Illustration 44Working of 3M Microloop system

The 3M Microloop system uses probes in the road, do determine passing objects. So this is not a VMD system. It is however interesting to take a look at the performance of this device in comparisson to our VMD tripline system. The biggest disadvantage of this system is off course the installation. The road needs to be closed for some time, since the probes need to be installed under the road. When constructing a road, those probes need to be placed, because they won't cause problems at that time.

7.2 VideoTrak

The Peek VideoTrak 900 is a video vehicle tracking and detection system. The camera used with this system was a Philips TC590 series high-resolution charged couple display (CCD) monochrome camera using a 1/3-inch format lens with an 8 mm focal length.The camera was equipped with an auto iris and infrared filter. It was installed the Peek 40 ft above the roadway on a 15-ft mast arm

7.3 SAS-1

The SAS-1 is a passive acoustic (listen only) detector that mounts beside the roadwaym with the

capability of monitoring up to five lanes from its sidefire orientation. The detector needs to be mounted as high as 35 ft above the roadway to accurately monitor five lanes. It was mounted 20 ft above the travel lanes because the detector was monitoring only two lanes and because of the mast arm's height. Its offset from the right lane was 25 ft (as measured at a 90-degree angle with the roadway). After test results became available, the vendor suggested that presence detection accuracy would have been better with a height of 25 ft to 30 ft and smaller offset.

7.4 AutoScope

AutoScope is a VMD-system. It uses several motion detection algorithms. The autoscope system is one of the best VMD-systems available today.

Illustration 45The AutoScope Camera

7.5 Inductive Loops

The inductive loop is actually the physical tripline. Real wires are put in the pavement. They detect cars/objects on the basis of induction. In our performance comparisons, we have incorporated two inductive loop variants, the ones manufactured by TTI and Hughes.

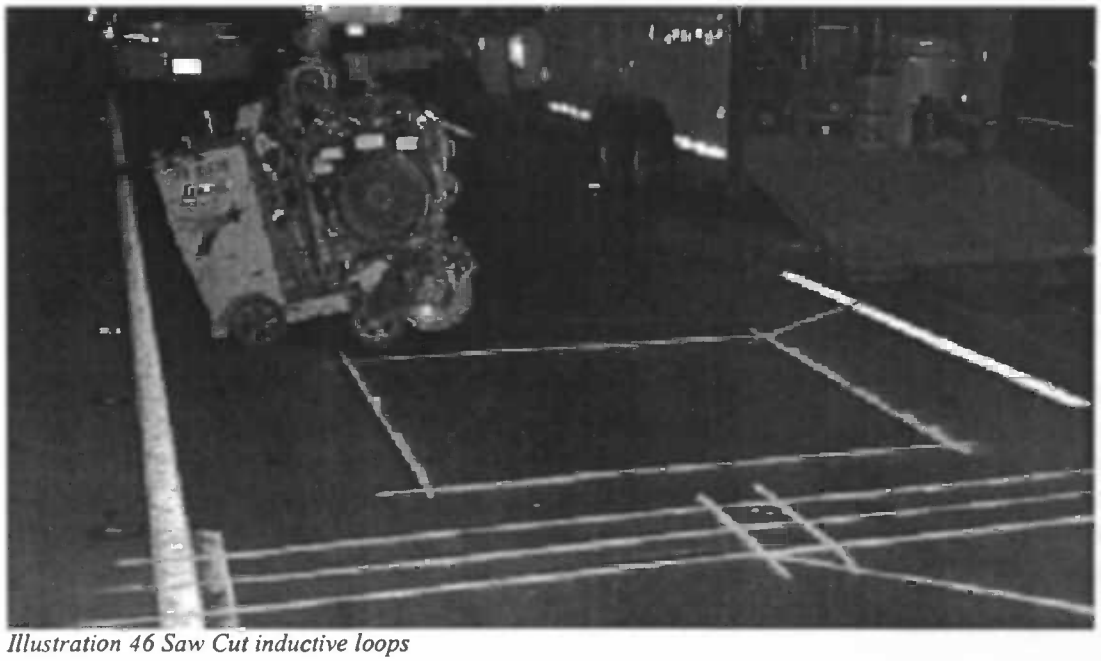


Illustration 46 Saw Cut inductive loops

8 Performance Measurement

Devising algorithms is one thing, performance measurements are a wholly different thing. In order to make statements about performance objective performance or error measurements need to be defined. In this chapter we will give an overview of several measurements, and argue about their applicability. First we will give an introduction to error measurement.

Two types of errors in the detection process can be distinguished :

- Falsely detected (FD-error): Nothing interesting happens, but the system issues an alarm.
- Falsely ignored (FI-error): A suspect (innocent until proven otherwise) walks by, but the system ignores him.

The system must be balanced between creating both errors. An important issue are the falsely ignored errors. These are much worse then falsely detected. The falsely detected-error only takes more resources: a human operator needs to examine the recorded data, or the data needs to be recorded on tape/hard drive. Respectively this takes time, tape or drive space. The real problems occur when important events are ignored. The corresponding data isn't recorded by the system, or inspected by a supervisor. There won't be any mean of identifying the burglar/intruder afterward. One can always record all the recorded data, but that's the reason the system was build: saving resources (tape).

In the following sections, the following abbreviations will be used:
GD : Good detected. An image that has been declared as "something" is reported as "something"
GI : Good Ignored. An image that should be ignored is ignored.
FD : False Detection. An image which should be ignored, is qualified as "something".
FI : False Ignored. An image that has been declared as something is reported as "nothing".

8.1 Good Classified

Away to measure the performance is to calculate the percentage of rightly identified images.

$$Good\ Classified = 100\,\% \ \frac{(GI + GD)}{(NrOfImages)}$$

This is a percentual performance measurement, so a value of 100% means a perfect score. One problem however arises. We can assume the number of images that should be qualified as nothing is much larger then the number of images that should be qualified as something. So, when the algorithm reports nothing, for all images, the performance will still be pretty good. A solution to this problem is separating the performance on "nothing" images, from the performance in "something images". The following performance measurements use this scheme.

$$GIPerc = \frac{GI}{(GI + FD)}$$

$$GDPerc = \frac{GD}{(GD + FI)}$$

$$Average\ Performance = 100\% \ \frac{GIPerc + GDPerc}{2}$$

$$Product\ Performance = 100\% \ \underline{GIPerc * GDPerc}$$

The GIPerc gives the performance on the nothing images, while the GDPerc gives the performance on the something images. We can combine these to percentual performance measurements. The first one averages both performances, while the second one uses the square root on the product of the two performances. In the following figures the difference between the both can be clearly observed:

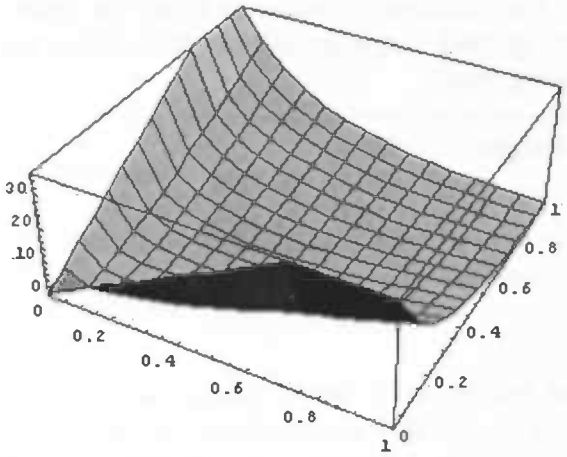


Illustration 47Performance average - Performance product against fd and fi

As we can see in illustration 47 the differences between the both performance measurements occur near the 0-lines of the FD and FI axes. This is logical, since the Product Performance gives 0 as results, while the Average Performance gives the half of the other error (either FD or FI). When the FD is about equal to the FI error, both performance measurements give almost the same result. This off course quite logical, since $a\ \underline{a} = \frac{a+a}{2} = a$.

8.2 Counting-error

Another way to measure performance is to count. Take for example a camera which is mounted hovering a freeway. The software can based on detection, count the number of cars that passed by. When comparing this to the ground truth count, we can use the following measurement:

$$E_{cnt} = 100\% \frac{|Cnt_{det} - Cnt_{groundtruth}|}{Cnt_{groundtruth}}$$

This measurement is often used to evaluate several systems, take for example [Texas]. This measurement however has a big disadvantage: two errors can combine to zero errors. If the algorithm ignores one car , but qualifies a shadow as a car, the count isn't changed (1-1=0). This disadvantage has often been ignored, but it completely disqualifies this method, to be of any use. Take for example the following result:

Total number of examined images	: 9854	
Correct	: 9773	98.32%
Good Detected	: 81	35.37%
Good Ignored	: 9487	99.8%
Falsely Detected	: 19	0.2%
Falsely Ignored	: 148	64.63%

To much images get ignored, but the count doesn't change, since some cars get counted twice. Unfortunately we are forced to use this measurement, so we can compare the results to other methods of detection (inductive loops, infrared detectors). The performances of those methods are available, using the count error. These are mostly available in this type of table:

	Lane		
Error Range (%)	EB Right	EB Left	WB
0 to 5	27 of 39 (69.2 %)	33 of 39 (84.6 %)	22 of 39 (56.4 %)
5 to 10	11 of 39 (28.2 %)	6 of 39 (15.4 %)	10 of 39 (25.6 %)
10 to 15	1 of 39 (2.6 %)	0	5 of 39 (12.8 %)
15 to 20	0	0	2 of 39 (5.2 %)

These are the results of 39 measurements in each of the three lanes (EB Right, EB left, WB). We need to determine the average error based on this figure. The simple way to do this, is just taking the average of the class in which it resides (0 to 5 results in 2.5). So we take for the EB Right lane the following calculation:

$$\frac{27}{39} 2.5 + \frac{11}{39} 7.5 + \frac{1}{39} 12.5 + \frac{0}{39} 17.5 = 4.17$$

This may however result in an error too, since the entries in the different classes aren't uniformly distributed. Specifically, you'd expect the entries to be closer to the left border, then to the right one. This is basically the right part of the Gaussian Distribution as shown in the next image.

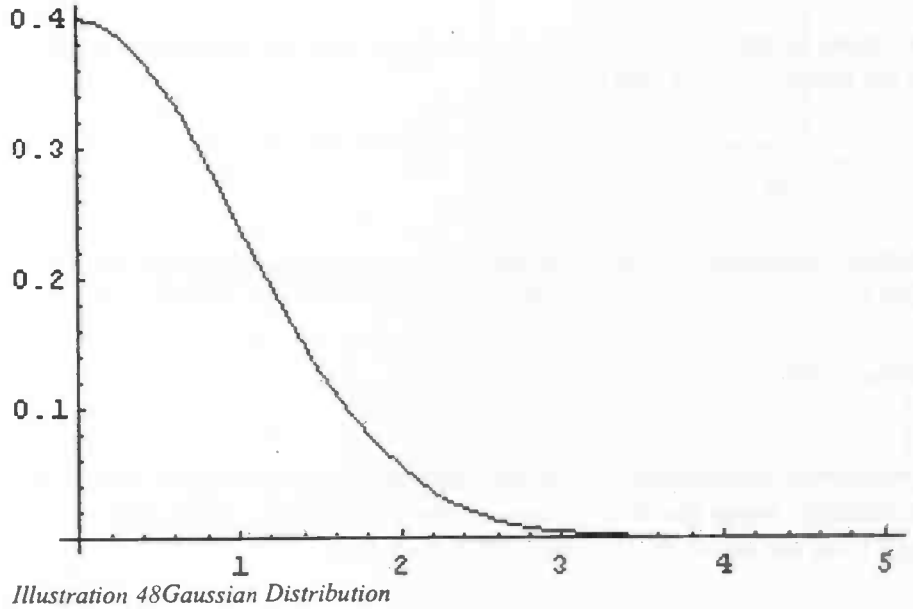


Illustration 48Gaussian Distribution

In order to transfer these class counts to a reasonable average, we try to match the distribution on a Gaussian half distribution. We then use this best matching distribution to determine the average. Let's take a look at the Gaussian distribution formula:

$$\frac{1}{s\,(2\,Pi)}\,e^{\frac{-x^2}{2\,s^2}}$$

Herein is s the standard deviation, and so the adjustable parameter. We need to estimate s, so the Gaussian distribution matches the real distribution. By using some estimation techniques, we were able to make a better guess at the actual performance of the discussed systems. This way we have devised the following Count errors for several systems, under two conditions: A sunny and a rainy day. These results can be found in the chapter performance.

8.3 Improved counting Error.

As we have argued in the previous section, the counting error doesn't work. We need to improve on the counting error method, by incorporating some time information. The object needs to be detected at approximately the right time, and it shouldn't be detected twice. Hence we want to make qualifications on passage level, instead of frame level. We can distinguish the following four possibilities, here "Something" and "nothing" are based on ground truth data, and "Detected" and "Not Detected" are based on the output of the algorithm.

	Something	***Nothing***
Detected	DS	DN
Not Detected	NS	NN

Table 3 Different measurements, the correct parts are grayed, all measurements are counts of their respective meaning.

We can use these values to define some new measurements, first the percentage rightly of detected cars, compared to the number of cars that passed by.

$$Detection\ Rate = 100\% \ \frac{DS}{DS + NS}$$

Second we will define a measurement, that inspects the number of rightly detected objects, when an object is detected.

$$Detection\ Right\ Rate = 100\% \ \frac{DS}{DS + DN}$$

When combining these two measurements, we have all info we need to know. There are off course the inverted measurements using the NN, instead of the DS, but they don't add anything, because they can be deduced from the above mentioned measurements.

Now we will focus our attention on how to determine the variables in table 3, based on the frame based variables. Normally the ground truth data for a car passing by will look like illustration 49: (0=*nothing*,1= *something).*

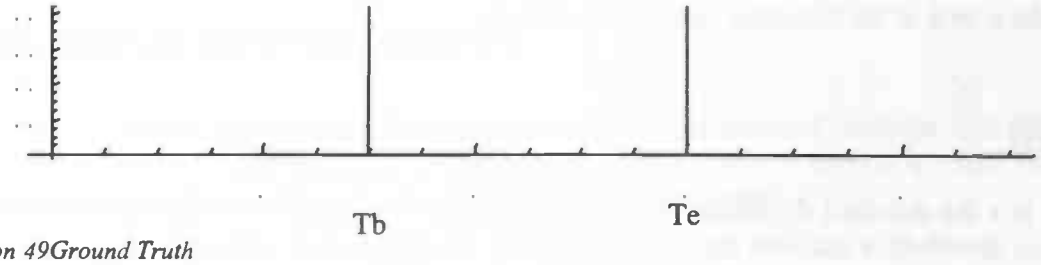


Illustration 49Ground Truth

Let's take a look at some output data, note that these graphs, are for algorithms, which report if something is present in the actual frame. So the ideal output would be Illustration 49.

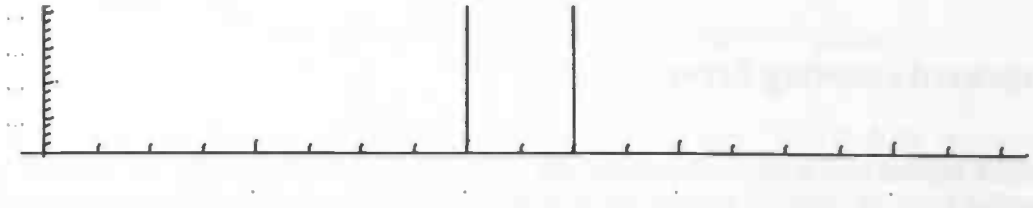


Illustration 50Output 1

Here the object is detected, but not for the whole time, it covers the scene. This isn't to bad, since we only need to detect it: DS gets increased by one.

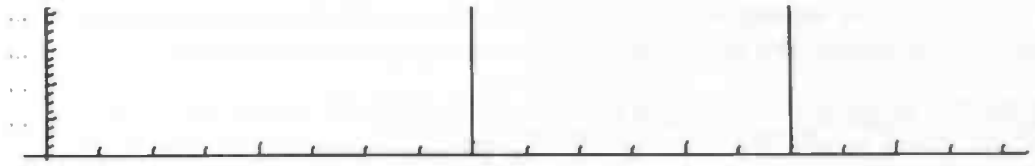


Illustration 51Output 2

Here the detection is late in both starting and ending. This isn't bad, so DS gets increased by one

Illustration 52Output 3

Now the detection starts in time, but it his prolonged way to long. This case is the most complex one. The car is detected, so DS should be increased. We however need to incorporate the fault of going on to long. We can do this by adding one to DN. The problem is finding an objective criterium to separate a bit to long, from much to long. We should relate this overshoot to the number of frames the car is actually on the tripline. Say we tolerate the overshoot if it's smaller then $a\ nrfr$, where *nrfr* is the number of frames the car is on the tripline. We can also use this measurement for undershoots. The a needs to be chosen carefully. In our performance measurements we use an a of 50%.

To formalize the previous, we present this formula:

$tr = te + \uparrow a\ (te - tb)\uparrow$ *The right-extended border*

$tl = tb - \downarrow a\ (te - tb)\downarrow$ *The left-extended border*

$Accepted = xb \le te \wedge xb \ge tl \wedge xe \ge tb \wedge xe \le tr$

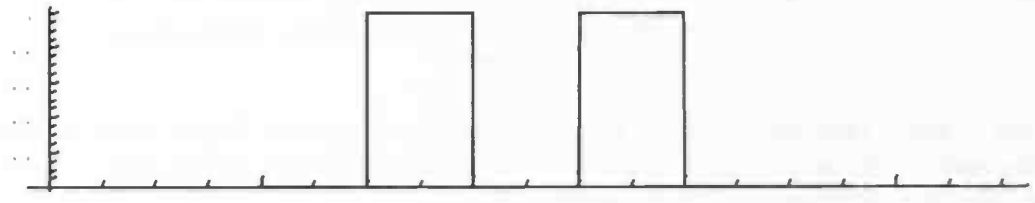


Illustration 53Output 4

Here the car gets detected twice, so DS and DN should both be increased with one. The actual fault is the ignore in the center.

8.4 Promptness and reproducibility

In the previous section we talked about over- and undershoot, the cases were a car gets detected to

late or to early. In this section we will define some measurements, concerning this matter. First the promtness of a detection. We define the promtness in terms of frame numbers.

$$Promptness = Frame\#_{detection} - Frame\#_{ground\ Truth}$$

Here $Frame\#_{detection}$ is the frame number in which the object is detected, and $Frame\#_{ground\ Truth}$ is the frame number of the start of the ground truth detection. Note that a negative promtness means the object is detected to early, and a positive promptness means that the object is detected to late. We can also calculate the average promptness in a test set. We can also take a look at the distribution of the promptness in a test set. If for example the promptness of all detections is a positive relative constant in a test set, so every object gets detected, at a relatively fixed number of frames past the real entrance on the tripline. In that case the whole output can be shifted in time, so the promptness will go to 0. We call this the reproducibility:

$$Reproducibility = \sigma(Promptness)$$

A reproducibility of 0, means that the promptness is constant among all detections, so the shift can be made without any problem. As the reproducibility gets higher, it gets more and more difficult to shift the results, so the performance would increase.

8.5 Using multiple Error measurements.

Another way to measure performance is to combine the above mentioned methods, and calculate an average over them. The only problem is the counting error, since it reports a value of *0* when everything is correct. That's why we have modified the counting error to:

$$Performance_{Counting} = \begin{cases} 100 - 50\ \dfrac{Cnt_{det} - Cnt_{groundtruth}}{Cnt_{groundtruth}} & Cnt_{det} \geq Cnt_{groundtruth} \\ 100 - 100\ \dfrac{Cnt_{groundtruth} - Cnt_{det}}{Cnt_{groundtruth}} & Cnt_{det} < Cnt_{groundtruth} \end{cases}$$

Detecting to many carts has a smaller influence then ignoring cars. This is done because an extra detection isn't as bad as an ignore to many. Now this measurement can be used for the averaging process. The new performance formula now becomes:

$$Performance_{average} = \frac{Performance_{Counting} + Good\ Classified + Average\ Performance + Product\ Performance + Detection\ Rate + \text{Detection Right Rate}}{6}$$

8.6 Separation measurement

Purely based on the output-values of an algorithm, we can give an estimate of the performance of such an algorithm. If we assume the the output values pro class (either True or False) are behaving like a Gaussian distribution, we can give this estimate based on the averages and Standard Deviation of the both classes. If we take a look of the following illustrations, we can see why.

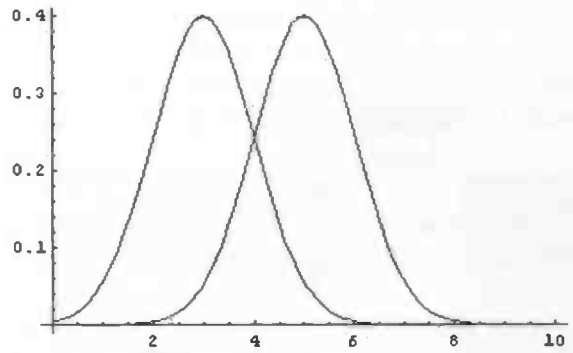


Illustration 54Gaussian Distribution of two classes

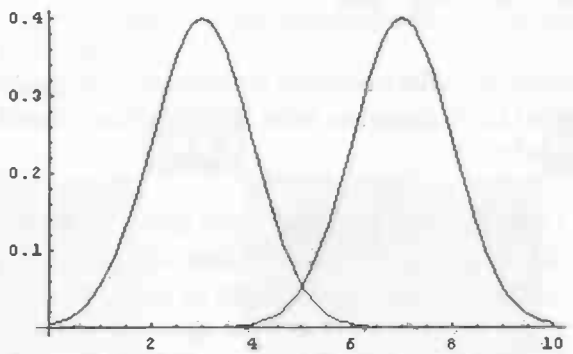


Illustration 55Gaussian Distribution of two classes

It is clear to see that the performance is better in the right image: the distributions are more separated. Based on this observation, we have come up with the following formula:

$$SepMeas = \frac{(\hat{t} - \hat{f})}{(sd\,t + sd\,f)}$$

Herein:

$\hat{t}$ = average of the True-class
$sd\,t$ = standard deviation of the True-class
$\hat{f}$ = average of the False-class
$sd\,f$ = standard deviation of the False-class

This is based on the cross point of both distributions. In fact SepMeas gives the number of Standard Deviations, that fit between the average (the peak) and the cross point.

9 Testing

In this chapter we will give an overview of our testing process, this includes a description of ground truth, and a overview of the used test sets.

9.1Ground Truth

In order to make objective performance measurements, we need to define a ground truth. This ground truth describes what the algorithm should report. There are two reliable ways to define this ground truth.

1. Use a physical inductive loop system. The inductive loop system makes virtually no mistakes in the detection of objects. When alligning our tripline with the loop position, we can use the output of the inductive loop system as our ground truth.
2. Use a human operator. This operator defines the ground truth for each frame. A human will make no mistakes in the detection of objects/cars in a frame.

Unfortunately there was no inductive loop output information avaialble for out data set. That's why we defined the ground truth by hand.

9.2 Three valued logic

All though the algorithms in the previous chapter all either report true or false, on the question of the detection, it isn't always crystal clear, whether to report a detection or not. If we look at the following image, it is hard to say, whether or not to report.

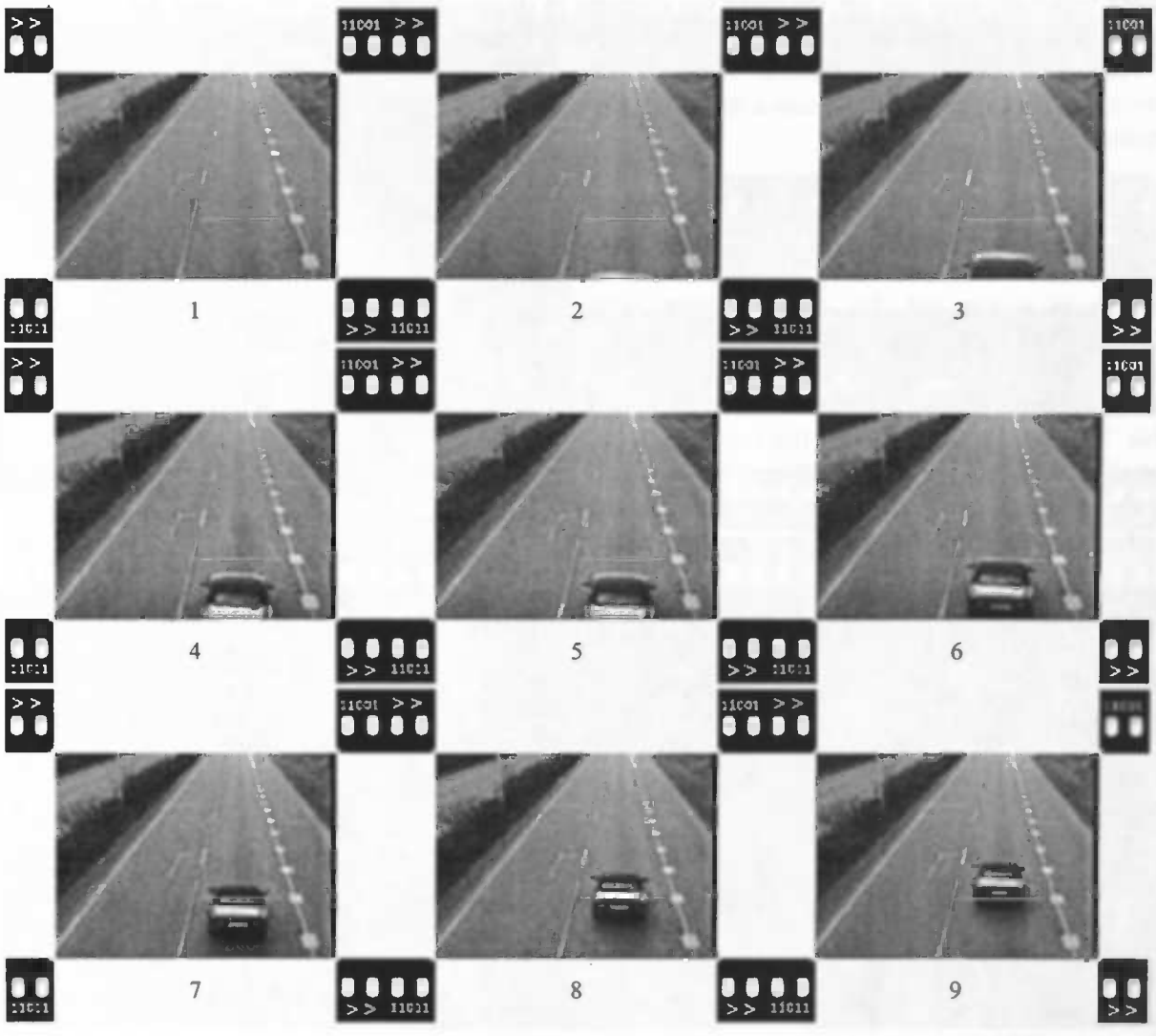
Illustration 56Tripline in doubt

The tripline is covering the wheels of the car, but a bigger part of the line covers the road between the wheels. The need to qualify this image right isn't there, the car will be detected in a subsequent frame. It doesn't matter what the algorithm reports, as long as afterward it is detected, and before it is ignored. The intermediate situation isn't very interesting. When we're defining the Ground Truth (see the previous section), we will define these cases as "Maybe". When doing the performance

measurements, we just ignore these cases.

9.3Test Sets

In this section we will give an overview of used test sets. The test set contains of approximately 40.000 samples, in 2 different image sequences. For each image sequence two triplines are defined. The first set is the traffic flow set. The traffic flow test set contains approximately 10.000 images at daytime of a highway in the Netherlands. This set looks like this:



It's a relatively easy test set, since no big changes in illumination occur. We expect the performance to be good of all proposed methods. In this set two triplines are defined, one on each track:

This set contains 9940 frames. The original movie file uses 25 FPS, and has a duration of 6 minutes and 38 seconds.

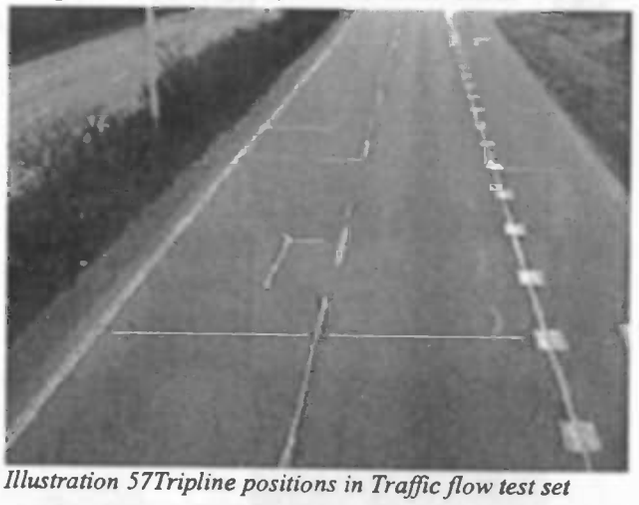


Illustration 57Tripline positions in Traffic flow test set

The second test set we call Dark. This dataset was recorded at night, at a gas station near Rotterdam, The Netherlands. This data set is as worse as it can get. It's a rainy night, with thunder and lightning, so the conditions change every frame. The camera is situated quite far from the viewable area, and it zoomed in quite a bit. The camera hangs under the roof of the gas station, and shakes quite often. On the next page a sequence of frames is shown. This test set contains 9000 frames. The original video

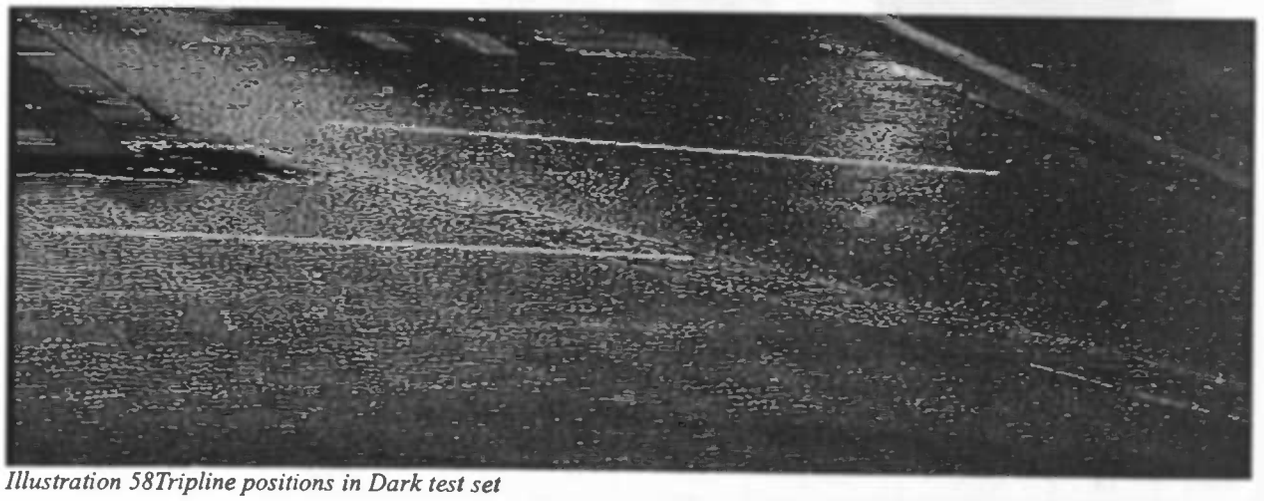
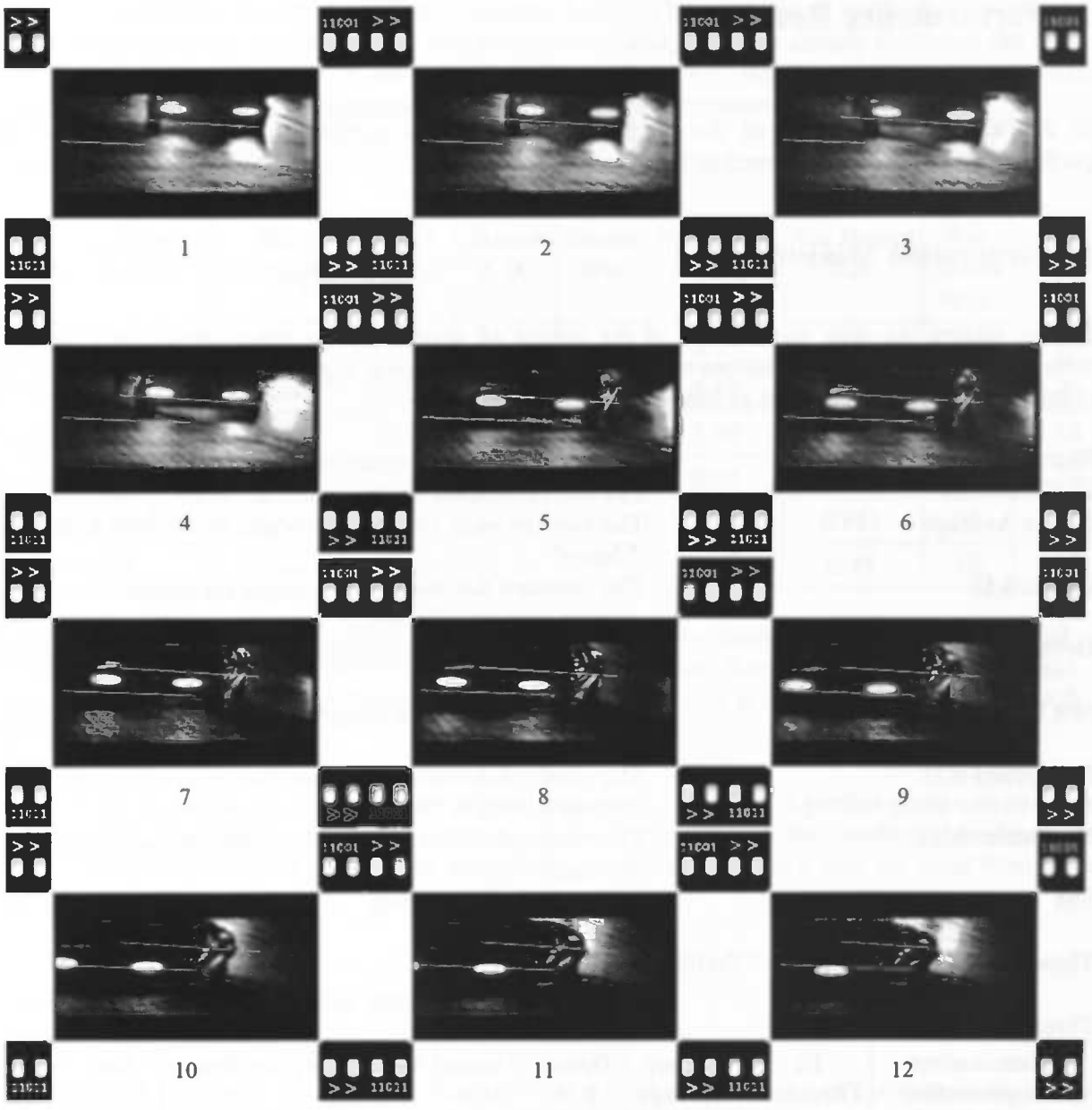


Illustration 58Tripline positions in Dark test set

stream was recorded at 25 FPS, and has a duration of 6 minutes and 40 seconds. Note that due to the heavy zooming of the camera, the car is larger in the screen, as oppose to the traffic flow test set. This results in the ground truth data file, to have more "Maybe" cases. Also the car leaves the tripline perpindicular to the triplines. This implies even more "Maybe" cases. Due to the difficulty of this set, we expect the performance to be significantly worse, then on the traffic flow test set. We also expect to see more differences in performance between the algorithms.


1
2
3
4
5
6
7
8
9
10
11
12

10 Performance Results

In this chapter a selection of the results of the different methods will be demonstrated. All performance results can be found in the appendices.

10.1Seperation Measurements

In this section we give an analysis of the output of several of our mentioned algorithms. The following tables are representations of the output-value of several algorithms mentioned before. The columns should be interpreted as followed:

Illumination Compensation : Type of illumination compensation used.
Ill. Threshold : The used threshold for the illumination compensation
Detect Average : The average output on images where the correct answer is "detect".
Detect S.D. : The standard deviation of the output on images where the correct answer is "detect".
Detect Min : The minimal output on images where the correct answer is "detect".
Not Detect Average : The average output on images where the correct answer is "nothing".
Not Detect S.D. : The standard deviation of the output on images where the correct answer is "nothing".
Not Detect Max : The maximal output on images where the correct answer is "nothing".
SM : Separation Measurement.

These results are based on the "Traffic Flow"test set.

Thresholded Difference:

Illumination Compensation	*Ill. Threshold*	*Detect Average*	*Detect S.D.*	*Detect Min*	*Not Detect Average*	*Not Detect S.D.*	*Not Detect Max*	*SM*
None	None	39.43	15.51	7	2.77	4.79	71	1.81
Linear Global	None	38.26	15.30	6	2.39	2.94	50	1.97
Quadratic Global	None	38.23	15.29	6	2.48	3.07	50	1.95
Linear Box (5)	None	12.33	5.16	4	1.46	0.59	11	1.89
Quadratic Box (5)	None	13.32	5.13	4	1.45	0.58	11	2.08
Pixel (Lin,2)	None	8.23	3.88	3	1.36	0.49	4	1.57
Pixel(quad,2)	None	8.51	3.86	3	1.35	0.47	4	1.61

For the not thresholded results, the box based quadratic method, gives the best results. When using

global compensation, the linear method is slightly better. The pixel based methods give very bad results, when using no threshold. This result doesn't surprise, since we already discussed the fading effect when using this type of compensation. It is also clear the Illumination Compensation methods, improve the output, compared to when using no illumination compensation.

Kolmogorov Smirnov

Illumination Compensation	*Ill. Threshold*	*Detect Average*	*Detect S.D.*	*Detect Min*	*Not Detect Average*	*Not Detect S.D.*	*Not Detect Max*	*SM*
None	None	52.50	20.18	3	2.17	10.51	100	1.64
Linear Global	None	51.62	18.12	3	2.18	9.99	100	1.76
Quadratic Global	None	51.76	17.94	3	2.49	11.41	100	1.68
Linear Box (5)	None	19.39	12.85	3	0.63	0.916	30	1.36
Quadratic Box (5)	None	19.40	12.84	3	0.62	0.911	30	1.37
Pixel (Lin,2)	None	13.05	11.56	0	0.69	0.75	10	1.00

When using the Kolmogorov Smirnov test statistic, the global methods outperform the box based versions. In fact, the none compensated algorithm performs better then the box based illumination compensation. The linear global method gives the best results. The pixel based are lagging behind again, when using no threshold.

Based on these results, we can conclude that without post processing, a perfect score can never be accomplished. The Detect minimum is almost everywhere lower then the "nothing" maximum, so we can't classify them both right, when using a simple threshold. That's why we need filtering to boost the score.

10.2Effects of Illumination compensation

In this section we will give an overview of the effects of illumination compensation on the performance of the Kolmogorov Smirnov test statistic. The full overview of the effects of illumination compensation on all algorithms can be found in the appendices. These performance figures were obtained using the Smirnov test statistic on the both triplines of the Dark dataset.

Illumination Compensation	*Ill. Threshold*	*Good Classified*	*Average perf.*	*Product perf.*	E_{cnt}	*Detect. Rate*	*Det. Right Rate*	*Perf* $_{average}$
None	None	84.92	52.27	38.61	56.73	38.00	19.59	48.35
Linear Global	None	94.40	51.71	23.46	97.12	36.00	32.73	55.90
Quadratic Global	None	85.02	47.44	24.60	58.65	42.00	22.11	46.64
Linear Box (5)	None	91.04	50.49	25.17	100.00	16.00	15.38	49.68
Quadratic Box (5)	None	94.47	56.33	38.44	99.04	24.00	22.64	55.82

Illumination Compensation	*Ill. Threshold*	*Good Classified*	*Average perf.*	*Product perf.*	E_{cnt}	*Detect. Rate*	*Det. Right Rate*	$Perf_{average}$
Pixel (Lin,2)	None	92.32	54.69	36.61	94.23	26.00	22.41	54.38
Pixel (Lin,2)	0.2	84.91	52.35	38.78	53.85	38.00	19.00	47.81

Table 4 Performance of Smirnov

It is clear the illumination compensation algorithms have a positive influence on the performance of this algorithm. The best scoring method is Linear global illumination compensation, but there is only a small margin to quadratic box based and the linear pixel based methods. The linear box based method delivers a mediocre result, it is better then without illumination compensation, but it lags behind, when compare to the best methods. Quadratic global and pixel based linear compensation deliver a bad result. They perfrom worse and about the same respectively as without illumination compensation. Unfortunately we can't say which scheme is the best, since it depends on which detection method you will use in the next step. In appendix B the full overview of results on the dark test set can be found, in those table it can be found which illumination compensation method fit which algorithm best.

10.3 Comparisson of algorithms

In this section we will make a comparisson of the performance of our algorithms. First we will give an overview of the results on the traffic flow tes set, using linear box based compensation.

Method	*Good Classified*	*Average perf.*	*Product perf.*	E_{cnt}	*Detect. Rate*	*Det. Right Rate*	$Perf_{rate}$ (average)
Thresholded Difference	98.89	96.42	96.38	96.40	95.00	95.80	96.48
Positive Thresholding	97.83	94.10	94.00	87.39	90.00	77.70	90.17
Smirnov	98.13	98.25	98.25	94.59	95.83	93.50	96.43
Modified Smirnov	98.22	98.68	98.68	95.05	95.83	94.26	96.79
Kolmogorov Smirnov	98.07	98.65	98.65	95.95	94.17	94.17	96.61
Approximate Entropy	97.69	95.44	95.41	91.82	98.29	89.84	94.75

Table 5 Results of methods on traffic flow test set.

All methods deliver approximately the same performance, only the positeve thresholding lags a bit behind. This is what we expected, this data set is relatively easy, especially when compared to the dark test set.

Method	*Good Classified*	*Average perf.*	*Product perf.*	E_{cnt}	*Detect. Rate*	*Det. Right Rate*	$Perf_{rate}$ (average)
Thresholded Difference	94.66	51.40	21.44	98.08	22.00	21.57	51.52
Positive Thresholding	58.22	53.06	52.77	73.08	28.00	17.50	47.10
Smirnov	94.40	51.71	23.46	97.12	36.00	32.73	55.90
Modified Smirnov	94.41	51.65	23.13	92.31	24.00	20.00	50.92
Kolmogorov Smirnov	94.63	56.49	38.67	97.12	30.00	27.27	57.36
Approximate Entropy	96.04	53.47	27.31	99.04	26.00	24.53	54.40

Table 6 Results of methods on "Dark" test set.

On our dark test set, the results are very bad. The best method (Kolmogorov Smirnov) scores 57.36% on average. The number two Smirnov, and number three approximate entropy are close, but the rest lags behind. We expected positive thresholding to outperform thresholded difference, because of the psoitive change a car causes in this data set. A car is lighter then the surroundings in this set. Unfortunately this is not the case, positive thresholding still scores worse the thresholded difference.

10.4 Effects of using alternative triplines

In this section we will give an overview of the effects of using the other types of triplines described in section 5.7. These results were obtained using linear global compensation with the Kolmogorov Smirnov test statistic, using the dark test set.

Type	*Good Classified*	*Average perf.*	*Product perf.*	E_{cnt}	*Detect. Rate*	*Det. Right Rate*	$Perf_{rate}$ (average)
0	95.03	52.57	25.71	98.08	28.00	27.45	54.47
1	94.66	52.82	27.39	92.31	32.00	26.67	54.31
2	94.92	52.51	25.70	100.00	28.00	26.92	54.68
3	95.05	52.43	25.11	100.00	26.00	25.00	53.93
4	95.39	49.52	0.00	98.08	4.00	3.92	41.82
5	94.97	52.84	26.87	98.08	28.00	25.93	54.45
6	94.63	52.89	27.66	91.35	34.00	27.87	54.73
7	95.04	52.27	24.49	98.08	26.00	25.49	53.56

Table 7 Alternative Triplines Results

As we can see in table 7, the usage of other types of triplines, has a neglectable effect on the performance of our system. The difference between the normal tripline ant the best tripline is only 0.26%, while the tripline has twice as many points on it. Type 4 gives the worst results, while still using the same amount of pixels on the tripline. A positive point is the performance of type 7 triplines. These triplines use only half of the number of pixels of a normal tripline, but still manage to get a result very close that of the normal tripline.

10.5 Effects of filtering

In this section we will illustrate the advantage of using filtering schemes as postprocessing mode.

Test Set	*Filter*	*Good Classified*	*Average perf.*	*Product perf.*	E_{cnt}	*Detect. Rate*	*Det. Right Rate*	$Perf_{rate}$ (average)
Dark	no	96.13	51.18	16.27	98.08	22.00	21.57	50.87
Dark	yes	94.14	54.81	34.65	84.62	36.00	26.47	55.12
Traffic	yes	98.40	96.11	96.08	93.69	95.00	91.20	95.08
Traffic	no	98.46	95.76	95.71	78.38	96.67	72.96	89.65

Table 8Effects of filtering

As we can see in the table, the traffic flow testset benefits more from the filtering techniques. The parameters of the fiters depend on the used dataset and detection algorithm. On the traffic flow test set, depending on which algorithm, a sequence filter of length 3 or 4, or a average filter with a window of 3 or 4, and a N of 2 give the best results. In the dark test set, the cars cover the tripline for more frames in a row. So we expect the parameters of the filters to be higher here. Surprisingly this doesn't hold. The best parameters are a sequence filter with length 1 or 2, or a average filter with a window of 2, and a N of 1. Though unexpected, this does explain the small difference in performance between the methods with and without filtering. The filters do not change much on the output, when using such small parameters.

10.6 Generalizing Capabilities

In this section we will investigate the generalizing capabilities of our methods. Can the same threshold and filter parameters be used on any test set? We tested this, by comparing the performance of our methods on our two test sets independently, and consecutively.

Method	*Perf on Dark*	*Perf on Traffic*	*Perf on total*	*GenCap*
Threshold Difference	48.00	96.48	72.63	100.0%
Positive Thresholding	42.67	90.17	66.32	99.8%
Kolmogorov Smirnov	55.12	95.08	68.58	91.3%
Smirnov	49.68	95.99	68.75	94.4%
Modified Smirnov	47.42	96.43	69.03	96.0%

Table 9Generalizing Capbilities (linear box based compensation)

Approximate entropy has been ignored here, since it will score very bad on this. The output is the number of pixels on the line which cause an alarm. This means the threshold is dependent on the length of the tripline, and results of the GenCap measurement will be low. The approximate entropy algorithm can off course be modified, so it will deliver a procentual measurement. This way it will score better on the generalizing capabilities.

10.7 Comparisson to other methods

In this section we will make a comparisson between our results, and the commercailly available systems as mentioned in chapter 7. As we have said before, this comparisson is done using the count error. In the following table some methods have only one error, while other have two. Our methods use our two data sets, easy is the traffic flow set, and hard the dark tes set. For the first three systems, easy is a daytime observation, while hard is a daytime with rain observation.

Detection method	E_{cnt} *(simple)*	E_{cnt} *(Hard)*
3M Microloop	2.37%	2.37%
VideoTrak	5.54%	14.21%(*)
SAS-1	4.88%	15% (*)
Autoscope 1	8.7%	
Autoscope 2	1.3%	
TTI inductive Loop	1.5%	
Hughes Inductive loop	1%	
Kolmogorov Smirnov	1.8%	5.77%
Thresholded Difference	3.6%	1.92%
Positive Thresholding	4.2%	2.53%
Smirnov	1.8%	5.67%
Modified Smirnov	1.2%	6.34%
Approximate Entropy	2.4%	1.92%

Table 10Comparisson to pother algorithms, using count error

When examining this table it is clear that the physical inductive loop systems offer superior performance. Another positive point about the inductive loop systems is their immunity to lighting changes. They aren't affect by day/night cycles, rain and so on. They just work. When comparing our performance to that of other vision based detection systems (Autoscop, VideoTrak) the results are quite good when using a simple data set (traffic flow). Unfortunately we couldn't obtain the performance results of these systems on as worse a test set as our dark test set. Only results on a rainy day were available (not for all systems). It is clear the VideoTrak system doesn't handle rain very well. This might indicate, that night sequences are also processed very bad, but that's off course only educated speculation. A problem that remains is the incorrectness of the count error, by just tweaking the threshold and filter parameters, we can obtain perfect results on the count error. To be fair, we set the parameters in a way, that the average over all introduced performance measurements was best. In some case this meant the count error was high, while other error measurements did give positive results. The table should threfore only be used to illustrate that our system is quite capable compared to other methods.

11 Impact of frame rate on performance

Until now, we have investigated the usage of triplines, in order to speed up the detection process. Another way to reduce the calculation costs is frame rate reduction. When we need to process fewer frames, the calculation costs will decrease accordingly. In this chapter we will investigate the effect of frame rate reduction. In order to test the impact of frame rate on performance, we need to reduce the frame rate of our used test sets. We can't increase the frame rate of our test sets, because we can't make up extra frames. We will reduce the frame rate by throwing away frames. To be realistic, we need the kept frames to be equidistant in the time line, so the time between frames is always the same. This way a reduction of 33% in the frame rate isn't possible, since the kept frames (keep 2, throw away 1) aren't equidistant in time. We so only consider the cases in which one frame is kept, and then some frames are thrown away.

Frames deleted	*Frame rate compared to original*	*First Example FPS*	*Second Example FPS*
0	100 %	25.00	12.50
1	50%	12.50	6.25
2	33%	8.33	4.17
3	25%	6.25	3.13
4	20%	5.00	2.50
N	$100\% \frac{1}{N+1}$	$\frac{25.00}{N+1}$	$\frac{12.50}{N+1}$

Table 11 Frame rate reduction

In table 11 an overview of the effect of deleting frames is shown.

11.1 Consequences of frame rate reduction

Now we know how to reduce the frame rate, we will investigate the effects of frame rate reduction. Since most algorithms work on an individual frame, not much differences when using these algorithms, only the sequence based algorithms are affected. They can however use more time as sequence, so they keep the same number of frames examined. This should counter this effect for these type of methods too. The post processing step (filtering) is however much more affected. These algorithms work on the basis of a car triggering in multiple sequential images. When reducing the frame rate, these sequences get shorter. Then there's off course the effect of skipping a car on the line in reality. First the car is below the tripline, and in the next frame the car will be above the tripline. This way, the algorithm is never able to detect the car. We need to avoid this scenario at all costs. That's why we need to determine the average time a car is on the tripline. Let's take a look at some of these results, note that there are 2 values available, one including the maybe (see Three valued logic), and one excluding it.

Test Set	*Average length including maybe*	*Minimal length, including maybe*	*Average length excluding maybe*	*Minimal length, excluding maybe*
Traffic Flow	10.98	4	9.55	4
Dark	25.82	14	12.31	4

Table 12Length of cars in frames on the tripline

It is clear that the dark test set contains more maybe's, since the difference between the length with and without the "maybe's" is much larger compared to the difference in the traffic flow set. This is logical, since in the dark test set the car first moves on the tripline, but then leaves the tripline on a trajectory perpendicular to the tripline. See the chapter test sets for more information on this. And now for some results. These results were obtained on the traffic flow test set, using the Kolmogorov Smirnov detection method.

Frames deleted	*#Cars in Ground Truth*	*Good Classified*	*Average perf.*	*Product perf.*	E_{cnt}	*Detect. Rate*	*Det. Right Rate*	*Perf (average)*
0	110	98.07	98.65	98.65	95.95	94.17	94.17	96.61
1	110	98.04	98.68	98.68	95.91	95.76	94.96	97.01
2	110	98.01	98.95	98.95	95.91	97.41	94.96	97.36
3	110	98.08	98.99	98.98	95.00	94.92	92.56	96.42
4	109	98.22	99.06	99.06	94.95	94.92	93.33	96.59
5	106	97.87	98.31	98.31	94.34	95.69	94.07	96.43
6	102	98.13	98.33	98.33	91.67	96.55	94.12	96.19
7	98	98.11	97.88	97.88	91.33	92.92	91.30	94.90
8	85	98.10	99.00	99.00	81.76	96.26	88.79	93.82

Table 13Results of frame rate reduction on Traffic flow test set

Until a reduction to 25% of the original framerate (3 frames deleted), every car covers the tripline at least one frame. When reducing the frame rate even further, some cars do not cover the line anymoore, so the number of cars in the ground truth gets smaller. The performance of the system decreases gradually. A reduction of 75% in framerate is still quite usable. Note that the performnace when reducing the frame rate even moore doesn't decrease much, all though these reductions aren't usable. This is logical since performance is measured compared to the ground truth. When cars dissapear from the ground truth, it isn't bad if they dissapear from the reports of the algorithm too. For practical uses, this off course doesn't hold.

Frames deleted	*#Cars in Ground Truth*	*Good Classified*	*Average perf.*	*Product perf.*	E_{cnt}	*Detect. Rate*	*Det. Right Rate*	*Perf (average)*
0	52	94.63	56.49	38.67	97.12	30.00	27.27	57.36
1	52	94.23	60.51	48.31	88.46	34.00	36.96	60.41
2	52	92.87	61.82	51.94	100.00	30.00	28.85	60.91
3	52	93.36	63.16	54.05	97.12	34.00	30.91	62.10
4	52	92.23	64.03	56.33	92.31	34.00	28.33	61.21
5	52	92.72	63.03	54.33	96.15	28.00	28.00	60.37
6	52	90.75	63.52	56.26	98.08	20.00	18.52	57.86
7	51	91.76	66.93	61.28	98.04	28.00	28.00	62.33
8	51	91.60	65.16	58.58	90.20	32.00	34.78	62.05
9	51	93.20	62.27	52.50	99.02	28.00	26.92	60.32

Table 14Results of frame rate reduction on Dark test set

Since a car covers the tripline for a larger number of frames, we can reduce the frame rate even more. It is when the frame rate gets reduced to 12.5% of the original frame rate, that we start to lose cars. Strangely enough the performance increases a bit, when reducing the framerate. Apparantly difficult frames are filtered out. The performance doesn't really decrease much when reducing the framerate. We only need to take into account, that all cars are still present on the tripline in at least one frame.

11.2 Frame rate reduction: The Conclusion

To answer the question of how far we can reduce the framerate, we need to investigate the data set on which the algorithm is run. Two factors determine the possible frame rate reduction:

1. The average and minimum number of frames a car covers the tripline. When a car covers the tripline for more frames, the frame rate can be reduced more.
2. The difficulty of the data set. When the detection precess is more difficult, the algorithm will rely on filtering to boost the performance of the system. Frame rate reduction implies the decrease in effectivity of the filtering algorithms.

12 Conclusions

Our tripline detection systems work reasonably well, as long as the data set, doesn't get to complicated. During daytime monitoring, almost perfect results are obtained. When the test set is more difficult, results get worse, and unusable. Before this kind of system van be commercially exploited, more investigations on bigger test sets are needed. This way, a more reliable performance estimation can be made. In dark environments the Kolmogorov Smirnov test statistic works best. When using a daytime surveillance system, all methods give approximately the same results. When comparing the performance to other commercially available systems, our performance is quite good when using simple data sets. Inductive loop systems give the best results, while VMD systems lag behind by a small margin. Illumination compensation gives the performance of our algorithms a big boost. Filtering the output has a positive influence on the performance.We discovered that linear box based compensation gives the best results.

13 Appendix A: All Results

Thresholded Difference:

Illumination Compensation	*Ill. Threshold*	*Detect Average*	*Detect S.D.*	*Detect Min*	*Not Detect Average*	*Not Detect S.D.*	*Not Detect Max*	*SM*
None	None	39.43	15.51	7	2.77	4.79	71	1.81
Linear Global	None	38.26	15.30	6	2.39	2.94	50	1.97
Quadratic Global	None	38.23	15.29	6	2.48	3.07	50	1.95
Linear Box (5)	None	12.33	5.16	4	1.46	0.59	11	1.89
Quadratic Box (5)	None	13.32	5.13	4	1.45	0.58	11	2.08
Pixel (Lin,2)	None	8.23	3.88	3	1.36	0.49	4	1.57

Posthresh:

Illumination Compensation	*Ill. Threshold*	*Detect Average*	*Detect S.D.*	*Detect Min*	*Not Detect Average*	*Not Detect S.D.*	*Not Detect Max*	*SM*
None	None	13.39	18.15	0	0.61	1.22	14	0.66
Linear Global	None	15.19	19.02	0	1.22	1.74	28	0.67
Quadratic Global	None	15.40	19.14	0	1.49	1.97	30	0.66
Linear Box (5)	None	6.33	3.80	1	0.75	0.75	5	1.23
Quadratic Box (5)	None	6.34	3.80	5	0.76	0.45	5	1.31
Pixel (Lin,2)	None	3.55	2.84	0	0.77	0.43	3	0.85

Again the quadratic box gives the best results, when using no threshold. The pixel based methods without threshold give very bad results. When using global compensation, only a small advantage is gained, compared to no illumination compensation. The Separation measurement might not be completely reliable in this situation. Since only positive changes are measured, the result will often be *0*. This results in the distribution being less Gaussian like. Since the Gaussian likeliness is very important for the separation measurement, these results will be unreliable.

Modified Smirnov

Illumination Compensation	*Ill. Threshold*	*Detect Average*	*Detect S.D.*	*Detect Min*	*Not Detect Average*	*Not Detect S.D.*	*Not Detect Max*	*SM*
None	None	2715	1710	111	37.10	261.7	5065	1.36
Linear Global	None	2662	1760	78	21.05	106.0	3054	1.42
Quadratic Global	None	2660	1765	78	22.34	109.2	3016	1.41
Linear Box (5)	None	408.0	279.1	51	6.57	12.71	411	1.38

Illumination Compensation	*Ill. Threshold*	*Detect Average*	*Detect S.D.*	*Detect Min*	*Not Detect Average*	*Not Detect S.D.*	*Not Detect Max*	*SM*
Quadratic Box (5)	None	407.5	277.9	52	6.57	6.57	411	1.41
Pixel (Lin,2)	None	189.2	156.1	25	4.35	4.35	136	1.15

This time the global methods outperform the box based ones, when using no threshold. The pixel based methods without thresholds are less weak when compared to other methods. The difference between compensating and not, is also less big, when compared to the other methods.

Kolmogorov Smirnov

Illumination Compensation	*Ill. Threshold*	*Detect Average*	*Detect S.D.*	*Detect Min*	*Not Detect Average*	*Not Detect S.D.*	*Not Detect Max*	*SM*
None	None	52.50	20.18	3	2.17	10.51	100	1.64
Linear Global	None	51.62	18.12	3	2.18	9.99	100	1.76
Quadratic Global	None	51.76	17.94	3	2.49	11.41	100	1.68
Linear Box (5)	None	19.39	12.85	3	0.63	0.916	30	1.36
Quadratic Box (5)	None	19.40	12.84	3	0.62	0.911	30	1.37
Pixel (Lin,2)	None	13.05	11.56	0	0.69	0.75	10	1.00

14 Results of daytime highway vehicle detection

In this section the performance results of a sequence consisting of approximately 10.000 frames, which were recorded at daytime on a highway in the Netherlands. 2 Triplines were defined in this sequence, the performance is given of those line's separately, and when combining the output of both lines. Then we also give the generalization capabilities of these methods, when comparing both triplines. Further on, we will define the generalization capabilities, based on all processed data.
Note that the best results are given, they were determined using filtering and brute force threshold selecting when using the $\text{Performance}_{\text{average}}$ method to optimize.
Thresholded Difference:

Illumination Compensation	*Ill. Threshold*	*Line*	*Good Classified*	*Average perf.*	*Product perf.*	E_{cnt}	*Detect. Rate*	*Det. Right Rate*	*Perf average*
None	None	1	98.54	97.12	97.11	93.06	90.48	92.68	94.83
None	None	2	96.73	97.98	97.97	98.00	96.15	96.15	97.17
None	None	1+2	97.63	97.27	97.27	95.95	94.17	94.17	96.08
Linear Global	None	1	98.91	96.67	96.64	90.28	95.24	93.02	95.13
Linear Global	None	2	97.13	98.26	98.25	97.33	96.15	94.94	97.01
Linear Global	None	1+2	98.07	97.60	97.60	94.59	95.83	93.50	96.20
Quadratic Global	None	1	98.97	95.43	95.35	91.67	92.86	92.86	94.52
Quadratic Global	None	2	97.20	98.30	98.29	98.00	96.15	96.15	97.35
Quadratic Global	None	1+2	98.12	97.44	97.44	94.14	95.83	92.74	95.95
Linear Box (5)	None	1	99.63	99.81	99.81	91.67	95.24	95.24	96.90
Linear Box (5)	None	2	98.01	95.57	95.53	98.00	96.15	96.15	96.57
Linear Box (5)	None	1+2	98.89	96.42	96.38	96.40	95.00	95.80	96.48
Quadratic Box (5)	None	1	99.64	99.82	99.82	90.28	97.62	95.35	97.09
Quadratic Box (5)	None	2	98.26	94.75	94.66	98.00	96.15	96.15	96.33
Quadratic Box (5)	None	1+2	99.04	95.59	95.52	97.30	94.17	96.58	96.37
Pixel (Lin,2)	None	1	99.71	98.79	98.78	95.83	90.48	97.44	96.84
Pixel (Lin,2)	None	2	98.79	93.55	93.34	98.00	94.87	94.87	95.57
Pixel (Lin,2)	None	1+2	99.27	93.86	93.67	98.20	92.50	96.52	95.67
Pixel(quad,2)	None	1	98.54	97.12	97.11	93.06	90.48	92.68	94.83
Pixel(quad,2)	None	2	96.73	97.98	97.97	98.00	96.15	96.15	97.17
Pixel(quad,2)	None	1+2	97.63	97.27	97.27	95.95	94.17	94.17	96.08
Pixel (Lin,2)	0.2	1	98.33	96.16	96.13	93.06	90.48	92.68	94.47
Pixel (Lin,2)	0.2	2	97.71	98.76	98.75	98.00	96.15	96.15	97.59
Pixel (Lin,2)	0.2	1+2	97.71	98.08	98.08	95.95	94.17	94.17	96.36

Table 15Performance of Thresholded Difference

Positive Thresholding:

Illumination Compensation	*Ill. Threshold*	*Line*	*Good Classified*	*Average perf.*	*Product perf.*	E_{cnt}	*Detect. Rate*	*Det. Right Rate*	*Perf average*
None	None	1	96.94	81.81	80.26	93.06	88.10	90.24	88.40
None	None	2	95.09	74.45	70.30	98.00	88.46	88.46	85.79
None	None	1+2	96.14	75.17	71.45	97.75	86.67	89.66	86.14

Illumination Compensation	*Ill. Thresh old*	*Line*	*Good Classified*	*Average perf.*	*Product perf.*	E_{cnt}	*Detect. Rate*	*Det. Right Rate*	*Perf* $_{average}$
Linear Global	None	1	97.58	81.93	80.26	87.50	97.62	91.11	89.33
Linear Global	None	2	95.79	75.90	72.14	94.67	88.46	83.13	85.01
Linear Global	None	1+2	96.63	76.57	73.24	95.95	86.67	86.67	85.95
Quadratic Global	None	1	97.48	80.81	78.90	87.50	95.24	88.89	88.14
Quadratic Global	None	2	95.70	73.22	68.19	98.00	87.18	87.18	84.91
Quadratic Global	None	1+2	96.42	76.75	73.55	95.50	85.83	85.12	85.53
Linear Box (5)	None	1	99.69	98.99	98.99	94.44	90.48	95.00	96.26
Linear Box (5)	None	2	95.95	92.07	91.95	84.00	89.74	70.71	87.40
Linear Box (5)	None	1+2	97.83	94.10	94.00	87.39	90.00	77.70	90.17
Quadratic Box (5)	None	1	99.69	98.99	98.99	94.44	90.48	95.00	96.26
Quadratic Box (5)	None	2	95.93	92.18	92.07	84.00	88.46	69.70	87.06
Quadratic Box (5)	None	1+2	97.82	94.19	94.10	87.39	89.17	76.98	89.94
Pixel (Lin,2)	None	1	99.46	96.74	96.70	88.89	95.24	90.91	94.65
Pixel (Lin,2)	None	2	96.93	83.46	81.91	90.00	93.59	81.11	87.83
Pixel (Lin,2)	None	1+2	98.11	84.01	82.53	93.69	88.33	84.80	88.58
Pixel(quad,2)	None	1	96.94	81.81	80.26	93.06	88.10	90.24	88.40
Pixel(quad,2)	None	2	95.09	74.45	70.30	98.00	88.46	88.46	85.79
Pixel(quad,2)	None	1+2	96.14	75.17	71.45	97.75	86.67	89.66	86.14
Pixel (Lin,2)	0.2	1	99.26	88.54	87.82	90.28	95.24	93.02	92.36
Pixel (Lin,2)	0.2	2	96.41	77.55	74.25	97.33	88.46	87.34	86.89
Pixel (Lin,2)	0.2	1+2	97.68	78.36	75.35	96.85	87.50	88.98	87.45

Table 16Results of positive thresholding

Kolmogorov Smirnov:

Illumination Compensation	*Ill. Threshold*	*Line*	*Good Classified*	*Average perf.*	*Product perf.*	E_{cnt}	*Detect. Rate*	*Det. Right Rate*	$Perf_{average}$
None	None	1	97.90	97.22	97.22	88.89	90.48	86.36	93.01
None	None	2	96.13	97.66	97.64	96.67	94.87	92.50	95.91
None	None	1+2	96.95	98.01	98.00	93.24	93.33	88.89	94.74
Linear Global	None	1	97.09	98.51	98.50	90.28	95.24	93.02	95.44
Linear Global	None	2	96.58	97.79	97.78	94.00	96.15	89.29	95.26
Linear Global	None	1+2	96.87	97.40	97.40	91.89	95.83	89.15	94.76
Quadratic Global	None	1	97.02	98.47	98.46	90.28	95.24	93.02	95.42
Quadratic Global	None	2	96.44	97.59	97.58	93.33	93.59	85.88	94.07
Quadratic Global	None	1+2	96.73	97.08	97.08	91.89	94.17	87.60	94.09
Linear Box (5)	None	1	99.76	99.67	99.67	91.67	95.24	95.24	96.87
Linear Box (5)	None	2	96.62	91.65	91.46	96.67	96.15	93.75	94.39
Linear Box (5)	None	1+2	98.40	96.11	96.08	93.69	95.00	91.20	95.08
Quadratic Box (5)	None	1	99.76	99.67	99.67	91.67	95.24	95.24	96.87
Quadratic Box (5)	None	2	97.31	94.24	94.16	94.67	96.15	90.36	94.48
Quadratic Box (5)	None	1+2	98.40	96.11	96.08	93.69	95.00	91.20	95.08
Pixel (Lin,2)	None	1	99.58	95.31	95.20	97.22	85.71	94.74	94.63
Pixel (Lin,2)	None	2	98.09	91.67	91.36	88.67	96.15	81.52	91.24
Pixel (Lin,2)	None	1+2	98.81	92.28	92.00	90.54	93.33	84.85	91.97
Pixel(quad,2)	None	1	97.90	97.22	97.22	88.89	90.48	86.36	93.01
Pixel(quad,2)	None	2	96.13	97.66	97.64	96.67	94.87	92.50	95.91
Pixel(quad,2)	None	1+2	96.95	98.01	98.00	93.24	93.33	88.89	94.74
Pixel (Lin,2)	0.2	1	98.56	96.28	96.25	93.06	90.48	92.68	94.55
Pixel (Lin,2)	0.2	2	97.58	98.63	98.62	98.00	96.15	96.15	97.52
Pixel (Lin,2)	0.2	1+2	98.07	98.65	98.65	95.95	94.17	94.17	96.61

Table 17Results of Kolmogorv Smirnov

Smirnov:

Illumination Compensation	*Ill. Threshold*	*Line*	*Good Classified*	*Average perf.*	*Product perf.*	E_{cnt}	*Detect. Rate*	*Det. Right Rate*	*Perf average*
None	None	1	98.27	96.35	96.33	91.67	90.48	90.48	93.93
None	None	2	96.82	97.92	97.91	98.00	96.15	96.15	97.16
None	None	1+2	97.40	98.34	98.34	94.59	94.17	91.87	95.79
Linear Global	None	1	99.05	96.96	96.93	90.28	97.62	95.35	96.03
Linear Global	None	2	97.32	98.54	98.53	97.33	96.15	94.94	97.14
Linear Global	None	1+2	98.13	98.25	98.25	94.59	95.83	93.50	96.43
Quadratic Global	None	1	99.08	96.97	96.94	90.28	97.62	95.35	96.04
Quadratic Global	None	2	97.72	98.04	98.04	97.33	96.15	94.94	97.04
Quadratic Global	None	1+2	98.11	98.43	98.43	94.14	95.83	92.74	96.28
Linear Box (5)	None	1	99.73	97.73	97.71	90.28	97.62	95.35	96.40
Linear Box (5)	None	2	97.75	95.43	95.39	98.00	96.15	96.15	96.48
Linear Box (5)	None	1+2	98.74	96.44	96.40	94.14	96.67	93.55	95.99
Quadratic Box (5)	None	1	99.74	97.74	97.72	90.28	97.62	95.35	96.41
Quadratic Box (5)	None	2	97.75	95.43	95.39	98.00	96.15	96.15	96.48
Quadratic Box (5)	None	1+2	98.75	96.44	96.40	94.14	96.67	93.55	95.99
Pixel (Lin,2)	None	1	99.80	99.90	99.90	87.50	97.62	91.11	95.97
Pixel (Lin,2)	None	2	98.66	95.03	94.93	98.00	96.15	96.15	96.49
Pixel (Lin,2)	None	1+2	99.07	96.13	96.08	93.69	96.67	92.80	95.74
Pixel(quad,2)	None	1	98.27	96.35	96.33	91.67	90.48	90.48	93.93
Pixel(quad,2)	None	2	96.82	97.92	97.91	98.00	96.15	96.15	97.16
Pixel(quad,2)	None	1+2	97.40	98.34	98.34	94.59	94.17	91.87	95.79
Pixel (Lin,2)	0.2	1	98.40	99.18	99.18	88.89	90.48	86.36	93.75
Pixel (Lin,2)	0.2	2	97.74	98.77	98.77	98.00	96.15	96.15	97.60
Pixel (Lin,2)	0.2	1+2	97.95	98.87	98.86	94.59	94.17	91.87	96.05

Table 18Results of Smirnov test statistic

Modified Smirnov:

Illumination Compensation	*Ill. Thresh old*	*Line*	*Good Classified*	*Average perf.*	*Product perf.*	E_{cnt}	*Detect. Rate*	*Det. Right Rate*	*Perf average*
None	None	1	98.34	96.80	96.79	91.67	90.48	90.48	94.09
None	None	2	97.41	98.18	98.17	98.00	96.15	96.15	97.34
None	None	1+2	97.77	98.30	98.30	95.05	94.17	92.62	96.03
Linear Global	None	1	98.70	98.48	98.48	90.28	97.62	95.35	96.48
Linear Global	None	2	97.82	98.34	98.34	98.00	96.15	96.15	97.47
Linear Global	None	1+2	98.22	98.68	98.68	95.05	95.83	94.26	96.79
Quadratic Global	None	1	99.08	97.82	97.81	90.28	97.62	95.35	96.33
Quadratic Global	None	2	97.84	98.83	98.82	98.00	96.15	96.15	97.63
Quadratic Global	None	1+2	98.33	98.12	98.12	94.59	96.67	94.31	96.69
Linear Box (5)	None	1	99.77	98.39	98.38	91.67	95.24	95.24	96.45
Linear Box (5)	None	2	98.36	95.70	95.65	98.00	96.15	96.15	96.67
Linear Box (5)	None	1+2	99.05	96.40	96.36	95.05	96.67	95.08	96.43
Quadratic Box (5)	None	1	99.77	98.39	98.38	91.67	95.24	95.24	96.45
Quadratic Box (5)	None	2	98.36	95.70	95.65	98.00	96.15	96.15	96.67
Quadratic Box (5)	None	1+2	99.01	96.38	96.34	95.05	96.67	95.08	96.42
Pixel (Lin,2)	None	1	99.75	98.60	98.59	88.89	97.62	93.18	96.10
Pixel (Lin,2)	None	2	99.06	95.55	95.45	98.00	96.15	96.15	96.73
Pixel (Lin,2)	None	1+2	99.42	95.79	95.71	95.50	95.83	95.04	96.22
Pixel(quad,2)	None	1	98.34	96.80	96.79	91.67	90.48	90.48	94.09
Pixel(quad,2)	None	2	97.41	98.18	98.17	98.00	96.15	96.15	97.34
Pixel(quad,2)	None	1+2	97.77	98.30	98.30	95.05	94.17	92.62	96.03
Pixel (Lin,2)	0.2	1	98.36	95.96	95.93	91.67	90.48	90.48	93.81
Pixel (Lin,2)	0.2	2	97.75	98.78	98.77	98.00	96.15	96.15	97.60
Pixel (Lin,2)	0.2	1+2	97.87	98.88	98.87	95.05	94.17	92.62	96.24

Table 19Results of Modified Smirnov

Approximate Entropy:

Illumination Compensation	*Ill. Threshold*	*Line*	*Good Classified*	*Average perf.*	*Product perf.*	E_{cnt}	*Detect. Rate*	*Det. Right Rate*	*Perf average*
None	None	1	98.64	98.24	98.24	91.67	95.12	92.86	95.79
None	None	2	97.20	94.40	94.34	94.59	98.68	91.46	95.11
None	None	1+2	97.69	95.44	95.41	91.82	98.29	89.84	94.75

Table 20Results of Approximate entropy

15 Appendix B: results on Dark test set

In this appendix the results on the dark test set will be given.
Thresholded Difference:

Illumination Compensation	*Ill. Threshold*	*Line*	*Good Classified*	*Average perf.*	*Product perf.*	E_{cnt}	*Detect. Rate*	*Det. Right Rate*	*Perf average*
None	None	1	93.61	62.16	52.27	100.00	34.62	34.62	62.88
None	None	2	92.21	49.03	13.75	76.92	20.83	13.16	44.32
None	None	1+2	93.43	49.41	13.48	100.00	12.00	11.54	46.64
Linear Global	None	1	94.79	53.42	29.90	96.15	38.46	40.00	58.79
Linear Global	None	2	94.21	50.88	18.93	100.00	29.17	26.92	53.35
Linear Global	None	1+2	94.66	51.40	21.44	98.08	22.00	21.57	51.52
Quadratic Global	None	1	95.09	53.40	29.37	73.08	61.54	40.00	58.75
Quadratic Global	None	2	93.38	48.70	0.00	100.00	16.67	15.38	45.69
Quadratic Global	None	1+2	93.26	48.57	5.50	100.00	12.00	11.54	45.14
Linear Box (5)	None	1	57.02	59.59	59.52	25.00	65.38	26.15	48.78
Linear Box (5)	None	2	65.96	43.40	35.77	34.62	66.67	26.67	45.51
Linear Box (5)	None	1+2	72.68	47.88	39.68	91.35	20.00	16.39	48.00
Quadratic Box (5)	None	1	55.20	58.64	58.53	40.38	53.85	24.56	48.53
Quadratic Box (5)	None	2	51.82	43.02	41.94	88.46	29.17	21.88	46.05
Quadratic Box (5)	None	1+2	78.38	44.45	25.17	82.69	22.00	15.71	44.73
Pixel (Lin,2)	None	1	62.69	51.28	49.81	92.31	23.08	20.00	49.86
Pixel (Lin,2)	None	2	89.46	46.65	0.00	96.15	12.50	10.71	42.58
Pixel (Lin,2)	None	1+2	72.91	46.72	37.19	90.38	18.00	14.52	46.62
Pixel (Lin,2)	0.2	1	93.74	63.76	55.11	88.46	42.31	34.38	62.96
Pixel (Lin,2)	0.2	2	92.24	49.04	13.75	76.92	20.83	13.16	44.32
Pixel (Lin,2)	0.2	1+2	93.45	49.49	14.03	100.00	12.00	11.54	46.75

Table 21Results of thresholded difference

Positive Thresholding:

Illumination Compensation	*Ill. Threshold*	*Line*	*Good Classified*	*Average perf.*	*Product perf.*	E_{cnt}	*Detect. Rate*	*Det. Right Rate*	*Perf average*
None	None	1	88.86	51.71	33.09	55.77	42.31	22.45	49.03
None	None	2	93.13	48.57	0.00	94.23	8.33	6.90	41.86
None	None	1+2	79.96	44.07	20.98	91.35	16.00	13.11	44.24
Linear Global	None	1	81.11	50.77	39.04	90.38	53.85	45.16	60.05
Linear Global	None	2	93.72	48.88	0.00	100.00	8.33	7.69	43.10
Linear Global	None	1+2	63.98	44.03	38.41	17.31	68.00	24.64	42.73
Quadratic Global	None	1	81.14	49.59	36.34	86.54	53.85	42.42	58.31
Quadratic Global	None	2	93.09	48.55	0.00	100.00	8.33	7.69	42.94
Quadratic Global	None	1+2	64.99	44.03	37.77	25.00	62.00	23.85	42.94
Linear Box (5)	None	1	93.35	48.25	0.00	96.15	0.00	0.00	39.62
Linear Box (5)	None	2	53.43	43.32	41.90	86.54	37.50	27.27	48.33
Linear Box (5)	None	1+2	96.32	50.16	5.59	1.92	2.00	100.00	42.67
Quadratic Box (5)	None	1	60.20	53.91	53.49	65.38	30.77	18.18	46.99
Quadratic Box (5)	None	2	66.85	49.65	45.97	100.00	37.50	34.62	55.76
Quadratic Box (5)	None	1+2	58.22	53.06	52.77	73.08	28.00	17.50	47.10
Pixel (Lin,2)	None	1	96.09	49.66	0.00	92.31	0.00	0.00	39.68
Pixel (Lin,2)	None	2	53.59	41.26	39.00	80.77	16.67	19.05	41.72
Pixel (Lin,2)	None	1+2	95.85	49.76	0.00	99.04	0.00	0.00	40.77
Pixel (Lin,2)	0.2	1	89.30	52.45	34.60	63.46	42.31	24.44	51.09
Pixel (Lin,2)	0.2	2	93.16	48.59	0.00	96.15	8.33	7.14	42.23
Pixel (Lin,2)	0.2	1+2	79.99	44.08	20.98	92.31	16.00	13.33	44.45

Table 22Results of positive thresholding

Kolmogorov Smirnov test statistic:

Illumination Compensation	*Ill. Threshold*	*Line*	*Good Classified*	*Average perf.*	*Product perf.*	E_{cnt}	*Detect. Rate*	*Det. Right Rate*	*Perf average*
None	None	1	94.64	58.27	43.38	100.00	38.46	38.46	62.20
None	None	2	93.86	50.43	17.38	96.15	16.67	16.00	48.41
None	None	1+2	94.06	50.86	20.28	89.42	24.00	19.05	49.61
Linear Global	None	1	95.56	56.19	37.21	96.15	42.31	39.29	61.12
Linear Global	None	2	93.75	49.57	11.72	86.54	25.00	18.18	47.46
Linear Global	None	1+2	95.03	52.57	25.71	98.08	28.00	27.45	54.47
Quadratic Global	None	1	95.66	54.55	32.26	98.08	38.46	37.04	59.34
Quadratic Global	None	2	94.08	49.06	0.00	100.00	8.33	7.69	43.19
Quadratic Global	None	1+2	93.33	49.28	12.90	98.08	16.00	15.69	47.55
Linear Box (5)	None	1	96.38	62.74	51.40	94.23	57.69	51.72	69.03
Linear Box (5)	None	2	95.45	49.78	0.00	100.00	4.17	3.85	42.21
Linear Box (5)	None	1+2	94.14	54.81	34.65	84.62	36.00	26.47	55.12
Quadratic Box (5)	None	1	96.50	61.27	48.32	96.15	46.15	42.86	65.21
Quadratic Box (5)	None	2	95.57	51.72	19.78	100.00	20.83	19.23	51.19
Quadratic Box (5)	None	1+2	89.50	56.08	42.93	97.12	30.00	27.27	57.15
Pixel (Lin,2)	None	1	93.11	64.62	56.99	73.08	50.00	32.50	61.72
Pixel (Lin,2)	None	2	95.21	49.92	7.47	96.15	4.17	4.00	42.82
Pixel (Lin,2)	None	1+2	94.63	56.49	38.67	97.12	30.00	27.27	57.36
Pixel (Lin,2)	0.2	1	94.66	58.28	43.38	100.00	38.46	38.46	62.21
Pixel (Lin,2)	0.2	2	93.81	50.40	17.38	100.00	16.67	15.38	48.94
Pixel (Lin,2)	0.2	1+2	94.17	50.92	20.29	91.35	22.00	18.03	49.46

Table 23Results of Kolmogorov Smirnov

Smirnov:

Illumination Compensation	*Ill. Threshold*	*Line*	*Good Classified*	*Average perf.*	*Product perf.*	E_{cnt}	*Detect. Rate*	*Det. Right Rate*	*Perf average*
None	None	1	93.44	62.41	52.86	98.08	34.62	33.33	62.46
None	None	2	93.22	49.56	13.82	90.38	16.67	12.90	46.09
None	None	1+2	84.92	52.27	38.61	56.73	38.00	19.59	48.35
Linear Global	None	1	94.88	54.65	33.69	100.00	34.62	34.62	58.74
Linear Global	None	2	94.15	50.58	17.41	100.00	29.17	26.92	53.04
Linear Global	None	1+2	94.40	51.71	23.46	97.12	36.00	32.73	55.90
Quadratic Global	None	1	95.39	55.43	35.28	96.15	42.31	39.29	60.64
Quadratic Global	None	2	93.63	48.83	0.00	100.00	8.33	7.69	43.08
Quadratic Global	None	1+2	85.02	47.44	24.60	58.65	42.00	22.11	46.64
Linear Box (5)	None	1	94.08	50.50	19.39	100.00	11.54	11.54	47.84
Linear Box (5)	None	2	83.86	48.30	28.83	71.15	62.50	36.59	55.21
Linear Box (5)	None	1+2	91.04	50.49	25.17	100.00	16.00	15.38	49.68
Quadratic Box (5)	None	1	95.97	56.92	38.66	100.00	38.46	38.46	61.41
Quadratic Box (5)	None	2	90.12	48.48	17.03	48.08	50.00	22.64	46.06
Quadratic Box (5)	None	1+2	94.47	56.33	38.44	99.04	24.00	22.64	55.82
Pixel (Lin,2)	None	1	93.65	67.28	61.09	73.08	46.15	30.00	61.87
Pixel (Lin,2)	None	2	83.80	47.06	24.73	67.31	41.67	23.26	47.97
Pixel (Lin,2)	None	1+2	92.32	54.69	36.61	94.23	26.00	22.41	54.38
Pixel (Lin,2)	0.2	1	93.59	62.15	52.27	100.00	34.62	34.62	62.87
Pixel (Lin,2)	0.2	2	93.15	49.52	13.82	90.38	16.67	12.90	46.07
Pixel (Lin,2)	0.2	1+2	84.91	52.35	38.78	53.85	38.00	19.00	47.81

Table 24Results of Smirnov

Modified Smirnov:

Illumination Compensation	*Ill. Threshold*	*Line*	*Good Classified*	*Average perf.*	*Product perf.*	E_{cnt}	*Detect. Rate*	*Det. Right Rate*	*Perf average*
None	None	1	92.38	62.55	53.79	84.62	46.15	35.29	62.47
None	None	2	92.19	49.02	13.74	92.31	16.67	13.33	46.21
None	None	1+2	93.30	49.41	14.02	100.00	12.00	11.54	46.71
Linear Global	None	1	94.24	51.94	25.50	96.15	38.46	35.71	57.00
Linear Global	None	2	93.85	50.56	18.15	98.08	33.33	29.63	53.93
Linear Global	None	1+2	94.41	51.65	23.13	92.31	24.00	20.00	50.92
Quadratic Global	None	1	94.64	51.13	21.14	94.23	38.46	34.48	55.68
Quadratic Global	None	2	93.09	48.55	0.00	100.00	16.67	15.38	45.62
Quadratic Global	None	1+2	93.26	48.57	5.50	100.00	10.00	9.62	44.49
Linear Box (5)	None	1	60.57	55.97	55.76	40.38	46.15	21.05	46.65
Linear Box (5)	None	2	68.19	43.76	34.73	51.92	66.67	31.37	49.44
Linear Box (5)	None	1+2	78.55	46.19	30.21	74.04	34.00	21.52	47.42
Quadratic Box (5)	None	1	61.95	50.39	48.85	69.23	34.62	21.43	47.74
Quadratic Box (5)	None	2	52.19	44.69	43.94	78.85	41.67	27.03	48.06
Quadratic Box (5)	None	1+2	74.28	46.68	35.92	69.23	24.00	14.29	44.07
Pixel (Lin,2)	None	1	62.54	50.87	49.31	100.00	19.23	19.23	50.20
Pixel (Lin,2)	None	2	89.66	47.70	13.55	90.38	25.00	19.35	47.61
Pixel (Lin,2)	None	1+2	74.55	47.27	36.97	84.62	20.00	14.71	46.35
Pixel (Lin,2)	0.2	1	92.36	62.54	53.79	84.62	46.15	35.29	62.46
Pixel (Lin,2)	0.2	2	92.21	49.03	13.75	92.31	16.67	13.33	46.22
Pixel (Lin,2)	0.2	1+2	93.30	49.41	14.02	100.00	12.00	11.54	46.71

Table 25Results of modified Smirnov

Approximate Entropy:

Illumination Compensation	*Ill. Thresh old*	*Line*	*Good Classified*	*Average perf.*	*Product perf.*	E_{cnt}	*Detect. Rate*	*Det. Right Rate*	*Perf average*
None	None	1	96.53	54.15	29.60	92.31	26.92	29.17	54.78
None	None	2	95.51	52.78	24.78	96.15	29.17	25.00	53.90
None	None	1+2	96.04	53.47	27.31	99.04	26.00	24.53	54.40

Table 26Results of Approximate Entropy

16 References

[AEH: AHMC] Kjersti Aas, Line Eikvil and Ragnar Bang Huseby, Department of Pattern Recognition and Image Processing, Norwegian Computing Center.

Applications of Hidden Markov Chains in Image Analysis.

citeseer.nj.nec.com/aas99applications.html

Pat. Recog. 32(4), pp. 703--713, 1999.

[DG: MRVCC] Nevenka Dimitrova and Forouzan Golshani, Arizona State university, Tempe

Motion recovery for video content classification

ACM Transactions on Information Systems, 13(4):408--439, October 1995

[EH:TSHMM] L. Eikvil and R.B. Huseby, Norwegian Computing Center, Oslo, Norway

Traffic Surveillance in Real-time using Hidden Markow Models

citeseer.nj.nec.com/eikvil01traffic.html

12th Scandinavian Conference on Image Analysis (SCIA). Proceedings. Bergen. June 11-14, 2001.

[FB:ERIMA] Ronan Fablet (Brown University, Providence) and Patrick Bouthemy (IRISA/ INRIA Rennes).

Extraction of regions of interest based on motion activity for video retrieval with partial query

9th Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU'2002, Annecy, July 2002

[G:VAHM] D.M. Gavrila, Image understanding Systems, Daimler-Benz Research

The Visual Analysis of Human Movement: A Survey

http://citeseer.nj.nec.com/gavrila99visual.html

D. Gavrila. The Visual Analysis of Human Movement: A Survey. Computer Vision and Image Understanding, 73(1):82-- 98, Jan. 1999.

[H: FSUAMM] Michael Harville, Hewlett-Packard Labs, Palo Alto

Foreground Segmentation Using Adaptive Mixture Modeld in Color and Depth

http://citeseer.nj.nec.com/harville01foreground.html

M. Harville, G. Gordon, and J. Woodfill. Foreground segmentation using adaptive mixture models in color and depth. In Workshop on Detection and Recognition of Events in Video., 2001.

[MR:MVBSS] Malik, Russel,Weber,Huang,Koller, Computer Science Sivision, University of California.

A Machine Vision Based Surveillance System for California Roads

citeseer.nj.nec.com/9139.html

Not published in print.

[MPMPA] Molina, Prades, Mossi, Pons & Albiol, Universidad Politecnia de Valencia Development of Video-Sensors for a surveillance System.

[MVSCD] Julio Pons, Josep Prades, Antonio Albiol &Jesus Molina, Universidad Politecnia de Valencia

Motion video sensor in the compressed domain.

CONFERENCE: 2001 SCS Euromedia Conference (Mediatec)
pp. 205-208 ISBN: 1-56555-217-2

[N:MDuAE] Phillip M. Ngan: Motion detection using Approximate entropy

DICTA/IVCNZ97, Massey University, New Zealand, December 1997, pp 379-384

[PD:DMO] Nikolaos Paragios and Rachid Deriche, INRIA, Sophia-Antipolis Cedex, France

Detection of moving objects: A Level Set Approach

citeseer.nj.nec.com/296877.html

International Symposium on Intelligent Robotics Systems, Jul. 97, Stocholm, Sweden.

[VMDTLK] http://www.videomotiondetectors.com/manuals/VMDtalk.htm

[YFH:CDPS] Stewart Young, Micheal Forshaw & Mark Hodgetts, Dept. Physics &Astronomy University College London

Change Detection for Perimeter Surveillance

http://ipga.phys.ucl.ac.uk/papers/detect1.pdf

Internal Document, not published

[YFH:ICMPS] Stewart Young, Micheal Forshaw & Mark Hodgetts, Dept. Physics &Astronomy University College London

Image comparison methods for perimeter surveillance

http://ipga.phys.ucl.ac.uk/papers/compare1.pdf

IEE 7th International Conf. on Image Processing and its Applications, Manchester UK, 12-15 July 1999, p.799-803.

[SHB:IPA] Milan Sonka, Vaclav Hlavac and Roger Boyle

Image processing, Analysis, and Machine Vision

ISBN: 0-534-95393-X

[Texas] INITIAL EVALUATION OF SELECTED DETECTORS TO REPLACE INDUCTIVE LOOPS ON FREEWAYS
Dan Middleton and Rick Parker
Texas Transportation Institute
http://tti.tamu.edu/product/catalog/reports/1439-7.pdf