Development of a computer-vision system for bicycles detection

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Studenti: Michele Rodolfi, Giulio Bacchiani

Professore: Luigi Di Stefano

Abstract

This project has been realized as a practical exercise on the subject of “Image Processing and Computer Vision”. Our goal was to develop a solution that could be also useful for the society where we live. For this reason we chose to create a system to count the number of bicycles passing in a particular street or bike trail.

This system, opportunely tuned, could be a powerful tool in the hands of municipalities interested in sustainable mobility, to detect which streets of the city are strongly used by bicycles, for instance to find the best places to build new bike lanes.

In fact, in our cities, it is easy to find video detection systems dedicated to motorized vehicles with the purpose of access regulation or traffic control.

Generally the cameras are combined with underground coils to signal the presence of a car in the street. More difficult is to find such systems dedicated to bicycles or pedestrians, because of the absence of pollution they produce, their smaller volume, and the impossibility to gain money for them with tickets since they don’t have plates.

This project aims to provide this functionality in a way that avoids the needs of invasive installation of coils in the streets or zones where we want to control the flux of bycicles.

Table of Contents

Introduction

The functionality of this system is to detect the number of bicycles that cross a virtual line positioned on the street. Many algorithms exist for this purpose: some of them are highly accurate but also complex, and they need a training phase and additional processing when the place changes. Others, because of the higher simplicity, are more flexible in spite of a minor number of information extracted from the video.

The peculiarity of this system is to be easily adaptable to different places, without the need of too much effort for reconfigure the system. This is important if you don’t know the place where it has to be implemented, or when the places to controls are more than one, as in case of city traffic control.

This is the reason why we have initially implemented the simplest and more flexible change detection algorithm: the two-frame difference. We have chosen this instead of the three-frame difference because the bigger obstacle to this system is due to the foreground aperture, which is more present in the three-frame difference algorithm.

Moreover, the major presence of ghosting in the two-frame algorithm does not add too many artifacts, rather it has been used to better discriminate motor bikes from bikes, as explained later.

Location

The choice of the location where placing the camera is important for the correct functionality of the system. A too high position reduces the screen-size of the vehicles passing on the street while a too low position will provide a too narrow recording view. At the same time a lateral position cause a problem when two vehicles are overlapped.

Another problem could be caused by the shadow of the objects, if the street is directly hit by the sun. In this situations the system needs to be tuned accurately taking into account the shape of the vehicles when they are extracted from the change detection algorithm, and maybe modify the discrimination policy taking into account the actual time.

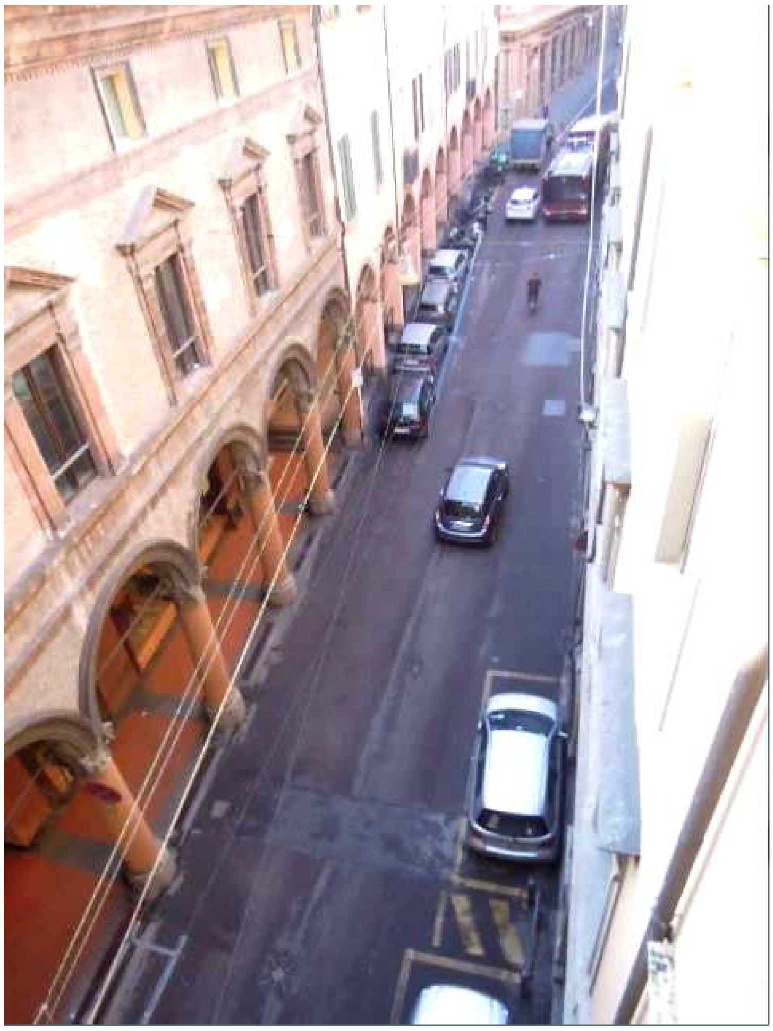

*Figure 1 Frame of the recorded video. The resolution is 640*480 pixel.*

The place used for our test video is on the top of a medieval tower in Via San Vitale in Bologna (Figure 1), but also a more ordinary one could be possible in general, as on the top of a lamp post, a traffic light, etc.

The Two-Frame Difference Algorithm

The two-frame difference is a simple change-detection algorithm. Its principle is to detect movement of objects by comparing two consecutive frames, and signaling those pixel whose values have changed by more than the threshold.

The flux diagram is shown in figure Figure 2.

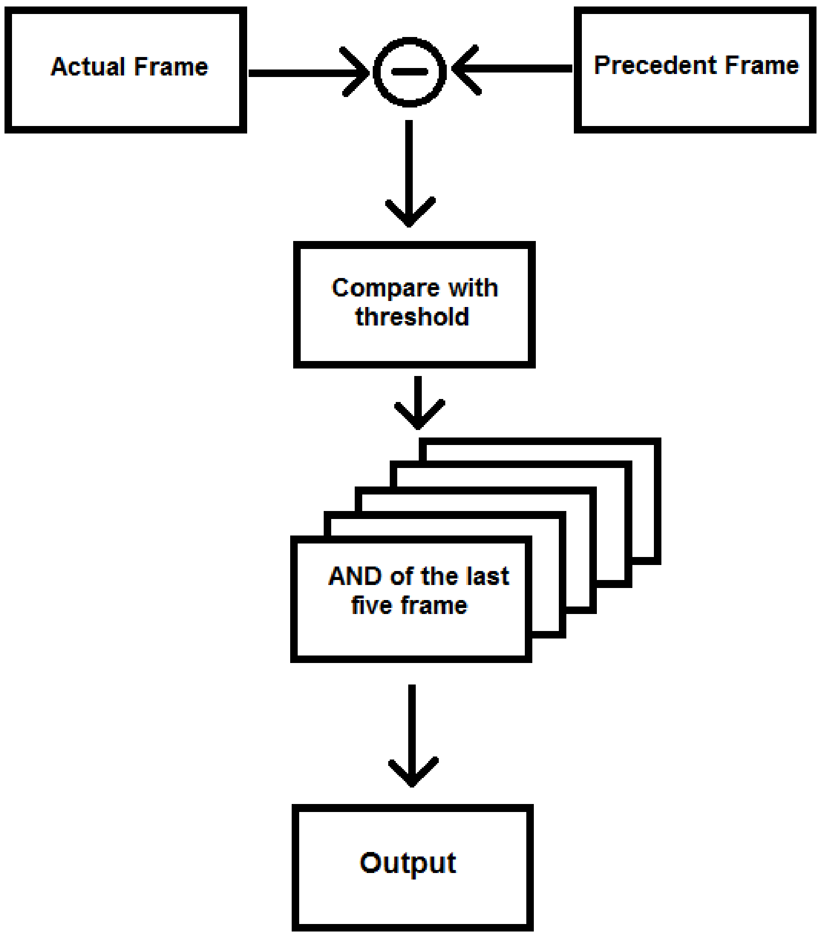

Figure 2 Flux diagram of the two-frame difference algorithm.

A high value of the threshold will produce no noise but the object will appear fragmented due to less pixel “activated”. On the contrary, a low threshold will cause too much noise and false detections.

We found a good compromise using a low threshold value (a pixel is signaled when its distance from the same pixel in the frame before is more than 3 units) and after that computing the logical operation “AND” between this result frame and the four before. This will result in a delay on the detection of a new object in the line, due to the lag of five frames (150ms) before all the five pixel of the further point of the object in the five frame will become active.

The frame before and after the noise reduction are shown in Figure 3.

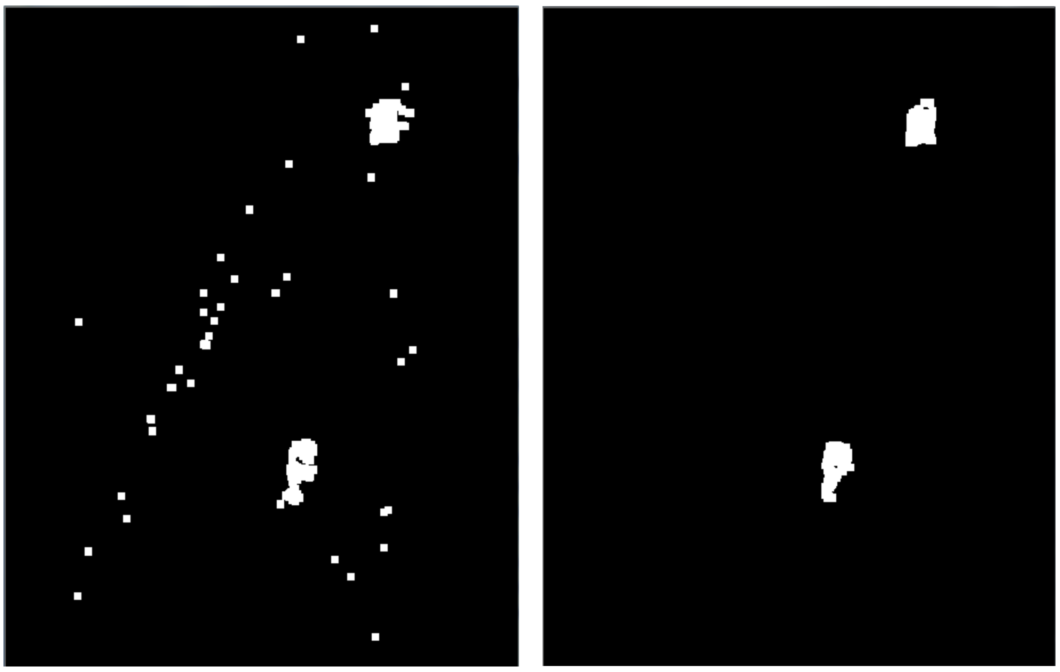

Figure 3 Comparison between the output of the two frame difference algorithm and the result of the AND operation among 5 different output frame of the same algorithm

A common problem of the two frame difference is *ghosting*, due to the activation of both the new position of the moving object and the older one. In this context, this is not considered as a problem; on the contrary it has been used as way to discriminate the motorbikes from bicycles. This was possible because *ghosting* is proportional to the velocity of the object, and generally motorbikes' speed is higher than bikes' speed, and so the resulting area of the former is higher than that of the latter.

Another problem of this category of algorithms is the foreground aperture, namely the non-activation of pixels of a moving object, caused by the uniformity of the surface of that object. This error appears often on the roof of cars or on the body of a bus, where there are not many features capable to create distinction between the two frame, as shown in figure Figure 4. Without this distinction, the two frame algorithm will consider those pixels as part of the background.

Luckily, this error is not present in bicycles or motorbikes, objects which we are more interested about in this project.

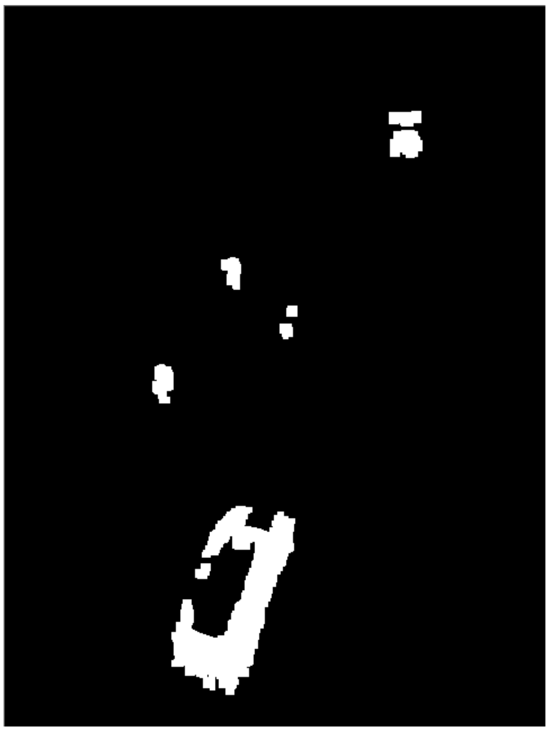

Figure 4 Example of foreground aperture on the roof of a car

The Background Subtraction Algorithm

The Background Subtraction Algorithm is an advanced change-detection algorithm. A representation of the static part of the video is computed and then subtracted from the current frame to detect which part of the frame are changing and therefore moving. The operation itself is similar to the two frame difference algorithm, but to get a good background there must be a prior background initialization phase, followed by a continuous background updating phase.

The flux diagram is shown in Figure 5.

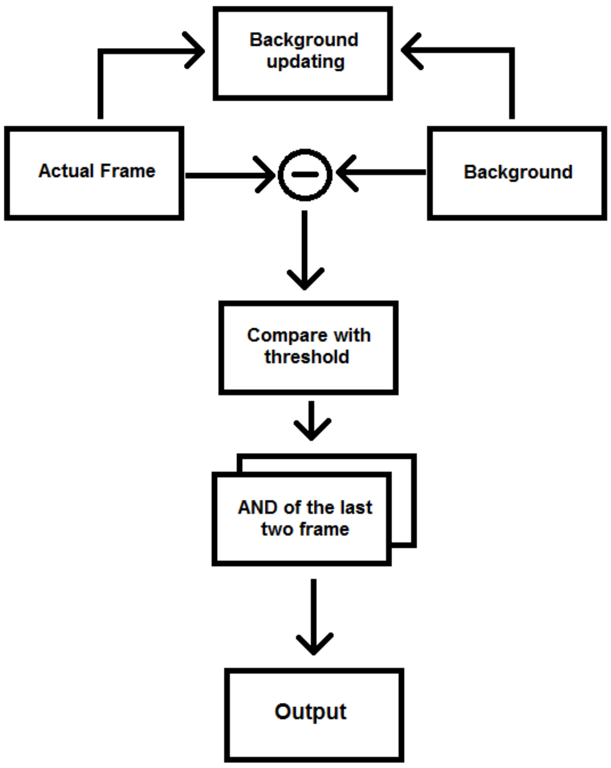

Figure 5 General flux diagram of the background subtraction algorithm

Because of these additional operations, the background subtraction algorithm is more expensive than the two frame difference one, but it resolves the main problem of the latter: the foreground aperture, which may split a moving object in the foreground mask, causing problems in counting he connected components passing through a line. Despite this achievement, the background subtraction algorithm introduce some new problems, as explained in the following.

We implemented the background initialization phase calculating the modal value for each pixel in the first 100 frames of the video; instead two alternative ways have been used as background updating phase: each of them with some pros and cons.

The first one is similar to the algorithm used by the background initialization. Basically every 100 frames the background is reinitialized and used to calculate the foreground mask for the next 100 frames. Because of the huge resource usage of the mode detection operation, we implemented the median detection rather than the modal, that is less resource hungry, but enough effective.

To add some robustness, we filtered the background updating with the change detection mask obtained from the two frame difference algorithm: if the pixel is in the foreground it will not be considered in the updating of the background.

This algorithm is effective and under stationary light condition it can be used without problems, as show in Figure 6 but it is not responsive to sudden light changes. For example, when a white car passes in the street, the sunbeams are reflected on its roof and spread on the buildings around and on the road, causing the foreground mask to become completely white. This behavior can be seen in Figure 7. If this condition lasts in time, the statistics for the background updating will be ruined and the background will be faked. Several frames will be needed to restore a valid background.

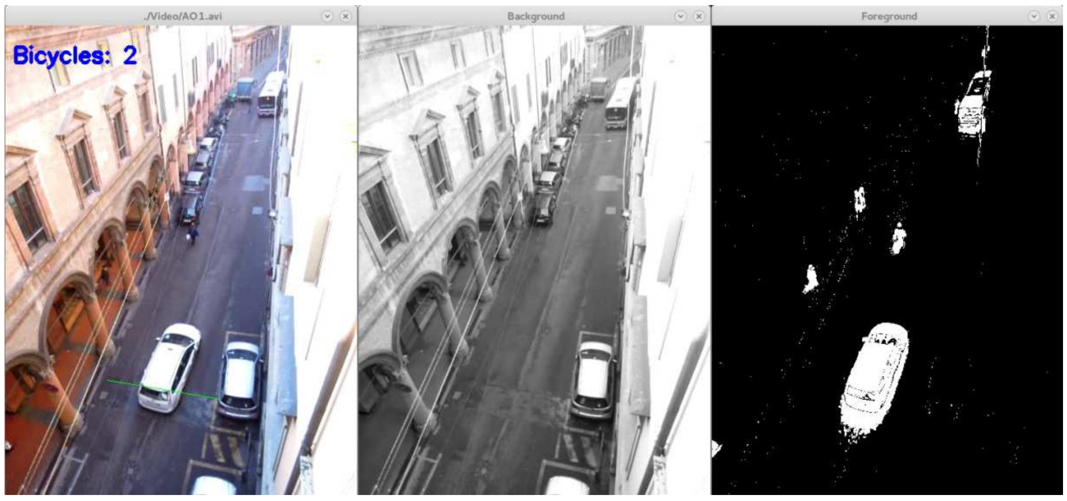

Figure 6 Current frame, background and foreground computed by background subtraction with median updating

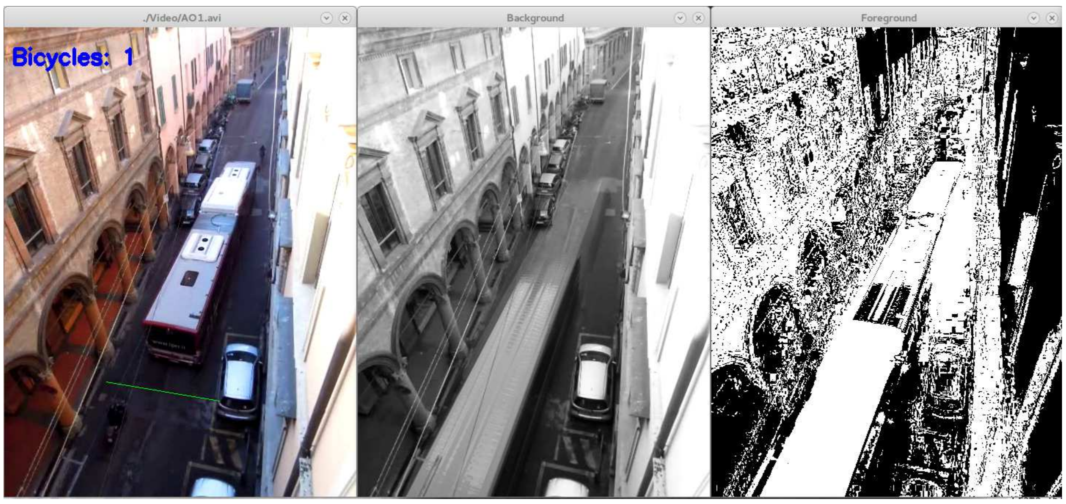

Figure 7 Current frame, background and foreground computed by background subtraction with median updating with a sudden light change due to the autobus' transit.

Afterwards we implemented a background updating algorithm based on counting how many consecutive times a pixel is different from the background. The idea is that if a pixel changes its gray-scale and remains different from the actual background for 100 consecutive frames, then this pixel should be part of the background. The mode of operation of this algorithm is clarified by the flux diagram in Figure 8.

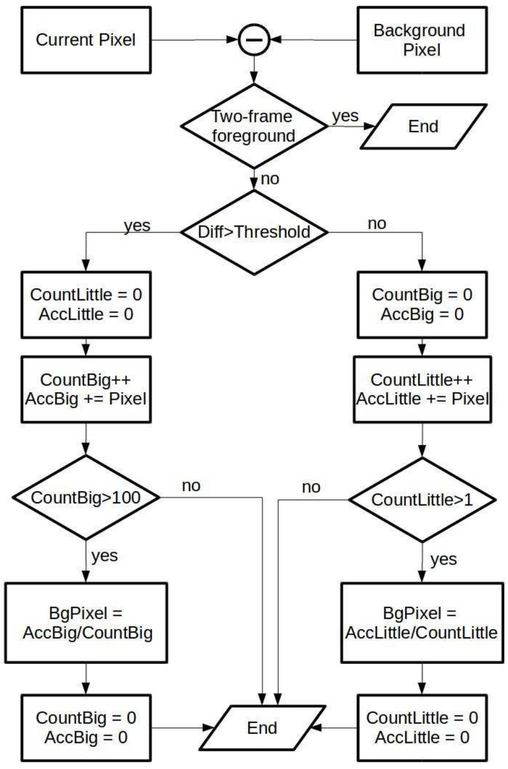

Figure 8 Flux diagram of the second background updating algorithm

This algorithm generate a less defined background and it is more subject to noise, but it can deal with light changes. In some cases it leaves a sort of trail behind the moving object, but in general this algorithm is more reliable than the previous one. An example can be seen in Figure 9.

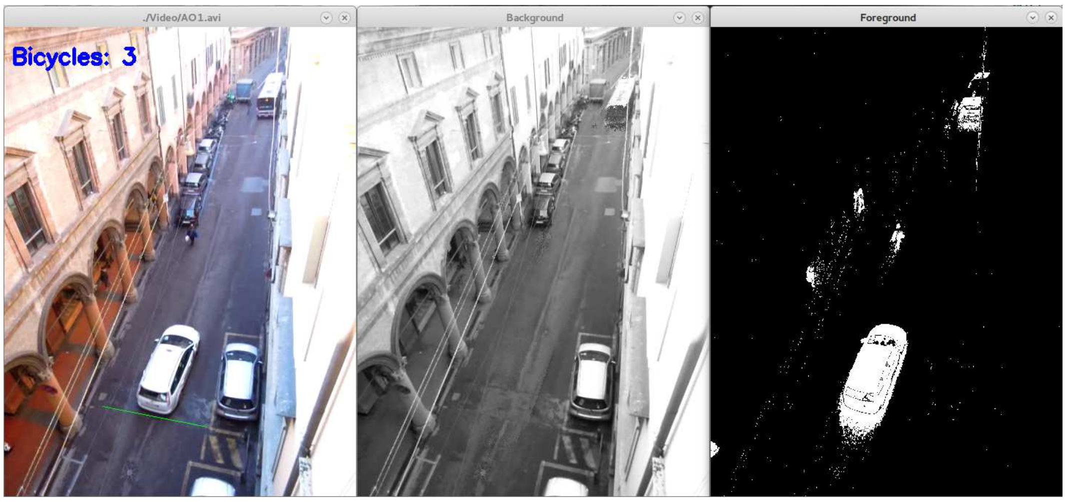

Figure 9 Current frame, background and foreground computed by the second algorithm

Triggering Action

A necessity of the software is to include a “system” that detects when a new object crosses the line (let’s call it “*trigger line*”) draws on the frame, and that does it only once. This because a multiple warning for the same object would create the need for a machine understanding that the object detected is the same, to avoid counting it more than once.

This is partially obtained by recording which pixels of the *trigger line* were excited in the previous frame.

As first step, we look if there are some foreground pixel on the *trigger line;* if it’s the case we have to understand if a new object activated them or not. For this reason we look on the contour of these pixels in the stored previous situation of the *trigger line.* If no close pixels were active it means that we are in presence of a new object, but we have still to realize if this new object has already been signaled from other pixels active in the actual *trigger line*. For achieve this result, we store as active a foreground pixel on the line only after we have controlled if some pixels are already active in his contour. Doing that we are sure that only the first time we find a new pixel in the trigger line the object will be signaled because no other pixels close to him are active, and after that the next new candidate (if belong to the same object) will find that pixel active, causing no trigger call and starting a chain reaction.

The all procedure, simplified, is shown in the flux diagram of Figure 10.

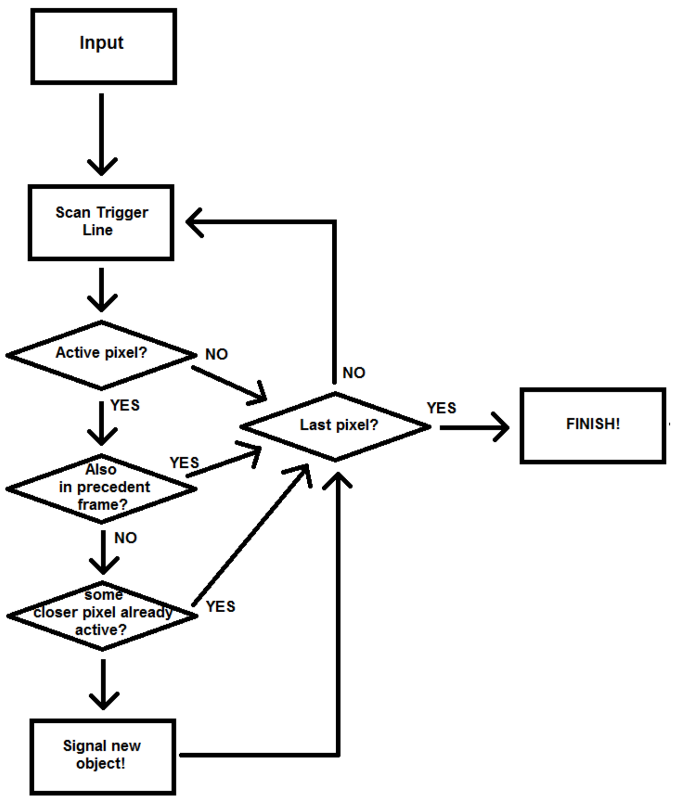

Figure 10 Flux diagram of the trigger part of the implemented software

This implies that a connected (or almost connected, because we look in a small area where pixels are considered to belong to the same object) group of pixel in the line are signaled only once, and that if this group of pixel was already active in the precedent frame no signal occurred.

In practice, this operations are obtained by scanning the line doing an AND operation of the result of the two-frame difference part and a mask where only the virtual line is present as white.

The general policy adopted for this trigger system is to prefer to generate a false signal rather than leave the possibility to lose the transit of an object. It will be then task of the functions that will analyze the pixel who has activated the trigger to understand if it was an interesting object or a false positive.

In fact, cars or other object subject to foreground aperture error like trams, will probably activate the trigger more than once. (and this introduces also the problem of not counting multiple times cars in the case that we will be interested also in them).

Reconstruction of the Object

Once we receive the call from the software trigger, the task becomes to reconstruct the object involved in this call. So, as first step, the software will scan the frame resulting from the two frame difference algorithm selecting those pixels connected to the pixels responsible of the activation of the trigger. This has been achieved by scanning multiple times that frame until no connected pixels are found, as shown in Figure 11. For optimization reasons, it will alternate a downward and an upward scan.

Each time a pixel is set, a counter is incremented: its final result it will be equal to the area.

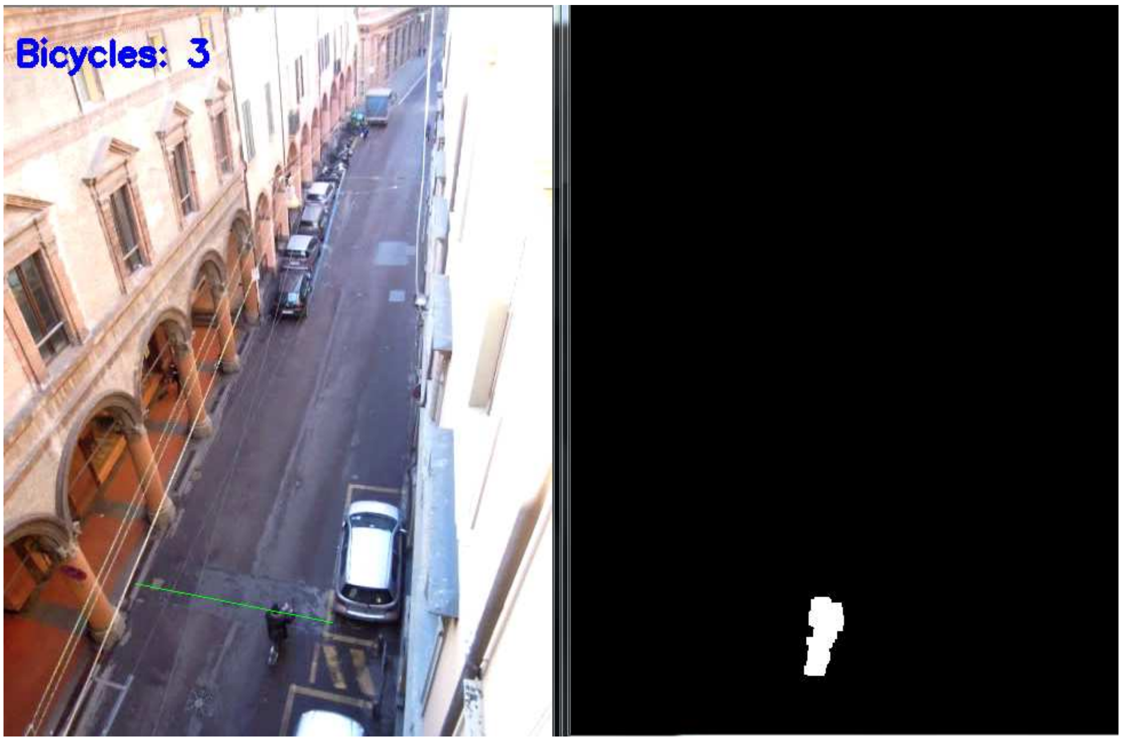

Figure 11 Reconstruction of the connected object which has been detected from the trigger.

Others operation computed are the calculus of both coordinates of the barycenter, which will be useful to measure the dimensions of the vehicle analyzed. These operations are simply the following:

$$x_B = \frac{1}{Area} \cdot \sum_{p \in object} x_p \; y_B = \frac{1}{Area} \cdot \sum_{p \in object} y_p$$

Where x_B and y_B are the coordinates of the barycenter; x_p and y_p are *x* and *y* coordinates of the generic point belonged to the object under investigation.

After that, we scan again the final image, and we set as black all those pixel which are completely surrounded by white pixel, leaving in this way only the perimeter, gaining another interesting parameter, used to find the Compactness ($\frac{Perimeter \cdot Perimeter}{Area}$).

Finding the perimeter gives us another interesting advantage, since the calculus of dimensions, length and width, needs to be done pixel by pixel. In fact, there is guarantee that the pixels farer from the barycenter, and so that define the dimensions, lie on the perimeter. Therefore, we perform the distance calculation only in those pixels belonging to the boundary, gaining efficiency.

To find the length we calculate the distance of each pixel from the line parallel to the *trigger line* passing for the barycenter and choosing the bigger (positive) and the smaller (negative) ones; the final value will be the sum of the two absolute values.

The same needs to be done for the length, but instead of doing with a line parallel to the *trigger line*, it has be done with a line perpendicular to it, always passing for the barycenter.

The equation of the trigger line has been extracted from the two points chosen by the user using the following simple equation $\frac{a}{b} = \frac{y_B - y_A}{x_A - x_B}$

Where *a* and *b* are the coefficients which multiply respectively the x and y variables, *A* and *B* are the two points known of the line. The coefficient *c* has been found by forcing the line to pass by the barycenter.

Finding the coefficients of a perpendicular of a line is an easy task, and it has been achieved by founding those coefficients that satisfy the following equation:

$$\frac{a}{b} = \frac{-b_{perpendicular}}{a_{perpendicular}}$$

Where *a* and *b* are the coefficients of the original line, and $a_{perpendicular}$ and $b_{perpendicular}$ the ones in which we are interested.

Now, with some more basic geometrical, we are able to draw the bounding box of the object related to a direction chosen as the normal direction of the traffic in the street.

The steps explained are shown in the sketch of Figure 12:

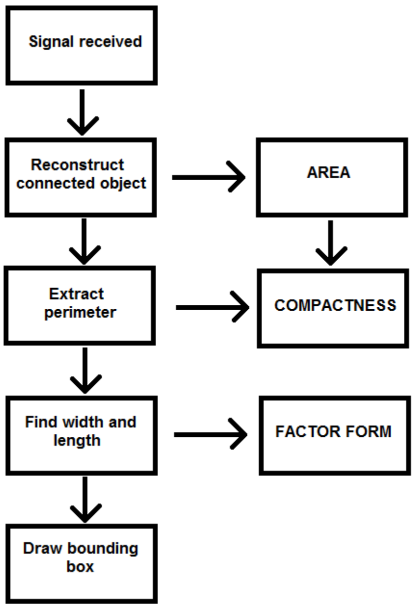

Figure 12 Sequence of steps executed for the reconstruction of the object

The final result of the reconstruction, is shown in Figure 13.

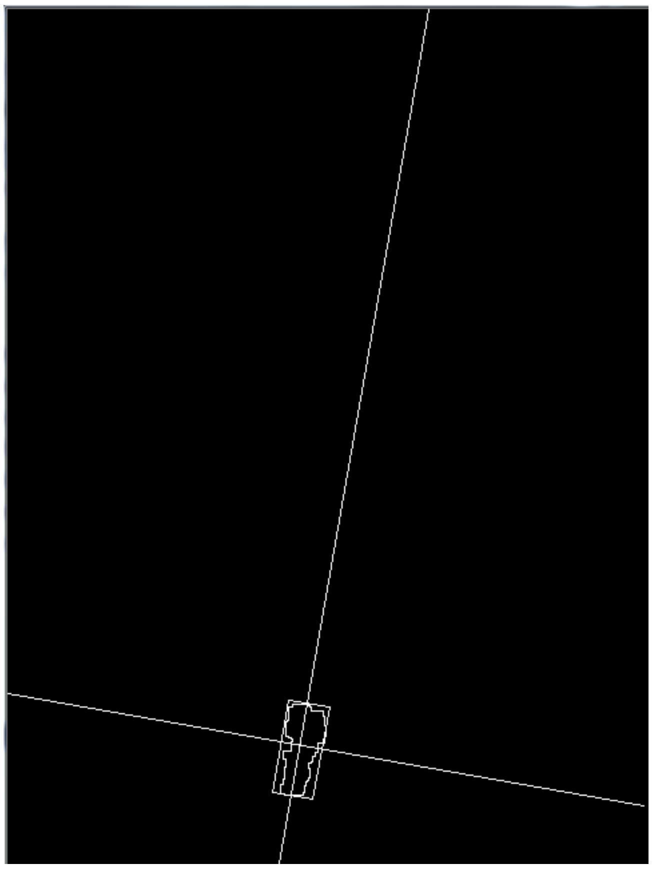

Figure 13 Final result of the reconstruction phase, including perimeter extraction and the drawing of the bounding box

Discrimination Policy

Actually the discrimination policy adopted is “direct” and “in or out”.

“In or out” it means that we count the object as a bicycle if it has all the controlled parameters in the chosen ranges, which are the area, the compactness and the factor form (length/wide). A possible alternative is to give a weight to each parameter, using it to give a score to the object, and then look if this score is enough to be classified as bicycle.

“Direct” it means that the ranges are chosen for a determined line, and they need to be rewritten if this line changes.

This strategy is summarize in Figure 14.

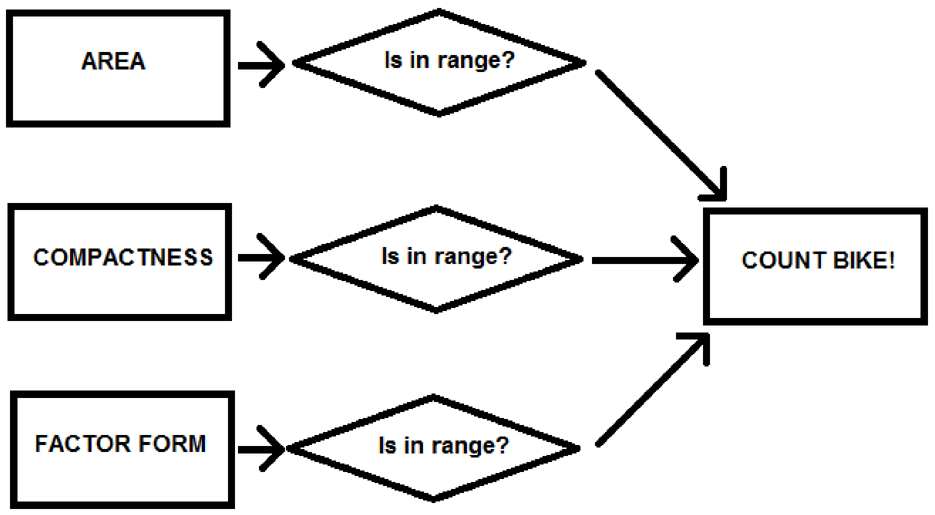

Figure 14 Discrimination policy used to detect a new bike

Another possibility is to choose ranges and discrimination policies as function of the line chosen. However, this strategy is not so practical, because if the place changes, this function needs to be completely rewritten, operation more complex and more time consuming than changing manually the parameters.

Conclusions

In this project two main strategy were applied to permit to count the number of bicycles passing through a line: one was based on the two frame difference algorithm while the second one on the background subtraction.

As often happen, we didn't find the "perfect" algorithm, because each one has some pro and cons. But we can state that in our scenario, the simpler one, the two frame difference, showed a better behavior than its more complex brother.

The former has the big limitation of manifesting foreground aperture problems in those moving objects having uniform surfaces, that may cause the wrong identification of bikes in the spurious part in which a connected uniform object could be split.

This problem is solved if background subtraction is used; on the other hand this algorithm has some drawbacks, as the presence of trail in moving objects due to the need of a fast background updating. This need comes from the necessity of manage sudden light changes cause by the reflection of sunlight in the roof of cars. This problem cause the incapacity to detect such bikes that are moving close behind a car or a truck.

Instead, robustness toward scene changes is one of the major benefits of the two frame difference algorithm, precisely because its simplicity and lack of long memory.

Another advantage comes from a characteristic that typically is seen as a problem of such kind of algorithms: ghosting. Ghosting, which is dependent on the shape and speed of the moving object, adds more information to our system, because it affects in different ways each typology of each target, permitting a correct discrimination between bikes, pedestrian and scooter.

Moreover, the identification problems due to foreground aperture have been coped by applying some morphology elaboration and by extracting and checking several parameters of the object under investigation, like its area, compactness and factor form.

For this reasons we have chosen to rely on the two frame difference algorithm as change-detection strategy, and we can say that our results are fully satisfying.

The overall flux diagram of the system implementing the two frame difference algorithm is shown in Figure 15.

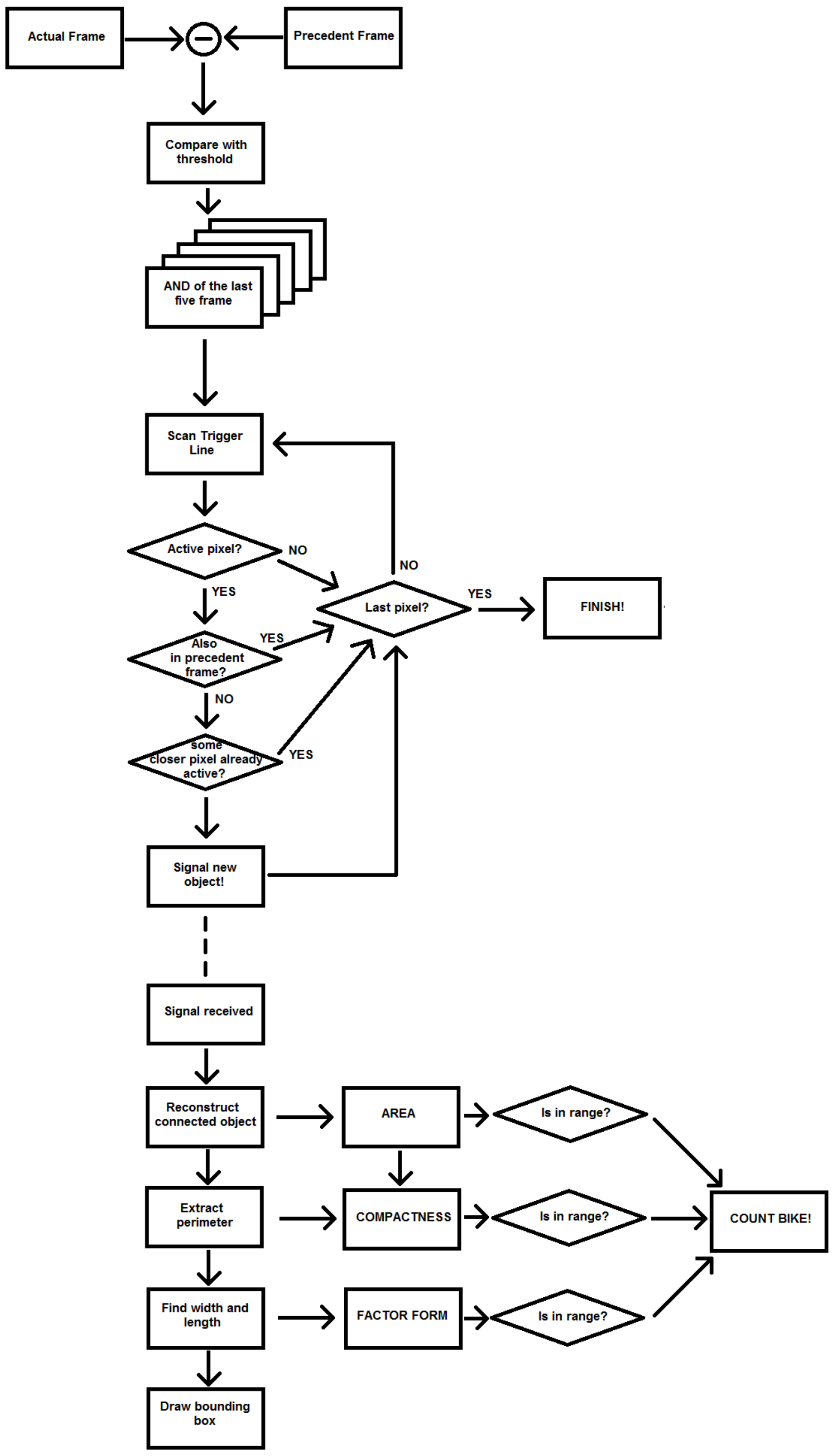

Figure 15 Overall flux diagram of the system

The system in action can be observed on Youtube at this link: https://www.youtube.com/watch?v=zRRuqOu6cgM.