TunnelVision: LHC Tunnel Photogrammetry System for Structural Monitoring

William Fallas C.
Summer Student
EN-STI-ECE
CERN
Switzerland
Email: wfallas@eie.ucr.ac.cr

***Abstract*—In this document an algorithm to detect deformations in the LHC Tunnel of CERN is presented. It is based on two images, one represents the ideal state of the tunnel and the other one the actual state. To find the differences between both, the algorithm is divided in three steps. First, an image enhancement is applied to make easier the detection. Second, two different approaches to reduce noise are applied to one or both images. And third, it is defined a group of characteristics about the type of deformation desired to detect. Finally, the conclusions show the effectiveness of the algorithm in the experimental results.**

I. Introduction

The inspection of a tunnel is one of the most expensive and laborious tasks needed for the prevention of damages and accidents. The LHC Tunnel of *CERN*, in Switzerland, has a length of approximately 27 km. To ensure that the experiments elaborated in this tunnel would not be interrupted by structural fails, it is necessary a complete periodic scan of the structure. In this scan, a group of people make a visual inspection of every meter of the tunnel looking for cracks and deformations in the walls and floor. This represents a great spend of money, time and human effort. Consequently, the development of an automatic system capable of the detection of cracks and deformations in the tunnel is necessary and mandatory.

August 11, 2015

II. Used Development Environment

For the development of this project a set of open source tools were used. The code was written in the programming language *C++*. Meanwhile *OpenCV*, a image processing open source library, was used as a support in the principal algorithm. Also, *Cmake* was chosen as the manager of the building process. Finally, every step of the development process and testing were done using the operative system *Ubuntu 14.04*.

III. Algorithm of Detection

The main program is divided in three different stages, each one with an specific purpose, but all with the same overall objective: to find the real differences between two given images. The first image presented in Fig. 1, witch will be call *State0* from now on, represents the natural state of the tunnel without any deformation or anomalies. In the meantime, the second image or *State1* can be observed in Fig. 2, witch presents a deformation on one of the walls.

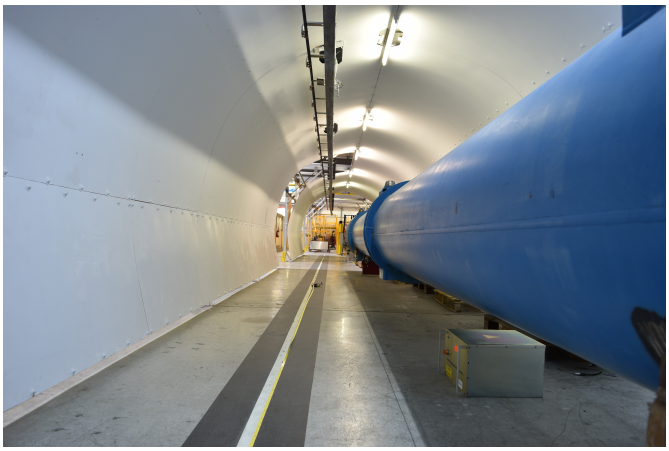

Fig. 1. Image without deformations: *State0*

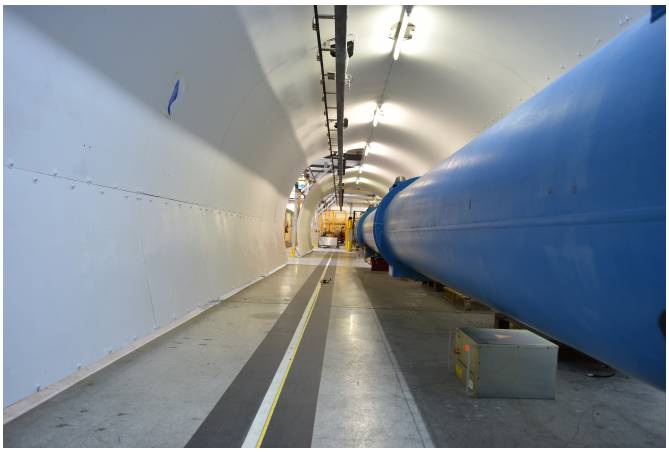

Fig. 2. Image without deformations: *State1*

Now, it is possible to generate a third image subtracting the other two previously presented. This third one displayed in Fig. 3 shows, in a grey-scale form, the absolute differences between *State0* and *State1*.

In order to recognize the real differences and those ones produced by other factors (different lighting conditions, different

Fig. 3. Resulted image of the subtraction of *State0* and *State1*

capture localization, etc), it is necessary to apply a series of methods which are presented as follows.

A. Image Enhancement

In interest of getting a first segmented image, it is convenient to apply a threshold binary operation over the image in Fig. 3. This is, to generate a binary image giving to each pixel a value of one or zero depending on a threshold number. The result is displayed in Fig. 4.

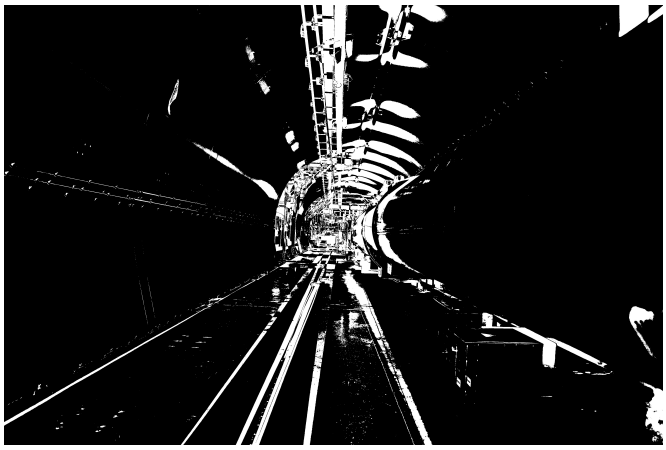

Fig. 4. Binary Image.

Secondly, It is necessary to use some morphological image processing techniques in order to define in a better way the objects presented in the scene. Therefore, a erosion operation is applied to the binary image as well as a *Gausian* smoothing operation to reduce noise and to obtain a clearer image. The resulted enhanced image is showed in Fig. 5.

At this point, the image is ready for a second stage of segmentation. Therefore, in order to identify more accurately the figures in the image, the *Canny Algorithm* to detect edges is applied. Later, with the resulted image, it is possible to find the contours of each object using the *OpenCV* funtion *findContours()*, obtaining finally a segmented image which is displayed in Fig. 9. This image has 4370 different contours which about 0.11% are real differences while the remaining 99.89% are noise. To reduce this noise is the objective in the following section.

Fig. 5. Enhanced Image.

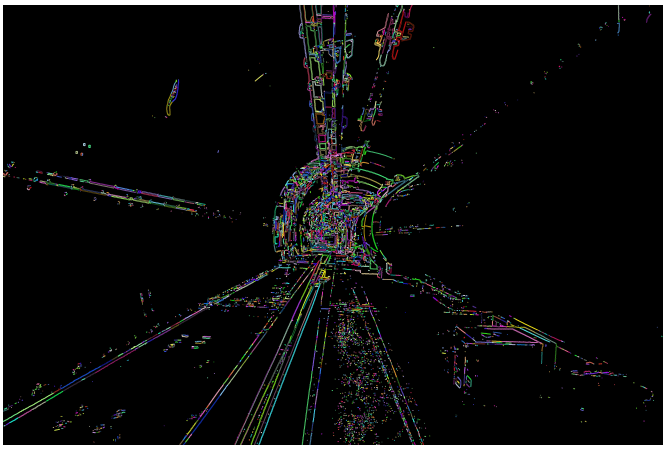

Fig. 6. Segmented Image.

B. Clean Up Noise

In theory, if both images were taken in the exact same place with the same environmental conditions, the image in Fig. 9 would show only real differences and it would be perfectly clear where the deformation is. Nevertheless, *State0* and *State1* were captured under different conditions and therefore it is necessary to correct this situation. Two different approaches were applied as showed below.

1) Position Offset: Observing the images *State0* and *State1* it is possible to note that there is a slightly different between the position were each one was taken. To correct this difference, it is necessary to quantify this offset. After that, it will be possible to transform the perception of one of the images to match it with the other one.

As a first step, *State0* and *State1* are scanned looking for the key factors of each one, all those particular points that would

help in the characterization of the image perspective. Later, each key factors of an image is compare with each one of the other image, if a pair of key factors posses a certain degree of similitude it is a match. At this point, it is convenient to draw a map displaying all the matches found as in Fig. 7.

Fig. 7. Matches between key factors.

With this information, it is easy to correct the perception of *State0* according to *State1* and vice versa. For example, if *State0* is corrected to match with *State1* the result is as showed in Fig. 8. It is observable that the corrected image losses a certain area depending on the degree of the offset, nevertheless in this specific case this area is immaterial.

Fig. 8. Corrected *State0* according *State1*.

Now, retaking all the steps since the subtraction of the images to the contour detection using the corrected *State0*, it is opportune to draw a new segmented image as showed in Fig. 9. This new image contains 877 contours, therefore approximately an 80% of the original noise was corrected by this approach. Despite this is a good result, it is not enough, so in the next section is presented another approach.

2) Common Features: In this approach the image *State0* is used as a feedback in order to reduce the noise. First, it is mandatory to extract the contours of *State0*. Later, each contour of *State0* will be contrasted with each contour of the segmented image. If the similitude between both is less than 0.25 (where 0 is the maximum similitude and 1 is the minimum) and the distance between both is less than 100 pixels, it is a match. Every match will be remove from the segmented image. Note that it is not possible to apply this approach using *State1* since it would erase the deformation.

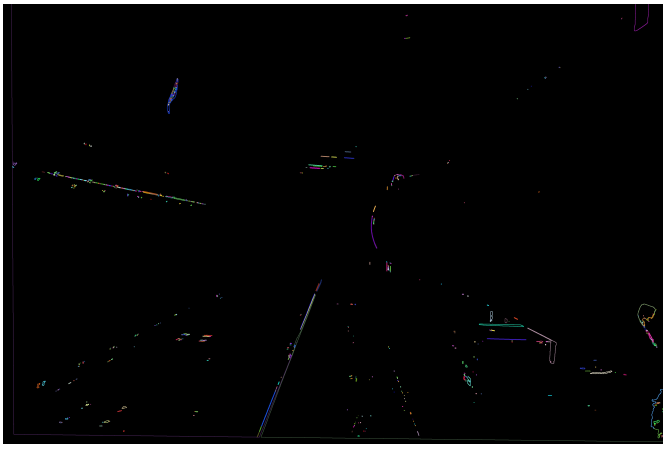

Fig. 9. Segmented image after correction of the position offset.

In Fig. 10 is presented the result of this approach using the images in Fig. 8 and Fig. 2, that is, the resulted image of the combination of both approaches. This image posses 190 contours, therefore approximately 96% of the noise have been removed.

Fig. 10. Segmented image after correction of the position offset and elimination of common features.

C. Final Detection

At this point, the only remaining step is to detect the crack or deformation. On the one hand it is true that the crack or deformation is unknown, being the objective of this project to detect it, and on the other hand it is possible to define a group of characteristics of how the deformation could be, in order to delimit the possibilities. In this case, this characteristics were define as follows:

- *Minimum and Maximum Area*: Deformations will be filtered depending on their area. This area cannot be more than the maximum value neither less than the minimum.
- *Minimum and Maximum Length*: In this case, the same as in the area but with the length of the deformation.
- *Location*: Because of the limited resolution of the camera in use, it is important to limit the scanning area to a

certain part of the image. In this specific case it will be part of the left wall and part of the floor.

- *Ellipse Proportion*: In order to distinguish between straight lines and real cracks or deformations, a filter based on the ellipse proportion is defined. It consists on finding a ellipse that fits the deformation, and with the length and wide information of the ellipse, a proportion of the type $P = \frac{L}{W}$ is defined. Later this proportion has to be in a certain range to be considered as a real deformation. Combining these four filter, the image presented in Fig. 11 is generated. And therefore, the detection is complete as shown in Fig. 12. Finally, in Fig. 13 is presented a simplified block diagram of the algorithm.

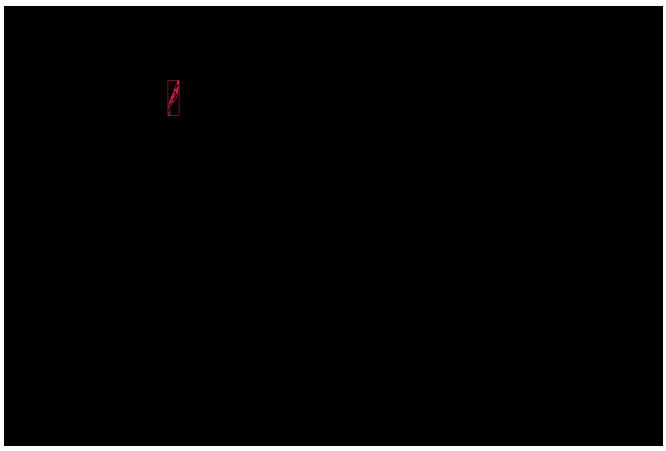

Fig. 11. Image filtered by area, length, localization and ellipse proportion.

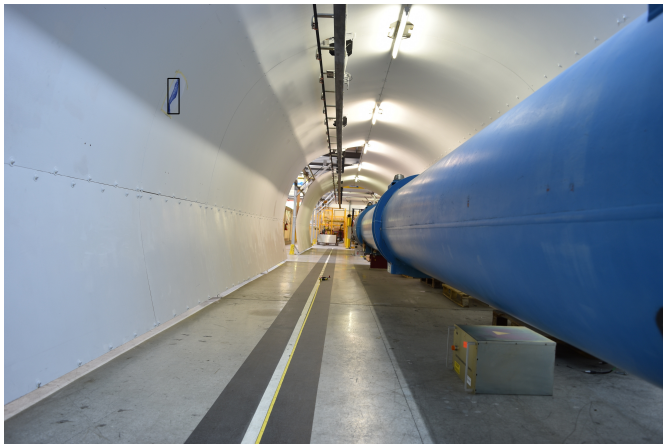

Fig. 12. Final result.

IV. Conclusion

It can be concluded that it is possible to detect different kind of deformation based on a group of basic characteristics that describe the deformation in general bases. Also the it can be observed that the methods of *Position Offset* and *Common Features*, used to reduce noise, are very effective when images cannot be capture in exactly the same position. Later, it would be necessary for future applications, to define a more complete group of characteristics that define the deformation to increase the accuracy and universality of the algorithm. Also, it would be a great addition for the algorithm to be capable of detecting deformation using images taken from different places, this is for example, to compare an image taken one meter of distance from the deformation with other one captured from two meters. To sum up, the development and improvement of this system would be a great contribution for *CERN* and science, saving money, time and human effort and maximizing the use of resources.

References

[1] OpenCV. 2014. *OpenCV 2.4.9.0 documentation*. [ONLINE] Available at: http://docs.opencv.org/. [Accessed 10 August 15].

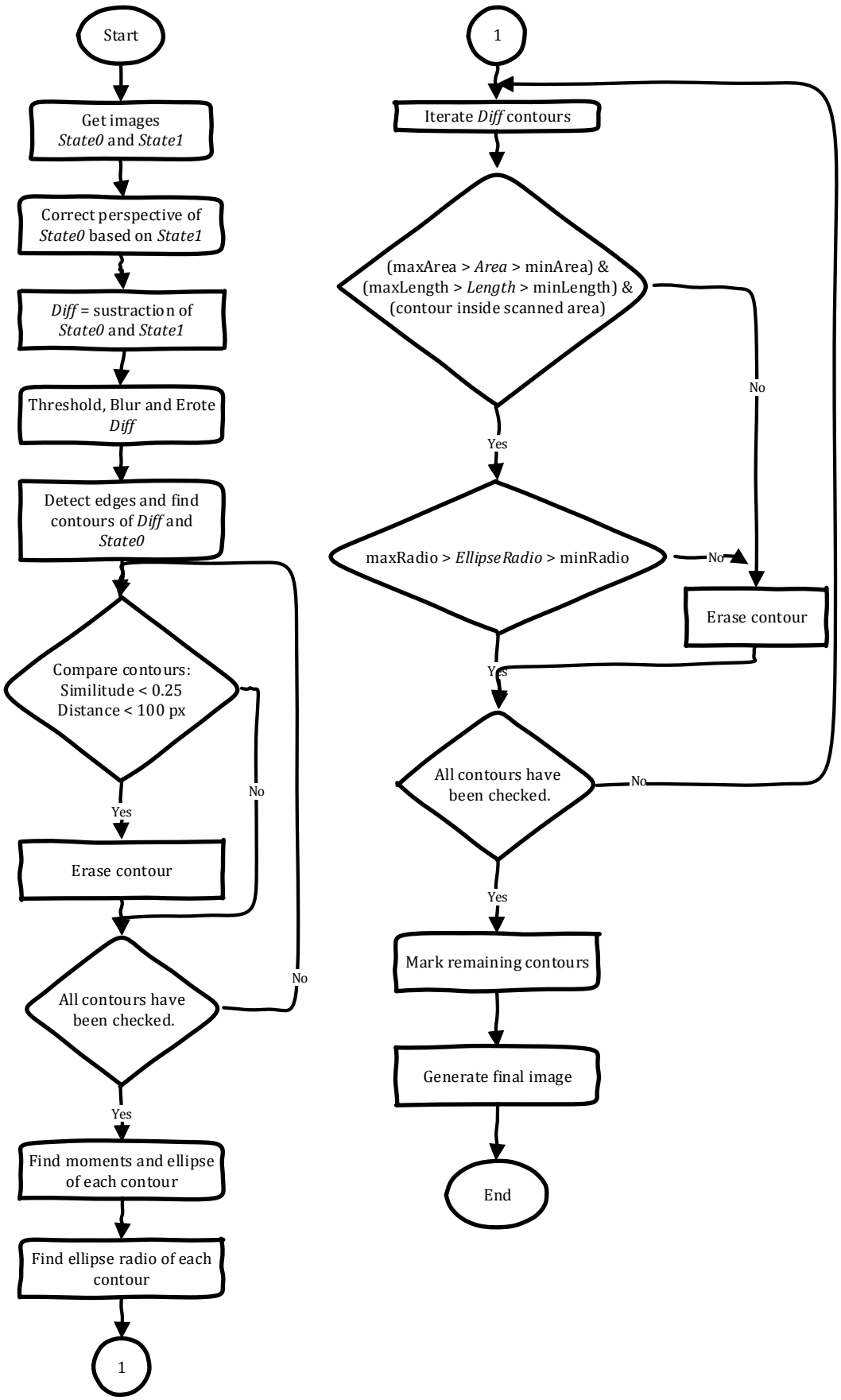

Fig. 13. Block diagram of the presented algorithm.