BACKGROUND RECONSTRUCTION TO REMOVE OBSTACLES FROM IMAGES

Alodie Boissonnet, Paul Jacob
Ecole Polytechnique
alodie.boissonnet@polytechnique.edu
paul.jacob@polytechnique.edu

January 5, 2020

ABSTRACT

This document is a report for our project taking place in the *INF573 : Image Analysis and Computer Vision* course at École Polytechnique. The goal of our project was to implement, research and test different methods in order to automatically remove temporary obstacles from a set of images, taken from the same place, without a human indicating where the obstacles on each picture are.

Several methods will be described in this document : at first, we tried to implement basic methods, that would take, for example, a random pixel, or the median pixel at each location in terms of intensity. But those method can be of little help when the temporary obstacles happen to be very often obstructing on some particular places in the picture.

We tried to bypass this problem by two ways : in the first time, by explicitly detecting obstructions, in order to give them as input to our algorithms and to fill the obstructed places by mixing the different images. In a second time, we implemented a graph-cut algorithm similar to the one presented in the *"Interactive Digital Photomontage"* research article from the *University of Washington* [1].

***K*eywords** Computer Vision · Image Analysis · Background Reconstruction · Obstruction · Graph Cut

1 Introduction

1.1 Presentation of the problem

The purpose of this project is the following : starting from a set of pictures showing the same place, we want to automatically remove the temporary obstacles and to reconstruct the scene without any obstruction. For instance, we want to eliminate people in front of monuments. This work can have useful applications, at the same time for individuals who aim at editing their holiday pictures, and for Computer Vision tools that needs to set the background of a scene.

1.2 Scope of study

We chose to work with images from a fixed camera, in order to avoid realignment issues. In order to find appropriate data sets, we mostly used images that we found on online public webcams. More particularly, the pictures we show in this paper are the results that we got from a skyline webcam of Juliette's house at Verone. In this data set, images are obstructed by people that are sufficiently large. We took images at Juliette's house at two different times : once when it was sunny and very obstructed, with lots of people in the garden. The second time, there was fewer people, but it was raining, and it is visible on the images. This represents a little additional difficulty for keeping a global consistency. We also tested our algorithms on other places such as highways, where we tried to remove cars. One of the difficulties of this data set was that the cars have very different sizes, according to their distance to the camera.

In the data sets we chose, some areas were more often obstructed than not obstructed. The problem was therefore more difficult than just taking the most obvious pixel in each image based on its frequency. We have deliberately chosen such data sets, to be able to test lots of different methods, and in order to develop algorithms that could give satisfying results for as many data sets as possible.

2 Probabilistic methods

2.1 Random method

When removing the foreground from images, it can be difficult to make transitions between images invisible, due to small variations in terms of luminosity, contrast, etc. To tackle this issue, a very simple approach is to determine the value of each pixels of coordinates (i, j) by randomly picking an image from the original data set, and picking its corresponding pixel at those coordinates. Therefore, there is no problems with transitions: as pixels are all mixed, we do not see switches.

The idea behind this method is that, supposing that all regions of the scene have several unobstructed appearances in the original data set, the probability of taking a "good" pixel is high, and the result will be mostly satisfying. However, this very simple method considers pixels from both the foreground and the background, and the foreground is not completely removed. And this effect is important when the pictures are often obstructed.

To strengthen this method, we can detect all obstructions areas (see subsection 3.1), and randomly choose pixels that do not belong to the foreground. However, it is possible that an area is detected as obstructed on all images. If this is the case, we choose to stop after we have tried two times the number of images. We have then a pretty efficient method, in which we do not have foreground pixels and where there is no visible transition.

Figure 1: Random method - with the foreground

Figure 2: Random method - without the foreground

2.2 Median method

The median method relies on the idea that, when an area of the scene is obstructed, the color of pixels at this area's coordinates are more or less random among all pictures. Therefore, if all areas show several occurrences of their unobstructed appearance, we suppose that the color of the background at (i, j) is approximately the median value of pixels of coordinates (i, j) from all images of the data set.

This method is very efficient when all areas are more often not obstructed. When it is not the case, results are still satisfying on the good areas, and the results can therefore be still good overall. When obstructions are redundant, and always of a similar color, we can obtains some errors (for example if someone is always at the same place in most of the pictures).

2.3 Histogram method

In some data sets, some areas are very often obstructed by people. For example, at the first time that we took pictures from the Juliette's house in Verone, some areas were obstructed most of the time. Thus, the median method had trouble

Figure 3: Result of the median method

on those areas, because it was most of the time the wrong pixel, and the median pixel was thus wrong. The histogram method aims at being more robust in those situations. The idea is the following : for each pixel, instead of taking the median value at every pixel location, we draw a histogram of red, green and blue values between 0 and 255 (with a previously defined number of intervals), and we then pick the pixel that is the closest to the maximum intensity in every histogram.

Even if the median pixel is not the good color, there is good hope that the right range of color is still the most present. Here are the results that we obtained in a part of the first data set : the histogram behaves slightly better on the areas of difficulty.

Figure 4: Median method on a difficult data set (lots of obstructions)

Figure 5: Histogram method on a difficult data set (40 intervals)

3 Methods with foreground detection

Another way to solve our initial problem is to detect explicitly the foreground, in order to reconstruct the background afterwards. This approach has the advantage to mix images only when it is necessary, and increases therefore the consistency of images. Different methods can be used to fill the obstruction zones. The first one consists in coping pixels from another image, while the second one copies gradients from another image. However, a special focus must be given to the first step of the algorithm, in order to detect with accuracy all blocked areas, even if they are most often obstructed than non-obstructed.

3.1 Detection of the foreground

To implement this kind of solution, we have to identify the foreground beforehand. To do so, we first used the previous per-pixel median method to determine if some pixels are too far from a median value. Indeed, we look at pixels of coordinates (i, j) of all images, and if the difference between the norm of a pixel and the median value is greater than a pre-defined threshold, we consider it as an outlier. We thus scan all pixels to determine areas of obstruction of images.

However, results were not fully satisfying. When a region was more often obstructed than not, it was not detected as such by our algorithm. Thus, this method required very binding conditions on images to obtain good results. To solve this issue, we implemented another method in which we compute the distance of pixels to their two nearest neighbours. If this distance is too large, we consider that the pixel is part of the foreground.

This approach is very efficient because it requires an area not to be obstructed on at least three images, independently of the total number of elements in the input data set. Indeed, when the area is obstructed, foreground pixels are very unlikely to be exactly similar. On the contrary, the background does not change, or very slightly, among images.

To finish, to avoid visible transitions when switching images from which pixels are computed, we define transition areas around obstruction areas. These areas are treated in the same way as obstructed areas, but they allow us to have a smoother result.

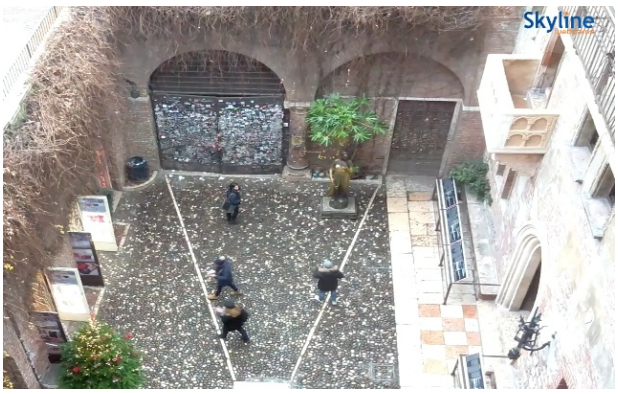

Figure 6: Image of study

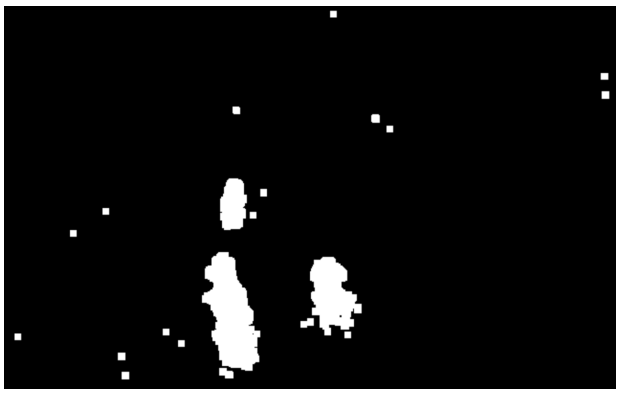

Figure 7: Obstruction areas

3.2 Copy method

The copy method relies on copying pixel from images where the area of interest is not obstructed. For this purpose, we have to identify the foreground, and we then fill the blocked areas. We start with the first image of our data set, and when we encounter an obstructed area, we scan the other images to retrieve the information we are missing. In this method, we copy pixels from another image, where the zone is not obstructed. This works pretty well when images are perfectly similar outside of obstacles.

However, when the global luminosity of images varies, switches between images are visible. To tackle this issue, we equalize images' intensity with an harmonization algorithm (see subsection 5.1). The copy method is then very powerful, providing that we are able to detect correctly all obstructions.

3.3 Gradient Method

Another solution to fill blocked areas is to copy gradients of another image, to recompute missing pixels. This method could be appropriate to smooth all switches between images. To do so, we use the same approach as for the copy method, but instead of copying pixels, we copy gradients.

We decided to compute gradients with the method by difference (1). As we scan our images pixels per pixels, we only know the value of the background for the pixels we have already studied, so we can only use these pixels to compute the value of the next one. And this is for this reason we choose the method by difference. The Sobel or Scharr operator requires to have information about all pixels around the one on study, but this is impossible for us.

Figure 8: Result of the copy method

$$G(x,y) = \frac{(I(x,y) - I(x-1,y)) + (I(x,y) - I(x,y-1))}{2} \tag{1}$$

The gradient method has good results, providing that we are able to detect correctly all obstructions, once again. However, a special attention must be paid to calculation errors. As we compute a pixel thanks to the previous one, calculation errors can be spread and worsen. It is therefore not appropriate to use this method when the obstruction area is too large. It can indeeed return a poor result.

Figure 9: Result of the gradient method

4 Graph-cut method

In order to take advantage of the global structure of areas in the pictures, we implemented a method based on graph-cut optimization. This method is presented in detail in the research article called *"Interactive Digital Photomontage"*, from the *University of Washington* [1].

The idea behind this method is to enhance the "pixel-wise" methods that could be used previously (such as the histogram, median, or random method), and to add some constraints of regularity that blocks too much mixes of different images.

We first define a target image for every pixel (such as, for instance, the image obtained by the previous methods). Then, we initialize a temporary resulting image as the first image of the data set, and by iterating on every image of the data set, we apply a graph cut algorithm on this temporary resulting image and the new image of the data set.

To that end, we define the energy E of a cut, and on this energy E, we apply a penalty on every pixel if it is too far from the target image (in terms of color for instance). To take advantage of the surrounding structure of every pixel, we also apply a penalty on neighbour pixels if they come from different images, depending on their distance in terms of color and gradients.

This approach gave us some slightly better results than the previous pixel-wise methods : however, it was not as significant as we hoped. A good point is that it helped in improving the overall consistency of the image, by less cutting people in pieces for example.

On the same difficult data set than in the subsection 2.3, here are the results that we obtained in comparison to the median, random and histogram methods.

Figure 10: Median method on a difficult data set (lots of obstructions)

Figure 11: Histogram method on a difficult data set (40 intervals)

Figure 12: Random method on a difficult data set

Figure 13: Graph-cut method on a difficult data set (optimized parameters)

5 Further improvements

5.1 Images intensity harmonization

For all methods, one of the big difficulties for a good result lies in smoothing image transitions, so as not to see them. Even if methods are pretty efficient by themselves, it is possible to improve results by equalizing the global intensity of images, outside of all obstruction areas. For example, this is beneficial for the copy and graph-cut methods not to see image switches, or for the median method to compare similar pixels. To do so, we compute the mean of the intensity of images outside of all obstruction areas, and we multiply images with an appropriate factor to obtain the same global intensity. This solution is very efficient to improve results of all the previous methods.

5.2 Sorting images

To improve results for the methods with foreground detection (see section 3) and the graph cuts method, it is also interesting to sort images according to the number of obstructed pixels they have. Indeed, we usually start with an initial image, from which we remove the foreground. The difficulty is mostly not to see transitions. Therefore the fewer obstruction areas we have, the fewer transitions we can see.

Conclusion

To conclude, obtaining an obstacle-free image from a set of images has proven to be a very interesting problem. It was indeed much more diverse than it might seem at first glance: the obstacles could be of different sizes, of different natures, they could be very frequent or not depending on the pictures, etc.

Also, with great variations in brightness and ambient contrast between the different images, a difficulty lays in keeping an overall consistency.

In order to overcome this diversity on the data sets that we have chosen, we have implemented different methods that are particularly effective on different cases. The basic median method will be sufficient on a data set with few obstructions (see subsection 2.2), but it becomes less robust when certain regions of the image often present obstacles. The histogram method can then be more robust in some cases (see subsection 2.3).

Explicit detection of obstructed regions (see section 3) has proven to be very effective on very obstructed data sets, provided that each region is correctly visible in at least three images. The copy method or the gradient propagation method is then applied, and assuming that the obstructions are found accurately, the results are very satisfying.

Finally, the graph-cut method (see section 4), which is an improvement of the pixel-per-pixel methods but that takes into account the surrounding pixel structure, allows a slight improvement on the data sets we have chosen, provided that we make a good choice of parameters in the graph weights. It also helps to keep a good overall consistency, because the similarity between neighboring pixels is taken into account. Also, since this method has a very large range of applications in photomontage, we could take advantage of this implementation for various purposes. For a cleaner rendering at the seams, a possible improvement would be to implement the gradient-domain fusion presented in the article.

References

[1] Assem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. Interactive digital photomontage. *ACM SIGGRAPH '04*.

6 Appendices

Another data set

We tested our different methods on different data sets to confirm what we could have found. Here are the results on another data set with completely different types of obstructions : cars of different sizes and at different distances from the camera.

Figure 14: Random method

Figure 15: Median method

Figure 16: Histogram method (5 intervals)

Figure 17: Histogram method (10 intervals)

Figure 18: Histogram method (20 intervals)

Figure 19: Histogram method (40 intervals)

Figure 20: Copy method

Figure 21: Gradient by difference method

Figure 22: Graph cuts gradient method