Removal of Unwanted Objects from Still Photographs

Rosana Bălănescu
Department of Computer Science
Babeş-Bolyai University
Cluj-Napoca, Romania
brie1861@scs.ubbcluj.ro

Adrian Sterca
Department of Computer Science
Babeş-Bolyai University
Cluj-Napoca, Romania
forest@cs.ubbcluj.ro

Ioan Bădărînză
Department of Computer Science
Babeş-Bolyai University
Cluj-Napoca, Romania
ionutb@cs.ubbcluj.ro

***Abstract*—In this paper we present a method for removing unwanted objects from still photographs. The method is based on taking several shoots of the same picture by either moving the camera or holding the camera fixed if the unwanted object is moving, so that afterwards, by careful comparisons between macroblocks from different shoots of the same picture, we are able to reconstruct the original picture without the unwanted object in it. The method operates in two scenarios, one when the unwanted object is moving in front of the camera (e.g. like a passing car) and another when the unwanted object is fixed (e.g. like electrical or communication wires). In the evaluation section we perform several tests that show our method is viable.**

***Index Terms*—image reconstruction, image object removal**

I. Introduction

We all know that moment when we are visiting a touristic site and we want to take a photo of a building with a beautiful architecture or of a monument, but unfortunately there are a lot of street cables in front of it. Or, maybe we are with a group of tourists, the guide tells us to hurry up and we are trying to snap a photo of some interesting view that we see, but a car passes between the camera and the object of our photo exactly in that moment. We will present an algorithm which takes as input a short video consisting of consecutive shoots of the same picture and from this video, we will re-construct the initial image without the intruder object.

There are two possible scenarios we consider for our method. The first one is when the unwanted object is in movement and the camera is (more or less) static. As more concrete examples, think about cars, birds, or people that accidentally enter in the field of view of the camera. In this approach, we need to analyze which macroblocks of pixels remain unchanged (or with a small relative error - the threshold for this error will be determined experimentally) from one frame to the next one, and keep them as they are in the resulted image. When this error is large, we try to find another pair of frames that satisfy the condition.

The other scenario is when the camera is moving and the unwanted object is static (electric cables, pillars, etc). In this case, we use the relative movement of the object with respect to the camera. The image in the background also appears to be moving with respect to the camera, but the displacement is way smaller in comparison with the displacement of the object which is closer to the camera. The processing of the images is similar as in the previous case, but we need to take into consideration the 3-D translation of the camera when trying to identify correspondent blocks of pixels from two frames. However, this method has its limitations, since it can be applied only in the situations when there is a certain ratio between the distances (camera - occluding object, camera - background) and only when the movement of the camera does not happen on parallel planes with the cables' plane.

The contribution of this paper is two algorithms of removing unwanted objects from a sequence of continues images by combining and searching missing information in neighboring frames. The first algorithm relies on the fact that the video camera is static and the intruder object is moving in the field of view and the second algorithm considers the opposite case, when the intruder object's position is fixed, but the video camera is moving.

The rest of the paper is organized as follows. In Section II we present related work. Following, Section III presents the main contribution of our paper, namely two algorithms for removing unwanted objects from photographs based on continuous shoots of the same picture. Section IV presents our evaluation experiments and the paper ends with conclusions in Section V.

II. Related work

Image recognition has been a very interesting topic for many years now and it attracted a lot of attention [1], [2], [3]. The most challenging visual task that we could ask for a computer to perform is the recognition of all the objects in an image. It is hard because the real world is a blend of objects that are obstructing each other, interfering with one another, appearing in different postures. Moreover, objects of the same class (e.g. cats) are widely varying depending on their race, so differences of color and shape are major, making it very improbable for a computer to perform a perfect and exhaustive match with some database samples [4]. In [5], the authors present the algorithm which solves the problem of filling large gaps in an image with a content that is plausible for the human eye. Previously, there were two types of algorithms that were solving this: texture synthesis algorithms [6] and inpainting techniques [7]. The first one is used for generating large regions with the same texture as the one in the unhidden area, while the second one

is used for filling small gaps, focusing on linear structures. The method presented in this paper combines the advantages of both types, by replicating simultaneously the texture and the structure. A key element in this process is the way in which the pixels are synthesized: they get a temporary color value together with a confidence value, which are both updated as a new patch of the image is reconstructed. The filling order is an important factor in achieving a successful result, thus, at every step, the algorithm computes the priority of every patch that has to be reconstructed. In comparison with other algorithms, this one produces better and clearer images. Since the reconstructed area did not suffer any diffusion to propagate the color values, as the classical inpainting algorithms were doing, the resulted pictures are not blurred at all. Another interesting approach of an algorithm for objects removal can be found in [8]. This particular algorithm, does not only complete the scene using just one image, but it also produces a consistent background content even in the cases of large foreground objects. This is done by inferring a possible structure of the hidden region in the image with the help of curve estimation, which means that pixels in the neighborhood of the "gap" are analyzed, looking more carefully at edges and trying to generate curves that could either traverse the gap, being continued on the other side of it, or that could end inside it. Then, an orientated patch matching algorithm is used to fill the gap, making a texture synthesis around the already estimated lines. Another important work in this domain is the one described in [9], image de-fencing. Digital photographers are using this technique to erase fence-like patterns from an image, without leaving any hint that a fence object ever existed. This would be similar to what we are seeing from a car with a high speed when passing by a fence. Moreover, image de-fencing is quite similar to one of the goals of the present paper, which is image "de-cabling". The task is not an easy one, due to factors like the complexity of the background and the large variety of fences. Based on a data-driven approach, it detects the fence pattern with the help of a feature descriptor: a histogram of oriented gradients (HOG), a technique which counts occurrences of gradient orientation situated in small areas of an image. However, the authors of this method modified this HOG to represent every particular pixel instead of a full patch. This seems a very good idea, since the evaluation of this algorithm shows that it detects a fence in an image with an accuracy greater than 98%. In comparison with this work, our method does not remove only small, line shaped, objects (like fences, cables) from an image, but also large objects like cars or persons.

In recent years there are a lot of papers trying to do image inpainting using convolutional neural networks [10]–[12]. Various CNNs and GANs (i.e. Generative Adversarial Networks) are used in order to infer missing pieces from a painting and generating a substitute from the surrounding pixels. Our method differs from these works based on neural networks because our method does not require any training (i.e. it is efficient) and it does not extract missing information from surrounding pixel patches in the same image, but instead it extracts missing pixel information from surrounding temporal frames in a video stream.

III. Algorithms

Unlike the methods presented in the previous section, we reconstruct the scene with the help of multiple frames extracted from a video. The process of shooting this small video does not need to be more elaborated than the process of shooting the initial image itself. For this reason, there is no need to guess or infer the color value of a pixel based on the information that we have from the rest of the picture. Instead of this, we try to find the information about the hidden area (the area behind the occluding object which has to be removed) in the other frames. The only issue that remains to be solved is how to make the correlation between the pixels in the hidden area of an image and the same "point" from the real world, which is visible in other images.

There are two phases that comprise our method:

- matching blocks of pixels from several frames and eliminating blocks that contain the intrusive object
- blending the matched blocks of pixels into the final image

A. Scenario 1

We begin the presentation of our method with the case when the photo camera is static and the object that has to be removed is moving. We assume each frame contains at least a small part of that object. After the frames are extracted from the video, the algorithm proceeds to compose the result image, which will be called "clean image", from now on. The algorithm is depicted in listing 1. We start with an empty clean image, CI. The reconstruction process is performed in multiple steps, one macroblock of pixels at a time (a macroblock is just a rectangular part of the image pixels). The basic idea of the algorithm is to compare corresponding macroblocks of pixels from consecutive frames and determine the pair of two macroblocks that are most similar - there is a high probability that these macroblocks do not contain the moving object that needs to be removed. Since the video camera does not move between successive frames in scenario 1, it is straightforward to find the macroblock from the next frame that corresponds to the current macroblock from the current frame (i.e. both macroblocks should start at the same (X, Y) coordinates). By searching for frames in which the macroblock looks very different than in the other frames, we can recognize the intrusive object (i.e. the object in movement). In this way, we can discard these macroblocks from those frames since they do not contain the whole information about the fixed background.

It is worth mentioning that our algorithm considers that the frames are in the YUV colorspace, and we only use the luminance pixel components (i.e. Y) in our computations (except the color blending part - line 14 of the algorithm). Lines 1-3 from the algorithm decomposes all the l frames into macroblocks which are saved in the MB_i sets, for $i \in [1, l]$; MB_i holds all the macroblocks from frame I_i. Following, in lines 4-16, for each macroblock of index/rank k, we find the pair of consecutive frames (I_i, I_{i+1}) for which the k-th

macroblocks are most similar. We compute this macroblock similarity using the MSE (Mean Square Error) formula [13]. The MSE of two macroblocks, $MB_i[k]$ and $MB_{i+1}[k]$, is computed in line 8. $MB_i[k]$ is the k-th macroblock from frame I_i and $MB_{i+1}[k]$ is the k-th macroblock from frame I_{i+1}. The minimum MSE score of the k-th macroblocks from two consecutive frames is computed in lines 7-13 and is saved in the min_MSE variable. At the same time min_idx stored the index of the frames that contain these minimum MSE macroblocks (actually, min_idx is the index of the first frame in the pair, the index of the second frame in the pair will be $min\_idx + 1$).

Next, after the minimum MSE pair for the k-th macroblock has been found across all l images, in line 14 we blend the pixel colors of this minimum MSE macroblock pair. Every pixel from macroblock M_k in line 14 will be obtained as the mean pixel of the corresponding pixels from the two macroblocks, $MB_{min\_idx}[k]$ and $MB_{min\_idx+1}[k]$. To determine a mean pixel, each RGB component is calculated as the arithmetic mean between the same component of P_1 and P_2 (P_1 being the pixel on the same position from the first macroblock and similarly P_2 as the pixel from the second macroblock). At an intermediary point in the execution of the program, the clean image will look like the one from Fig. 1 where P_1, P_6 are pixels from the macroblock $MB_{min\_idx}[k]$ and Q_1, Q_6 are the corresponding (i.e. situated on the same positions) pixels from the macroblock $MB_{min\_idx+1}[k]$.

Finally, in line 15, the k-th color blended macroblock, M_k which should not contain moving pixels, is added to the clean image, CI.

Algorithm 1 The object removal algorithm for static camera scenario

Input:
I_i: the i-th frame; $i \in [1, l]$
I_i has m x n pixels having only Y components in the YUV colorspace
Output:
CI: the clean image reconstructed (without unwanted objects)

The Unwanted Object Removal algorithm is:

1: **for** $i = 1$ **to** l **do**
2: $MB_i = GenerateMacroblocks(I_i)$
3: **end for**
4: **for** $k = 1$ **to** $size(MB_1)$ **do**
5: $min\_MSE = \infty$
6: $min\_idx = 1$
7: **for** $i = 1$ **to** $l - 1$ **do**
8: $mseval = MSE(MB_i[k], MB_{i+1}[k])$
9: **if** $mseval < min\_MSE$ **then**
10: $min\_MSE = mseval$
11: $min\_idx = i$
12: **end if**
13: **end for**
14: $M_k = BlendColors(MB_{min\_idx}[k], MB_{min\_idx+1}[k])$
15: $AddMacroblockToCleanImage(M_k, CI)$
16: **end for**

The steps described above will be applied for every macroblock, thus obtaining the whole image, the background, as a combination of smaller local backgrounds. The choice of the size of the macroblock is very important, since it can influence decisively the quality and the accuracy of the clean image. In this paper, the optimal size of the macroblock is determined experimentally.

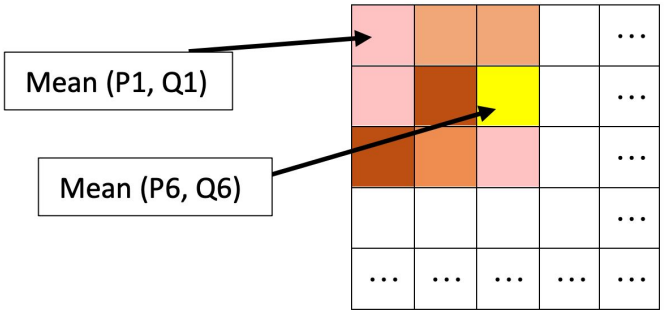

Fig. 1. Clean Image (in progress)

B. Scenario 2

The second scenario adds more complications to the method. The pixels that correspond to the same "point" of the reality are not anymore on the same positions in all the frames. It is not feasible and not even accurate to compare every pair of points from two images to find the matches between them. We extract ORB keypoints (Oriented FAST and Rotated BRIEF) [14] [15] from both frames and then find matching keypoints in both frames using the Euclidean distance. After we have found a set of matching ORB keypoints, we compute the homography matrix between the two frames (i.e. the transformation between two planes). Having this information, it is possible to calculate the new coordinates of a translated and rotated block of pixels. Basically, this method is just a generalization of the first one, in which we consider as movement the cables and the pillars that "do not move in the same rhythm" as the building (relative to the camera). The process of reconstruction begins with choosing a reference image RI, the one that will be cleaned. The best choice for this is the frame in the middle, since we have the highest chances of finding its macroblocks in the other frames. Then we compute the homography matrix between it and every other frame. We will note with H_i the homography matrix needed to transform the plane of the RI image into the plane of the i-th image (I_i). Thus, we will obtain a list of homographies, $[H_1, ..., H_n]$, where n is the number of frames in the video. As in the first scenario, now the reference image is split in macroblocks of pixels, and for each of them we apply the following operations:

1) Apply a perspective transformation with the matrix H_i on the coordinates of that macroblock to find the new coordinates of the transposed macroblock of pixels in the image I_i (for every i from 1 to n).
2) If, for an image, the transposed macroblock is not entirely included in that image (its coordinates are less than 0 or greater than the width or height of the image,

respectively), it means that this image does not contain the entire information about that macroblock, therefore it can be discarded (in practice, when transforming the coordinates of the macroblock, if there exist pixels that are out of the boundaries of the accepted range only with some units, that image is not discarded and we use the color value of the closest pixel instead).

3) From the remaining images, we find the pair with the minimum MSE between their transposed macroblocks, M_1 and M_2. In the ideal case, the minimum MSE is obtained from two images that do not contain the occluding object in that macroblock.
4) For each pixel from the macroblock of the RI image, its new color value is computed as the mean between the values of its corresponding pixels in M_1 and M_2.

In the end, the RI image is reconstructed with small parts from the other images. However, it is possible that for some macroblocks, the algorithm discards all the other images at step 2. Then, for those blocks, the algorithm will keep the values of the original pixels (the one from the RI image). This is a limitation of the algorithm, which does not remove any bits of objects that exist in those areas, since it can not find any information of what is behind them in the other images.

The detailed algorithm used for scenario 2 is depicted in listing 2. It is similar to the first algorithm, but now before we start comparing macroblocks, in lines 1-7 we translate each frame $I_i, i \in [1, l]$ to real world coordinates relative to the reference image, RI. We first extract ORB keypoints from the reference frame, RI, then we extract ORB keypoints from the remaining frames. We match keypoints for each pair (RI, I_i) and compute the homography matrix H_i for translating the coordinates of image I_i to the coordinates of RI. Then we apply the homography matrix H_i to the frame I_i. The rest of the algorithm (i.e. lines 8-28) is very similar to the first algorithm, with (min_MB1, min_MB2) being the minimum MSE macroblock pair. $FindCorrespMacroblock(MB_i[k], I_{i+1})$ returns the macroblock from frame I_{i+1} that corresponds to (i.e. is placed at approximately the same real-world coordinates as) macroblock $MB_i[k]$.

IV. Experiments

In this section we present some of the experiments we have performed in order to validate our method. Unfortunately it is very hard to evaluate quantitatively an algorithm that performs artistic object removal in a picture [10]–[12]. This is because the object to be removed can be replaced with many artistic textured pixels, each having its own value in the eye of the viewer. Furthermore, for our specific technique is is even harder to assess it quantitatively because we could not find any available dataset with short video streams of continuous shoots of the same photograph and also to include in it the ground truth picture (i.e. the picture without the intruder object). Instead we performed a qualitative evaluation by showing the visual result of our algorithms.

Algorithm 2 The object removal algorithm for camera displacement scenario

Input:
$I_1, ..., I_l$; the set of all frames of the video stream not including RI
RI: the reference image (i.e. the frame from the middle of the video stream)
Output:
CI: the clean image reconstructed (without unwanted objects)

The Unwanted Object Removal algorithm is:

1: $F_{RI} = ExtractORBKeypoints(RI)$
2: **for** $i = 1$ **to** l **do**
3: $F_i = ExtractORBKeypoints(I_i)$
4: $MatchSet = MatchKeypoints(F_{RI}, F_i)$
5: $H_i = ComputeHomography(MatchSet)$
6: $I_i = I_i \cdot H_i$
7: **end for**
8: **for** $i = 1$ **to** l **do**
9: $MB_i = GenerateMacroblocks(I_i)$
10: **end for**
11: $MB_{RF} = GenerateMacroblocks(RI)$
12: **for** $k = 1$ **to** $size(MB_{RF})$ **do**
13: $min\_MSE = \infty$
14: $min\_MB1 = null$
15: $min\_MB2 = null$
16: **for** $i = 1$ **to** $l - 1$ **do**
17: $mb_1 = MB_i[k]$
18: $mb_2 = FindCorrespMacroblock(MB_i[k], I_{i+1})$
19: $mseval = MSE(mb_1, mb_2)$
20: **if** $mseval < min\_MSE$ **then**
21: $min\_MSE = mseval$
22: $min\_MB1 = mb_1$
23: $min\_MB2 = mb_2$
24: **end if**
25: **end for**
26: $M_k = BlendColors(min\_MB1, min\_MB2)$
27: $AddMacroblockToCleanImage(M_k, CI)$
28: **end for**

A. Scenario 1 (Still camera and moving object)

Case 1 Moving object behind still object

First, we tried to assess our method in the first scenario. We took a shoot of a video of approximately 4 seconds containing images of the same physical coordinates, but a person is moving in front of the camera (behind the object of interest which is another person, in this case). The parameters of our algorithm were:

- Number of frames: 2 frame/sec => 9 frames;
- Macroblock size: 32x32 pixels; 64x64 pixels; 128x128 pixels;
- Comparison step: 3.

Some of the frames extracted by the algorithm can be seen in Fig. 2. First we tried with less then 9 frames per video, but the results were unsatisfactory. Then we took approximately 4 seconds of video, 2 frames per second, hence the algorithm extracted 9 frames, which seems to be enough for this case, since it managed to recreate a completely clean image. The clean image can be seen in Fig. 3. In fact, it worked very good even with very large block sizes (such as 128). Therefore,

Fig. 2. Frames used for Case 1 (moving object behind still object)

Fig. 3. The reconstructed, clean image for Case 1

increasing the number of frames is a better way of refinement that decreasing the size of the block.

Case 2 Moving object is in front of still object

Next, we applied our method to the same scenario, but this time the intrusive object moves in front of the still object (i.e. a person). The video shoot has approximately 3 seconds, 2 frames per second. The parameters of our algorithm were:

- Number of frames: 2 frame/sec => 7 frames;
- Macroblock size: 32x32 pixels ; 16x16 pixels ;
- Comparison step: 3.

The frames extracted by the algorithm can be seen in Fig. 4. As seen in Fig. 5, the person in orange is removed completely. This result was obtained taking the block size of 16x16. In another try, with a block of 32x32, the result was very similar. Thus, without much effort, in just 3 seconds of staying still we obtained a beautiful touristic picture.

B. Scenario 2 (Moving camera and still object)

In the second scenario, we tried to apply our method in order to remove cables in front of a building. In our first attempt, the input video has approximately 4 seconds, depicting a building with street cables in front of it. The frames used are depicted in Fig. 6 and the parameters of our algorithm were:

- Number of frames: 2 frame/sec => 9 frames;
- Macroblock size: 8x8 pixels ;
- Comparison step: 8 (so the comparison was an exhaustive one)

In Fig. 7 we can see that the street cables were removed in proportion of 95% (we can still see parts of them in the

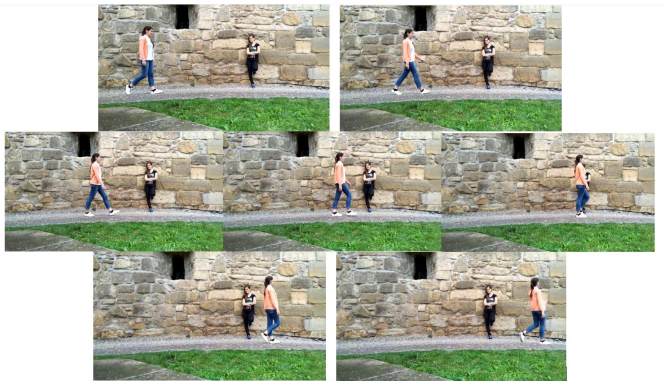

Fig. 4. Frames used for Case 2 (moving object in front of still object)

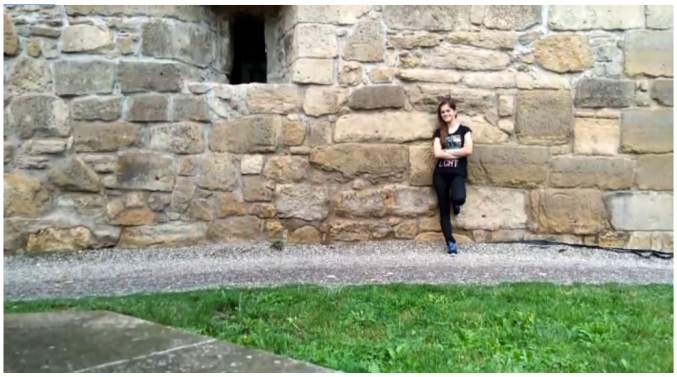

Fig. 5. The reconstructed, clean image for Case 2

low-left corner). The building seems to be well reconstructed, with just some small deviations noticeable at the windows. However, the left-most part, the sky and the upper edge look quite pixelated. This is a sign that the choice of the block size was too small.

In a second attempt we have used the same video stream as previously, approximately 4 seconds long, depicting a building with street cables in front of it. The parameters of our algorithm were this time the following:

- Number of frames: 2 frame/sec => 9 frames;

Fig. 6. Frames used for the 2nd scenario

Fig. 7. The reconstructed, clean image for the 2nd scenario; first attempt

Fig. 8. The reconstructed, clean image for the 2nd scenario; second attempt

- Macroblock size: 32x32 pixels;
- Comparison step: 8.

The percentage of cables removed dropped to around 90%. After testing it with different scenarios, we can draw the conclusion that it manages to provide a successful outcome in 95% of the situations that fall under the first case (static camera and moving object), which is quite satisfactory. Meanwhile, for the second case (static object and moving camera), many constraints should be imposed on the input video in order to obtain a good result, and even then, the image does not look completely natural. Concretely, the conducted experiments showed that when street cables are parallel to the movement of the camera, it is impossible for the algorithm to remove them. Moreover, for the algorithm to be able to detect the cables, the distance between the camera and the cables should be significantly smaller than the distance between the camera and the building/monument/view in the background. Another weak point of the algorithm is that, even when it manages to completely eliminate the occluding objects, the view behind them contains some misplaced blocks of pixels (parts of the straight lines appear deviated).

V. Conclusions

In this paper we detailed a mechanism of correcting a still photograph by removing unwanted objects from it like passing cars or persons or electrical/networking cables. Such a mechanism is very useful in tourism when visiting foreign places and taking photo shoots of various monuments. Our method operates in two scenarios. In the first scenario, the unwanted object is moving in front of the camera, like a moving car or a passing person. In the second scenario, the unwanted object is fixed like electrical cables. In both scenarios, our method relies on taking several photo shoots of the same picture, dividing all pictures into macroblocks and matching corresponding macroblocks from different pictures and blending them together in a single, corrected, photo. We performed some specific experiments, applying our method to real-life photographs and we showed that our method is viable. As future work, we plan to apply our method on more real-life photographs, especially in different illuminating conditions and also to see if our method works on commercial movies. Also, we want to update our algorithms so that they take advantage of recent parallel processing techniques [16].

References

[1] P. Arbelaez, "Boundary extraction in natural images using ultrametric contour maps," in *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, 2006, pp. 182–182.

[2] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001, pp. 341–346.

[3] Y. Li, Y. Wang, and Y. Piao, "Extraction of thin occlusions from digital images," in *Selected Papers of the Chinese Society for Optical Engineering Conferences held October and November 2016*, vol. 10255. International Society for Optics and Photonics, 2017, p. 1025540.

[4] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[5] A. Criminisi, P. Perez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Transactions on Image Processing*, vol. 13, no. 9, pp. 1200–1212, 2004.

[6] H. Igehy and L. Pereira, "Image replacement through texture synthesis," in *Proceedings of International Conference on Image Processing*, vol. 3, 1997, pp. 186–189 vol.3.

[7] S. Ravi, P. Pasupathi, S. Muthukumar., and N. Krishnan, "Image inpainting techniques - a survey and analysis," in *2013 9th International Conference on Innovations in Information Technology (IIT)*, 2013, pp. 36–41.

[8] J. Yang, K. Hua, Y. Wang, W. Wang, H. Wang, and J. Shen, "Automatic objects removal for scene completion," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2014, pp. 553–558.

[9] M. Khalid, M. M. Yousaf, K. Murtaza, and S. M. Sarwar, "Image defencing using histograms of oriented gradients," *Signal, Image and Video Processing*, vol. 12, no. 6, pp. 1173–1180, 2018.

[10] Y. Jiahui, L. Zhe, Y. Jimei, S. Xiaohui, L. Xin, and S. H. Thomas, "Generative image inpainting with contextual attention," in *2018 Conference on Computer Vision and Pattern Recognition*, 2018.

[11] A. Y. Raymond, C. Chen, Y. L. Teck, G. S. Alexander, H.-J. Mark, and N. D. Minh, "Semantic image inpainting with deep generative models," in *2017 Conference on Computer Vision and Pattern Recognition*, 2017.

[12] S. Yuhang, Y. Chao, L. Zhe, L. Xiaofeng, H. Qin, L. Hao, and K. C.-C. Jay, "Contextual-based image inpainting: Infer, match, and translate," in *2018 European Conference on Computer Vision*, 2018.

[13] Zhou Wang and A. C. Bovik, "A universal image quality index," *IEEE Signal Processing Letters*, vol. 9, no. 3, pp. 81–84, 2002.

[14] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," vol. 2, 11 2005, pp. 1508 – 1515 Vol. 2.

[15] "Opencv documentation - orb (oriented fast and rotated brief)," accessed: 09-June-2020. [Online]. Available: ttps://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_orb/py_orb.html?highlight=orb

[16] N. Virginia, B. Darius, and S. Adrian, "Mpi scaling up for powerlist based parallel programs," in *Euromicro International Conference on Parallel, Distributed and Network-based Processing*, 2019.