ISSN No:-2456-2165

Image Inpainting with Global Optimization using Parallel Computing

Berkan Höke
Huawei Turkey R&D Center
Istanbul, Turkey

Abstract:- A framework for image inpainting is presented in this work. Inpainting is performed using discrete global optimization problem with an objective function rather than existing techniques. An iterative approach relies on extended belief propagation, called Priority BP has been used to solve the optimization. Although Priority-BP provides serious speedup by reducing complexity, inpainting an image is very intensive task. Hence, this study aims to provide further speedup improvements by using the power of GPUs through Compute Unified Device Architecture (CUDA). The effect of parallel processing framework is demonstrated on the real images for inpainting tasks.

Keywords:- *Image Inpainting, Image Restoration, GPU, CUDA.*

I. INTRODUCTION

Image inpainting [1], [2] is the process of restoring missing data in a specific region of an image or video using the information of known pixels. There are many applications including removing specific objects from a scene and re-touching deteriorated photographs. The main goal is to reconstruct the image in which the inpainted region is filled with values with the surrounding texture in a way that is not visible to human eye. In this work, we propose an extension to the exemplar - based global optimization approach for digital inpainting which attempts to extends the method proposed by [3] with parallel capabilities of graphical processing unit(GPU). Exemplar-based approaches outperformed statistical-based [4] or PDE-based methods [1] up to now. Simplest approach is filling the missing data simply by interpolating known pixels of the image to reconstruct the unknown ones [5], [6]. However, these approaches suffers from visual inconsistencies. Global approaches have been proposed to overcome this problem by using an EM-like scheme in order to optimize the completion process [7], [8]. Nevertheless, EM relies on initialization and can converge to local minima. Some methods require user interaction. For example, Sun et al. [9] needs user input to indicate the curves on which the most remarkable missing structures reside, similarly Drori et al. [10] utilize points of interest. On the other hand, [11] relies on segmentation, but segmentation on natural images is an extremely difficult task for every object.

Finally, exemplar-based methods also place emphasis on the order by which the image synthesis proceeds, usually using a confidence map for this purpose [10], [12]. However, these two techniques have significant disadvantage. First, the confidence map computation is not a generalized case. Secondly, the value of reconstructed pixel is not updated once a known value has been assigned which may lead to visual inconsistencies. Komodakis and Tziritas proposed [3] exemplar-based approach for image completion, texture synthesis and image inpainting without user interaction. Original belief propagation algorithm is extended with two major improvements besides standard belief propagation: “label pruning” and “priority-based message scheduling” which decreases computational cost of BP dramatically. Both extensions are applicable to any image and they are used to optimize any Markov Random Field (MRF). Above all, cost calculations and MRF based approach allows parallel computable which means we could use computation power of the modern GPUs using Compute Unified Device Architecture (CUDA) for significant speedup.

With the development of recently proposed neural network approaches, their applications derived as well as for inpainting tasks. Such methods basically initialize the target region by filling the pixels with a constant value such as mean ImageNet pixel values [13]. Then convolutional network takes the image as input, and obtain a result with artifacts which are then repaired in the post-processing. Pathak et al. [14] proposed a method to replace hole regions with semantic image understanding approach called context encoders. Yang et al. [15] extended context encoders by propagating texture information of source region pixels as a post processing step. Similarly, Poisson blending is applied as a post-process [16]. Following this work, post processing is done by using refinement network with attention layers [17], [18]. In [19], generative adversarial networks are used to avoid any dependence of external training set, nevertheless hyperparameters need to be tweaked for each image separately and it suffers from need for many iterations to obtain satisfactory results. By using standard convolutional layers, encoders become vulnerable to noise and incorrect hole initialization. Harley et al. [20] proposed a soft attention mask for semantic segmentation task in order to solve standard convolution vulnerabilities. Further, [21] makes use of a partial convolution operation, which prioritizes valid inputs. Network based approaches are able to complete any image semantically, still they require excessive amount of time to be trained and focus on specific image context.

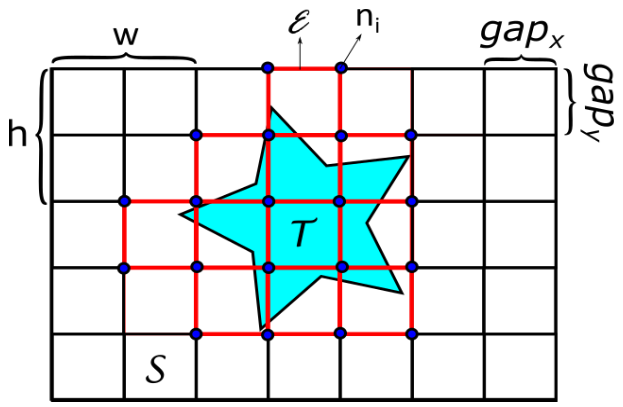


Fig 1: Illustration of the source and target regions, nodes and edges. Here, gapx and gapy are chosen w=2 and h=2, respectively.

II. IMAGE INPAINTING AS A DISCRETE GLOBAL OPTIMIZATION PROBLEM

Given an input image $\mathcal{I}_0$, target region T and a source region S where $(S \subset \mathcal{I}_0 - T)$, the main goal of image inpainting is to reconstruct the pixels in T in a visually acceptable way using the information obtained from S by using the following discrete Markov Random Field (MRF):

Each label L of MRF represents the patches from the source region S, whose size is $w \times h$. For this purpose, windows with width gap_x and height gap_y are employed. Intersection of corner points of these windows and target region T in $w \times h$ neighborhood, constructs a MRF node n_i, 4-neighborhood paths between nodes construct an edge $\mathcal{E}$ as demonstrated in Fig .

$V_i(l)$ denotes the cost for replacing patch $l \in \mathcal{L}$ with n_i and indicates the difference between patch and the source region around n_i using sum of squared differences (SSD):

$$V_i(l) = \sum_{k=1}^{\infty} \mathcal{M}(n_i + p)\left(\mathcal{I}_0(n_i + p) - \mathcal{I}_0 - \mathcal{I}_0(l + p)\right)^2$$

$$p \in \left[-\frac{w}{2}, \frac{w}{2}\right] \times \left[\frac{-h}{2}, \frac{h}{2}\right]$$

where $M(.)$ is a binary mask with the following equation:

$$M(n_i) = \begin{cases} 0, & if\ n_i \in T \\ 1, & if\ n_i \in S \end{cases}$$

Similarly, the pairwise cost $V_{ij}(l, l')$ indicates the score of the match between due to placing patches l, l' at the resulting region of overlap and will again be given by the SSD over neighbors n_i, n_j. Note that gap_x and gap_y are set $\frac{w}{2}$ and $\frac{h}{2}$ respectively, so that overlap exists.

Using this approach, a label needs to be assigned $\hat{l} \in \mathcal{L}$ to each node n_i so that the total energy $E(\hat{l}_i)$ of the MRF is minimized using the following equation:

$$E(\hat{l}_i) = \sum_{i=1}^{N} V_i\left(\hat{l}_i\right) + \sum_{(i,j)\in\mathcal{E}} V_{ij}(\hat{l}_i, \hat{l}_j)$$

Intuitively, region $\mathcal{T}$ is revealed by optimizing this energy function.

III. PRIORITY-BP

Moreover, belief propagation algorithm could be applied to energy function. However, computational cost of BP is intolerable due to the huge number of existing labels. A modified MRF optimization scheme called Priority-BP is proposed in order to solve computation cost, by adding two improvements to BP algorithm. First one is dynamic label pruning which dramatically reduce the number of labels. The important thing to note is that only the beliefs calculated by BP are used for that purpose. Second improvement is priority-based message scheduling which utilizes label pruning and provides sending messages faster. Furthermore, it significantly improves BP's convergence, thus accelerating completion even further. In addition to these, Priority-BP is generic and could be applied with MRF energy function, thus computation cost is significantly decreased in case of huge number of labels.

Algorithm 1 Priority-BP Psseudcode

```
Assign priorities to nodes and declare them uncommitted
K: number of iterations
N: number of nodes
**for** k = 1 to K **do**
    **for** time = 1 to N **do**
        $n_i$ = "unvisited" node of highest priority
        Apply "label pruning" to node $n_i$
        forwardOrder[time] = $n_i$
        Mark n_i as "visited"
        **for** any unvisited neighbor $n_j$ of node $n_i$ **do**
            Send messages $m_{ij}(.)$
            Update $b_j(.)$ and node $n_j$ priority
        **end for**
    **end for**
    **for** time = N to 1**do**
        $n_i$ = forwardOrder[time]
        Mark $n_i$ as unvisited
        **for** any "visited" neighbor $n_j$ of node $n_i$ **do**
            send all messages $m_{ij}(.)$from node $n_i$ to node $n_j$
            update beliefs $b_j(.)$ as well as priority of node $n_j$
        **end for**
    **end for**
Assign to each node $n_i$ its label $\hat{l}_i$ that maximizes $b_i(.)$
**end for**
```

Similar approaches proposed but they only cover the restricted classes of MRF [22]. Priority-based message scheduling scheme can be used solely as a general method for reducing the convergence time of BP.

A. *Priority-based Message Scheduling*

BP is an iterative algorithm which propagates messages from one node to its neighbor nodes of an MRF [23], [24]. Messages sent from node n_i to node n_j denoted as $m_{ij}(l), l \in L$ which indicates that n_i votes for n_j to be assigned label l. Furthermore, messages are updated until it converges as follows

$$m_{ij}(l) = \min_{l_i \in \mathcal{L}} \left\{ V_i(l_i) + V_{ij}(l_i, l) + \sum_{k:\, k \neq j, (k,i) \in \mathcal{E}} m_{ki}(l_i) \right\}$$

Based on these messages, belief $b_i(l),\ l \in L$ is computed for each node, where belief $b_i(l)$ is defined as follows:

$$b_i(l) = -V_i(l) - \sum_{k:\, (k,i) \in \mathcal{E}} m_{ki}(l)$$

$b_i(l)$ estimates maximum a posteriori (MAP) at node n_i and thus results with decision how likely to assign label $\hat{l}_i$ to this node. On this basis, the label with maximum belief is assigned to the node, namely $\hat{l}_i = arg \max_{l \in \mathcal{L}} b_i(l)$. BP always results in the optimal solution for tree structured graphs, but it guarantees to converge to local optima, for graphs, such as Markov models. If we denote the total number of labels as$|\mathcal{L}|$, a single message update from node n_i to another node n_j costs $\mathcal{O}(|L|^2)$ time. As we have many labels due to high number of patches, pairwise cost matrix $V_{ij}(.,.)$ between any pair of adjacent nodes n_i, n_j gets larger and become resource hungry. To overcome this problem, we reduce the number of labels by utilizing the beliefs $b_i(l)$ calculated by BP. However, all of the nodes are not appropriate to apply label reduction. For the nodes where label pruning is applicable, we reduce the number of labels such that maximum number of labels after pruning isL_{max}, where$L_{max} \ll |L|$. Finally, message propagation from a signal node would take $\mathcal{O}(L_{\max})$ time. Message scheduling scheme provides two significant improvements in terms of speed. First, it makes label pruning possible and tends to send messages with less effort. Secondly, it boosts the convergence time of BP algorithm. Thereabout, message scheduling scheme is associated with the priorities assigned to each node of Markov Random Field. Node priorities rely on the node's confidence about which labels to choose, and it is continuously being updated during the execution of the algorithm. Basically, message scheduling scheme assign highest priority to the node to propagate messages which is most confident about its labels. By this way, node with highest confidence are able to tolerate the losses due to label pruning before propagating messages and messages will be cheaper (sends less data). On the other hand, the other nodes became more tolerant to pruning. Moreover, propagation of the most informative messages in a graph model helps belief propagation algorithm to converge faster, so that it needs less number of iterations to converge optimal solution.

A pseudocode of Priority-BP is represented in Algorithm **1**. The algorithm performs a forward and a backward pass in each iteration, respectively. Message scheduling scheme and label pruning occurs within the forward pass, where half of the messages propagated. During this pass, all nodes are visited in priority order. On each visited node n_i, it is marked as "visited" which implies that it must not be visited during the current forward pass. We also prune its labels and corresponding node transmit its messages to all of its neighbors except the visited nodes. We update the priorities of all neighbors who received a new message and then we continues with the next unvisited node that have highest priority until all nodes are visited. Backward pass aims to transmit the other half of the messages as well. During the backward pass, we do not utilize node priorities but only visit the nodes in reverse order with respect to the order we followed in the forward pass. Remaining unsent messages from each node are transmitted through this process. Thus, we do not apply label pruning during the backward pass. Node's priority is based on the current belief that node possess. The priorities are updated so that we can use them on the next iteration of forward pass. Updating the priorities is such an efficient task, since only the nodes who recently received messages must be updated.

B. *Node Priorities*

Belief $b_i(l)$ indicates whether label l suitable for node n_i or not. For the evaluation of the confidence, the number of labels whose belief exceeds the fixed threshold value b_{conf}. The main reason for this is that, if there are much labels which likely to be assigned to the node n_i , it implies that corresponding node turns out less confident about which explicit label to choose. On the contrary, if there are small number above the threshold, choosing the label for n_i becomes more confident as it only considers among a small set of labels.

We only consider relative beliefs denoted as $b_i^{rel}(l) = b_i(l) - b_i^{\max}$ where $b_i^{\max} = \max_{l \in \mathcal{L}} b_i(l)$ and confusion set of node n_i denoted as $CS(n_i) = \left| l \in \mathcal{L} :\ b_i^{rel}(l) \geq b_{conf} \right|$. Then, the priority of corresponding node n_i is computed as in Equation 1.

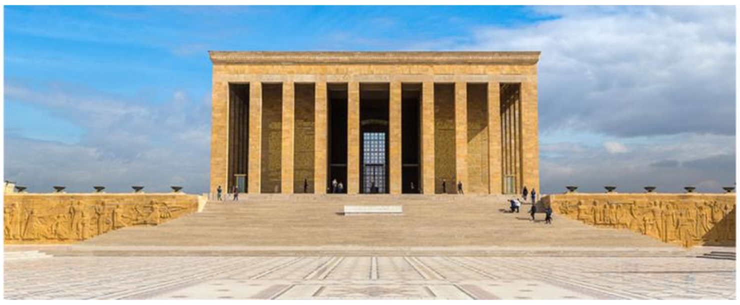
$$priority(n_i) = \frac{1}{|CS(n_i)|} \qquad (1)$$

Since belief of a node could only change when any of the received message changes, we update priorities that corresponds to the nodes which have new incoming messages during the both passes (forward and backward). Priority definition justifies this update mechanism.

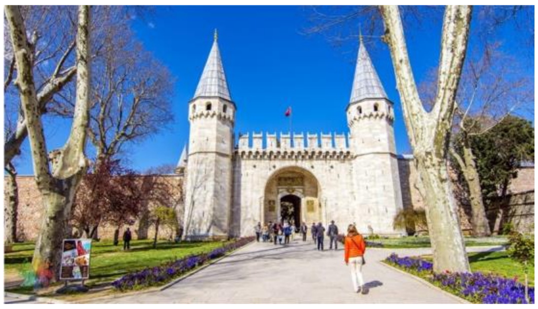
IV. RESULTS

Inpainting algorithm is performed on 2 corrupted images as shown in Figure 2. We applied a mask to remove the objects in the frame. Masked region is demonstrated in Figure 3.

Algorithm has been run on both images with 2.5GHz Intel i7-4710HQ processor and Nvidia Geforce GTX860M which has 640 CUDA cores. GPU and CPU implementations gave the

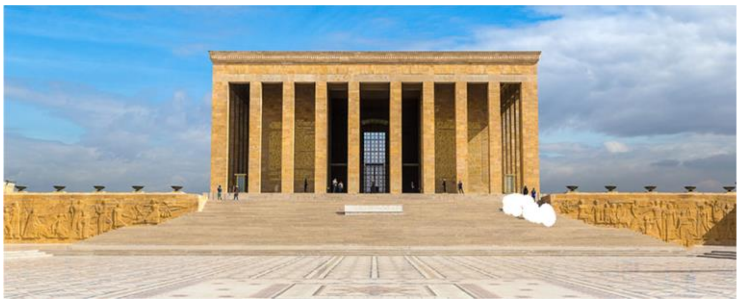


a) First sample

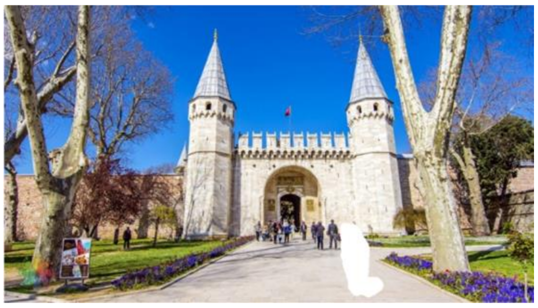


b) Second sample

Figure 2: Original Images

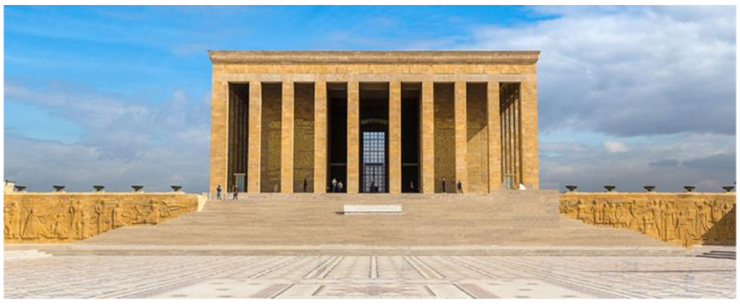


a) First Sample

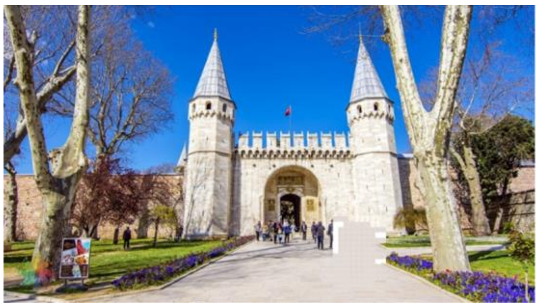


b) Second sample

Figure 3: Masked Region for Removing Objects



a) First sample



b) Second sample

Figure 4: Inpainted Images

same output for both images. Inpainted results for both of the images are shown in Figure 4.

For completeness, we implemented the speedup process on GPU under varying window and target region size. Number of node and iteration count increases as target region size increases and patch width and height decreases. Patch window size $w \times h$ and gap_x and gap_y changes as differently from target size increment. We need to paste more patches in exchange for high quality inpainted images.

Error! Reference source not found. demonstrates the varying mask size(target region size) and patch width and height sizes and their corresponding number of nodes in the image.

		Patch Width/Height (w = h)		
		8	4	2
Target Region Size	16×16	16	64	256
	32×32	64	256	1024
	64×64	256	1024	4096

Table 1: Number of nodes corresponding to the mask size (Target Region Size), width and height of the image patches

In both image we observe that there are major visual artifacts, since the mask image M is a minimum bounding matrix that that includes the target region. Hence, algorithm neglect the mask extent and reconstruct this region by using outer region. Mask should be considered solely for better

Target Region Size	Patch Size($w \times h$)	# of Patches	Single-Thread CPU (ms)	GPU / CUDA (ms)
16×16	16×16	6	2.2	2.8
32×32	16×16	40	139	9
64×64	16×16	180	6,241	126
16×16	8×8	40	139	9.7
32×32	8×8	180	6,254	117
64×64	8×8	748	859,518	8,238
16×16	4×4	144	3491	79
32×32	4×4	672	444,061	6,348
64×64	4×4	2880	91,448,426	365,798

Table 2: Performance comparison of sequential CPU and CUDA implementations

results. The parameters w and h is set to 16, and $b_{conf} = -SSD_0$ where SSD_0 is average score for a single node in 4-connectivity neighborhood. Size of the image does not affect the execution time, since we only consider the target region. Hence, number of generated patches and number of nodes inside target region are the main factor that affect the speed. Number of iterations are set to 100. Execution Time of CPU and GPU implementations are held in **Error! Reference source not found.**.

Execution time of GPU implementation includes transfer time require to copy data from CPU to GPU. As expected, GPU speedup factor increases as number of patches increased. Such an acceleration with more number of nodes is remarkable. Nevertheless, GPU acceleration is unsurprising, since message transferring between nodes is such a paralellizable process. GPU does not have a speed up when only 6 patches are available. However, the speedup factor drastically increases as the number of patches grow. With 2880 patches GPU provides up to 250x speedup.

Result of the inpainting algorithm for varying window sizes and node sizes are shown in **Error! Reference source not found.**. Target region increases from 16 to 64 and number of patches increase accordingly. We adjust patch size so that patch per window changes. As we analyze the images, inpainting quality increases as patch size decreases. We expect the image where we use 64×64 window size and node size 4×4 become the best among these image, however there is a bluish artifact on that image. Since we process the gray image, SSD calculation may find different colors close to each other. On the 16×16 nodes, people removed successfully, but patch borders are visible (seems not natural). 4×4 nodes provides quite smooth images for both 32×32 and 16×16 target regions.

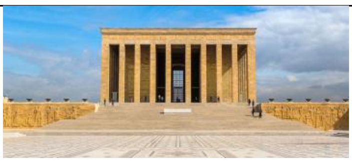
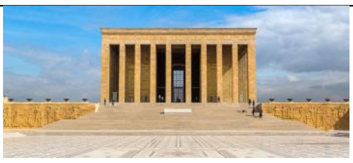
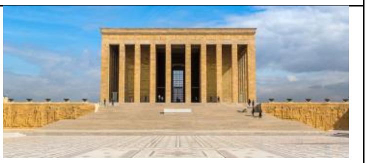
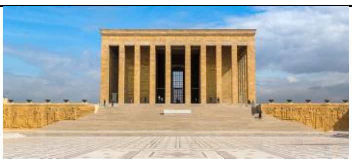
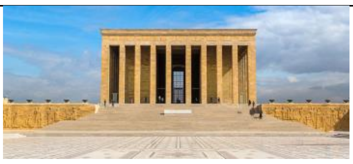
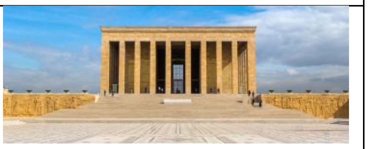
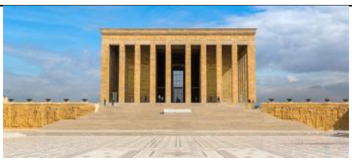
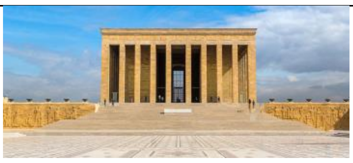
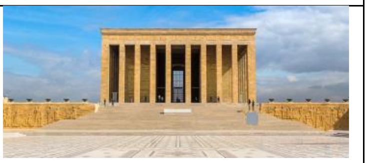
Patch Size / Node Size	16	8	4
16			
32			
64			

Table 3: Comparison of Various Window and Patch Sizes

V. CONCLUSIONS

An existing approach which performs image inpainting has been presented. In this approach, a modified belief propagation algorithm, called Priority-BP, has been utilized that extends the capabilities of standard BP with two improvements: priority-based message scheduling and label pruning. Algorithm is generic and could be applied any image without any prior knowledge. Finally, the algorithm is implemented using the power of modern graphical processing units, as algorithm contains many processes that is possible to run parallel. Significant speedup is observed with this improvement. However, the mask should be chosen carefully for more detailed enhancements. On the other hand, the algorithm may use a patch with different patterns which causes artifacts. The reason is we only use grayscale image for inpainting which does not consider RGB properties of the patch. Therefore, it pastes the patches from any other color. A color image will be replaced in a future work.

REFERENCES

[1]. M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, “Image inpainting,” in Proceedings of the 27th annual conference on Computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 417–424.

[2]. G. Emile-Male, The restorer’s handbook of easel painting. Van Nostrand Reinhold New York, 1976, vol. 31.

[3]. N. Komodakis, “Image completion using global optimization,” in 2006 IEEE Computer Society Conference on Computer Vision and Pattern

[4]. Recognition (CVPR’06), vol. 1. IEEE, 2006, pp. 442–452.

[5]. J. Portilla and E. P. Simoncelli, “A parametric texture model based on joint statistics of complex wavelet coefficients,” International journal of computer vision, vol. 40, no. 1, pp. 49–70, 2000.

[6]. A. A. Efros and T. K. Leung, “Texture synthesis by non-parametric sampling,” in Proceedings of the seventh IEEE international conference on computer vision, vol. 2. IEEE, 1999, pp. 1033–1038.

[7]. V. Kwatra, A. Sch¨odl, I. Essa, G. Turk, and A. Bobick, “Graphcut textures: image and video synthesis using graph cuts,” in ACM Transactions on Graphics (ToG), vol. 22, no. 3. ACM, 2003, pp. 277–286.

[8]. V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, “Texture optimization for example-based synthesis,” in ACM Transactions on Graphics (ToG), vol. 24, no. 3. ACM, 2005, pp. 795–802.

[9]. Y. Wexler, E. Shechtman, and M. Irani, “Space-time video completion,” in Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004., vol. 1. IEEE, 2004.

[10]. J. Sun, L. Yuan, J. Jia, and H.-Y. Shum, “Image completion with structure propagation,” in ACM Transactions on Graphics (ToG), vol. 24, no. 3. ACM, 2005, pp. 861–868.

[11]. I. Drori, D. Cohen-Or, and H. Yeshurun, “Fragment-based image completion,” in ACM Transactions on graphics (TOG), vol. 22, no. 3. ACM, 2003, pp. 303–312.

[12]. J. Jia and C.-K. Tang, “Image repairing: Robust image synthesis by adaptive ND tensor voting,” in 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings., vol. 1. IEEE, 2003, pp. I–I.

[13]. A. Criminisi, P. Perez, and K. Toyama, “Object removal by exemplar based inpainting,” in 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings., vol. 2. IEEE, 2003, pp. II–II.

[14]. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein et al., “Imagenet large scale visual recognition challenge,” International journal of computer vision, vol. 115, no. 3, pp. 211–252, 2015.

[15]. D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, “Context encoders: Feature learning by inpainting,” in Proceedings ofthe IEEE conference on computer vision and pattern recognition, 2016, pp. 2536–2544.

[16]. C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li, “High resolution image inpainting using multi-scale neural patch synthesis,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 6721–6729.

[17]. S. Iizuka, E. Simo-Serra, and H. Ishikawa, “Globally and locally consistent image completion,” ACM Transactions on Graphics (ToG), vol. 36, no. 4, pp. 1–14, 2017.

[18]. J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, “Generative image inpainting with contextual attention,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 5505–5514.

[19]. Y. Song, C. Yang, Z. Lin, H. Li, Q. Huang, and C.-C. J. Kuo, “Image inpainting using multi-scale feature image translation. corr abs/1711.08590 (2017),” 2017.

[20]. D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Deep image prior,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9446–9454.

[21]. A. W. Harley, K. G. Derpanis, and I. Kokkinos, “Segmentation-aware convolutional networks using local attention masks,” in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 5038–5047.

[22]. G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, “Image inpainting for irregular holes using partial convolutions,” in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 85–100.

[23]. P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," International journal of computer vision, vol. 70, no. 1, pp. 41–54, 2006.

[24]. W. T. Freeman, E. C. Pasztor, and O. T. Carmichael, "Learning low level vision," International journal of computer vision, vol. 40, no. 1, pp. 25–47, 2000.

[25]. J. Pearl, Probabilistic reasoning in intelligent systems: networks of

[26]. plausible inference. Elsevier, 2014