Guided Image Completion by Confidence Propagation*

Xiao Tan[a], Changming Sun[†b], Kwan-Yee K. Wong[a], Tuan D. Pham[c]

[a] *The University of Hong Kong, Pokfulam, Hong Kong.*
[b] *CSIRO Computational Informatics, Locked Bag 17, North Ryde, NSW 1670, Australia.*
[c] *Aizu Research Cluster for Medical Engineering and Informatics, The University of Aizu, Fukushima 965-8580, Japan.*

Abstract

This paper presents a new guided image completion method which fills any missing values by considering information from a guidance image. We develop a confidence propagation scheme that allows the filling process to be carried out globally without the need of deciding the filling order. We conduct experiments in several applications where the problem can be formulated into a guided image completion problem, such as interactive segmentation and colorization. The experimental results show that our method provides good results and succeeds in situations where conventional methods fail. In addition, as all building blocks are parallel processes, our method is much suitable for hardware acceleration.

Keywords:
Confidence propagation, Image completion, Image segmentation, Colorization

1. Introduction

Many computer vision and graphics applications involve an image completion process to restore damaged parts of an image or to infer pixel values in the unknown parts based on the known parts, so that the resultant images look natural. Such techniques are often used in applications for filling holes which are left behind after removing objects from photographs, or for colorizing images based on some color strokes provided by users.

*This work was mainly carried out while the first author was with CSIRO Australia.
†Corresponding author

A specific case of image completion is the application of image inpainting where nothing is known about the missing regions, and the missing regions are therefore filled according to some hypothesizes, e.g., the consistency in the pixel values and/or the gradients values (pixel based inpainting) or the consistency in the textural structure (texture based inpainting) between missing regions and neighboring regions. To address the pixel based inpainting problem, Bertalmio *et al.* [1] employed the idea of anisotropic diffusion to ensure that the filling process can transport the smoothness information along the isophotes direction which is perpendicular to image gradients. A faster technique employs the fast marching method to determine the inpainting order based on a distance map which is progressively updated as the filling boundaries move forward. Some other techniques [1, 2, 3, 4] address the pixel inpainting problem by propagating both the gradient directions and the intensity values of the surrounding pixels into the missing regions. However, these techniques do not convey texture information in the missing regions. One direct solution is adding synthetic texture to the regions [5]. However, this method fails to propagate strong oriented structures. To solve this problem, an exemplar-based texture inpainting method is proposed by introducing the strength of the gradient along the boundary into consideration. Additional related techniques include those in [6, 7, 8].

1.1. From Image Inpainting to Guided Image Completion

All the conventional image inpainting methods mentioned above fill values in the missing regions based on the inpainting input image only. Guided image completion, on the other hand, takes advantage of a given guidance image to aid the completion process. For example, filling missing values in a depth map under the guidance of a color image of the same scene, and filling missing values in chrominance channel under the guidance of the luminance channel. In these applications, when neighboring pixels have similar values in the guidance image, the values of these pixels in the image being completed are usually similar, and thus it is natural to exploit the guidance image during the completion. More formally, a guided image completion process involves a guidance image I, an input image being completed P, and an output image Q. The pixels whose values in P are given beforehand in some regions are denoted by Θ ($\Theta \subset P$). The goal of the guided image completion is to fill the values of pixels in $P \backslash \Theta$ under the guidance of I. We formulate this process

as

$$Q(i) = \begin{cases} P(i) & i \in \Theta \\ U(i) & \text{otherwise} \end{cases} \tag{1}$$

where U is the resultant image determined by the guided image completion algorithm.

Similar to the texture inpainting methods, some guided image completion methods, such as those in [7, 9], involve a texture synthesis step during the filling process to hallucinate texture in the filled region. While methods proposed for applications such as depth map completion or colorization often take into account the similarity of pixel values. In this paper, we focus on the pixel value based guided image completion problem where the consistency of pixel values in P roughly indicates the consistency in Q regardless of the noises existing in the known region of Q. A good algorithm for addressing the pixel value based guided image completion problem is expected to be (1) edge preserving (edges in U are coincident with those in I); (2) content preserving: pixels in homogenous regions (pixels having similar color or texture) of the guidance image are filled with similar values in U (in this paper we only consider color similarity); and (3) robust against unreliable pixels in Θ.

1.2. Literature Review

Previous studies have developed a variety of techniques to solve specific graphics problems which are viewed as particular cases of the image completion problem. One of the most popular methods is the quadratic function minimization based scheme as used in [10, 11, 12], where the completing is done by minimizing a quadratic function:

$$E(U) = \sum_{i \in I} \left(U(i) - \sum_{k \in \mathcal{N}_i} w_{ki} U(k) \right)^2 \tag{2}$$

where $U(i)$ is the value to be assigned to pixel i; $\mathcal{N}_i$ is a set of pixels around i; w_{ki} is a weighting function which is defined such that $\sum_{k \in \mathcal{N}_i} w_{ki} = 1$. The weighting function measures the similarity of pixels according to the guidance image, which reaches a high value in homogenous regions and reduces rapidly at the color/intensity boundaries. The functionality of weights aims to assign similar pixel values in homogenous regions while keep the edges in U coincident with those in I. There are many ways to define the weighting function. For example, it can be defined based on normalized correlation [13],

squared difference [10, 12], or the "matting affinity" [11, 14]. As any constant vector is a trivial solution to Eq. (2), $E(U)$ is minimized subject to an additional constraint: $U(i) = P(i)$, $\forall\ i \in \Theta$. The closed-form matting algorithm in [11] takes user inputs (the pixels in Θ) as soft constraints instead of hard constraints [12, 10]. Minimizing Eq. (2) involves inverting a sparse graph Laplacian matrix. However, the complexity of inverting this sparse matrix grows rapidly with the increase of non-zero elements. Such algorithms therefore often define $\mathcal{N}_i$ as a small neighborhood, e.g., 3×3 or 5×5 squared windows.

Some other techniques perform guided image completion by filling pixels sequentially in a similar manner as conventional inpainting methods [7, 4]. Superior to the conventional inpainting methods, these techniques modify the sampling process or the filling order by considering the color similarity of neighboring pixels in the guidance image. This modification enables pixels with similar color of the neighboring known pixels to be filled earlier, and thus being likely to assign similar pixel values in the homogenous region.

Yatziv and Sapiro used the geodesic distance to measure how pixels are related and proposed using a weighted blending method to carry out the guided image completion [15]. They applied this approach in the image and video colorization. Since blending all pixel values from known regions is computationally expensive, only serval known values with high weights are used during the blending to compute the filling value for a given pixel. Another popular practise of the guided image completion is the interactive segmentation where users provide some strokes on different objects and the algorithms will automatically segment the image into different parts. Carsten formulated this problem as an energy minimization problem over a Markov random field (MRF) model, which is solved using the graph-cuts energy minimization method [16].

The main shortcoming of these approaches is the fact that they may struggle when the guidance image contains interleaving homogenous regions (see Fig. 1). There are multiple reasons for this. Some are caused by the inefficiency of the model, for example the 3×3 or 5×5 neighborhoods in Eq. (2) fail in propagating values from the known region to unknown regions far away. Some lie in the inappropriate filling scheme, for example the weighted averaging filling scheme in [4] will cause trailing or fatting artifacts. This paper proposes a new guided image completion method by introducing a self-evaluate mechanics called confidence propagation which evaluates how a pixel is filled after each filling step and exploits the evaluation to make the

filling process bias towards those reliably filled pixels and thus provides more convincing results. Moreover, it is easy for parallel acceleration since the pixels are processed independently in each filling step. Previous approaches are designed for one or two specific problem(s) and may work (well) only for problem(s). For example, the method proposed in [16] works only with discrete filling values but not with continuous filling values. By contrast, our method provides an unified framework to formulate different graphic problems, which will be shown later in the experimental section of this paper.

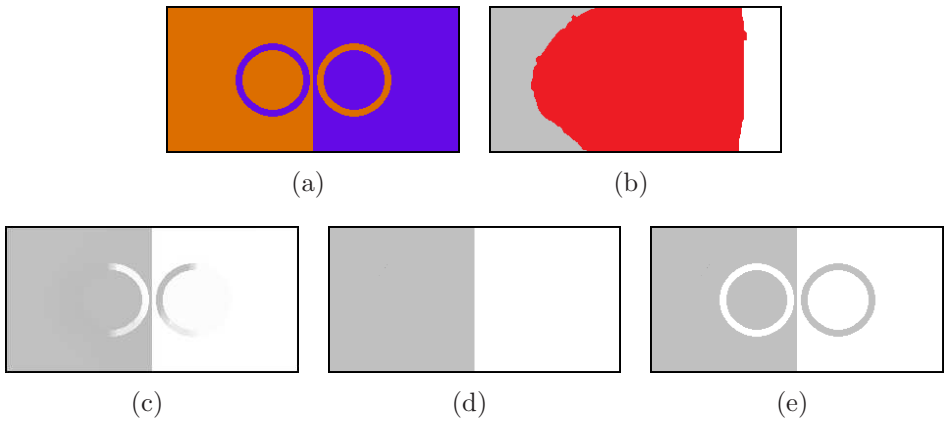


Figure 1: Guided image completion on a synthetic image. (a) Guidance image. (b) input image for completion. Regions to be filled are marked in red. (c) Results by sequentially filling method [4]. (d) Results by minimizing a quadratic function [10, 11]. (e) Our results.

2. Method

We define the following notations in our algorithm: the confidence of the assigned value at pixel i is defined as C_i; the set of neighboring pixels in a squared window centered at i with size $2r + 1$ as $\mathcal{N}_i$.

Our algorithm iteratively performs the following two steps in an interleave manner until the terminating criterion meets: (1) filling step: fill or update values of pixels in missing regions; and (2) propagation step: update the confidence values of pixels.

2.1. Computing Filling Values Using Confidence

We assume that the confidence value of pixels after the previous iteration are known. In the filling step, a value is assigned to pixel i at the tth iteration

by a filtering-like process:

$$U^t(i) = \mathcal{F}\left(i, Q^{t-1}, W\right) \tag{3}$$

Here, Q^{t-1} is the completion results in the $(t-1)$th iteration. $\mathcal{F}(i, Q^{t-1}, W)$ is the output of adaptive filtering at pixel i with W being the weights and Q^{t-1} being the inputs. For example, the weighted average filtering at i is explicitly expressed as:

$$\mathcal{F}\left(i, Q^{t-1}, W\right) = \frac{\sum\limits_{k \in \mathcal{B}_i} W_{ki} Q^{t-1}(k)}{\sum\limits_{k \in \mathcal{B}_i} W_{ki}} \tag{4}$$

where $\mathcal{B}_i \subseteq \mathcal{N}_i$ are pixels with known values (pixels in Θ or pixels whose values have been filled perviously before the tth iteration). Unlike conventional filtering methods which define W only by the local property around a pixel, our method incorporates a term called confidence for helping fill values globally. W_{ki} is defined here as the product of two terms:

$$W_{ki} = w_{ki}\left(C_k^{t-1}\right)^{\gamma} \tag{5}$$

where C_k^{t-1} is the confidence value at pixel k in the $(t-1)$th iteration, and γ is a non-negative constant controlling the strength of the confidence; w_{ki} is a weight of k to i computed from their pixel value similarity in the guidance image. In this study, w_{ki} is defined using the bilateral filtering kernel [17] with Gaussian functions:

$$w_{ki} = \exp\left(-\frac{|I_i - I_k|^2}{\sigma_r^2}\right) \exp\left(-\frac{|i-k|^2}{\sigma_s^2}\right) \tag{6}$$

Here I_i is the value of i in I. In general, the confidence measures how sure of assigning a value to pixels in the previous iteration. The weighting function w_{ki} pulls the filling results towards those pixels which are close to i and whose are similar to that of i in the guidance image I. By using the confidence and the weighting function together, our method favors assigning value to i with the values of pixels having high confidences and being similar to i in I.

2.2. Confidence Propagation

Confidence propagation is carried out with the steps of the filling process. Before describing confidence propagation, we introduce a reliability map $\mathcal{R}$

which contains the initial confidence value of the pixels in Θ. The reliability map, an optional input, is defined based on some application dependent a priori. For example, in depth completion where the scene is usually piecewise smooth, the reliability of a pixel can therefore be defined based on the number of pixels which have the similar depth values around it (see Section 3.1 for details). The reliability ranges from 0 (unreliable) to 1 (reliable). As the value of pixel $i \in P \backslash \Theta$ is not given, its reliability is 0. The reliability map enables us to incorporate additional information or hypothesis for achieving robust image completion. If there is no such information or hypothesis, we can just set $\mathcal{R}(i)$ to 1 for all i in Θ. The reliability of a pixel is fixed after the assignment.

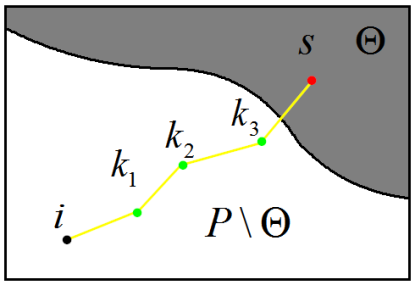


Figure 2: Path sinking at pixel $i \in P \backslash \Theta$. The red pixel s is the source, and the black pixel i is the sink. The path is colored in yellow, and the pixels on the path are colored in green. The reliability of the source is $\mathcal{R}_i^t = \mathcal{R}(s)$, and m_i^t, the minimum value of all weights on the edges of the path, is given by $m_i^t = \min(w_{ik_1}, w_{k_1k_2}, w_{k_2k_3}, w_{k_3s})$.

Suppose that for a pixel of interest i, we find a path where the source is a pixel in Θ and the sink is i. This path is recursively constructed and updated as the completion process goes. Denote the path we find in the tth iteration by $\mathcal{P}_i^t$. For any two neighboring points p and q along the path, we compute the weight w_{pq} between them based on the similarity measured from the guidance image. For simplicity, this weight is also computed by using Eq. (6). We introduce m_i^t to denote the minimum weight among all weights along $\mathcal{P}_i^t$ (see Fig. 2 for details). The confidence value of i in the tth iteration is defined as:

$$C_i^t = \begin{cases} \mathcal{R}(i) & i \in \Theta \\ m_i^t \mathcal{R}_i^t & \text{otherwise} \end{cases} \tag{7}$$

where $\mathcal{R}_i^t$ is the reliability value of the source of $\mathcal{P}_i^t$. According to the definition, we could see that m_i^t is large only if pixels on the path are similar with

each other or, in other words, all pixels along the path are in the homogenous region of the guidance image. That is to say if the path crosses region boundaries, m_i^t will be small. An alterative for defining m_i^t would be the geodesic distance or shortest path based method as used in [15]. However, it does not perform well for propagating the confidence over long distances when noises exist in the guidance image, because geodesic distance accumulates the distance between neighboring two pixels along a path, and hence the impact of the noise will also be accumulated. Unlike the method in [15], our method defines m_i^t based on the minimum value, and the two neighboring pixels on the path are not limited to be 4 or 8 connected. Therefore, as will be shown in our experiments, our method performs well even where the homogenous regions are interleaving with each other in the guidance image. The confidence for $i \in \Theta$ is defined by its own reliability and is fixed. For $i \in P \backslash \Theta$, its confidence updates with the change of its path configuration which is being updated as the iteration goes.

During initialization, each pixel in Θ is assigned one path and pixels in $P \backslash \Theta$ are not assigned to any path. To show how to update the path configuration, we focus on a single iteration. Suppose the path for pixel i is being updated. Q^{t-1} (filling results in the previous iteration) are known, and $U^t(i)$ have already been computed by using the method as described in Section 2.1. The path is updated in such a way that the filled values of two neighboring pixels on a path are similar, and the confident value of the path is high. For this purpose, we add i to the path of another pixel j chosen from pixels whose filling results in the previous iteration are similar to $U^t(i)$. More specifically, denote the minimum difference between $U^t(i)$ and the filling results of its neighboring pixels by $\Delta_i^t = \min_{k \in \mathcal{B}_i} |Q^{t-1}(k) - U^t(i)|$. The set of candidates is given by:

$$\Phi_i = \left\{ k \middle| k \in \mathcal{B}_i \wedge \left| Q^{t-1}(k) - U^t(i) \right| < \Delta_i^t + \delta \right\} \tag{8}$$

where δ controls the selection of candidates and is set to 1 by default when the filling values are integers (see Fig. 3). To some extent, δ allows the changing of filled values of pixels along the path. We pick a pixel among the candidates in Φ_i such that

$$j = \arg\max_{k \in \Phi_i} \left(\hat{m}_k^t \mathcal{R}_k^{t-1} \right) \tag{9}$$

where

$$\hat{m}_k^t = \min \left(w_{ki}, m_k^{t-1} \right) \tag{10}$$

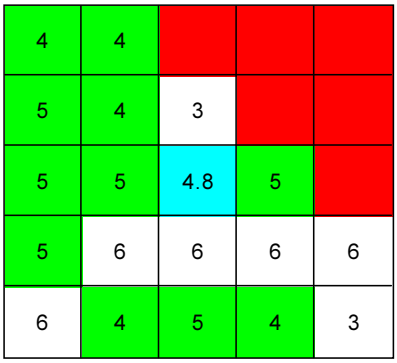


Figure 3: Diagram of candidates in Φ_i out of a 5×5 neighborhood. Pixel i is colored in blue and $U^t(i) = 4.8$. Filled values in the previous iteration (Q^{t-1}) are shown in the grid. $\Delta_i^t = 5 - 4.8 = 0.2$. Pixels whose values have not been filled yet are denoted in red. Candidates in Φ_i for $\delta = 1$ are colored in green.

For all pixels in $P \backslash \Theta$, $\hat{m}_i^0$ is initialized to be 0. If j computed from Eq. (9) has a higher $\hat{m}_j^t$ value than $\hat{m}_i^{t-1}$, we will add i to the path of j from behind, and call j the *preceding pixel* of i; otherwise we do nothing to i. Recall that $\hat{m}$ is initialized to be 0, and thus if i is not involved in any path, it will be involved by the first path passing through its neighborhood; however when another path passing through the neighborhood with a higher confidence value, i may be attached to this path and be removed from the old path. Once the (new) *preceding pixel* of i is found, we accept the new confidence value. That is,

$$\begin{cases} \mathcal{R}_i^t = \mathcal{R}_j^{t-1},\ m_i^t = \hat{m}_j^t & \text{if } \mathcal{R}_i^{t-1} m_i^{t-1} < \mathcal{R}_j^{t-1} \hat{m}_j^t \\ \mathcal{R}_i^t = \mathcal{R}_i^{t-1},\ m_i^t = m_i^{t-1} & \text{otherwise} \end{cases} \tag{11}$$

and

$$C_i^t = m_i^t \mathcal{R}_i^t \tag{12}$$

This updating rule is simple and makes it possible to compute the confidence value of a pixel without explicitly computing the configuration of the path. This is similar to using the dynamic programming or the Dijkstra algorithm [18, 19, 20] for computing the geodesic weight which is defined by the shortest path, and these methods compute the weight without explicitly

computing the path. However, if one does want to know the path configuration of a pixel, it can be achieved by recursively tracing the *preceding pixel* until a pixel in Θ is reached, and the pixel found in Θ is the source of the path. In addition, since w_{ki} in Eq. (10) is the same as that used in Eq. (5), we can therefore reuse the weights when computing Eq. (10).

2.3. Implementation Issues and Algorithm Steps

The complexity of our algorithm consists of two parts: performing the value filling (Eq. (3)) and finding the pixel which has the maximum confidence (Eq. (9)). In general, the complexity of both operations for one pixel in one iteration is approximately $O(|\mathcal{N}|)$, so the complexity of the algorithm is $O\left(|\mathcal{N}| \sum_{i \in I} T_i\right)$ where T_i is the number that pixel i has been visited until the algorithm stops.

We propose two schemes to reduce the algorithm complexity by reducing the number of visits to a pixel. The first scheme is based on the fact that the value of a pixel i inside the missing region cannot be estimated when no pixel in $\mathcal{N}_i$ is filled. This observation implies that when filling values we do not need to visit all pixels in $P \backslash \Theta$ but only visit those near the filling boundaries. Thus, we only consider pixels within 3-pixel-distance to the filling boundary in each iteration. A similar approach called the "onion peel" method has been used in previous exemplar-based inpainting works [6]. However, in their studies the pixels or patches are filled sequentially using best pixel (or patch) first method [4, 7, 21] and the value is fixed once filled.

In addition, we notice that most pixels whose values stay unchanged in two sequential iterations will keep unchanged in all following iterations unless the filling values and the confidence values of its neighboring pixels are both changed. This observation follows from the fact that the confidence and filling values of a pixel is computed from those values of pixels within its neighborhood using Eq. (4) and Eq. (11). Hence, if there is no value change happened within its neighborhood, the values of this pixel will not change. On the other hand, when value change happens within the neighborhood of a pixel, we will revisit this pixel to check whether the value change affects this pixel by recomputing Eq. (4) and Eq. (11). Therefore, after the assignment to a pixel, we do not revisit it unless there is at least one neighboring pixel (within 3-pixel distance) whose filling value and confidence value are both changed during the previous iteration. These two schemes greatly reduce the

total number of visits to a pixel. Denote pixels to be visited at the tth iteration by Ω^t. Recall that the confidence value is updated in a non-decreasing manner, which means that if a pixel being revisited has a high confidence value, its (confidence and filling) value is likely to be kept unchanged, while if it has a low confidence value, it is likely to be modified. Since the filling value for a pixel with low confidence value is usually incorrect, our method is likely to correct this value when the pixel is revisited in subsequent iterations.

Recall that we do not revisit a pixel if no confidence value or no value assignment of its neighboring pixel is changed. Therefore, once the filling results in two sequential iterations are equal: $Q(i)^{t-1} = Q(i)^t$ for all i, our algorithm will stop. However, it is not guaranteed that there is a certain t so that $Q(i)^{t-1} = Q(i)^t$ for all i. Fortunately, it is guaranteed that confidence values of all pixels will be fixed after a certain number of iterations, because the confidence value of a pixel is computed in a non-decreasing way during the propagation (Eqs. (11) and (12)). In addition, the confidence value is not greater than 1. As a result, the value of confidence is a bounded monotonic sequence whose value will be fixed at its limit.

Algorithm 1 shows the steps of the proposed method.

Algorithm 1. Steps of Guided Image Completion:

Inputs: guidance image I, input image for completion P with partially known values for pixels in Θ, a weighting function w, a filtering function $\mathcal{F}$, a reliability map $\mathcal{R}$ (optional).

Parameters: the radius of the squared window r, the strength of confidence γ.

Outputs: Guided completion output Q.

1. Initialize $t = 1$ and set the confidence value $C_i^0 = \mathcal{R}(i)$.
2. Select pixels (Ω^t) to be updated by picking the pixels within 3-pixel-distance to the filling boundary or to the pixels whose filling values and confidence values are changed in the previous iteration.
3. For each i in Ω^t
   a. compute the value $U^t(i)$ using Eq. (3).
   b. find the pixel that maximizes the confidence from neighboring pixels using Eq. (9).
   c. update the confidence value by Eqs. (11) and (12).

4. If there is at least one pixel whose confidence value and filling value are both changed during step 3, then $t := t + 1$ and go to step 2; otherwise go to step 5.
5. Output results: $Q(i) = \begin{cases} P(i) & i \in \Theta \\ U^t(i) & \text{otherwise} \end{cases}$

3. Experiments and Applications

We tested our algorithm on a wide variety of computer vision and graphics applications. All experiments were run on a 32 bits PC with a 3.0 GHz CPU and 4.0 GB RAM under the standard C++ implementation without using any optimization or acceleration techniques. Furthermore, unless otherwise specified, the reliability of pixels in Θ was initialized to 1. In all the following experiments, we used constant parameters: $\gamma = 7.0$, $r = 9$, $\sigma_r = 0.1$, and $\sigma_s = 11$.

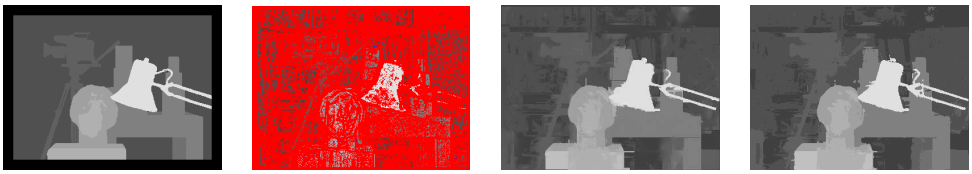
Codes of our algorithm will be released with the publication of this article.

(a) (b) (c) (d) (e)

Figure 4: Results of depth map completion for Kinect cameras. (a) Color image. (b) Depth map from the Kinect cameras. (c) Corresponding reliability map calculated from Eq. (14) which is normalized such that the maximum value equals 255 and the minimum value equals 0. (d) Results by the sequentially filling method [4] (note that the depth of the chair is incorrectly filled). (e) Our results.

3.1. Depth Map Completion

In depth map completion, a depth map (P) with missing regions and a color image (I) of the same scene are given beforehand. In this application, the weighted median filter is chosen as $\mathcal{F}$ for obtaining sharp boundaries. The weighted median filter starts with building a histogram for the pixel of interest i by summing up weights over B_i, where the bin of the histogram is



(a) (b) (c) (d)

Figure 5: Results of depth map completion using sparse control points. (a) Groundtruth depth map. (b) Sparse control points. Pixels containing noises are colored in blue. (c) Results from [12] (error pixels: 2.41%). (d) Our results (error pixels: 1.98%).

indexed by the pixel (depth) value. The value of this histogram in the lth bin is given by [22]:

$$H\left(i,l\right)=\sum_{k\in B_i}W_{ki}\delta\left(Q\left(k\right)-l\right) \tag{13}$$

where l ranges from valid depth values, and $\delta\left(\cdot\right)$ is a Dirac delta function. The median filter accumulates this histogram until it reaches the median value of the summation over all bins, that is to find l' such that $\sum_{l<l'}H\left(i,l\right)<\frac{1}{2}\sum_{l}H\left(i,l\right)$, and $\sum_{l\leq l'}H\left(i,l\right)\geq\frac{1}{2}\sum_{l}H\left(i,l\right)$. The median filter then outputs l' at pixel i. Now consider defining the reliability map. Take the application of completing depth map from the Kinect [4] as an example, where the color image is used as a guidance image. As the depth map provided by the Kinect cameras contains noises, we define the reliability of pixel $i\in\Theta$ as:

$$\mathcal{R}\left(i\right)=\frac{1}{\left|\mathcal{N}_i\right|}\sum_{k\in\mathcal{N}_i}\exp\left(-\frac{\left|I_i-I_k\right|^2}{\sigma_r^2}\right)\delta\Big(P\left(i\right)-P\left(k\right)\Big) \tag{14}$$

This reliability map is shown in Fig. 4(c). The sequential filling method which is based on the best pixel first manner does not perform well in regions with a large area of missing values, while our method is able to recover these regions (see Fig. 4). All results are filtered by guided filtering [14] for better performance at boundaries.

Another type of depth map completion as proposed in [12] is using a set of sparse ground control points for leveraging stereo matching based on the least squared method. The result of our method and that of [12] are provided in Fig. 5. For quantitative evaluation, if the difference between the filling result

of a pixel and the ground truth depth value is larger than 1, we will regard the pixel as an error pixel. As demonstrated in Fig. 5, our method is more robust to erroneous ground control points and provides better results than that of [12]. To compare with the fast marching method in [4] (FMM) and the quadratic minimization based method [12] (QMM), we carried out more experiments on the Middlebury datasets [23]. In this experiment, we randomly selected regions to be completed and removed the depth information there. The completion results were checked against the ground truth depth maps to calculate the rate of error pixels. The quantitative results are listed in Table 1, and the visual comparison with different methods are given in Figs. 6 and 7. From Table 1, we see that our method provides quantitatively better results in most datasets. In Figs. 6 and 7, we find that our method outperforms the other two methods at object boundaries.

Table 1: The comparison on the rate of error pixels (the best performance is colored in red). The average errors over all 20 datasets are provided in the last entry.

Images	Methods			Images	Methods		
	QMM [12]	FMM [4]	Ours		QMM [12]	FMM [4]	Ours
Aloe	0.291	0.154	0.123	Baby1	0.291	0.109	0.107
Baby2	0.041	0.024	0.046	Baby3	0.163	0.112	0.091
Bowling1	0.203	0.173	0.251	Bowling2	0.228	0.120	0.142
Cloth1	0.034	0.108	0.183	Cloth2	0.420	0.126	0.131
Teddy	0.280	0.282	0.050	Lampshadel1	0.274	0.125	0.070
Lampshade2	0.356	0.214	0.054	Midd1	0.340	0.216	0.099
Monopoly	0.143	0.061	0.041	Plastic	0.109	0.073	0.012
Rocks1	0.377	0.364	0.128	Rocks2	0.097	0.088	0.087
Moebius	0.091	0.054	0.056	Dolls	0.093	0.096	0.092
Reindeer	0.185	0.139	0.123	Flowerpots	0.201	0.150	0.166
				Averaging	0.211	0.139	0.103

3.2. Colorization

In image colorization, color information is added to pixels through some color strokes provided by users. In this application, a gray scale luminance image is used as the guidance image and the chrominance channels are the completion inputs. The colorization is performed on the YUV color space, where Y contains the luminance information, while U and V channels encode the color. In this application, we adopt the weighted averaging filter as $\mathcal{F}$. In this experiment, we used strokes picked from a color image and

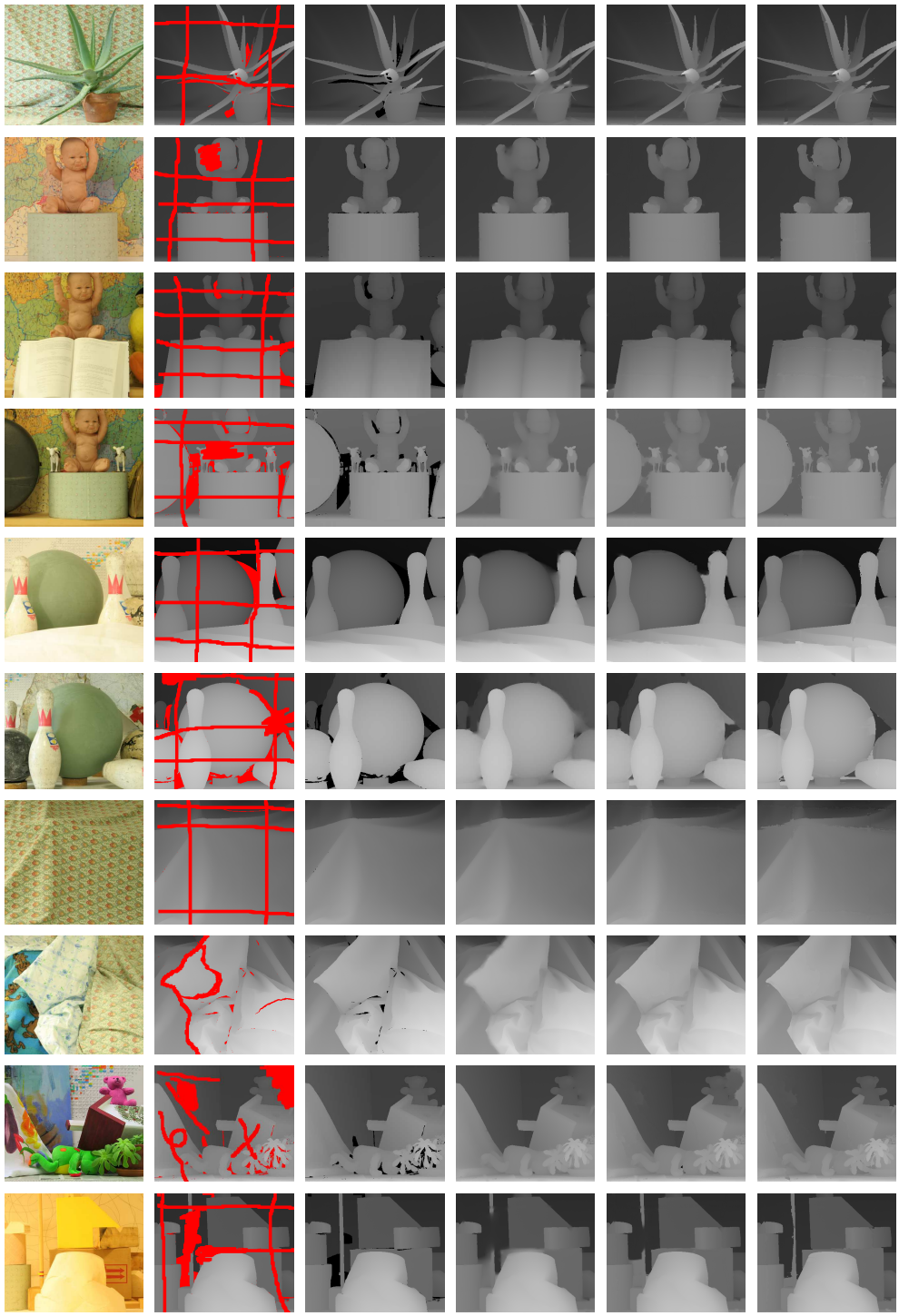


Figure 6: Results of depth map completion on Middleburry datasets [23] (to be continued in Fig. 7). (a) Color image. (b) Impaired depth map. (c) Ground truth (depths in dark points are missing). (d) Results from QMM [12]. (e) Results from FMM [4]. (f) Our results.

Figure 7: Results of depth map completion on Middleburry datasets [23] (continue from Fig. 6). (a) Color image. (b) Impaired depth map. (c) Ground truth (depths in dark points are missing). (d) Results from QMM [12]. (e) Results from FMM [4]. (f) Our results.

used the luminance channel of the color image as the guidance image. To quantitatively evaluate the results, we compared the results with the original color image under the measure of normalized mean square error (NMSE) which is given by: $\frac{1}{\{I\}} \sum_{i \in I} \frac{\|\mathbf{U}_i - \mathbf{I}_i\|}{\|\mathbf{U}_i\| + \|\mathbf{I}_i\|}$ where $\mathbf{U}_i$ and $\mathbf{I}_i$ are the vector of "RGB" value at pixel i in the resultant image of colorization and the original image respectively, and $\{I\}$ is the number of pixels in the image. The smaller the NMSE value is, the better the result will be. We compared our method with the quadratic minimization based approach [10] (QMM) and the geodesic weights based approach [15] (GEO). See Fig. 8 for visual comparison and Table 2 for quantitative comparison.

Because method in [10] performs colorization by minimizing a constrained quadratic function, and the computational complexity of inverting a huge sparse matrix grows rapidly with the size of its non-zero elements, this method can only handle small size neighborhoods (e.g., 3×3 or 5×5 neighborhoods). Unfortunately, small neighborhoods usually lead to the"bleeding" artifact at weak boundaries. This artifact also exists in the results from the geodesic blending method proposed in [15]. This artifact is demonstrated in the test image "Mountain" in Fig. 8 where our method does a much better job in suppressing this "bleeding" artifact. Besides, the method in [10] fails to correctly assign values to pixels which are far from strokes. For example, the blue strokes locating at the top of the background near the words in test image "Mountain" only contributed to its neighboring pixels. Many pixels of the sky are incorrectly colored in red due to the failure of propagating the value of the blue strokes to them. In addition, test image "Flower" in Fig. 8 illustrates another case that we believe to be challenging for competing methods. Here, the challenge is the region where the guidance image varies dramatically while the completion results should not vary. Let us look at the test image "Flower": the luminance values of pixels on leafs and stems vary a lot over the image, while their color is approximately constant (green color). Our method has better performance in these regions compared with methods in [10, 15], since our method groups non-connecting pixels which have a similar luminance value into a path and filling them with a similar color.

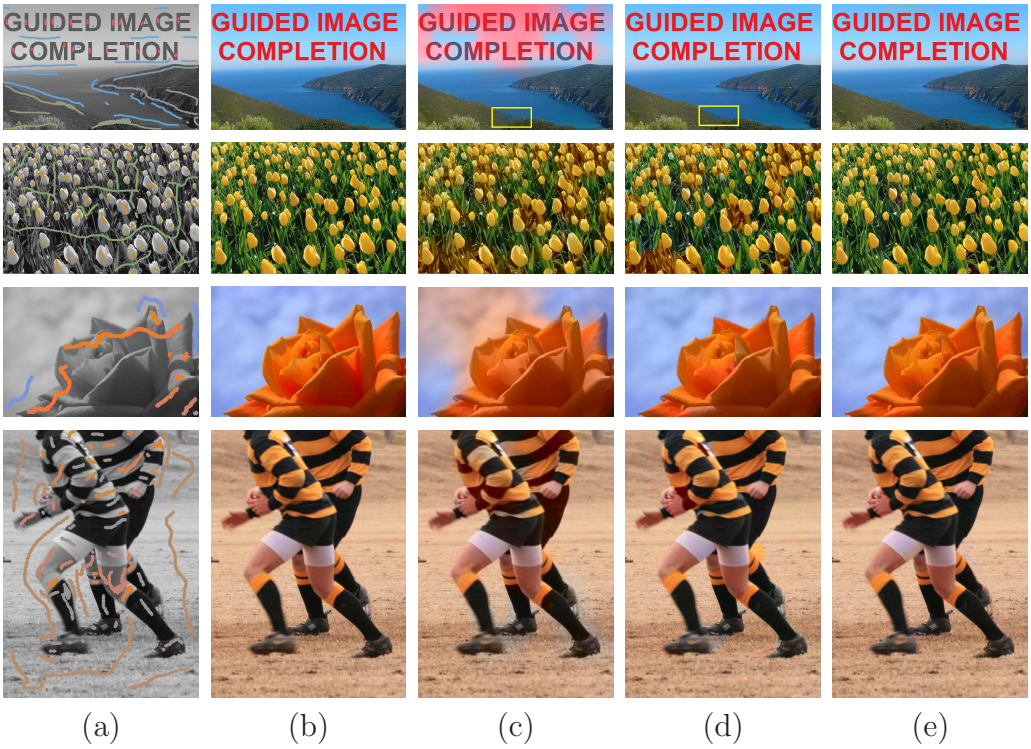


Figure 8: Results of colorization (better viewed in the electronic version). From the top to the bottom are test images: Mountain, Flower, Rose, Sport. (a) Gray-scale image with color strokes, (b) Reference color image, (c) Results from [10], (d) Results from [15], and (e) Our results. The background pixels on the blue sky in the"Mountain" image in (c) are incorrectly colored because they are far from the blue strokes. The green grass in yellow squares in (c) and (d) are incorrectly colored in blue due to the "bleeding" artifact.

Table 2: The comparison on the NMSE value (the best performance is colored in red).

Images	Methods		
	QMM [10]	GEO [15]	Ours
Mountain	0.097	0.021	0.018
Flower	0.148	0.131	0.077
Rose	0.078	0.028	0.027
Sport	0.049	0.025	0.019
Averaging	0.093	0.051	0.035

3.3. Interactive Segmentation

In interactive segmentation, an image I is given with some known labels $U(i) = l_j$ ($l_j \in \mathcal{L} = \{l_1, l_2, \cdots, l_n\}$) indicating the segment where pixel i belongs. Typically, $n = 2$ for the foreground and background segmentation as in [16] and [24]. The interactive segmentation can also be formulated into the guided image completion framework where the completion value is the label of a pixel. In this problem, the label indicates the classification of pixels and does not have numerical meaning. Therefore, the weighted mode filter [25, 26] is suitable. In this filter, the output of i is determined by maximizing a histogram: $H(i, U^{t-1}, W)$ where its lth bin is given by:

$$H\left(i, U^{t-1}, W, l\right) = \sum_{k \in \mathcal{N}_i, U^{t-1}(k)=l} W_{ki} \tag{15}$$

and the output of the weighted mode filter at i is:

$$\mathcal{F}\left(i, U^{t-1}, W\right) = \arg\max_{l \in \mathcal{L}} H\left(i, U^{t-1}, W, l\right) \tag{16}$$

The energy based methods as in [16, 24] and [27] currently lead the fashion of interactive segmentation. Another alterative way of performing interactive segmentation is the cost volume filtering [28]. In these methods, a data term is first computed from the Gaussian mixture model (GMM) or the color histogram. A filtering-like process is then carried out on the cost volume [28], or a cost function which incorporates the data term and the smoothness term for penalizing the boundary disagreement is minimized [27, 16]. However, minimizing an energy function which penalizes boundary disagreement is particularly prone to boundary shrinking. The cost volume filtering, on the other hand, does not always provide globally reliable results because of its locally optimizing characteristic. Moreover, the accuracy of these two methods highly depends on the data term. In our implementation, we do not restrict $n = 2$, and we also try to segment the image into more regions, i.e., $n = 3$. See Fig. 9 for typical foreground and background segmentation cases ($n = 2$), and Fig. 10 for $n = 3$ cases. The data term is computed using GMM as described in [16]. The method in [16] successively recomputes the GMM after each iteration, while this GMM model is fixed in other methods. For comparison fairness, we disabled the model re-computation step and used the results from one iteration. Our method provides better results than the other two methods and does not require computing the data terms.

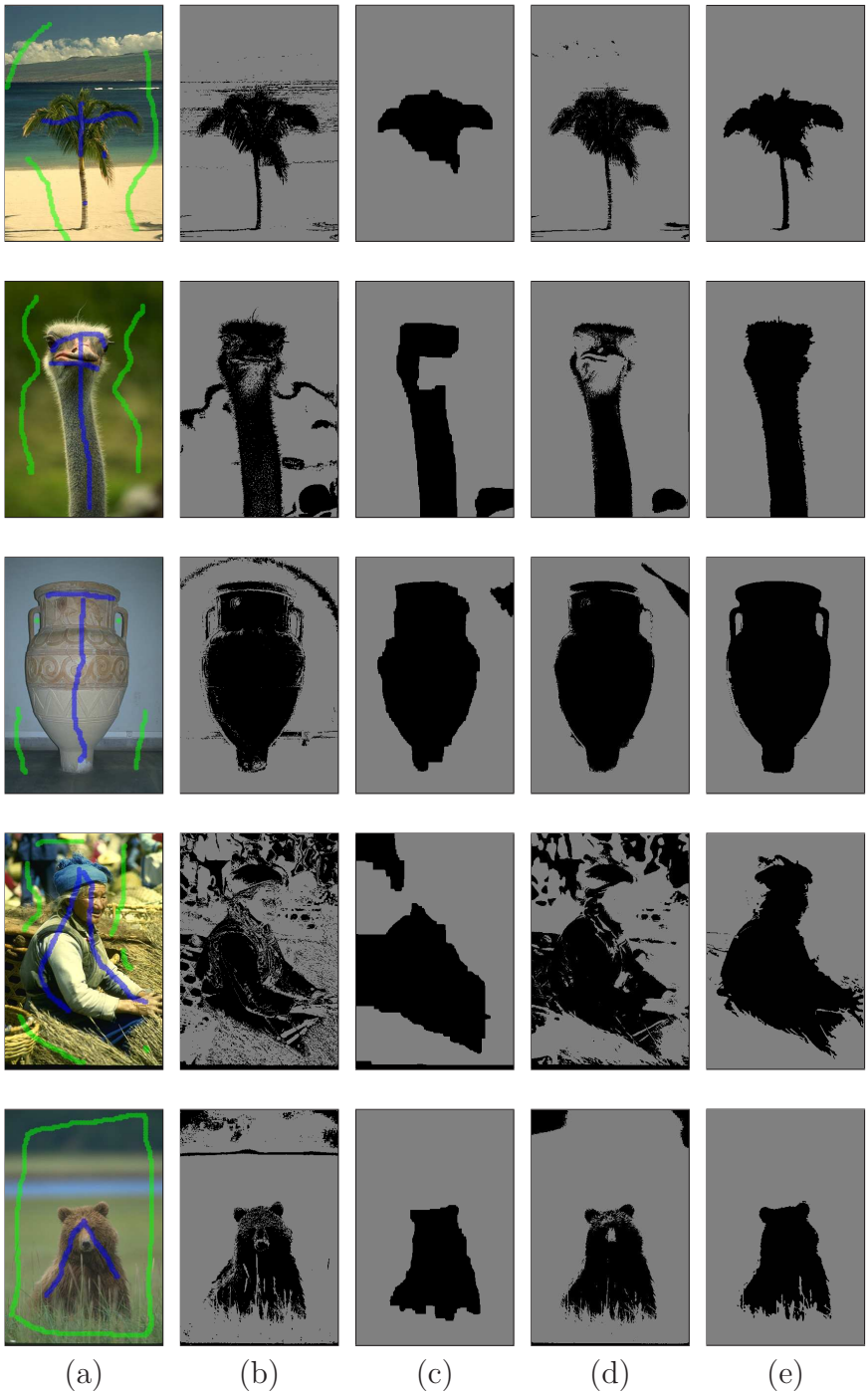


Figure 9: Results for foreground and background interactive segmentation. Figure courtesies from [29]. (a) Original image and strokes provided by users. (b) Results by applying winner-take-all method on the data term. (c) Results by using GraphCuts (one iteration of [16]). (d) Results by [28]. (e) Our results. Note that our method produces correct segmentation results and preserves object boundaries very well.

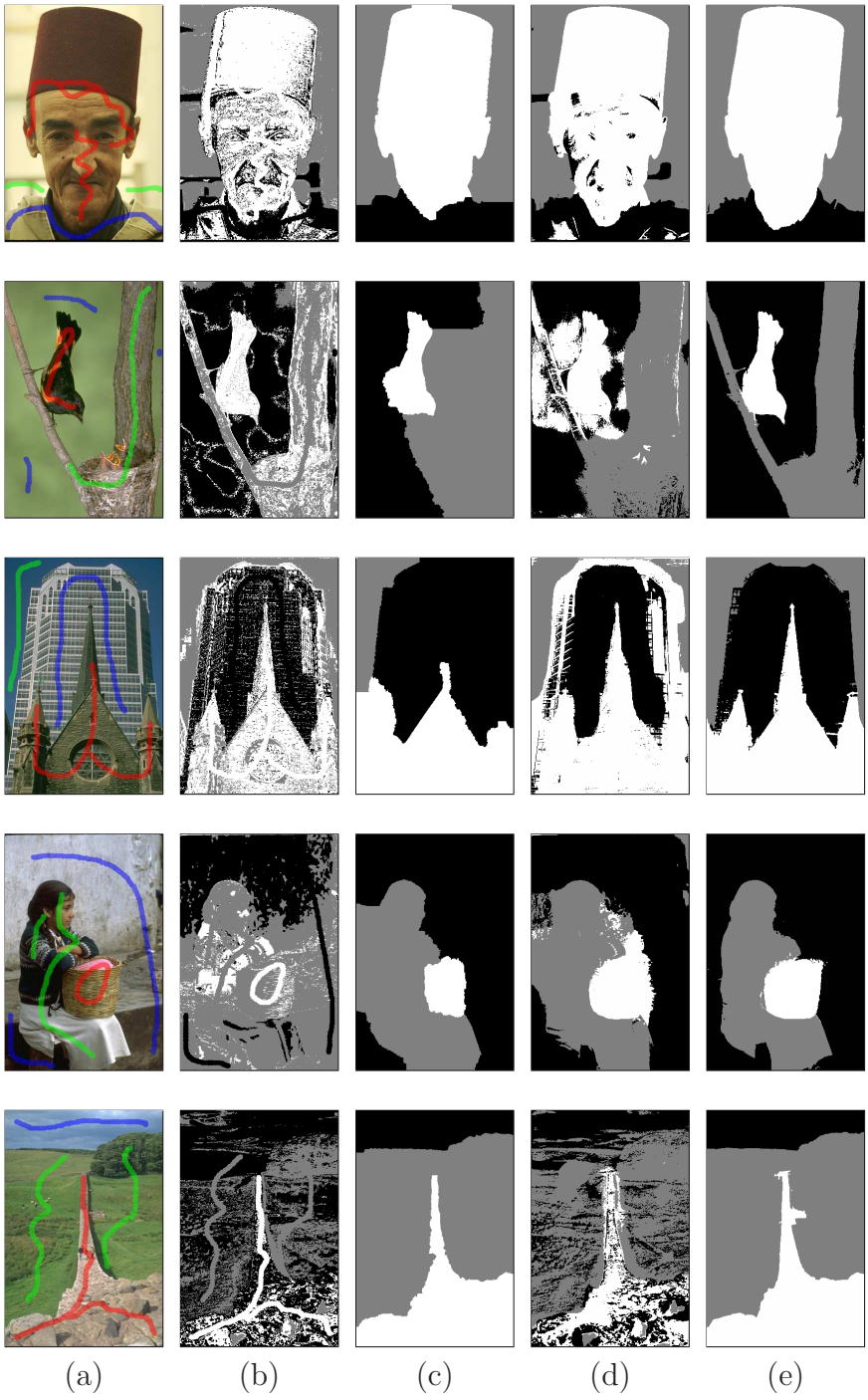


Figure 10: Comparison of interactive segmentation results for $n = 3$. Figure courtesies from [29]. (a) Original image and strokes provided by users. (b) Results by applying winner-take-all method on the data term. (c) Results by using GraphCuts (one iteration of [16]). (d) Results by [28]. (e) Our results. Note that our method produces correct segmentation results and preserves object boundaries very well.

3.4. Discussions

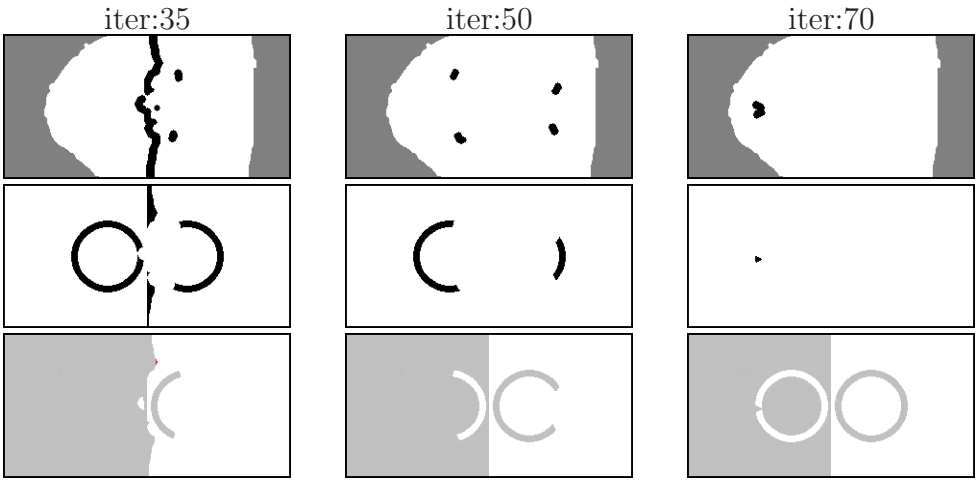
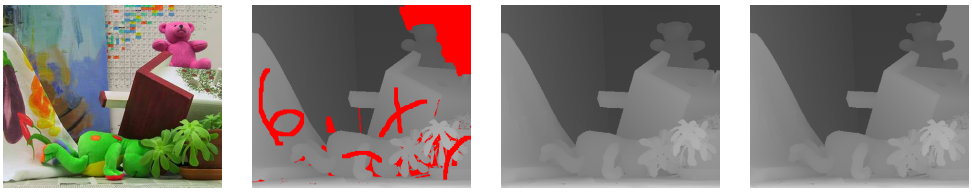


Figure 11: Confidence propagation on a synthetic image. From the left to the right are results in iteration: 35, 50, and 70. First row: pixels in Θ are colored in gray, pixels in $P \backslash \Theta$ are colored in white, and pixels in Ω^t (pixels to be updated) are colored in black. Second row: confidence maps where higher intensity indicates higher confidence. Last row: filling results where pixels that have not been visited until the current iteration are colored in red. The guidance image, input image for completion and the final results are given in Fig. 1.

We have demonstrated the advantage of our method with several experiments. To better understand why our method performs well and also to show the effectiveness of confidence propagation, an experiment on a synthetic image with 30000 pixels being filled carried out (see Fig. 1). The intermediate result, the confidence map, and the pixels to be updated in each iteration are given in Fig. 11. We noticed a distinguishing property of our guided image completion: it performs very well in regions which correspond to interleaving homogenous regions of the guidance image. The reason is as follows. First, recall that the confidence value of a pixel is updated according to the confidence value of a neighboring pixel which is filled with a similar value and has the maximum confidence value (Eq. (9)). This enforces confidence propagating along the path where pixels are filled with similar values. Second, the confidence value increases with the increase of the color similarity between two neighboring pixels in the guidance image. As a result, it is expected that pixels with a high confidence value, are those whose values are similar not only in the guidance image I but also in the completion results. Therefore, an



(a) (b) (c) (d)

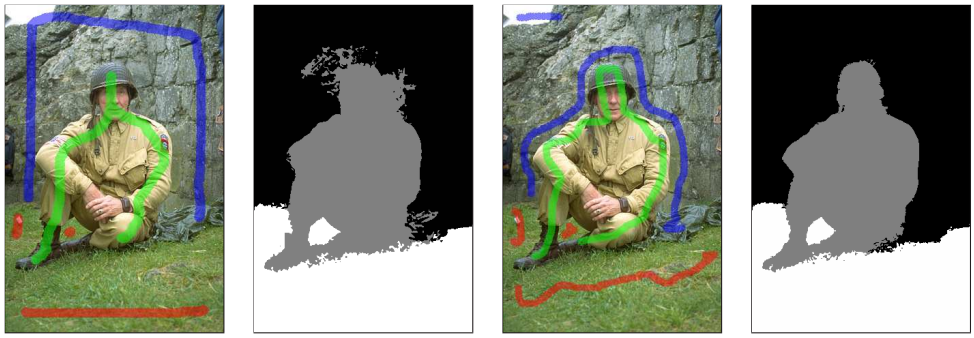
Figure 12: Effect of parameter γ. (a) Guidance image. (b) Input depth map for completion (red regions are missing). (c) Results by $\gamma = 7.0$, $r = 9$, $\sigma_r = 0.1$, and $\sigma_s = 11$. (d) Results by using $\gamma = 1.0$. Note that large γ is robust to noise-like textures on the background wall.

incorrectly filled pixel whose confidence value is usually low will be corrected as the confidence is propagated from pixels with high confidence values. For example, two circles in Fig. 11 which are initially filled by incorrect values will be corrected when high confidence values propagate into pixels of these circles.

Our method took 10 seconds on the test image in Fig. 1, which is lower than the method of solving a sparse linear system (1.2 seconds) and sequentially filling method (3 seconds). However, our method fills values in a parallel manner which is ideal for execution on parallel architectures compared with the other two methods.

Our method has 4 parameters: σ_r, σ_s, r, and γ. The effects of σ_r and σ_s are similar to that in bilateral filtering. Large values for σ_r and σ_s make results smooth but may lead to ambiguity around boundaries, whereas small values yield clear boundaries but may cause noise like isolated regions in the results. Parameter r decides the search range of homogenous regions. For example, if the two circles in Fig. 1 are further away from the boundary, r should be increased to cover the distance between circles and boundaries so that the circles can be filled correctly. A large γ helps to find strong boundaries globally and prevents the algorithm from being trapped by weak boundaries or noises (see Fig. 12).

One limitation is with the situation of filling the same value to regions with homogenous texture rather than with homogenous color in the guidance image. An example is shown in Fig. 13.



(a) (b) (c) (d)

Figure 13: Limitations. (a) Guidance image and strokes. Different objects are discriminated by texture rather than color values around the head of the solider. (b) Fuzzy boundaries will be obtained around the head of the solider. (c) Moving strokes closer to desired boundary improves the results. (d) Improved results.

4. Conclusion and Future Work

This paper presented a novel method for addressing a generic guided image completion problem which is widely used in computer vision and graphics fields. The proposed method introduces confidence to evaluate completion results and carries out the completion process via updating the confidence. Our method has the property of preserving local details and filling values globally. That is, pixels within homogenous regions in the guidance image are filled with similar values even when they are far from each other. In addition, the process of our method is carried out on individual pixels independently in each iteration. This attribute makes our method ideal for execution on parallel architectures. Experimental results have demonstrated that our method outperforms state-of-the-art methods in many applications.

Since the proposed guided image completion is a type of edit propagation method, we would like to apply advanced patch based model into confidence propagation to achieve texture based image completion in future studies. We also would like to incorporate current studies on constant filtering [30, 31, 14] to reduce the complexity of filtering-like filling process from $O(|\mathcal{N}|)$ to $O(1)$.

References

[1] M. Bertalmio, G. Sapiro, V. Caselles, C. Ballester, Image inpainting, in: Proceedings of the 27th Annual Conference on Computer Graphics and

Interactive Techniques, 2000, pp. 417–424.

[2] M. Bertalmio, A. L. Bertozzi, G. Sapiro, Navier-Stokes, fluid dynamics, and image and video inpainting, in: IEEE Conference on Computer Vision and Pattern Recognition, Vol. 1, 2001, pp. 355–362.

[3] A. Telea, An image inpainting technique based on the fast marching method, Journal of Graphics Tools 9 (1) (2004) 23–34.

[4] X. Gong, J. Liu, W. Zhou, J. Liu, Guided depth enhancement via a fast marching method, Image and Vision Computing 31 (10) (2013) 695–703.

[5] M. Bertalmio, L. Vese, G. Sapiro, S. Osher, Simultaneous structure and texture image inpainting, IEEE Transactions on Image Processing 12 (8) (2003) 882–889.

[6] R. Bornard, E. Lecan, L. Laborelli, J.-H. Chenot, Missing data correction in still images and image sequences, in: Proceedings of the 10th ACM International Conference on Multimedia, 2002, pp. 355–361.

[7] L. Wang, H. Jin, R. Yang, M. Gong, Stereoscopic inpainting: Joint color and depth completion from stereo images, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.

[8] Z. Xu, J. Sun, Image inpainting by patch propagation using patch sparsity, IEEE Transactions on Image Processing 19 (5) (2010) 1153–1165.

[9] D. Doria, R. J. Radke, Filling large holes in LiDAR data by inpainting depth gradients, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2012, pp. 65–72.

[10] A. Levin, D. Lischinski, Y. Weiss, Colorization using optimization, ACM Transactions on Graphics 23 (3) (2004) 689–694.

[11] A. Levin, D. Lischinski, Y. Weiss, A closed-form solution to natural image matting, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (2) (2008) 228–242.

[12] L. Wang, R. Yang, Global stereo matching leveraged by sparse ground control points, in: IEEE Conference on Computer Vision and Pattern Recognition, 2011, pp. 3033–3040.

[13] A. Torralba, W. T. Freeman, Properties and applications of shape recipes., in: IEEE Conference on Computer Vision and Pattern Recognition, 2003, pp. 383–390.

[14] K. He, J. Sun, X. Tang, Guided image filtering, in: European Conference on Computer Vision, 2010, pp. 1–14.

[15] L. Yatziv, G. Sapiro, Fast image and video colorization using chrominance blending, IEEE Transactions on Image Processing 15 (5) (2006) 1120–1129.

[16] C. Rother, V. Kolmogorov, A. Blake, GrabCut: Interactive foreground extraction using iterated graph cuts, ACM Transactions on Graphics 23 (3) (2004) 309–314.

[17] C. Tomasi, R. Manduchi, Bilateral filtering for gray and color images, in: IEEE International Conference on Computer Vision, 1998, pp. 839–846.

[18] J. J. Helmsen, E. G. Puckett, P. Colella, M. Dorr, Two new methods for simulating photolithography development in 3D, in: SPIE's 1996 International Symposium on Microlithography, International Society for Optics and Photonics, 1996, pp. 253–261.

[19] J. A. Sethian, A fast marching level set method for monotonically advancing fronts, Proceedings of the National Academy of Sciences 93 (4) (1996) 1591–1595.

[20] J. N. Tsitsiklis, Efficient algorithms for globally optimal trajectories, Automatic Control, IEEE Transactions on 40 (9) (1995) 1528–1538.

[21] A. Criminisi, P. Perez, K. Toyama, Object removal by exemplar-based inpainting, in: IEEE Conference on Computer Vision and Pattern Recognition, Vol. 2, 2003, pp. 721–728.

[22] Z. Ma, K. He, Y. Wei, J. Sun, E. Wu, Constant time weighted median filtering for stereo matching and beyond, in: IEEE International Conference on Computer Vision, 2013, pp. 49–56.

[23] http://www.vision.middlebury.edu/stereo/ (2013).

[24] Y. Boykov, G. Funka-Lea, Graph cuts and efficient ND image segmentation, IJCV 70 (2) (2006) 109–131.

[25] J. Van de Weijer, R. Van den Boomgaard, Local mode filtering, in: IEEE Conference on Computer Vision and Pattern Recognition, Vol. 2, 2001, pp. 428–433.

[26] D. Min, J. Lu, M. N. Do, Depth video enhancement based on weighted mode filtering, IEEE Transactions on Image Processing 21 (3) (2012) 1176–1190.

[27] Y. Boykov, M.-P. Jolly, Interactive graph cuts for optimal boundary & region segmentation of objects in ND images, in: IEEE International Conference on Computer Vision, Vol. 1, 2001, pp. 105–112.

[28] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, M. Gelautz, Fast cost-volume filtering for visual correspondence and beyond, in: IEEE Conference on Computer Vision and Pattern Recognition, 2011, pp. 3017–3024.

[29] P. Arbelaez, M. Maire, C. Fowlkes, J. Malik, Contour detection and hierarchical image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (5) (2011) 898–916.

[30] F. Porikli, Constant time O(1) bilateral filtering, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.

[31] Q. Yang, K.-H. Tan, N. Ahuja, Real-time O(1) bilateral filtering, in: IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 557–564.