IPSJ Transactions on Computer Vision and Applications Vol. 2 235–252 (Dec. 2010)

Research Paper

Interactive Removal of Shadows from a Single Image Using Hierarchical Graph Cut

Daisuke Miyazaki,[†1,∗1] Yasuyuki Matsushita[†2]
and Katsushi Ikeuchi[†1]

We propose a method for extracting a shadow matte from a single image. The removal of shadows from a single image is a difficult problem to solve unless additional information is available. We use user-supplied hints to solve the problem. The proposed method estimates a fractional shadow matte using a graph cut energy minimization approach. We present a new hierarchical graph cut algorithm that efficiently solves the multi-labeling problems, allowing our approach to run at interactive speeds. The effectiveness of the proposed shadow removal method is demonstrated using various natural images, including aerial photographs.

1. Introduction

Shadows in an image reduce the reliability of many computer vision algorithms, such as shape-from-X, image segmentation, object recognition and tracking. Also, shadows often degrade the visual quality of the images, e.g., causing inconsistencies in a stitched aerial photograph. Shadow removal is therefore an important pre-processing step for computer vision algorithms and image enhancement.

Decomposition of a single image into a shadow image and a shadow-free image is a difficult problem to solve unless additional prior knowledge is available. Although various types of prior information have been used in previous approaches, the task of shadow removal remains challenging. Because the previous techniques do not use a feedback loop to control the output, it has not been possible to refine the output in the intended manner. As a result, it is still a time-consuming task to remove the shadows, especially from the more difficult examples. To address this problem, we developed an efficient computation method for the shadow removal task. Unlike the previous shadow removal methods, our method allows the user to interactively and incrementally refine the results. The interaction speed is achieved by using a new formulation for shadow removal in a discrete optimization framework.

The chief contributions of this paper are as follows:

MRF formulation for shadow removal We, like Nielsen and Madsen [1], formulated the problem of shadow matte computation in a Markov random field (MRF) framework. Unlike their approach, we used the user-supplied hints fully as prior information, while discrete optimization techniques can find the best solution using the prior information.

Hierarchical graph cut To achieve the interactive speed, we developed a hierarchical optimization method for the multi-labeling problem. The method produces a sub-optimal solution, and the number of applying max-flow/min-cut algorithm for each iteration is $2\log_2 n$, while that of the α-expansion [2] is n. Here, n represents the number of labels.

Interactive optimization Our system interactively and incrementally updates the estimates of the shadow matte. The estimates are refined by the user via a stroke-based user interface.

We validated the effectiveness of our technique quantitatively and qualitatively using various different input images.

1.1 Prior Work

Shadow removal algorithms can be categorized into two classes: multiple-image and single-image methods. Weiss [3] proposed a multiple-image method for decomposing an input image sequence into multiple illumination images and a single reflectance image. This method was extended by Matsushita, et al. [4] to produce multiple illumination images and multiple reflectance images. These methods require several images taken from a fixed viewpoint, which limits their application.

Both automatic and interactive techniques have been proposed for the removal of shadows from a single image. Finlayson, et al. [5] presented an automatic

†1 Institute of Industrial Science, the University of Tokyo
†2 Microsoft Research Asia
∗1 Presently with Hiroshima City University
This work was done while the first author was a visiting researcher at Microsoft Research Asia.

method that detects the shadow edge by entropy minimization. Fredembach and Finlayson [6] extended that method to improve the computational efficiency. These two methods aim to detect the shadow edges using physics-based methods, but they require the illumination chromaticity of the shadow region to be different to that of the non-shadow region. On the other hand, Tappen, et al. [7] took a learning-based approach by creating a database of edge images to determine the shadow edges robustly.

Instead of using the edge information, other works use the brightness information. Baba, et al. [8] estimated gradually changing shadow opacities, assuming that the scene does not contain complex textures. Conversely, the method proposed by Arbel and Hel-Or [9] can handle scenes with complex textures, but it does not handle gradual changes in the shadow opacity. Nielsen and Madsen [1] proposed a method that can estimate gradually changing shadow opacities from complex textured images. However, their method remains limited due to the simple thresholding method used to detect the shadow edges.

Recently, interactive methods have been gaining attention, enabling the user to supply hints to the system to remove shadows from difficult examples. Wu and Tang's method [10),11] removes shadows when given user-specified shadow and non-shadow regions. It adopts a continuous optimization method that requires many iterations to converge. As a result, it is not straightforward to use their method in an interactive and incremental manner. Our method solves this problem by formulating the problem in an MRF framework.

1.2 Overview of Our Approach

The overview of our shadow removal method is illustrated in **Fig. 1**. First, we automatically over-segment the input image to produce a set of super-pixels. In the next stage, *region segmentation stage*, the user specifies the shadow, non-shadow, and background regions using a stroke-based interface such as Lazy Snapping [12], and the regions are segmented by the GrabCut algorithm [13]. Using the likelihood of the non-shadow region as prior information, our method removes the shadows by representing them as a multi-label Markov random field (MRF). The shadow removal is performed using a hierarchical graph cut algorithm, and the output is shown to the user at a responsive speed. The default parameters of the hierarchical graph cut are used at this *initial removal stage*. To further improve the results, the user can specify areas where the shadow was not perfectly removed. The parameters of the hierarchical graph cut algorithm are updated by additional user interaction at this *interactive refinement stage*, and the improved output is displayed to the user rapidly.

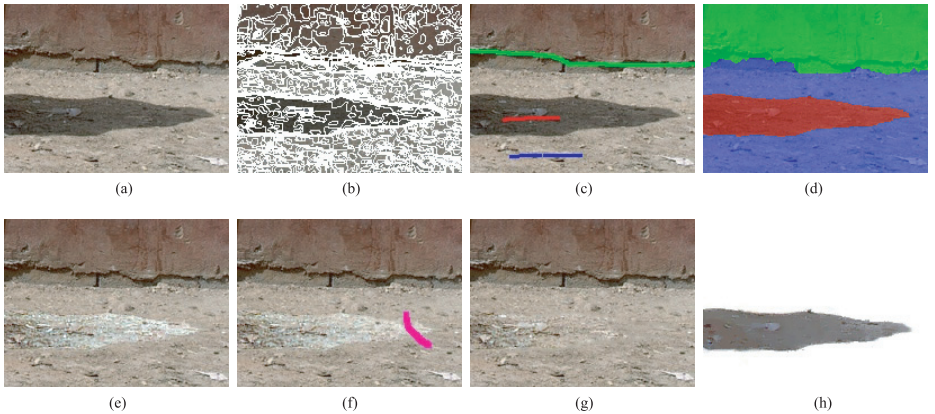


Fig. 1 Illustration of the shadow removal process. (a) The input is a single image. (b) The image is segmented into small super-pixels that are used in steps (c) and (d). (c) The user draws strokes to specify the shadow, non-shadow, and background regions. We denote the region where the shadow appears in the input image as shadow region, and the region which has no shadow and has the same texture as the shadow region as non-shadow region. The background region is the area where our shadow removal software is not applied. (d) The shadow, non-shadow, and background regions after region segmentation stage. (e) Our graph cut shadow removal algorithm is applied to the image using default parameters. (f) The user specifies any defective areas in the results from step (d), and the graph cut shadow removal algorithm recalculates a shadow-free image using the updated parameters. We denote the image where the shadow is removed as shadow-free image, and the image which represents the shadow opacity as shadow image. (g) The resulting shadow-free image. (h) The resulting shadow image.

2. MRF Formulation of Shadow Removal

This section describes our graph cut shadow removal algorithm. We begin with the image formation model of Barrow and Tenenbaum [14]. The input image I can be expressed as a product of two intrinsic images, the reflectance image R and the illumination image L, as

$$I = RL. \tag{1}$$

The illumination image L encapsulates the effects of illumination, shading, and shadows. We can further decompose the illumination image into $L = \beta L'$, where β represents the shadow image, defined as a function of the shadow opacity, and L' represents the other factors of L. Hence, Eq. (1) can be written as $I = \beta RL'$, or more simply,

$$I = \beta F, \tag{2}$$

as in Wu and Tang [10]. Here, F represents the shadow-free image. β and F are interdependent, i.e., if we know β, we also know F. Therefore, our problem is the estimation of β from the input image.

We designed an energy function characterized by four properties:

(1) the likelihood of the color histogram (D^t);
(2) the likelihood of the umbra (D^u);
(3) the smoothness of the shadow-free image F (D^f); and
(4) the smoothness of the shadow image β (V^b).

The total energy $E(\beta)$ of all nodes $\mathcal{P}$ and edges $\mathcal{N}$ are represented as follows:

$$E(\beta) = \sum_{p\in\mathcal{P}} \left\{\lambda_t D_p^t(\beta_p) + \lambda_u D_p^u(\beta_p) + \lambda_f D_p^f(\beta_p)\right\} + \sum_{\{p,q\}\in\mathcal{N}} \lambda_b V_{p,q}^b(\beta_p, \beta_q), \tag{3}$$

where the parameters λ_t, λ_u, λ_f, and λ_b are the weight factors of the corresponding cost variables. p and q represent the node indices. The weight of each term in Eq. (3) is automatically determined by the interactive parameter optimization described in Section 3.

The likelihood cost of the color histogram D^t is related to the probability density function (pdf) of the non-shadow region. Assuming that the likelihood P of the non-shadow region is the same as the likelihood of the shadow-free image F, the cost function can be formulated as

$$D_p^t(\beta_p) = -\log P(I_p/\beta_p), \qquad p \in \mathcal{P}, \tag{4}$$

where I/β represents F. We represent the pdf P as a 1D histogram for each 1D color channel. We do not estimate all three of the color channels using a 3D pdf, since the solution space will be too large (e.g., 64 labels for 1D pdf and $64 \times 64 \times 64$ labels for 3D pdf).

Dividing the average intensity of the non-shadow region by the average intensity of the shadow region yields a good initial estimate for β, which we denote as β_0. At the initial removal stage, we, like Wu and Tang [10], use β_0 as an initial value for β. The inner part of the shadow region (the *umbra*) has a value which is close to β_0, while the shadow boundary (the *penumbra*) varies from β_0 to 1, with 1 representing a non-shadow region. In order to express the above characteristics, we introduce the following cost function for D^u:

$$D_p^u(\beta_p) = |\beta_p - \beta_0|^{0.7} + |\beta_p - 1|^{0.7}, \qquad p \in \mathcal{P}. \tag{5}$$

β_0 is a constant value, and is not changed throughout the interactive refinement stage. L0.7-norm is useful for separating two types of information [15], and we also use it here to decompose the input image into the shadow and shadow-free images.

We also employ a smoothness term for the shadow image β. L1-norm is a good estimator in order to avoid outliers [16]; thus we set the smoothness term of the shadow image β as follows:

$$V_{p,q}^b(\beta_p, \beta_q) = |\beta_p - \beta_q|, \qquad \{p, q\} \in \mathcal{N}. \tag{6}$$

Although the smoothness term defined in Eq. (6) can be solved using either the α-expansion or Ishikawa's method [17], we solve it using a hierarchical graph cut in order to reduce the computation time.

We also set the smoothness term of the shadow-free image. Our hierarchical graph cut assumes that the smoothness term is represented by L1-norm, but because $F = I/\beta$, we cannot represent the smoothness term by L1-norm. We therefore calculate the smoothness term of the shadow-free image as follows and add it to the data term:

$$D_p^f(\beta_p) = \left| I_p/\beta_p - \overline{I_p/\beta_p} \right|^2, \qquad p \in \mathcal{P}. \tag{7}$$

The term D^f represents the blurred shadow-free image. The value $\overline{I_p/\beta_p}$ is the value $F_p = I_p/\beta_p$ which is blurred befor each iteration: (1) Before starting each iteration, we blur the shadow-free image calculated in previous iteration; (2) for $D_p^f(\beta_p)$, we use the difference between the shadow-free image calculated from β_p

(a) User input (b) Region segmentation (c) Over-segmentation

Fig. 2 Results of image segmentation. (a) Shadow, non-shadow, and background regions specified by red, blue, and green strokes drawn by the user. (b) The image segmented into shadow, non-shadow, and background regions represented as red, blue, and green regions. (c) The image segmented into small super-pixels.

and the blurred shadow-free image calculated in step (1); (3) after applying the graph cut to Eq. (3) for one iteration, we go back to step (1) if the graph cut is not converged.

2.1 Image Segmentation

Region segmentation

The cost functions D^t and D^u in Eq. (3) are calculated from the shadow and non-shadow regions. In order to construct the likelihoods, we have to segment the image into shadow and non-shadow regions before applying our shadow removal algorithm. Like previous image segmentation methods [12),13)], we ask the user to mark each region using a stroke-based interface, as shown in **Fig. 2** (a). After that, we separate the image into three regions: shadow, non-shadow, and background. The background region is not used in further calculations. The segmented region shown in Fig. 2 (b) is obtained by using the α-expansion algorithm. The boundary of the background region should be specified clearly so that this region would not be updated. However, the boundary between the shadow region and the non-shadow region does not need to be clearly segmented because the software can revise the results.

Over-segmentation

To accelerate the region segmentation stage, we segment the image into small super-pixels in the over-segmentation stage, as shown in Fig. 2 (c). In this stage, we reduce the colors of the image by k-means clustering in the RGB color space, and then connect neighboring pixels which have the same color into one small super-pixel. These super-pixels are also used in the initial removal stage. The boundary of each super-pixel sometimes produces defects, so we apply our graph cut shadow removal algorithm pixel by pixel in the interactive removal stage.

3. Interactive Parameter Optimization

In this section, we explain how to interactively update the weighting parameters λ introduced in Eq. (3). In the initial removal stage, we apply our graph cut shadow removal algorithm (Section 2) with the default weighting parameters. The default parameters $\lambda_t = 10$, $\lambda_u = 1$, $\lambda_f = 0.01$, and $\lambda_b = 1$, however, are not always optimal. Therefore, we use the user input to find a better parameter set.

The main concept of the interactive parameter optimization algorithm is represented in the following formula.

$$\hat{\mathbf{\Lambda}} = \underset{\mathbf{\Lambda}}{\operatorname{argmin}} \sum_{p \in \Omega_0} |\hat{\beta}_p - \beta_c|^2, \quad s.t. \{\hat{\beta}_p | p \in \Omega_0\} = \text{graph\_cut}(\beta_p | p \in \Omega_0; \mathbf{\Lambda}), \tag{8}$$

where $\mathbf{\Lambda} \equiv \{\lambda_t, \lambda_u, \lambda_f, \lambda_b\}$. The system automatically updates the parameters $\mathbf{\Lambda}$ so that the shadow image β_p of pixel p will be close to the user-specified constant β_c. The constant value β_c of pixel p is specified from the starting point of the stroke input by the user. The area Ω_0 to be examined is specified by the user. Here, graph_cut is the function to solve Eq. (3) using the parameters $\mathbf{\Lambda}$, the current shadow image β, and the hierarchical graph cut explained in Section 4. By considering the trade-off between the precision and the computation speed, we limited the iterations of Eq. (8) to 4: We explain the detailed implementation in Section 3.2. Eq. (8) represents the case for the shadow image β, and the case for the shadow-free image F is similar. The system increases λ_t and λ_f for smooth textures, and increases λ_u and λ_b for constant shadow areas (**Fig. 3**). For example, if λ_f is large, the image will be blurred due to the term D^f, which is effective to remove the shadow when the image contains the scene with constant color. The discussion about the formulation of Eq. (8) is provided in Section 3.1, and the actual implementation of it is shown in Section 3.2.

3.1 Objective Function

In our implementation, we use Eq. (8) for the interactive parameter optimiza-

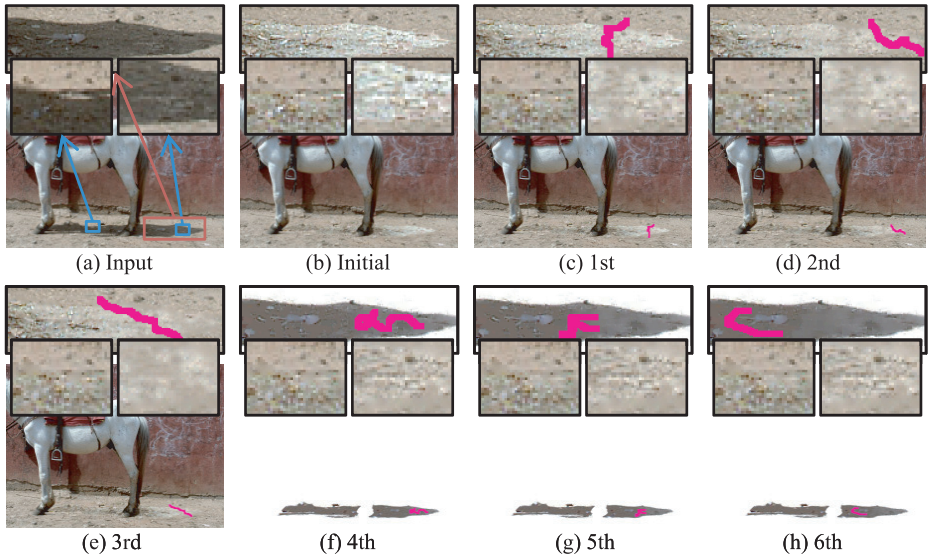


Fig. 3 The image enhanced by user-specified strokes. According to the user's strokes, the algorithm automatically finds the parameters which reflect the user's intentions. The strokes are represented by magenta pixels. In this example, 1–3rd strokes are added to the shadow-free image and 4–6th strokes are added to the shadow image. (a) is the input image, (b) is the initial state, and (c)–(h) are the results after the 1–6th strokes.

tion. In this section, we discuss the validity of Eq. (8). In Eq. (8), the term $\sum |\hat{\beta}_p - \beta_c|^2$ represents the constraint given by the user stroke. The software tries to make the shadow opacity β to be close to the value β_c specified by the user. The function graph_cut represents the minimization of the cost function, Eq. (3). Consequently, Eq. (8) tries to minimize both the Eq. (3) and the term $\sum |\hat{\beta}_p - \beta_c|^2$ in order to obtain the optimal value for the shadow opacity β and the parameters λ_t, λ_u, λ_f, and λ_b.

Eq. (8) can be expressed in other form as follows.

$$\begin{aligned} \hat{\mathbf{\Lambda}} &= \operatorname*{argmin}_{\mathbf{\Lambda}} U(\hat{\beta}) \,, \\ \{\hat{\beta}_p | p \in \Omega_0\} &= \operatorname*{argmin}_{\{\beta_p | p \in \Omega_0\}} E(\mathbf{\Lambda}, \beta) \,, \end{aligned} \tag{9}$$

or more simply,

$$\begin{aligned} \hat{\mathbf{\Lambda}} &= \operatorname*{argmin}_{\mathbf{\Lambda}} U(\hat{\beta}(\mathbf{\Lambda})) \,, \\ \text{where} \quad \hat{\beta}(\mathbf{\Lambda}) &= \operatorname*{argmin}_{\beta} E(\mathbf{\Lambda}, \beta) \,. \end{aligned} \tag{10}$$

Here, $E(\mathbf{\Lambda}, \beta) \equiv E(\mathbf{\Lambda}, \{\beta_p | p \in \Omega_0\})$ represents the cost function, Eq. (3), and $U(\hat{\beta}) \equiv U(\{\hat{\beta}_p | p \in \Omega_0\})$ represents the constraint $U(\hat{\beta}) = \sum_{p \in \Omega_0} |\hat{\beta}_p - \beta_c|^2$ given by the user. The variable $\mathbf{\Lambda}$ is a global parameter which affects the $|\Omega_0|$ number of the shadow opacity $\hat{\beta}_p$, while the variable $\hat{\beta}_p$ is a local parameter set for each pixel. Since the nature of these parameters are quite different, optimizing them at the same time does not work stably. In order to solve the problem stably, the algorithm is represented by the nested loop as Eqs. (8), (9), and (10). The outer loop minimizes U with respect to $\mathbf{\Lambda}$, and the inner loop minimizes E with respect to β. The inner loop minimizes E with the graph cut; thus, we can use the same source code used in the initial removal stage. The outer loop does not contain E, namely, the outer loop does not require the graph cut; thus, the computation time is reduced. The solution to $\operatorname{argmin}_{\mathbf{\Lambda}} E$ is $\lambda_t = 0$, $\lambda_u = 0$, $\lambda_f = 0$, and $\lambda_b = 0$ (c.f., Eq. (3)), and these values are not the values that we want. Note that we want the parameter $\mathbf{\Lambda}$ which satisfies the user requirement, U. The solution to $\operatorname{argmin}_{\beta} U$ is $\beta_p = \beta_c$, (c.f., Eq. (8)), and this value is not the value that we want. Note that we want the shadow opacity which minimizes the cost function of the shadow removal problem, E, namely, Eq. (3).

3.2 Detailed Implementation

The detailed algorithm of the interactive parameter optimization algorithm is described in **Fig. 4**. The behavior of the algorithm is illustrated in **Fig. 5**. The first half of the algorithm is to estimate the parameter $\mathbf{\Lambda}$ (c.f., line 3–7 in Fig. 4 and Fig. 5 (a)–(b)). The user specifies some strokes as is shown in Fig. 3, and the system automatically optimize the parameters which are consistent to the user's intention. The user can draw strokes to both the shadow image β and the shadow-free image F; however, we only explain the case where the user drew strokes to the shadow image β here. The second half of the algorithm is to apply the graph cut shadow removal algorithm to other area using the updated parameters (c.f., line 8–12 in Fig. 4 and Fig. 5 (b)–(c)). The system automatically select some regions, and applies the graph cut shadow removal algorithm.

1: $\{\beta_p | p \in \mathcal{P}\} \Leftarrow$ initial value
2: $\mathbf{\Lambda} \equiv \{\lambda_t, \lambda_u, \lambda_f, \lambda_b\} \Leftarrow$ initial value
3: $\Omega_0 \Leftarrow$ user specified area
4: $\beta_c \Leftarrow$ user specified β // algorithm for F is also similar
5: $\{\beta_p | p \in \Omega_0\} \Leftarrow \beta_c$
6: $\Omega_0 \Leftarrow \Omega_0 \cup$ surrounding 4px-width area
7: $\hat{\mathbf{\Lambda}} \Leftarrow \mathrm{argmin}_{\mathbf{\Lambda}} \Sigma_{p \in \Omega_0} |\hat{\beta}_p - \beta_c|^2$
s.t. $\{\hat{\beta}_p | p \in \Omega_0\} \Leftarrow \mathrm{graph\_cut}(\beta_p | p \in \Omega_0; \mathbf{\Lambda})$
8: $\{\beta_a, \beta_b\} \Leftarrow \{\beta_p, \hat{\beta}_p\}$ s.t. $|\beta_p - \hat{\beta}_p| > T_0$, $p \in \Omega_0$
// T_0 is adjusted if $\{\beta_a, \beta_b\}$ is not found
9: $\Omega_1 \Leftarrow \{p\}$ s.t. $|\beta_p - \beta_a| < T_1$, $p \in \mathcal{P}$
// T_1 is adjusted if $|\Omega_1|$ is too big or too small
10: $\{\hat{\beta}_p | p \in \Omega_1\} \Leftarrow \beta_b$
11: $\Omega_1 \Leftarrow \Omega_1 \cup$ surrounding 4px-width area
12: $\{\beta_p | p \in \Omega_1\} \Leftarrow \mathrm{graph\_cut}(\hat{\beta}_p | p \in \Omega_1; \hat{\mathbf{\Lambda}})$
13: $\mathbf{\Lambda} \Leftarrow \hat{\mathbf{\Lambda}}$
14: **if** user is still unsatisfied **then** goto 3

Fig. 4 Interactive parameter optimization.

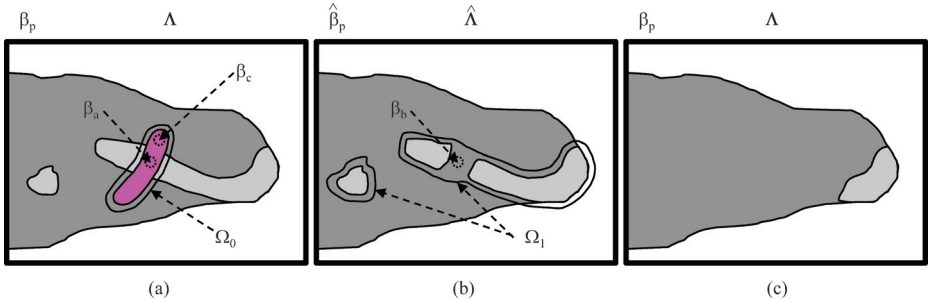


Fig. 5 A sketch of the interactive parameter optimization. (a) This illustration represents the shadow image β_p. Magenta stroke represents the user specified area. The shadow opacity of the starting point of the stroke is represented by β_c. The area for parameter estimation is represented by Ω_0. After the parameter optimization, the parameters $\mathbf{\Lambda}$ becomes $\hat{\mathbf{\Lambda}}$, and the shadow opacity β_p becomes $\hat{\beta}_p$. At a certain point randomly chosen from Ω_0, the shadow opacity β_a changed to β_b. (b) The area where shadow opacity β_p is close to β_a is represented by Ω_1. (c) This illustration represents the result after applying graph cut shadow removal algorithm to the area Ω_1 with the updated parameters $\hat{\mathbf{\Lambda}}$.

Parameter optimization

The first half of the algorithm is to optimize the parameter $\mathbf{\Lambda}$ under the constraint specified by the user (Fig. 5 (a)–(b)).

Fig. 4 line 3 The user draws strokes to specify incorrectly removed areas. The region specified by the user is indicated by Ω_0.

Fig. 4 line 4 We represent the shadow opacity of the starting point of the stroke as β_c. The purpose of the process in this section is to estimate the parameter $\mathbf{\Lambda}$ so that the shadow opacity will be as close as possible to β_c.

Fig. 4 line 5 Before applying the graph cut algorithm, we set the initial value of the shadow opacity. The initial value is set to be β_c for the area Ω_0 specified by the user.

Fig. 4 line 6 The algorithm is also applied to a region surrounding the area Ω_0 specified by the user. The region Ω_0 is expanded by 4 pixels width. The value "4" is determined empirically. The shadow opacity of the surrounding area is considered to be correct while it is not considered to be correct in the user specified area. The shadow opacity of the surrounding area works as a soft constraint.

Fig. 4 line 7 We apply our method with 4 different parameter sets that are randomly modified from the current parameters $\mathbf{\Lambda}$. We compare the results $\hat{\beta}_p$ obtained using the 4 parameter sets and choose the result which is closest to the starting point β_c of the stroke. In other words, we first randomly make 4 different parameter sets, $\mathbf{\Lambda}_1$, $\mathbf{\Lambda}_2$, $\mathbf{\Lambda}_3$, and $\mathbf{\Lambda}_4$, which is slightly different from the current parameters $\mathbf{\Lambda}$. Next, we solve the graph cut and obtain the shadow image $\hat{\beta}_{1p}$, $\hat{\beta}_{2p}$, $\hat{\beta}_{3p}$, and $\hat{\beta}_{4p}$ for the parameters $\mathbf{\Lambda}_1$, $\mathbf{\Lambda}_2$, $\mathbf{\Lambda}_3$, and $\mathbf{\Lambda}_4$, respectively. Then, we find the minimum among the following 4 values, $\sum_{p \in \Omega_0} |\hat{\beta}_{1p} - \beta_c|^2$, $\sum_{p \in \Omega_0} |\hat{\beta}_{2p} - \beta_c|^2$, $\sum_{p \in \Omega_0} |\hat{\beta}_{3p} - \beta_c|^2$, and $\sum_{p \in \Omega_0} |\hat{\beta}_{4p} - \beta_c|^2$, and choose the parameter set $\hat{\mathbf{\Lambda}}$ which makes this value minimum. This process is represented in Eq. (8). The value "4" is determined empirically by considering the trade-off between the precision and the computation speed. We do not change the parameters to the steepest descent but change the

parameters randomly in order to limit the number of computing the cost function. The convergence of the algorithm is slow but it will converge close to the minimum and slightly oscillate around the minimum. The global minimum of Eq. (8) is the shadow opacity β_c specified by the user; thus, we do not iterate the computation until convergence, and only evaluate 4 different parameters. We minimize Eq. (8) for estimating the parameters $\mathbf{\Lambda} \equiv \{\lambda_t, \lambda_u, \lambda_f, \lambda_b\}$. Detailed discussion is provided in Section 3.1.

Shadow removal applied to the selected region

After changing the parameters, we also update the shadow of the areas not specified by the user. It is time consuming if we apply the graph cut shadow removal algorithm to the whole area $\mathcal{P}$; thus, we limit the size of the region Ω_1 to be processed. The second half of the algorithm is to choose some pixels to be processed, and apply the graph cut shadow removal algorithm to the chosen pixels using the updated parameter $\hat{\mathbf{\Lambda}}$.

Fig. 4 line 8 In line 7, the function "graph_cut" updates the shadow opacity β_p of pixel p to be $\hat{\beta}_p$ using the parameter set $\hat{\mathbf{\Lambda}}$. The shadow opacity has drastically changed if the value $|\beta_p - \hat{\beta}_p|$ is larger than the threshold T_0, and it has slightly changed if the value $|\beta_p - \hat{\beta}_p|$ is smaller than the threshold T_0. We randomly choose a certain pixel p inside the region Ω_0, and check whether the difference of the shadow opacity before applying the graph cut and after applying the graph cut is larger than the threshold T_0. We denote the chosen values β_p and $\hat{\beta}_p$ as β_a and β_b, respectively. Since we choose a pixel p randomly, β_a and β_b are expected to be the values of the shadow opacity before and after the graph cut, which are distributed in wide area. The threshold T_0 is dynamically adjusted so that these values β_a and β_b can be found; namely, if we could not find the pixels which satisfy the condition, we relax the condition by decreasing the threshold T_0 so that we can find the pixels which satisfy the condition. We choose one set of these values; thus, β_a and β_b are constant values.

Fig. 4 line 9 If the pixel p has a shadow opacity close to β_a, it is likely to become β_b if we apply our graph cut shadow removal algorithm. The shadow opacities of some pixels outside the user specified region Ω_0 are also equal to β_a. We denote the set of pixels whose shadow opacity is close to β_a within a threshold T_1 as Ω_1; namely, if the difference of the shadow opacity between the pixel p, β_p, and β_a is less than the threshold T_1, such pixel can be included in the set Ω_1. If we apply the graph cut shadow removal algorithm to the region Ω_1, the shadow opacity of some pixels in Ω_1 becomes close to β_b. Note that it is not guaranteed that the shadow opacity of all the pixels in region Ω_1 becomes β_b: Unless the graph cut shadow removal algorithm is applied, we do not know what value the shadow opacity will be for the pixels in region Ω_1. It is time consuming if the number of pixels in Ω_1, namely $|\Omega_1|$, is too large, and the change through this process is little if $|\Omega_1|$ is too small. Therefore, we dynamically change the threshold T_1 so that the number of pixels to be chosen, namely $|\Omega_1|$, is limited in a certain range. The range of $|\Omega_1|$ is set empirically considering the trade-off between the computation speed and the speed of the convergence.

Fig. 4 line 10 Before applying the graph cut shadow removal algorithm, the initial value of the shadow opacity $\hat{\beta}_p$ in region Ω_1 is set to be β_b. The shadow opacity in region Ω_1 is close to β_a; thus, setting the initial value as β_b is considered to be effective.

Fig. 4 line 11 We also apply the graph cut shadow removal algorithm to the surrounding region of Ω_1. The region Ω_1 is expanded by 4 pixels width. The value "4" is determined empirically. The shadow opacity of the surrounding area is considered to be correct while it is not considered to be correct in the selected area. The shadow opacity of the surrounding area works as a soft constraint.

Fig. 4 line 12 We apply the function "graph_cut" to the region Ω_1 using the current parameter set $\hat{\mathbf{\Lambda}}$. Note that, the parameter $\hat{\mathbf{\Lambda}}$ obtained in the first half of the algorithm is used. Also note that the graph cut shadow removal algorithm, namely graph_cut, is applied to the region Ω_1 which is not always the same as the user specified region Ω_0.

Summary of the algorithm

To summarize, the parameters $\mathbf{\Lambda}$ are updated (line 7) from the user's strokes Ω_0 (line 3). The output image in the area Ω_1 which the user did not specify ($\Omega_1 \neq \Omega_0$) is also improved by the updated parameters $\hat{\mathbf{\Lambda}}$ (line 12). Since the size of the area to be computed is smaller than the whole set of pixels ($\Omega_0 \in \mathcal{P}$

and $\Omega_1 \in \mathcal{P}$), the system can display the output image immediately.

4. Hierarchical Graph Cut

In order to improve the computation speed of the n-label graph cut, we propose a hierarchical graph cut. The algorithm uses a coarse-to-fine approach to run more quickly than both the α-expansion method [2] and Ishikawa's method [17].

We first explain the benefits of our approach as compared with the previous methods. A hierarchical approach to region segmentation or image restoration has been previously studied [18)–22)]; however, few methods have employed a hierarchical approach which can be applied to other applications. Juan, et al. [23] use an initial value before solving a graph cut to increase the computation speed, but it is only twice as fast as the α-expansion. A method called LogCut proposed by Lempitsky, et al. [24] is much faster, but it requires a training stage before it can be applied. On the other hand, the method proposed by Komodakis, et al. [25] does not need any training stages, but the method only improves the computation time of the second and subsequent iterations, not the first iteration. We propose a hierarchical graph cut algorithm which is faster than the α-expansion when applied to a multi-label MRF.

The pseudo-code of the hierarchical graph cut is described in **Fig. 6**, and that of the α-expansion is shown for comparison in **Fig. 7**. In Fig. 7 and Fig. 6, the function "graph" adds nodes and edges to the current graph g under the rule shown in **Fig. 8**. For each iteration, the α-expansion solves the 2-label MRF problem, where one is the current label and the other is stated as α. Our hierarchical graph cut uses multiple "α"s for each iteration (line 12 and line 13 in Fig. 6). The list of multiple αs is represented by A in line 3, and is defined as:

$$A_0 = \{0\},\ A_1 = \left\{\frac{n}{2}\right\},\ A_{2j} = \bigcup_{k=0}^{2^{j-1}-1} \left\{\frac{1+4k}{2^{j+1}}n\right\},\ A_{2j+1} = \bigcup_{k=0}^{2^{j-1}-1} \left\{\frac{3+4k}{2^{j+1}}n\right\}, \tag{11}$$

where j represents the level of the hierarchical structure, and n represents the number of labels. The interval of each label at level j is $\frac{4}{2^{j+1}}n$. For example, the list of labels for $n = 64$ will be as follows.

```
1:  B ≡ {β_p | p ∈ P} ⇐ initial value
2:  n ⇐ /* number of labels comes here */
3:  A ⇐ Eq. (11)
4:  success ⇐ 0
5:  for i = 0 to 2 log_2 n − 1 do
6:      g ⇐ null
7:      for all p ∈ P do
8:          α_p ⇐ argmin_{α∈A_i} |β_p − α|
9:          g ⇐ g ∪ graph(α_p, β_p) // see Fig. 8
10:     end for
11:     for all {p, q} ∈ N do
12:         α_p ⇐ argmin_{α∈A_i} |β_p − α|
13:         α_q ⇐ argmin_{α∈A_i} |β_q − α|
14:         g ⇐ g ∪ graph(α_p, α_q, β_p, β_q) // see Fig. 8
15:     end for
16:     B′ ⇐ max-flow(B, g)
17:     if E(B′) < E(B) then
18:         B ⇐ B′
19:         success ⇐ 1
20:     end if
21: end for
22: if success = 1 then goto 4
```

Fig. 6 Hierarchical graph cut.

$$A = \big\{\{0\},\{32\},\{16\},\{48\},\{8,40\},\{24,56\},\{4,20,36,52\},\ldots \\ \ldots,\{3,7,11,15,19,23,27,31,35,39,43,47,51,55,59,63\}\big\}. \tag{12}$$

A represents the hierarchical structure since the number of A increases exponentially:

$$|A_i| = \max(1, 2^{\lfloor \frac{i}{2} \rfloor - 1}), \tag{13}$$

where $|\cdot|$ represents the number of items included in the list. For example, the number of items for $n = 64$ will be as follows.

1: $B \equiv \{\beta_p | p \in \mathcal{P}\} \Leftarrow$ initial value
2: $n \Leftarrow$ /* number of labels comes here */
3: $A \Leftarrow \{0, 1, 2, \ldots, n-1\}$
4: success $\Leftarrow 0$
5: **for** $i = 0$ to $n-1$ **do**
6: $\quad \alpha \Leftarrow A_i$
7: $\quad g \Leftarrow$ null
8: $\quad$ **for all** $p \in \mathcal{P}$ **do**
9: $\quad\quad g \Leftarrow g \cup \text{graph}(\alpha, \beta_p)$ // see Fig. 8
10: $\quad$ **end for**
11: $\quad$ **for all** $\{p, q\} \in \mathcal{N}$ **do**
12: $\quad\quad g \Leftarrow g \cup \text{graph}(\alpha, \beta_p, \beta_q)$ // see Fig. 8
13: $\quad$ **end for**
14: $\quad B' \Leftarrow \text{max-flow}(B, g)$
15: $\quad$ **if** $E(B') < E(B)$ **then**
16: $\quad\quad B \Leftarrow B'$
17: $\quad\quad$ success $\Leftarrow 1$
18: $\quad$ **end if**
19: **end for**
20: **if** success $= 1$ **then** goto 4

Fig. 7 α-expansion.

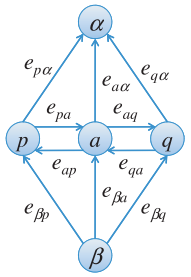
$$\{|A_0|, |A_1|, \ldots, |A_{11}|\} = \{1, 1, 1, 1, 2, 2, 4, 4, 8, 8, 16, 16\}. \quad (14)$$

Note that our algorithm can express all n numbers of the labels:

$$\sum_{i=0}^{2\log_2 n - 1} \max\left(1, 2^{\lfloor \frac{i}{2} \rfloor - 1}\right) = n. \quad (15)$$

Our method only requires $2\log_2 n$ times for each iteration (line 5 in the algorithm 6) thanks to the hierarchical approach, while the α-expansion needs n times for each iteration. Although Ishikawa's method does not require any iterations, the computation time is almost the same as for the α-expansion, since the number of nodes for the computation is n times larger than for the α-expansion.

Recently, Li and Huttenlocher [26], and Scharstein and Pal [27] took a machine learning approach in order to model the cost function (especially the smoothness function) which can represent the good stereo matching as possible. They estimated the parameters of the smoothness function using the training data of



edge	weight	for
$e_{p\alpha}$	$D(\alpha_p)$	$p \in \mathcal{P}$
$e_{\beta p}$	$D(\beta_p)$	
$e_{\beta a}$	$V(\beta_p, \beta_q)$	$\{p, q\} \in \mathcal{N},\ \alpha_p = \alpha_q$
$e_{a\alpha}$	$V(\alpha_p, \alpha_q)$	
e_{pa}, e_{ap}	$V(\alpha_p, \beta_p)$	
e_{aq}, e_{qa}	$V(\alpha_q, \beta_q)$	
$e_{\beta a}$	$V(\beta_p, \beta_q)$	$\{p, q\} \in \mathcal{N},\ \alpha_p \neq \alpha_q$
$e_{a\alpha}$	$V(\alpha_p, \alpha_q)$	
e_{ap}, e_{aq}	∞	$\{p, q\} \in \mathcal{N},\ \alpha_p \neq \alpha_q,$ $V(\beta_p, \beta_q) \leq V(\alpha_p, \alpha_q)$
e_{pa}	$[V(\alpha_p, \beta_q) - V(\beta_p, \beta_q)]^+$	
e_{qa}	$[V(\beta_p, \alpha_q) - V(\beta_p, \beta_q)]^+$	
e_{pa}, e_{qa}	∞	$\{p, q\} \in \mathcal{N},\ \alpha_p \neq \alpha_q,$ $V(\beta_p, \beta_q) \geq V(\alpha_p, \alpha_q)$
e_{aq}	$[V(\alpha_p, \beta_q) - V(\alpha_p, \alpha_q)]^+$	
e_{ap}	$[V(\beta_p, \alpha_q) - V(\alpha_p, \alpha_q)]^+$	

Fig. 8 Graph construction for our graph cut. $[x]^+ = \max(0, x)$. The nodes p and q are the neighboring nodes, which also represents the pixel position. The node a is the auxiliary node added in order to set the weights properly. The nodes α and β are the sink and source nodes, respectively, which also represent the labels used temporarily in the graph cut algorithm and the label to be estimated, respectively. $\mathcal{P}$ and $\mathcal{N}$ are the pixel set and the set of the couples of the neighboring pixels. e represents the edge weight between two nodes of the graph. V and D are the smoothness and the data term, respectively.

stereo matching results in order to obtain better results. Modeling appropriate smoothness function is important in the stereo matching problem since the disparity changes discontinuously when there are many occlusions. Our shadow removal software only treats the gradually changing shadow, and the discontinuous shadow opacity is not assumed. The discontinuity of the shadow appears if the scene is illuminated by multiple point light sources; thus, we are also interested in further improvement of our method using their methods [26),27)] to treat with this problem.

Though our algorithm is faster than α-expansion method [2)], its quality is slightly lower than α-expansion method [2)] and Ishikawa's method [17)]. However, the results in Section 5.1 indicate that our results are quite similar to the results of α-expansion method and Ishikawa's method. Recently, Pock, et al. [28)] proposed a method which produces slightly better results than Ishikawa's method when the problem is spatially continuous. Pock's work is devoted to the study of the variational problem,

$$\min_u \left\{ \int_\Omega |\nabla u(\mathbf{x})| \, d\mathbf{x} + \int_\Omega \rho\left(u(\mathbf{x}), \mathbf{x}\right) d\mathbf{x} \right\}, \tag{16}$$

where u is the label represented in MRF framework, Ω is the image domain, $\mathbf{x} = (x, y)^T$ is the pixel coordinate, and ρ is the data term. The left term of Eq. (16) is the Total Variation of the label u, represented as follows.

$$|\nabla u(\mathbf{x})| = \sqrt{\left(\frac{\partial u(\mathbf{x})}{\partial x}\right)^2 + \left(\frac{\partial u(\mathbf{x})}{\partial y}\right)^2}. \tag{17}$$

The TV term allows for sharp discontinuities and preserves edges in the solution. Therefore, the shadow boundary of hard shadow can be improved if we use Pock's method. However, we are interested in improving the results of soft shadow rather than hard shadow in our future work, since our method is more suitable for hard shadow than soft shadow as is discussed in Section 6.

We believe that improving the cost function (Eq. (3)) would be the most important future work than improving the graph cut algorithm. Later, in Section 5.1, we show the hierarchical graph cut results compared to other graph cuts. Section 5.1 shows that it is difficult to distinguish the difference between our result and other results at a glance. On the other hand, in Section 5.2, we show that our shadow removal result is apparently better than other shadow removal result obtained by Finlayson's method [5)] and Wu's method [11)]. One of the difference between our shadow removal algorithm and other shadow removal algorithms is that we employ graph cut for optimization, and the other is that we elaborately defined the cost function of the shadow removal problem. Since improving the graph cut would give only a small progress in the quality of the output as shown empirically in Section 5.1, we are directing our attention to improving the cost function in our future work. Though using better graph cuts like Pock's method improves our results, we believe that there is a much room in improving the cost function of the shadow removal problem.

4.1 Graph Structure

We construct the graph structure as described in Fig. 8. The edge weight e_{pa}, e_{ap}, e_{aq}, and e_{qa} in Fig. 8 should be a positive number; thus, we truncate the edge weight if the weight is negative. Now, we discuss how the truncation affects the cost function defined by the engineer.

Consider the case where $V(\beta_p, \beta_q) \geq V(\alpha_p, \alpha_q)$. The smoothness cost when the label of pixel p is α (i.e., α_p) and the label of pixel q is β (i.e., β_q) is represented as follows (Fig. 8):

$$[V(\alpha_p, \beta_q) - V(\alpha_p, \alpha_q)]^+ + V(\alpha_p, \alpha_q). \tag{18}$$

Here, $[x]^+ = \max(0, x)$ represents the truncation of the negative value. From line 12, line 13 in Fig. 6, and Eq. (11), the following inequalities hold.

$$0 \leq |\alpha_p - \beta_p| \leq \frac{2}{2^{j+1}} n, \qquad 0 \leq |\alpha_q - \beta_q| \leq \frac{2}{2^{j+1}} n. \tag{19}$$

From $V(\beta_p, \beta_q) \geq V(\alpha_p, \alpha_q)$, Eqs. (6), (11), and (19), we obtain the following property.

$$-\frac{2}{2^{j+1}} n \leq V(\alpha_p, \beta_q) - V(\alpha_p, \alpha_q) \leq \frac{2}{2^{j+1}} n. \tag{20}$$

If $V(\alpha_p, \beta_q) - V(\alpha_p, \alpha_q) \geq 0$ holds, the truncation does not occur. In this case, Eq. (18) becomes $V(\alpha_p, \beta_q) - V(\alpha_p, \alpha_q) + V(\alpha_p, \alpha_q)$; namely, the proper cost $V(\alpha_p, \beta_q)$ is set to the graph; thus, the max-flow computation [29)] (line 16) can obtain a correct result. On the other hand, the proper cost is not set to the graph if $V(\alpha_p, \beta_q) - V(\alpha_p, \alpha_q) \leq 0$, and the maximum error for this term is

$\frac{2}{2^{j+1}}n$. This error value is the half of the interval of the αs $\frac{4}{2^{j+1}}n$ (c.f., Eq. (11)); meaning that the graph construction shown in Fig. 8 can solve the problem up to a discretization error. We can also prove the same property for the cases which are not $V(\beta_p, \beta_q) \geq V(\alpha_p, \alpha_q)$.

The above property indicates that the algorithm cannot always estimate the global optimum of the given cost function. On the other hand, the estimation error is limited with a small value; thus, the algorithm is expected to produce a solution close to the global optimum.

5. Experiments

5.1 Hierarchical Graph Cut

Comparison to other graph cut methods

First, we experimentally validate the performance of the hierarchical graph cut algorithm in three domains: shadow removal, image restoration, and stereo matching. We used the stereo data sets introduced in Ref. 27). The results

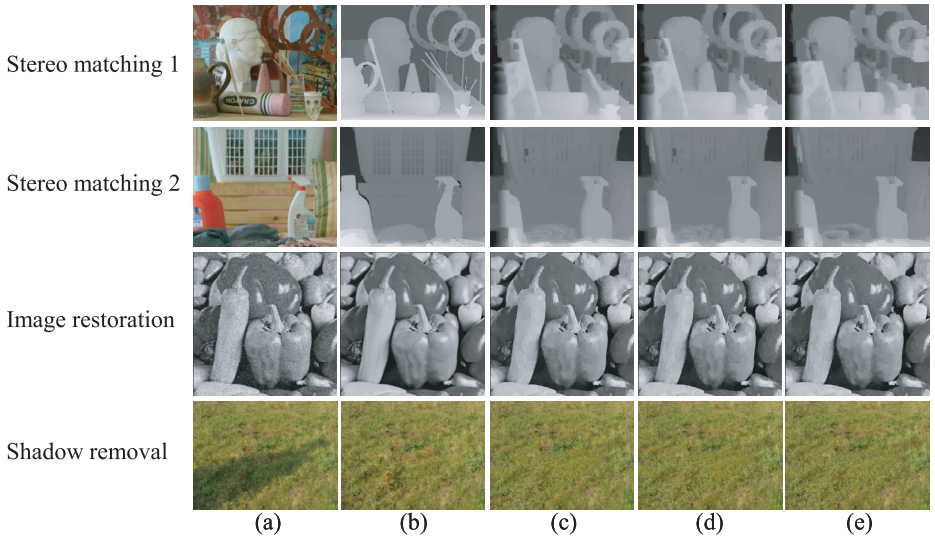


Fig. 9 Results of our hierarchical graph cut. (a), (b), (c), (d), and (e) show the input, the ground truth, the result for Ishikawa's method, the result for α-expansion, and the result for the hierarchical graph cut, respectively.

Table 1 Computation speed of our *H-Cut*. The first row specifies the experiments. The second and third rows show the number of labels used and the image size. The fourth and fifth rows show the ratio of computation times for *X-Cut vs. H-Cut* and *A-Cut vs. H-Cut*. The error differences for *H-Cut* minus *X-Cut* and *H-Cut* minus *A-Cut* are shown in the sixth and seventh rows, where a positive value means that the other methods outperform our method. The number of iterations required until convergence occurs is shown in the eighth and ninth rows for *A-Cut* and *H-Cut*. *X-Cut* does not need iterations. The memory size required is shown in the tenth, eleventh, and twelfth rows for *X-Cut*, *A-Cut*, and *H-Cut*.
[notations] *X-Cut*: Ishikawa's exact optimization. *A-Cut*: α-expansion. *H-Cut*: Hierarchical graph cut.

Problem		Stereo matching 1	Stereo matching 2	Image restoration	Shadow removal
Labels		128	128	256	64
Image size		543 × 434	480 × 397	256 × 256	640 × 480
Speed-up	*vs. X-Cut*	×5.2	×5.9	×8.0	×6.4
	vs. A-Cut	×6.8	×11.4	×16.6	×3.4
Error difference	*vs. X-Cut*	+5.0%	+3.2%	+0.6%	+0.0%
	vs. A-Cut	+4.6%	+3.0%	−0.4%	+0.0%
Iteration	*A-Cut*	8	7	10	4
	H-Cut	10	5	7	6
Allocated memory	*X-Cut*	6,748 MB	5,537 MB	5,306 MB	4,538 MB
	A-Cut	106 MB	111 MB	65 MB	482 MB
	H-Cut	106 MB	120 MB	64 MB	482 MB

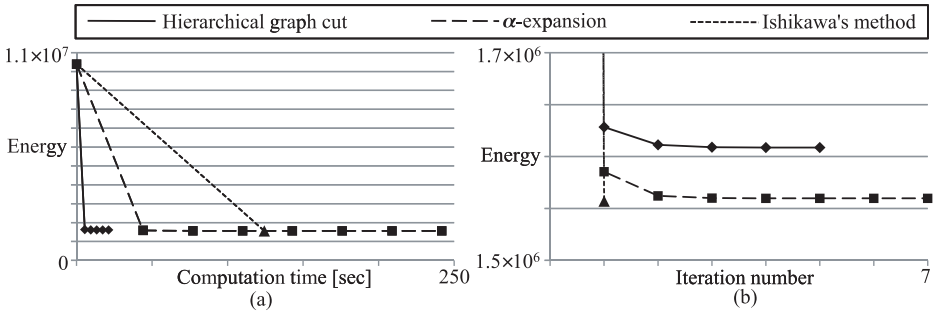


Fig. 10 The plot of the value of overall cost of experiment "stereo matching 2" *vs.* the time (a) or iteration (b). The initial value of the cost is the cost of conventional stereo.

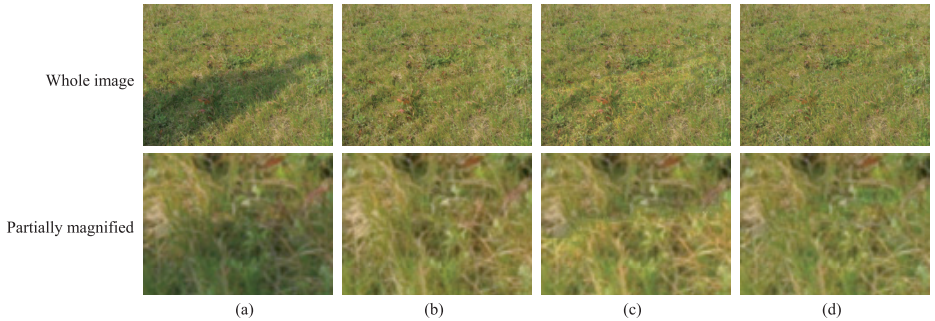


Fig. 11 The shadow removal result of our method and the gradient descent approach. (a), (b), (c), and (d) show the input, the ground truth, the result for gradient descent method, and the result for the hierarchical graph cut, respectively.

Table 2 Computation speed of our method and the gradient descent approach. The first and second rows show the number of labels used and the image size. The third row shows the ratio of computation times for the gradient descent method *vs.* our hierarchical graph cut method. The error difference for our method minus the gradient descent method is shown in the fourth row, which indicates that our method is better than the gradient descent method. The number of iterations required until convergence occurs is shown in the fifth and sixth rows for our method and the gradient descent method. The memory size required is shown in the seventh and eighth rows for our method and the gradient descent method.

Labels		64
Image size		640 × 480
Speed-up		×1.8
Error difference		−3.4%
Iteration	*Gradient descent*	19
	Hierarchical graph cut	6
Allocated memory	*Gradient descent*	485 MB
	Hierarchical graph cut	482 MB

shown in **Fig. 9** indicate that our algorithm produces similar results to the α-expansion [2] and to Ishikawa's method [17]. **Table 1** shows that the hierarchical graph cut is 3 to 16 times (*or* 5 to 8 times) faster than the α-expansion (*or* Ishikawa's method). The change in the value of the cost function at the first iteration is large, while it is negligible after the second iteration, as shown in **Fig. 10**. The disadvantage of the hierarchical graph cut is that the result depends on the initial value when there are many local minima, as shown in the stereo matching result. Due to its fast computation speed, we use the hierarchical graph cut in our shadow removal algorithm instead of the α-expansion and Ishikawa's method. The detailed discussion for image restoration and stereo matching is shown in Appendix A.1.

Fig. 12 Our shadow removal results. The first and fourth columns show the input images, the second and fifth columns show the shadow-free images, and the third and sixth columns show the shadow images.

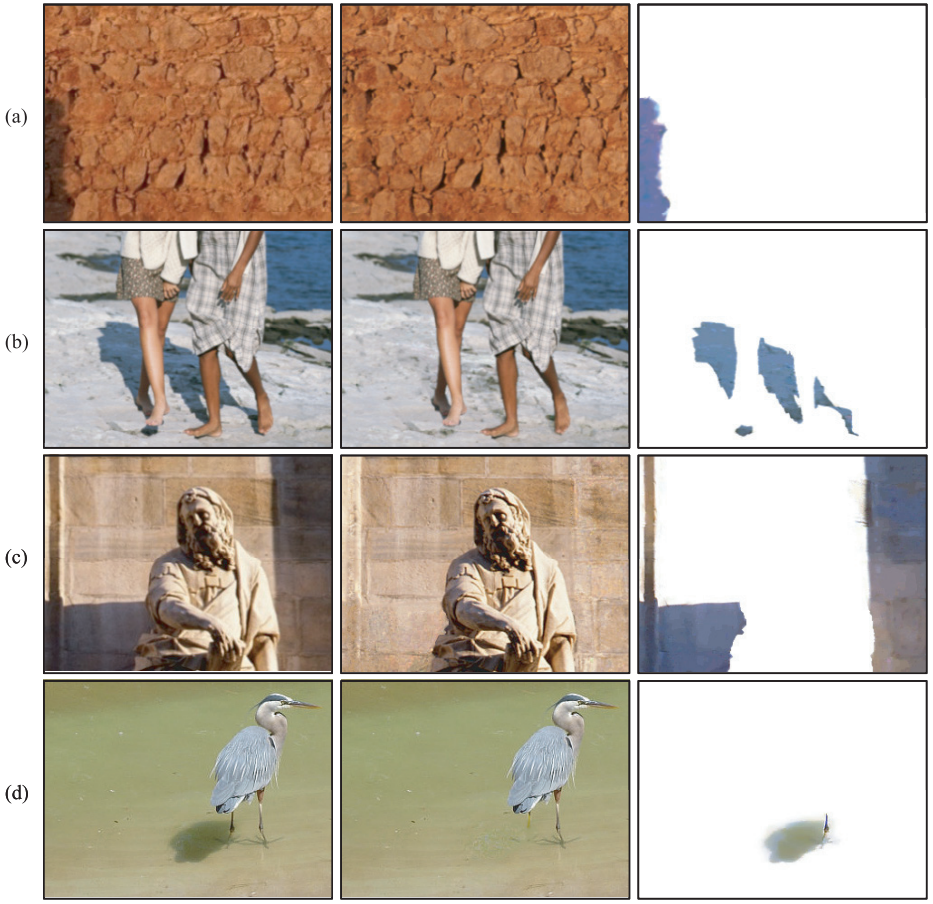


Fig. 13 Our shadow removal results. The first and fourth columns show the input images, the second and fifth columns show the shadow-free images, and the third and sixth columns show the shadow images.

Comparison to gradient descent approach

We also applied the method based on gradient descent to the shadow removal problem. **Figure 11** (c) is the result of the gradient descent based method. Compared to the result of our method shown in Fig. 11 (d), the gradient descent based method does not work well at the shadow boundary. The error difference shown in **Table 2** also shows that the image quality of Fig. 11 (c) is lower than that of Fig. 11 (d). Gradient descent finds a local minimum which is close to the initial value; while most of the graph cut methods can find a global minimum. The experiment shown in Table 1 indicates that our hierarchical graph cut (c.f., H-Cut) can produce the result close to the global minimum (c.f., X-Cut). Therefore, our method removes the shadow at the boundary in higher quality than the method based on gradient descent.

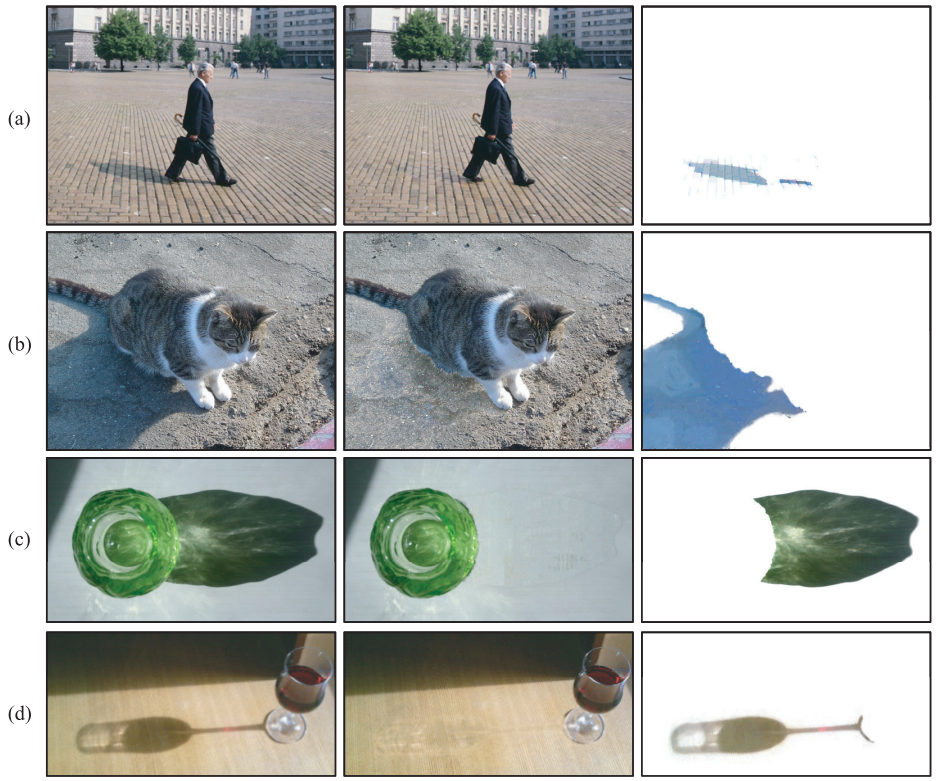


Fig. 14 Our shadow removal results. The first and fourth columns show the input images, the second and fifth columns show the shadow-free images, and the third and sixth columns show the shadow images.

Table 3 Computation time for our shadow removal. The first column gives the images from Figs. 12–14. The second column shows the size of each image. The third column shows the number of user interactions used for parameter optimization. The fourth column shows the computation time for each user interaction, using 3 GHz desktop computer.

Image	Image size	Strokes†	Average time per stroke
Fig. 12 (a) grass	640 × 480 [px]	N/A	N/A
Fig. 12 (b) family	640 × 480 [px]	1	1.4 [sec]
Fig. 12 (c) standing	320 × 240 [px]	4	1.2 [sec]
Fig. 12 (d) horse	640 × 480 [px]	6	2.1 [sec]
Fig. 13 (a) rock	640 × 480 [px]	8	1.5 [sec]
Fig. 13 (b) women	640 × 480 [px]	12	2.0 [sec]
Fig. 13 (c) statue	320 × 240 [px]	15	1.9 [sec]
Fig. 13 (d) bird	320 × 240 [px]	2	1.5 [sec]
Fig. 14 (a) walking	640 × 480 [px]	26	2.0 [sec]
Fig. 14 (b) cat	640 × 480 [px]	27	5.8 [sec]
Fig. 14 (c) glass	640 × 360 [px]	32	6.3 [sec]
Fig. 14 (d) wine	640 × 320 [px]	40	2.8 [sec]

† = Number of strokes for parameter optimization

5.2 Shadow Removal

Natural image

Our shadow removal results are shown in **Figs. 12–14**. The number of user interactions required for parameter optimization is listed in **Table 3**. The system displays the output image at the responsive speed. The shadows were removed effectively while the complex textures of the images were preserved. We express the shadow opacity using 64 discrete values, but these results show that this discretization does not cause any strong defects. Figure 12 (a) did not require the stroke for the interactive parameter optimization, since the default parameters successfully removed the shadows. The shadows of the cracks in Fig. 13 (a) and the statue in Fig. 13 (c) are not removed since the user did not specify these shadows to be removed. The leg of the bird in Fig. 13 (d) is disappeared since the brightnesses of the shadow and the leg are similar, and this is always a problem for most of the shadow removal softwares. If the shadows are complex (Fig. 14 (c) and Fig. 14 (d)), the user has to add many strokes to extract them.

Aerial images

In aerial images, the shadows of buildings fall both on the ground and on neighboring buildings. Neighboring aerial images are often taken at different times, so that when they are stitched together, there may be a seam where the different images meet. Thus, it is important to remove the shadows in the aerial images. After many strokes are added to this complex scene, we finally obtain the images which are shown in **Fig. 15**.

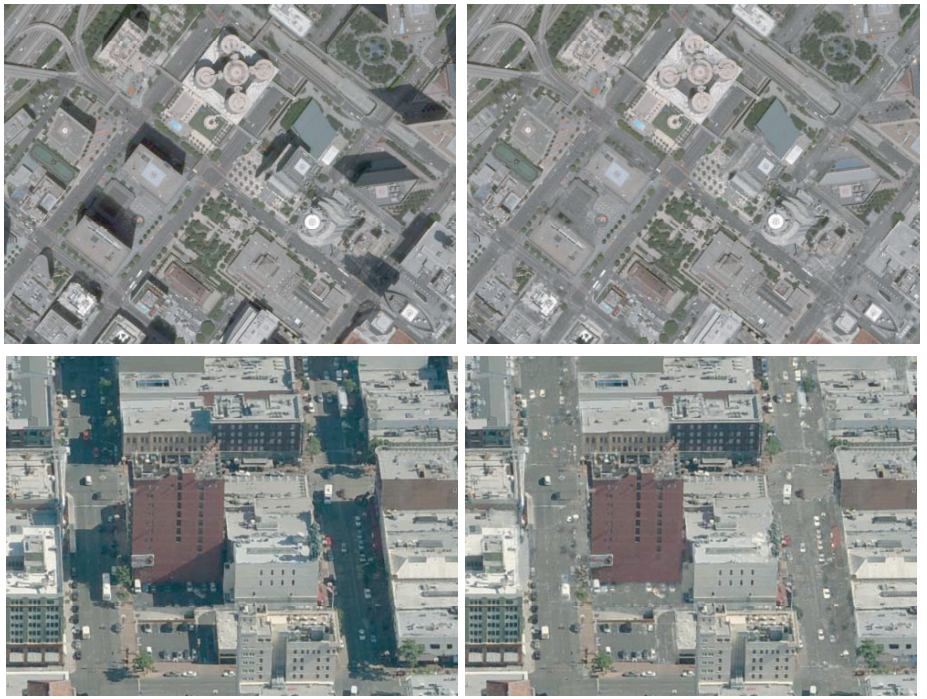


Fig. 15 Application to aerial images. The input image and the shadow-free image are shown.

Evaluation

In **Fig. 16**, we show how our method benefits from the user interaction. The results are evaluated quantitatively using the ground truth. Our results improve gradually when the user interacts with the system. The computation times for the results shown in Fig. 16 (a) using a 3 GHz desktop computer are 21 [sec], 336 [sec],

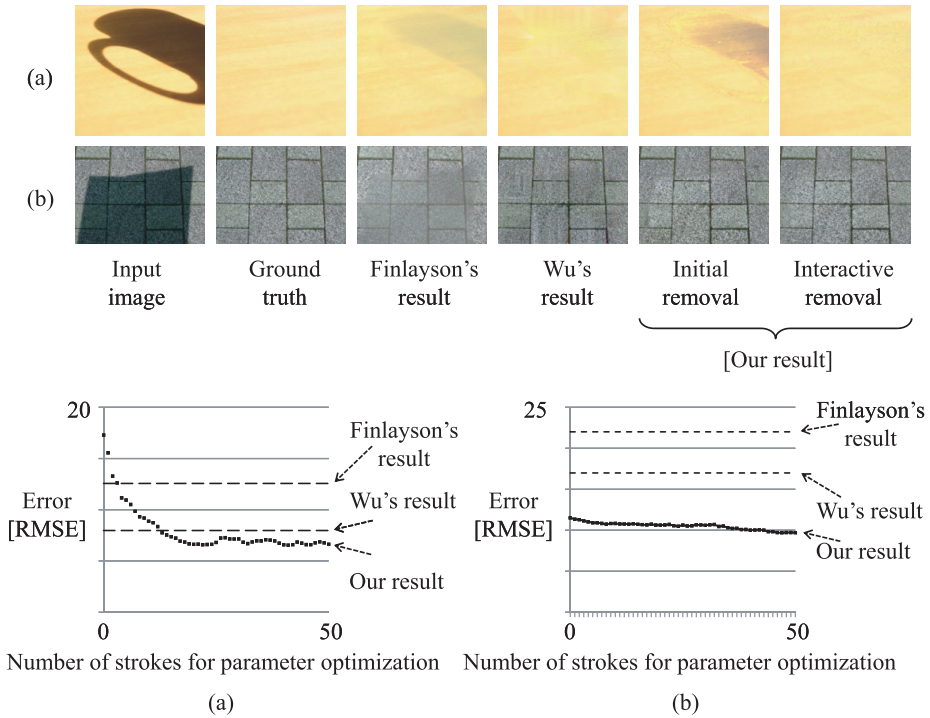


Fig. 16 Comparison between our method, Finlayson's method, and Wu's method, when applied to indoor scene (a) and outdoor scene (b). The root mean square error (RMSE) is calculated by comparison with the ground truth. The solid line represents our results and the dashed lines represent Finlayson's results and Wu's results.

and 65 [sec] for the main part of the algorithm for Finlayson's method [5], Wu's method [11], and our method until convergence, respectively; while the computation times for Fig. 16 (b) are 8 [sec], 649 [sec], and 53 [sec].

6. Conclusions and Discussions

We present a method for user-assisted shadow removal from a single image. We have expressed the shadow opacity with a multi-label MRF and solved it using a hierarchical graph cut. Our hierarchical graph cut algorithm allows the system to run at interactive speeds. The weighting parameters for each cost term are automatically updated using an intuitive user interface.

Using user-supplied hints, the coefficients of each cost term are adjusted, and the method can be applied to both hard shadows and soft shadows. We set the default coefficients so that it can successfully remove the hard shadow. Since the best coefficients for soft shadow is difficult to find, we require the users to interact to the system. Automatic shadow removal which works well with soft shadow is our future work.

The hierarchical graph cut solves multi-label MRF problems 3 to 16 times faster than α-expansion [2] and Ishikawa's graph cut [17]. However, the precision of this graph cut method is often slightly worse than α-expansion. It often falls into local minimum; thus, a good initial value is necessary. Considering both the computation speed and the robustness is a hard task, and we leave it as an open problem.

Another limitation is that the method requires the labels to have a numerical order. It cannot be used, for example, to segment image pixels into the foreground or background. However, fast solutions already exist for such foreground/background cut problems.

References

1) Nielsen, M. and Madsen, C.B.: Graph cut based segmentation of soft shadows for seamless removal and augmentation, *Proc. Scandinavian Conf. Image Anal.* (*SCIA*), pp.918–927 (2007).
2) Boykov, Y., Veksler, O. and Zabih, R.: Fast approximate energy minimization via graph cuts, *IEEE Trans. Patt. Anal. and Mach. Intell.*, Vol.23, No.11, pp.1222–1239 (2001).
3) Weiss, Y.: Deriving intrinsic images from image sequences, *Proc. Int'l Conf. Comp. Vis.* (*ICCV*), Vol.2, pp.68–75 (2001).
4) Matsushita, Y., Nishino, K., Ikeuchi, K. and Sakauchi, M.: Illumination normalization with time-dependent intrinsic images for video surveillance, *IEEE Trans. Patt. Anal. and Mach. Intell.*, Vol.26, No.10, pp.1336–1347 (2004).
5) Finlayson, G.D., Drew, M.S. and Lu, C.: Intrinsic images by entropy minimization, *Proc. European Conf. Comp. Vis.* (*ECCV*), pp.582–595 (2004).
6) Fredembach, C. and Finlayson, G.: Simple shadow removal, *Proc. Int'l Conf. Patt. Recog.* (*ICPR*), pp.832–835 (2006).
7) Tappen, M.F., Freeman, W.T. and Adelson, E.H.: Recovering intrinsic images from a single image, *IEEE Trans. Patt. Anal. and Mach. Intell.*, Vol.27, No.9, pp.1459–

1472 (2005).
8) Baba, M., Mukunoki, M. and Asada, N.: Shadow removal from a real image based on shadow density, *ACM SIGGRAPH Posters*, p.60 (2004).
9) Arbel, E. and Hel-Or, H.: Texture-preserving shadow removal in color images containing curved surfaces, *Proc. Comp. Vis. and Patt. Recog.* (*CVPR*) (2007).
10) Wu, T.-P. and Tang, C.-K.: A Bayesian approach for shadow extraction from a single image, *Proc. Int'l Conf. Comp. Vis.* (*ICCV*), Vol.1, pp.480–487 (2005).
11) Wu, T.-P., Tang, C.-K., Brown, M.S. and Shum, H.-Y.: Natural shadow matting, *ACM Trans. Gr.*, Vol.26, No.2, p.8 (2007).
12) Li, Y., Sun, J., Tang, C.-K. and Shum, H.-Y.: Lazy snapping, *Proc. ACM SIGGRAPH*, pp.303–308 (2004).
13) Rother, C., Kolmogorov, V. and Blake, A.: GrabCut — Interactive foreground extraction using iterated graph cuts, *Proc. ACM SIGGRAPH*, pp.309–314 (2004).
14) Barrow, H.G. and Tenenbaum, J.M.: Recovering intrinsic scene characteristics from images, *Computer Vision Systems*, pp.3–26 (1978).
15) Levin, A., Zomet, A. and Weiss, Y.: Separating reflections from a single image using local features, *Proc. Comp. Vis. and Patt. Recog.* (*CVPR*), pp.306–313 (2004).
16) Miyazaki, D., Hara, K. and Ikeuchi, K.: Median photometric stereo as applied to the Segonko Tumulus and museum objects, *Int'l J. of Comp. Vis.*, Vol.86, pp.229–242 (2010).
17) Ishikawa, H.: Exact optimization for Markov random fields with convex priors, *IEEE Trans. Patt. Anal. and Mach. Intell.*, Vol.25, No.10, pp.1333–1336 (2003).
18) Darbon, J. and Sigelle, M.: Image restoration with discrete constrained total variation, *J. Math. Imaging Vis.*, Vol.26, No.3, pp.261–276 (2006).
19) D'Elia, C., Poggi, G. and Scarpa, G.: A tree-structured Markov random field model for Bayesian image segmentation, *IEEE Trans. Image Processing*, Vol.12, No.10, pp.1259–1273 (2003).
20) Feng, W. and Liu, Z.-Q.: Self-validated and spatially coherent clustering with net-structured MRF and graph cuts, *Proc. Int'l Conf. Patt. Recog.* (*ICPR*), pp.37–40 (2006).
21) Lombaert, H., Sun, Y., Grady, L. and Xu, C.: A multilevel banded graph cuts method for fast image segmentation, *Proc. Int'l Conf. Comp. Vis.* (*ICCV*), Vol.1, pp.259–265 (2005).
22) Nagahashi, T., Fujiyoshi, H. and Kanade, T.: Image segmentation using iterated graph cuts based on multi-scale smoothing, *Proc. Asian Conf. Comp. Vis.* (*ACCV*), pp.806–816 (2007).
23) Juan, O. and Boykov, Y.: Active graph cuts, *Proc. Comp. Vis. and Patt. Recog.* (*CVPR*), pp.1023–1029 (2006).
24) Lempitsky, V., Rother, C. and Blake, A.: LogCut - Efficient graph cut optimization for Markov random fields, *Proc. Int'l Conf. Comp. Vis.* (*ICCV*) (2007).
25) Komodakis, N., Tziritas, G. and Paragios, N.: Fast, approximately optimal solutions for single and dynamic MRFs, *Proc. Comp. Vis. and Patt. Recog.* (*CVPR*) (2007).
26) Li, Y. and Huttenlocher, D.P.: Learning for stereo vision using the structured support vector machine, *Proc. Comp. Vis. and Patt. Recog.* (*CVPR*) (2008).
27) Scharstein, D. and Pal, C.: Learning conditional random fields for stereo, *Proc. Comp. Vis. and Patt. Recog.* (*CVPR*) (2007).
28) Pock, T., Schoenemann, T., Graber, G., Bischof, H. and Cremers, D.: A convex formulation of continuous multi-label problems, *Proc. European Conf. Comp. Vis.* (*ECCV*), pp.792–805 (2008).
29) Boykov, Y. and Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision, *IEEE Trans. Patt. Anal. and Mach. Intell.*, Vol.26, No.9, pp.1124–1137 (2004).
30) Scharstein, D. and Szeliski, R.: vision.middlebury.edu/stereo.
31) Scharstein, D. and Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, *Int'l J. of Comp. Vis.*, Vol.47, No.1-3, pp.7–42 (2002).

Appendix

A.1 Image Restoration and Stereo Matching

The main topic of this paper is the shadow removal; thus, we examine the experiments shown in Fig. 9 and Table 1 in the appendix.

The initial value of image restoration is the input image itself. The data term is defined as follows.

$$D_p(\beta_p) = \min(t_1, |\beta_p - I_p|), \tag{21}$$

where t_1 is a constant value, and I_p is the pixel brightness of the input image. The smoothness cost is as follows.

$$V_{p,q}(\beta_p, \beta_q) = t_2 \exp(-|\nabla I_p|/t_3)|\beta_p - \beta_q|, \tag{22}$$

where t_2 and t_3 are constant values, and ∇I_p represents the edge strength which is calculated by Sobel operator.

The initial value of stereo matching is the result of conventional stereo which uses shiftable window [31]. The data term is defined as follows.

$$D_p(\beta_p) = \min(t_4, s(\beta_p)), \tag{23}$$

where t_4 is a constant value, and $s(\beta_p)$ is the SAD (sum of absolute difference) between the stereo image pair calculated by shiftable window whose size is 5×5. The smoothness cost is as follows.

Table 4 Middlebury stereo evaluation [30], where error threshold is 1. Tsukuba, Venus, Teddy, and Cones are the names of input image pair. The smaller the numerical value is, the better the algorithm's performance is. "Avg. rank" represents the overall performance of the algorithm. "H-Cut" represents the proposed method. "SSD+MF [1a]," "DP [1b]," "SO [1c]," and "GC [1d]" represent the stereo matching algorithms which use "SSD + min-filter," "Dynamic programming," "Scanline optimization," and "Graph cuts using alpha-beta swaps," respectively [31].

Algorithm	Avg. rank	Tsukuba	Venus	Teddy	Cones
GC [1d]	57.5	1.94	1.79	16.5	7.70
H-Cut	59.4	2.85	1.73	10.7	5.46
DP [1b]	65.1	4.12	10.1	14.0	10.5
SSD+MF [1a]	69.0	5.23	3.74	16.5	10.6
SO [1c]	70.7	5.08	9.44	19.9	13.0

$$V_{p,q}(\beta_p, \beta_q) = \left(t_5 \exp(-|\nabla I_p|^2/t_6) + t_7\right) |\beta_p - \beta_q|, \tag{24}$$

where t_5, t_6, and t_7 are constant values, and ∇I_p represents the edge strength which is calculated by Sobel operator. The subpixel refinement and the occlusion detection are not implemented in our current software.

We have compared our stereo matching result with other methods [30]. The results are shown in **Table 4**. As expected, our result is worse than conventional graph cut stereo, but is better than DP (dynamic programming) stereo and other conventional stereos. The comparison with state-of-art stereo algorithms can be found in the Middlebury stereo webpage [30].

(Received February 18, 2010)
(Accepted September 27, 2010)
(Released December 15, 2010)

(Communicated by *Stephen Maybank*)

Daisuke Miyazaki received his B.S. degree in science from the University of Tokyo in 2000, M.S. degree in information science and technology from the University of Tokyo in 2002, and Ph.D. degree in information science and technology from the University of Tokyo in 2005. He is a lecturer at Hiroshima City University, Japan. After working at the University of Tokyo for three years, and the Microsoft Research Asia for a half year, he joined Hiroshima City University. He received the Best Overall Paper Award from VSMM in 2000. His research interests include computer vision and computer graphics. He is a member of ACM and IEEE.

Yasuyuki Matsushita received his B.S., M.S. and Ph.D. degrees in EECS from the University of Tokyo in 1998, 2000, and 2003, respectively. He joined Microsoft Research Asia in April 2003, and now he is a Lead Researcher in Visual Computing Group. His areas of research are computer vision (photometric techniques, such as radiometric calibration, photometric stereo, shape-from-shading), computer graphics (image relighting, video analysis and synthesis). Dr. Matsushita served as an Area Chair for IEEE Computer Vision and Pattern Recognition (CVPR) 2009 and International Conference on Computer Vision (ICCV) 2009, and he is on the editorial board member of International Journal of Computer Vision (IJCV) and IPSJ Journal of Computer Vision and Applications (CVA). He also serves as a Program Co-Chair of PSIVT 2010.

Katsushi Ikeuchi received his B.Eng. degree in mechanical engineering from Kyoto University, Kyoto, Japan, in 1973, and Ph.D. degree in information engineering from the University of Tokyo, Tokyo, Japan, in 1978. He is a professor at the Interfaculty Initiative in Information Studies, the University of Tokyo, Tokyo, Japan. After working at the AI Laboratory at the Massachusetts Institute of Technology for three years, the Electrotechnical Laboratory for five years, and the School of Computer Science at Carnegie Mellon University for 10 years, he joined the University of Tokyo in 1996. He was selected as a Distinguished Lecturer of IEEE Signal Processing Society for the period of 2000–2001, and a Distinguished Lecturer of IEEE Computer Society for the period of 2004–2006. He has received several awards, including the David Marr Prize in ICCV 1990, IEEE R&A K-S Fu Memorial Best Transaction Paper Award in 1998, and best paper awards in CVPR 1991, VSMM 2000, and VSMM 2004. In addition, in 1992, his paper, "Numerical Shape from Shading and Occluding Boundaries," was selected as one of the most influential papers to have appeared in the Artificial Intelligence Journal within the past 10 years. He is a fellow of IEEE.