Dense Labeling with User Interaction: an Example for Depth-Of-Field Simulation

Ana B. Cambra
acambra@unizar.es
Adolfo Muñoz
adolfo@unizar.es
José J. Guerrero
jguerrer@unizar.es
Ana C. Murillo
acm@unizar.es

Instituto de Investigación en Ingeniería
de Aragón
Universidad de Zaragoza
Zaragoza, Spain

Abstract

This work presents a method to achieve dense labeling in a very simple and efficient way, faster and with more flexibility than related approaches. We use an initial superpixel graph and reformulate its constraints as a sparse linear system of equations, which is efficiently solved as a linear least squares problem. We demonstrate our approach with an interactive application for depth-of-field simulated effects based on dense depth estimation from a single image. This kind of interactive application requires to continuously obtain a dense labeling estimation, therefore it is necessary to solve the task with high efficiency. Our experiments show our method obtains comparable results for a dense depth estimation against related approaches, while providing a more generic and powerful representation of the problem. The proposed dense labeling system opens new opportunities to design interactive applications that require dense labeling estimation of any other image property.

1 Introduction

Plenty of problems in computer vision and image-processing applications rely on *labeling* techniques. Given a set of initial values for a few pixels, the estimation process assigns the best value for every image pixel. We find plenty of applications for this formulation, such as obtaining depth information in panoramic images [19], semantic labeling of RGB-D images [23], assigning semantic and geometric labels in conventional images [33] or tools to assist the user in manual image filtering and editing [14, 18]. Our work is focused on this latter group of interactive applications. We should note that common dense labeling techniques present too high computational cost for interactive applications, where the feedback to the user must be at interactive rates. We find other editing image applications which, using different techniques, already provide an interactive feedback to facilitate intuitive editing, e.g. to manipulate the appearance of material objects [3, 8] or to separate a video into its reflectance and illumination intrinsic images [5].

Pages 102.1-102.13
DOI: https://dx.doi.org/10.5244/C.30.102

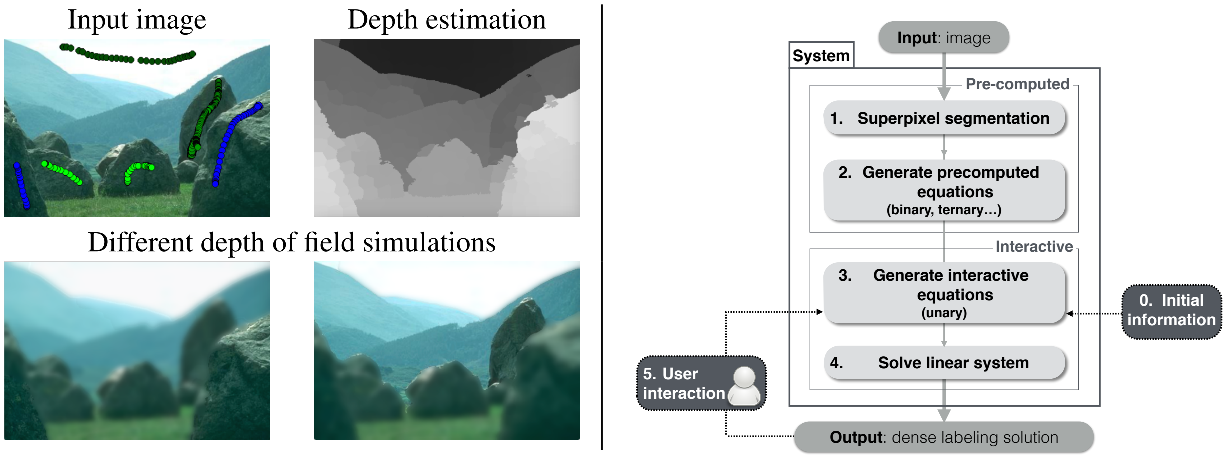
Input image
Depth estimation
Different depth of field simulations
Input: image
System
Pre-computed
1. Superpixel segmentation
2. Generate precomputed equations (binary, ternary...)
Interactive
3. Generate interactive equations (unary)
0. Initial information
4. Solve linear system
5. User interaction
Output: dense labeling solution

Figure 1: Left: Steps of the interactive D-o-F application: the user provides a few strokes over the image and with each new edition, the application re-estimates the depth map estimation and applies a depth-of-field effect. Right: Steps of our dense-labeling pipeline.

The main contribution of this work is a novel pipeline for interactive dense labeling, which provides a framework that can be applied in any application that involves dense labeling and user interaction. Our approach is focused on efficient dense labeling estimation and is particularly well suited for the use of continuous magnitudes. The dense labeling is formulated as a linear system of equations over superpixels and then solved as a linear least squares problem. Our experiments show that our approach is the fastest to obtain a solution compared to related approaches while keeping comparable quality in the results. Besides, we demonstrate how our pipeline is suitable for interactive applications developing an interactive application for depth-of-field simulated effects from a single image, Fig. 1, which requires a fast dense depth estimation. Our approach provides sufficient accuracy in the results at the necessary speed to achieve a good response to the user interaction.

2 Related Work

Dense labeling. Many related works take advantage of the use of superpixels to help with different labeling problems, such as estimating dense depth information in multiple panoramic images [19] or assigning semantic and geometric labels in conventional images [33]. Assigning the labels according to superpixels rather than individual pixels allows us to reduce the complexity of the dense-labeling optimization although implicitly introduces an accuracy penalty. Markov Random Fields (MRF) are common ingredient on solutions for dense labeling problems [30] but they still present too high computational cost to achieve results at interactive rates. As our approach, MRF superpixel-based approaches [19, 23, 33] or more efficient recent work [7], follow the same aim to improve the execution time. Our formulation is less restrictive and our experiments prove that we achieve a faster response. Another important group of dense labeling solutions is inspired by the Random Walker (RW) algorithm [9, 27, 32]. The RW algorithm can be used in an an interactive segmentation tool [11] where the user marks a few pixels with an arbitrary number of labels. This technique computes the probability of each label for each pixel as the probability of a random walk starting at the pixel that first reaches a seed with that label. All probabilities may be computed analytically by solving a system of linear equations and therefore, it can be slow for user interaction when a high number of labels is used. We find multiple works towards efficient solutions, based on offline pre-computation [2], or on the use of superpixels [10].

As opposed to typical dense labeling formulations, which deal with a discrete label set, our approach is designed to represent continuous magnitudes. Then, our approach is more related to complex MRF modifications [28, 35] or works where the Random Walker probability has been used as a continuous label [21].

Computer vision tasks with a human-in-the-loop. *Human-in-the-loop* approaches have been shown very successful in a large variety of computer vision problems. We find incremental learning systems from the user input, for tasks such as fine grained categorization of conventional image content [34] or activity recognition [17], and crowdsourcing based applications, such as medical applications built from crowdsourced human knowledge [22]. In both cases, as well as in our work, each user input is a very small piece of information that is not enough on its own to solve the problem, but integrated in an automatic system facilitates an efficient and high quality solution.

Besides the general problem of dense labeling, our work is also related to depth estimation from a single image. Using a single-view to generate depth information is a very challenging task in computer vision which has seen many advances in the last years [12, 15, 24, 29]. However, our algorithm provides a tool to obtain dense depth information but in very different settings, interactive applications with a human-in-the-loop. Differently from these works, we aim to achieve a dense depth labeling without any prior model assumed or learned. We seek to infer the information just from the user interaction, as targeted in related previous interactive solutions for example towards image segmentation [20] or depth labeling [16]. Our motivation is that many final applications using dense depth maps are actually interactive image editing or modeling applications, such as the depth-of-field filter application demonstrated as part of this work.

3 Dense labeling algorithm

This section explains our proposed interactive dense labeling. Right Fig. 1 shows a summary of its main steps.

3.1 Problem formulation.

We model the image as a graph, where the set of nodes N are the superpixels and the edges E represent the established relationships between superpixels. Given a set of labels L, a dense-labeling problem consists in assigning a label, $l \in L$, to each superpixel in the image. Assigning a label to a superpixel is equivalent to assigning the same label to all pixels inside the superpixel. Given this representation, we define a linear system of equations that can be solved at interactive rates, leading to the desired dense labeling from a sparse input.

While standard labeling formulation forces to choose a discrete set of labels, our formulation considers continuous label sets, e.g., we use real numbers in [0,1]. Our proposed solution to this labeling problem is based on a linear system of equations where the unknowns are the labels. Each equation contributes to the label value of one or more superpixels. A *unary* equation assigns a specific value as the label of a superpixel. A *binary* equation establishes a relation between the label of two superpixels connected in the graph. Note that linear equations involving three or more unknowns could be trivially formulated and included.

We generate all the equations to compose a linear system, $Ax = b$, where, A and b contain coefficients and independent terms respectively from all linear equations and x is the labeling solution. Since the number of equations and unknown is usually different, the system is turned into $(A^T A)x = A^T b$, where we find solution with a common least squares method, minimizing the error. The $(A^T A)$ matrix is symmetric by definition and positive semidefinite. Therefore, the equation can be solved applying Cholesky decomposition, specifically, LDLT decomposition, with the advantage of being fast and numerically stable.

This formulation opens the possibility of posing multiple unary equations per superpixel or multiple binary equations per edge, so the graph is in practice a multigraph. This could even lead to contradictory equations, in the sense that the corresponding coefficients of two equations may lead to different solutions if considered independently. This is not a problem for our solver, as it is not exact but a linear least squares minimization.

3.2 Initialization and interactive pipeline steps

The presented minimization formulation has the advantage of being linear. This allows us to split the system $Ax = b$ in the equivalent:

$$\begin{pmatrix} A_p \\ A_i \end{pmatrix} \cdot x = \begin{pmatrix} b_p \\ b_i \end{pmatrix} \tag{1}$$

The proposed pipeline (Fig. 1) enables a relatively expensive initialization step, but helps to minimize the calculations that depend on each user interaction, reaching interactive rates.

The *initialization* step first obtains superpixels and generates A_p and b_p. A_p and b_p contain the equations that are independent of the user interaction and therefore are only once. The included equations are just dependent on pixel positions and values, in particular, they correspond to the binary equations described in next Sec. 4.1.

The *interactive* step generates the rest of the system, matrix A_i and vector b_i, which contains the equations related to user interactions (the unary equations of Sec. 4.1.)

4 Application of the interactive dense labeling

The application described in this section is a proof of the use of our interactive dense labeling pipeline for interactive real applications. It simulates depth-of-field effects for a single image, which are generated from the depth estimation interactively updated by a human. Figure 3 and the supplementary material show results from our interactive application. As input, the user indicates, by means of strokes, some region of the picture in the foreground (blue) and in the background (different ranges of green, the darker, the further). Each user stroke is associated to a depth value. The closest objects are assigned with a depth value 1, while objects marked as the furthest get depth value 0. Each pixel affected by a user stroke generates an unary equation on the superpixel where it is contained. These equations are added to the system and combined with the pre-computed binary equations, and the system is efficiently solved to provide a dense depth map. Once the depth map is estimated, the user can choose the focus point, and the simulated image is generated by applying a variable Gaussian blur filter. All the process is interactive, hence the user can refine the input and get immediate feedback of the filter changes.

4.1 Steps to estimate a dense depth map from user interaction.

Our input is the original image and a sparse set of user-provided approximate depth values. The output of the dense-labeling is a depth map that we later use to apply the Depth of Field effect.

Superpixel segmentation. We considere that each superpixel has a single depth value. The segmentation used is independent to our pipeline and any other implementation could be used. We chose to use the algorithm SLIC [1], which groups pixels based on color properties. The only parameter used for this segmentation algorithm is the approximate number of superpixels. This number adjusts the superpixel size depending on the image resolution. The number of superpixels defines the number of unknowns in the linear system (number of depth labels) and therefore the size of the problem. As a consequence, the number of superpixels influences the speed of the solver at the interactive stage of the algorithm.

Binary equations. As we mentioned before, these equations are grouped into the *initialization* step (A_p and b_p). We establish binary relationship between depth values of connected superpixels. We consider two superpixels are connected following 8-neighbors connectivity. Given two connected superpixels p and q, their binary:

$$w_b(l_p - l_q) = 0, \tag{2}$$

where l_p and l_q are the unknown depth values (labels) of superpixels p and q respectively. Note that all binary equations tend to label equally neighboring superpixels. Therefore, we add a preconditioning factor w_b to prioritize the connections whose border pixel colors are similar in the CIE-Lab color space and secondly to limit the label propagation across the object boundaries. We assume for instance that neighbouring superpixels with similar color belong to the same object. Then, in our particular problem, they have high probability to be at the same depth. This factor w_b is defined as:

$$w_b = \begin{cases} w_c & \text{if } d_{pq} \leq D_{MAX} \\ 1 - w_c & \text{otherwise,} \end{cases} \tag{3}$$

where $w_c \in [0,1]$, and D_{MAX} is set experimentally to 0.05. Distance d_{pq} represents the color similarity between the boundary pixels of superpixels p and q. The Euclidean distance on each channel. The mean of each channel is normalized and obtained from the pixels from p which are in the boundary with q. w_c modulates the effect of the CIE-Lab color similarity over the depth propagation. Lower values of w_c tend to ignore color boundaries and therefore blur the whole depth map, while higher values (close to 1) may isolate certain superpixels. We experimentally set this value to $w_c = 0.99$.

Unary equations. The unary equations link the user input with superpixel depths which are grouped into the *interactive* step (A_i and b_i). With each user edition, we add new unary equations. Our depth-of-field application examples use initial values from user strokes, and each stroke has associated a real value in [0,1]. In particular, our application provides 5 buttons so the user can choose the desired depth for each stroke. We use uniformly distributed values for the 5 buttons. For each depth value z_i applied to a pixel i that belongs to a superpixel p ($i \in p$) we include the following equation into the system:

$$l_p = z_i, \tag{4}$$

where l_p is the (still) unknown depth label of the superpixel. When a user adds a stroke, our algorithm does not guarantee that the pixels underneath such stroke will keep the value after the propagation, due to the final system being solved as a linear least squares minimization. Each user stroke can affect several superpixels or different strokes can affect the same superpixel, generating contradictory (or redundant) superpixel constraints. Then, the user input can include many equations per superpixel (by applying larger strokes) which in practice enforces that depth value, while smaller strokes lead to fewer equations and therefore their overall weight on the system is smaller. This is an expected and desired behavior, which is not as straightforward in other labeling formulations. After each stroke the system is interactively updated and a dense depth map is obtained.

Global preconditioning. The global effect of unary and binary equations is image-dependent and rather unpredictable. We force a global system preconditioning by multiplying both sides of unary equations (4) by $\frac{w_u}{\#u}$ and both sides of binary equations (2) by $\frac{1-w_u}{\#b}$, where $\#u$ and $\#b$ are the total number of unary and binary equations and $w_u \in [0,1]$ is a user parameter that controls the weight of unary equations over binary ones. A value of $w_u = 0.4$ has been found appropriate and is used on all the results.

5 Experiments

This section evaluates the proposed interactive application and presents a quantitative and exhaustive evaluation of the performance of our pipeline.

5.1 System performance and parameters.

Execution time. An essential goal of our labeling approach is to guarantee the response time requirements for applications that interact with a user. Table 1 shows the execution time of each pipeline step in a typical execution measured in a desktop computer (Intel Core i5 2,5 GHz) with a sample image.

The *initialization steps* are executed only once at the beginning, while the image is being loaded to the application. The superpixel segmentation is the most expensive step from this stage, although the segmentation used is independent from our pipeline and any other implementation could be used. The number of superpixels affects directly the execution time to solve our system and therefore the execution time of the interactive loop. However, we limit the maximum number of superpixels and keep the interactivity of the system independent of the image size used. We set this maximum number of superpixels to 600. As we can see in Table. 1, this value allows our system to be interactive even with high resolution images.

The *interactive loop steps* (building new unary equations according to new user input, solving the system and re-estimating the depth map) are executed after each user interaction. All these steps are executed in less than a second (670 milliseconds), enough for this application to respond to the user at interactive rates.

Influence of Parameters in the Performance. As described in previous Sections 3 and 4, our formulation includes some control parameters which have been set experimentally to provide the best trade-off for our application (w_u=0.4, D_{MAX}=0.04 or w_c=0.99). Note that

Table 1: Typical execution time per step with an image of 2008x1340. Percentage values represent the contribution of each pipeline step over the total runtime.

Step	**Time** (seconds)	**Percentage**	**Step**	**Time** (seconds)	**Percentage**
Initialization stage			*Interactive loop*		
1. Superpixel segmentation (532 superpixels)	11.59	75 %	3. Add unary equations (1517 equations)	0.04	0.25 %
2. Add binary equations (234 equations)	3.25	21 %	4. Solve system	0.42	2.5 %
			5. Build depth map	0.21	1.25 %
Total:	14.84		**Total:**	0.67	

these recommended values correspond to our particular case of dense depth estimation. If the pipeline is used for estimating a different real magnitude they would probably change. The supplementary material includes the analysis run to determine these values.

5.2 Interactive dense labeling evaluation

We focus on the part of the pipeline that estimates the labeling (step 4 from Fig. 1), since it is the only common part in all the approaches compared. We compare our results against state-of-art dense labeling approaches using two very different inputs (Fig. 2): a dense but unreliable input labeling obtained automatically from two stereo images and a sparse but more reliable input labeling obtained from a few user strokes. Our work is focused on generating a good estimation using just a few pieces of information from the user. However, MRF-based methods are focused on obtaining an accurate solution using an available initial estimation, typically distributed all over the image. Therefore, they are suboptimal when a very sparse input is used, as we confirmed with the results in this section.

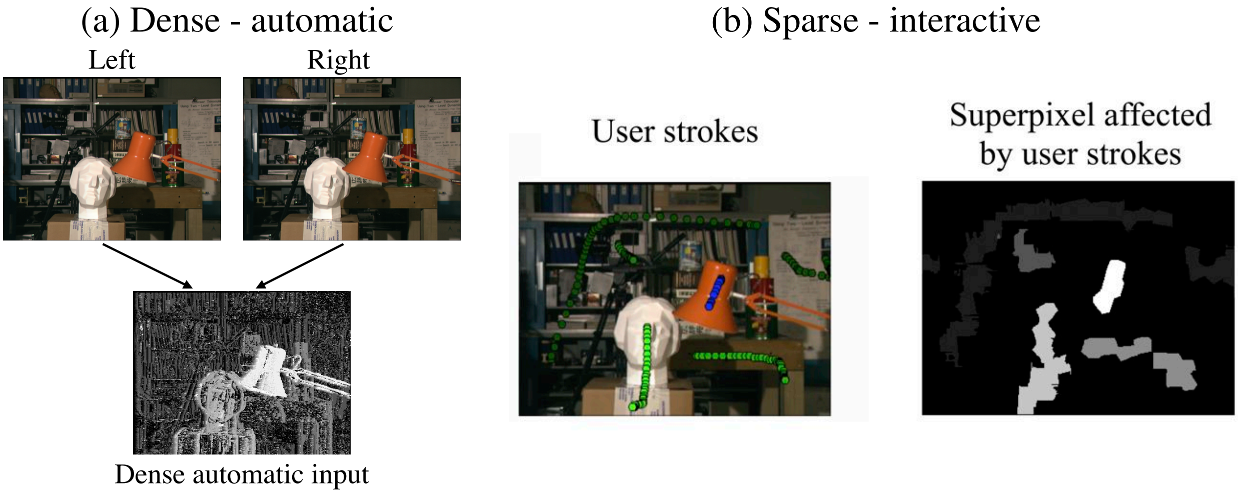

Figure 2: Types of labelling initialization used in our experiments.

Reference labeled data used. For these labeling experiments we use well known public data [25] [26] designed to evaluate stereo algorithms. In this dataset the ground truth labels represent the disparity between corresponding pixels from two images. We chose this dataset because it had public results for recent dense labeling related methods.

State-of-art approaches considered. We compare our algorithm to the following state-of-the-art algorithms described in a well known comparative of MRF-based dense labeling approaches [30]: iterated conditional modes (*ICM*) [4]; Graph Cut (*GC*) with α-expansion moves (*Expansion*) and $\alpha\beta$-swap moves (*swap*) [6]; two implementations (*BP-S* and *BP-M*) of loopy belief propagation [31], and Sequential three-re-weighted message passing (*TRW-S*) [13]. Besides, since our focus is on speed and interactive properties, we include the recently presented block coordinate descent algorithm (*BCD*) [7], which was developed with an emphasis on speed, and a Random Walk based approach, the implementation provided by [11], which was designed as an interactive segmentation tool. This formulation is very close to ours and it is also based on a linear equation system. Note that both MRF and RW implementations are based on pixel-wise formulations, while our work is superpixel-based, therefore it has a disadvantage over pixel-wise techniques in terms of accuracy but presents higher efficiency without much penalization in accuracy, as we can see in the following experiments.

We also include a superpixel-based version of *GC*, which is the only related method whose available implementation can be directly adjusted to support a superpixel formulation. We adapt its MRF-graph edges to use superpixels as nodes, as our approach does, and adapt its MRF unary and binary cost functions to include the same constraints as our unary and binary equations.

Error measure. To compare the quality of the different algorithms solutions, we measure the error obtained in each solution as the *mean of the differences (or mean error)* between each pixel in the solution and the same pixel in the ground truth.

Dense labeling input. As we mentioned, the MRF based methods use all the initialization information they can extract from two stereo images, by running an automatic disparity estimation algorithm for stereo. In order to evaluate our pipeline and the RW approach in similar settings than the other MRF studied approaches we need an initial disparity estimation. As initial disparity map, we use the same initial map utilized in the ICM algorithm [30], Fig. 2 (a). Note that this noisy and unreliable input is not the target case for our method.

Sparse user input. In this test, the input for all methods evaluated is the same set of pixels initialized from user strokes, Fig. 2 (b). As the user associates each pixel stroke with a depth value, we can create a sparse initial depth map to initialize all the methods compared here. In this experiment, the user input consists on 5 different levels of depth assigned to different user strokes. Note that the input image in Fig. 2 (b) highlights the whole superpixels affected by user strokes, but actually we are only including one unary equation per pixel affected.

Discussion of the results. Table 2 shows error and execution time (measured on a standard desktop machine Intel Core i5 2,5 GHz) for the dense labeling estimation in three of the tests run. In the supplementary material, we show the final disparity estimation obtained by each of the evaluated methods.

As previously mentioned, MRF-based methods are focused on obtaining an accurate solution using an dense initial estimation and, as we can see in Table 2 (a), these methods obtain the best estimation in terms of accuracy. However, our algorithm is faster than any of the other algorithms in this comparative, while keeping comparable accuracy than the fastest ones (which still are one order of magnitude slower). Note that the direct comparison with RW implementation is somehow unfair as the available code is implemented in Matlab, while

Table 2: Execution time (*seconds*) and mean error (*err*) for dense labeling obtained for 3 test images (name and resolution in each column) with automatic input *(a)* and user input *(b)* initialization. *#Labels*: number of disparity levels considered on each test.

(a) AUTOMATIC DENSE INPUT

Method	**Tsukuba** (384x288) #Labels=16		**Venus** (434x383) #Labels=20		**Teddy** (450x375) #Labels=60	
	time	**err**	**time**	**err**	**time**	**err**
Pixel-based						
ICM [4]	0.520	0.12	0.460	0.10	1.900	0.13
Expansion [6]	2.220	0.02	6.940	0.02	19.90	0.05
Swap [6]	2.250	0.02	7.010	0.02	12.60	0.05
TRW-S [13]	8.840	0.02	115.0	0.02	158.0	0.05
BP-S [31]	1.370	0.02	8.690	0.03	21.20	0.05
BP-M [31]	13.30	0.02	–	–	193.0	0.05
BCD [7]	0.920	0.09	1.500	0.17	2.760	0.08
RW [11]	0.200*	0.12	0.400*	0.20	0.600*	0.16
Superpixel-based						
Expansion	3.090	0.06	6.320	0.10	6.210	0.08
Ours	**0.002**	0.06	**0.005**	0.10	**0.005**	0.09

–: the method did not converge to a solution
*: execution time is measured in Matlab.

(b) SPARSE USER INPUT

Method	**Tsukuba** (384x288) #Labels=5		**Venus** (434x383) #Labels=5		**Teddy** (450x375) #Labels=5	
	time	**err**	**time**	**err**	**time**	**err**
Pixel-based						
ICM [4]	N/A	N/A	N/A	N/A	N/A	N/A
Expansion [6]	N/A	N/A	N/A	N/A	N/A	N/A
Swap [6]	N/A	N/A	N/A	N/A	N/A	N/A
TRW-S [13]	N/A	N/A	N/A	N/A	N/A	N/A
BP-S [31]	N/A	N/A	N/A	N/A	N/A	N/A
BP-M [31]	24.40	0.14	34.20	0.09	35.10	0.18
BCD [7]	–	–	–	–	–	–
RW [11]	0.500*	0.13	0.600*	0.20	0.700*	0.09
Superpixel-based						
Expansion	4.170	0.06	6.370	0.14	7.960	0.09
Ours	**0.002**	0.06	**0.005**	0.15	**0.005**	0.06

–: can't provide to available implementation.
*: execution time is measured in Matlab.

the rest use C++, so we could expect the speed up of around one order of magnitude typically observed between these two environments. Theoretically, the RW [11] requires solving a linear system per label in the solution, while our approach only solves one, independently of the number of labels. For example, in tests of Table 2 (b) where we use five labels, the RW solver would be around five times slower than our approach.

Table 2 (b) shows our approach obtains also the best estimation in terms of accuracy, although we could say that the three best approaches obtained results of comparable quality. We could say that by definition, superpixel-based approaches are typically less accurate than pixel-based methods but faster to compute. Our work is superpixel-based and therefore has advantage in terms of time cost over pixel-wise techniques. However, this time advantage is achieved at a small sacrifice on accuracy, as it can be seen from the quality of final estimations obtained in the Table 2.

An important advantage of our method is the flexibility. First, as we use a continuous label set, we do not need to choose the number of possible final labels a priori. Second, only MRF-based methods can include in the final solution intermediate label values like us (but at much higher cost), but RW can only assign a choice among the input labels, i.e., if five depth values are given as input, only those five values will compose the final solution. Another interesting advantage of our method is that the execution time does not fluctuate a lot with image dimensions or number of labels, since is not directly depending on any of those magnitudes, but on the number of superpixels.

5.3 Interactive depth-of-field application examples

Figure 3 shows examples of the results obtained with our application (more results on supplementary material). The proposed pipeline uses our interactive dense labeling estimation as key step, in this case depth estimation, which has good quality compared to state of the art methods. The final result of the interactive pipeline and its quality are subjective to the user so we can only show qualitative evaluation of this application. The video included demonstrates the real time execution and behavior of the application.

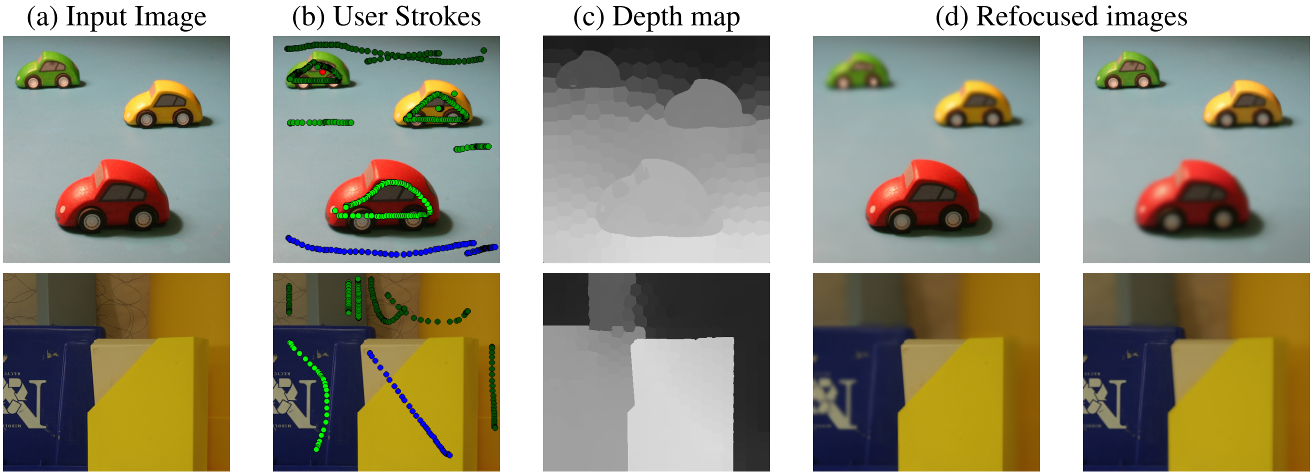

Figure 3: Results from interactive depth-of-field application. With a few user strokes (b) over the original image (a), the user can quickly get different depth-of-field effects in the image (d). The estimated depth map (c) is used to generate the depth-of-field effects.

6 Conclusions

We have developed an efficient approach for dense labeling based on a superpixel linear least squares formulation. As demonstration of the suitability of our approach for interactive applications, we have presented an interactive application that, given a single image without any additional information, enables an unskilled user to easily generate synthetic depth-of-field blur effect on the image. The depth-of-field post-processing requires a per-pixel depth value, which is obtained from a very sparse and approximate depth user input. The proposed dense labeling technique has great flexibility to model this problem. Since our dense labeling system models the problem by means of linear equations, the connectivity graph model can be easily augmented without much computational cost.

Furthermore, our formulation has the advantage of providing an interactive solver, which is key to applications such as our proposed depth-of-field editor.

Our accuracy is limited by the number of superpixels and by the accuracy of the input, which, coming from the user, it is probably rather approximate. Still, we have shown that we yield results which are accurate enough for many applications and often comparable to other slower state of the art methods. Besides, since we target an interactive technique the user can always refine and improve the input iteratively.

We believe that our approach will inspire future research for interactive editing applications based on dense labeling. The interactivity rates of our approach, together with the aforementioned expanded flexibility, provide great potential for other applications dealing with tasks such as semantic labeling, image restoration, intrinsic imaging or inpainting.

7 Acknowledgments

This work has been funded by Project DPI2015-65962-R (MINECO/FEDER, UE). The authors would like to thank Diego Gutierrez for his insightful comments and discussions on this research.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[2] Shawn Andrews, Ghassan Hamarneh, and Ahmed Saad. Fast random walker with priors using precomputation for interactive medical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, pages 9–16. Springer, 2010.

[3] Stephan Bergmann, Tobias Ritschel, and Carsten Dachsbacher. Interactive appearance editing in rgb-d images. In *19th International Workshop on Vision, Modeling and Visualization*, pages 1–8. Eurographics Association, 2014.

[4] Julian Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 259–302, 1986.

[5] Nicolas Bonneel, Kalyan Sunkavalli, James Tompkin, Deqing Sun, Sylvain Paris, and Hanspeter Pfister. Interactive intrinsic video editing. *ACM Transactions on Graphics (TOG)*, 33(6):197, 2014.

[6] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(11): 1222–1239, 2001.

[7] Qifeng Chen and Vladlen Koltun. Fast mrf optimization with application to depth reconstruction. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 3914–3921, 2014.

[8] Francesco Di Renzo, Claudio Calabrese, and Fabio Pellacini. Appim: linear spaces for image-based appearance editing. *ACM Transactions on Graphics (TOG)*, 33(6):194, 2014.

[9] A Fabijanska and J Goclawski. The segmentation of 3d images using the random walking technique on a randomly created image adjacency graph. *IEEE Trans. Image Processing*, 24(2):524, 2015.

[10] Daniel Freedman. An improved image graph for semi-automatic segmentation. *Signal, Image and Video Processing*, 6(4):533–545, 2012.

[11] Leo Grady. Random walks for image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783, 2006.

[12] Derek Hoiem, Alexei A Efros, and Martial Hebert. Recovering surface layout from an image. *International Journal of Computer Vision*, 75(1):151–172, 2007.

[13] Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006.

[14] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 689–694. ACM, 2004.

[15] Miaomiao Liu, Mathieu Salzmann, and Xuming He. Discrete-continuous depth estimation from a single image. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 716–723, 2014.

[16] Angeles Lopez, Elena Garces, and Diego Gutierrez. Depth from a Single Image Through User Interaction. In *Spanish Computer Graphics Conf.*, pages 11–20. The Eurographics Assoc., 2014.

[17] Ching-Hu Lu, Yu-Chen Ho, Yi-Han Chen, and Li-Chen Fu. Hybrid user-assisted incremental model adaptation for activity recognition in a dynamic smart-home environment. *IEEE Trans. on Human-Machine Systems*, 43(5):421–436, Sept 2013.

[18] Qing Luan, Fang Wen, Daniel Cohen-Or, Lin Liang, Ying-Qing Xu, and Heung-Yeung Shum. Natural image colorization. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 309–320. Eurographics Association, 2007.

[19] Branislav Mičušík and Jana Košecká. Multi-view superpixel stereo in urban environments. *International Journal of Computer Vision*, 89(1):106–119, 2010.

[20] Jifeng Ning, Lei Zhang, David Zhang, and Chengke Wu. Interactive image segmentation by maximal similarity based region merging. *Pattern Recognition*, 43(2):445–456, 2010.

[21] Raymond Phan and Dimitrios Androutsos. Robust semi-automatic depth map generation in unconstrained images and video sequences for 2d to stereoscopic 3d conversion. *Multimedia, IEEE Transactions on*, 16(1):122–136, 2014.

[22] Orod Razeghi, Guoping Qiu, Hywel Williams, Kim Thomas, and IMA VIPLAB. Skin lesion image recognition with computer vision and human in the loop. *Medical Image Understanding and Analysis (MIUA)*, pages 167–172, 2012.

[23] Xiaofeng Ren, Liefeng Bo, and Dieter Fox. Rgb-(d) scene labeling: Features and algorithms. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 2759–2766. IEEE, 2012.

[24] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. 3-d depth reconstruction from a single still image. *International Journal of Computer Vision*, 76(1):53–69, 2008.

[25] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47 (1-3):7–42, 2002.

[26] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *IEEE Conf. on Computer Vision and Pattern Recognition*, volume 1, pages I–195, 2003.

[27] Rui Shen, I Cheng, Xiaobo Li, and A Basu. Stereo matching using random walks. In *International Conference on Pattern Recognition*, pages 1–4, 2008.

[28] Dheeraj Singaraju, Leo Grady, and René Vidal. P-brush: Continuous valued mrfs with normed pairwise distributions for image segmentation. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1303–1310, 2009.

[29] Hao Su, Qixing Huang, Niloy J. Mitra, Yangyan Li, and Leonidas Guibas. Estimating image depth using shape collections. *Transactions on Graphics (Special issue of SIGGRAPH 2014)*, 2014.

[30] Richard Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall Tappen, and Carsten Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 30(6):1068–1080, 2008.

[31] Marshall F Tappen and William T Freeman. Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters. In *Computer Vision, 2003. Proc. 9th IEEE Int. Conf. on*, pages 900–906. IEEE, 2003.

[32] Olivier Teboul, Loic Simon, Panagiotis Koutsourakis, and Nikos Paragios. Segmentation of building facades using procedural shape priors. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 3105–3112, 2010.

[33] Joseph Tighe and Svetlana Lazebnik. Superparsing. *International Journal of Computer Vision*, 101(2):329–349, 2013.

[34] C. Wah, S. Branson, P. Perona, and S. Belongie. Multiclass recognition and part localization with humans in the loop. In *IEEE Int. Conf. on Computer Vision*, pages 2524–2531, Nov 2011.

[35] Koichiro Yamaguchi, Tamir Hazan, David McAllester, and Raquel Urtasun. Continuous markov random fields for robust stereo estimation. *Computer Vision- ECCV*, pages 45–58, 2012.