# Joint Affinity Propagation for Multiple View Segmentation

Jianxiong Xiao Jingdong Wang Ping Tan Long Quan
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
{csxjx, welleast, ptan, quan}@cse.ust.hk

## Abstract

*A joint segmentation is a simultaneous segmentation of registered 2D images and 3D points reconstructed from the multiple view images. It is fundamental in structuring the data for subsequent modeling applications. In this paper, we treat this joint segmentation as a weighted graph labeling problem. First, we construct a 3D graph for the joint 3D and 2D points using a joint similarity measure. Then, we propose a hierarchical sparse affinity propagation algorithm to automatically and jointly segment 2D images and group 3D points. Third, a semi-supervised affinity propagation algorithm is proposed to refine the automatic results with the user assistance. Finally, intensive experiments demonstrate the effectiveness of the proposed approaches.*

## 1. Introduction

Structure from motion takes an image sequence of a rigid (or static) object as the input and recovers the camera poses and a cloud of 3D points. After many years of continuous research, nowadays, the structure from motion algorithm, such as [6], can robustly and accurately recover hundreds of thousands of points and all the camera poses.

The reconstructed 3D points, however, are unstructured in space, therefore are not yet sufficient for creating a geometric model of the underlying objects. To structure the available 3D points and registered 2D images, recent researches [10, 11] show that a joint segmentation of the reconstructed 3D points and the multiple 2D images is fundamental for the subsequent modeling applications. Obviously, the concept of object is subjective, and learning from the user assisted 2D image segmentation gives the object segmentation more useful information. Hence, we wish to segment 3D points and 2D images into groups, where each group represents a distinct object. This segmentation can be regarded as a post-processing step of structure from motion, which provides semantic organizations of the recovered 3D points. And this is very useful for the subsequent 3D modeling of the scene.

### 1.1. Related work

The segmentation can be performed individually on the 3D points or 2D images. The 3D points, without considering the 2D images, is a little like the range data. The segmentation is usually based on local geometric characterizations, which is insufficient to obtain semantic segmentation. On the other hand, segmenting 2D natural images is a well-studied topic, and many successful methods, such as [13, 15], were proposed. It is a choice to individually segment each image of the multiple view sequence. For example, a pure 2D segmentation approach to reconstruct small-leaf trees is proposed in [14]. However, it definitely misses very rich 3D information or 2D correspondence information, and hence can not obtain satisfactory results on large objects with more complex color, texture and shape information.

Some methods were proposed to utilize the motion information for multiple image segmentation. The layered approaches originated from [16] usually do not directly adopt 3D reconstruction information. In [17, 18], motion estimation and segmentation on the extracted correspondences between frames are performed, then layer assignment (i.e. pixel label) is obtained through propagating the labels of the corresponding pixels. In [9], the joint inference of motion estimation and labeling is solved using the Expectation Maximization algorithm.

The modern stereo matching framework, such as [5], is very similar to the layered approach, and it in essence discretizes the 3D space into a few layers. Bilayer segmentation of binocular stereo video in [4], a simplest layered representation, probabilistically fused the stereo cues and learned appearance model to separate the figure from the background.

### 1.2. Our approach

In this paper, we explore for multiple view segmentation by jointly utilizing and grouping the 2D and 3D data. The availability of both 2D and 3D data can bring complementary information for segmentation. For instance, some

Proceedings of 11th IEEE International Conference on Computer Vision (ICCV2007)

objects are obviously separable in 3D whereas others are clearly cut out by image boundary information even if they are closely connected in space.

In this paper, we follow the joint segmentation framework proposed in [11] for the multiple view segmentation. The main contributions of this paper include the similarity based on the joint 2D and 3D information, and the two clustering algorithms: hierarchical sparse affinity propagation and semi-supervised graph-based affinity propagation.

The rest of this paper is organized as follows. The preliminary processes are presented in Sec. 2. And in Sec. 3 we formulate the joint segmentation problem. Sec. 4 introduces the interactive strategy learning from the user assistance. And the novel optimization method is proposed in Sec. 5. The experimental results are presented in Sec. 6, Finally, Sec. 7 concludes this paper.

## 2. Preliminary Processes

We capture multiple view images from a number of different overlapping views around an object using a hand-held camera. We use the approach described in [6] to compute the camera poses and a quasi-dense cloud of reliable 3D points in space.

Each reconstructed 3D point corresponds to several 2D points in different views. To utilize the texture information, we may also associate patches with each 3D point. We perform the competitive region growing algorithm on each image by taking projected 2D points as the seeds. In this way, each 3D point is associated with image patches in some views.

An observation is that although structure from motion (SFM), the quasi-dense approach in our case, can recover the 2D correspondences and the corresponding 3D position, inaccuracy is unavoidable. Furthermore, SFM, which is based on interest point matching, generate points partially on the object boundary. In our case, the quasi-dense points are obtained by propagating the points of interest. Consequently, a small error around the object boundary may result in a large color difference. We have to use some techniques to enhance the robustness of the 2D projection estimation. Hence, we need to process the set of patches to reduce the error induced by the 2D projection inaccuracy. Here, we propose a robust patch filtering process to remove some outlier patches. The algorithm is shown in Alg. 1.

**Algorithm 1** Robust patch filtering

1. Compute the mean color $\mathbf{c}_i$ for the patch in $i$-th image.
2. Compute the similarity between the patch pairs by $s_{ij} = -\left\|\mathbf{c}_i - \mathbf{c}_j\right\|^2$.
3. Perform affinity propagation [1] to cluster this mean color set $\{\mathbf{c}_i\}$ to obtain exemplars.
4. Keep the cluster corresponding to the exemplar that has the largest number of supporting patches as the representative patch vectors, and denoted as $\{P_i\}$.

## 3. Formulation

Let $I = \{I_i\}$ be the set of $n$ images with $i = 1, \ldots, n$. Each image $I_i$ is represented by a set of regions, i.e. $I_i = \{(\mathbf{u}_k, P_k)\}$ with $k$ up to the number set by the visible projections of the quasi-dense points in this view, and $\mathbf{u}_k$ is the projection coordinate in 2D image space and $P_k$ is the corresponding patch. It is assumed that all the images are fully calibrated with respect to a common coordinate frame. We define a *joint point* $\mathbf{x}$ to be a vector composed of the 3D coordinates $(x, y, z)$ of a point in space and all its corresponding patches $P_i$ in all images, i.e. $\mathbf{x} = ((x, y, z), (\mathbf{u}_1, P_1), \cdots, (\mathbf{u}_n, P_n))$, where each projection satisfies $\mathbf{u}_i = \mathbf{P}_i(x, y, z, 1)^T$ for the projection matrix $\mathbf{P}_i$ of the $i$-th camera. The correspondence information is encoded in the joint point representation. And each joint point $\mathbf{x}$ is associated with an $n$-dimensional visibility vector $\mathbf{v}$ with binary values to indicate that $\mathbf{u}_i$ is visible in the $i$-th image if the $i$-th component is 1, and invisible otherwise. A segmentation is a set of labels $L = \{l_k\}$, and each of them $l_k$ assigns a set of joint points to a common group. Here $X = \{\mathbf{x}_j\}$ is the given set of joint points, $V = \{\mathbf{v}_j\}$ is the given set of visibilities, and $X$ and $V$ are given by the quasi-dense reconstruction in our case. We now want to get the inference of $L$, given $X$, $V$ and $I$. Similar to [11], we define a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with joint points as nodes, in which edge weights denote a local similarity measure between the two joint points in the graph $\mathcal{G}$. Different from [11], we generalize the joint point from a pixel level to a region (superpixel) level to help the definition of the joint similarities.

### 3.1. Graph construction

The set of edges $\mathcal{E}$ is constructed using the $k$-Nearest Neighbor ($k$-NN) technique. To guarantee that the joint points $i$ and $j$, $(i, j) \in E$, must be both visible at least in one view, each view is associated with a set of joint points that are visible in this view. We then build for each view a $k$-NN network on the corresponding set of joint points according to the 3D Euclidean distance. Finally, we combine those networks together to reach a graph on the entire joint points.

### 3.2. Joint similarity

The joint use of 3D and 2D information for better segmentation, since all our images and 3D data are perfectly

registered, is discovered by [10, 11]. All these useful information is encoded in the weights on the edge. For similar nodes, similar labels should be selected for them. Therefore, a similarity is defined on each edge to characterize the smoothness of the labels. The quality of a segmentation fundamentally depends on the similarity, and hence we seek to define it jointly from both 3D and 2D features.

**3D similarity** The points that are closer in space tend to have a higher probability belonging to the same group, i.e. the distance between the points of the same group is smaller than that of the points in different groups. We naturally take this spatial distance as a similarity measure $s_{3d}(i,j) = -\frac{||\mathbf{p}_i - \mathbf{p}_j||^2}{2\sigma_{3d}^2}$, where $\sigma_{3d}^2$ is the expectation $E(||\mathbf{p}_i - \mathbf{p}_j||^2)$ and $\mathbf{p} = [x\ y\ z]^T$. In addition to the 3D Euclidean distance, the normal directions are also important for shape smoothness. We incorporate the difference between normal directions into the similarity and define $s_{3n}(i,j) = -\frac{||\mathbf{n}_i - \mathbf{n}_j||^2}{2\sigma_{3n}^2}$, where $\mathbf{n}_j$ is the normal direction vector of point $j$, approximately estimated from its neighbor points, and $\sigma_{3n}^2$ is the expectation $E(||\mathbf{n}_i - \mathbf{n}_j||^2)$. The final 3D similarity is given by $s_3(i,j) = s_{3d}(i,j) + s_{3n}(i,j)$.

**2D color similarity** Since a joint point $\mathbf{x}$ is associated with the image colors, we can define a similarity function encoding the color differences as $s_c(i,j) = -\frac{||E(\mathbf{c}_i) - E(\mathbf{c}_j)||^2}{2\sigma_c^2}$, where $\sigma_c^2 = E(||E(\mathbf{c}_i) - E(\mathbf{c}_j)||^2)$, and $E(\mathbf{c}) = \frac{1}{|\mathbf{v}|_1}\sum_{i=1}^{n}\mathbf{c}_i$. This color consistency between joint points is intuitively estimated using their average colors, since different points may have different numbers of visible color features. Averaging the colors leads to a more stable solution. However, this similarity function only makes sense between the objects with apparent different colors.

In case of apparent similar colors, image contour features, similar to [8], should be incorporated into the similarity. It is assumed at present that each pixel $\mathbf{u}$ in view $I_v$ is associated with a response $g_v(\mathbf{u})$ to show the degree of the pixel lying on a contour point. The endpoints of the edge $(i,j)$ must both be visible at least in one view, meaning that the line segment $[i,j]$ must correspond to a line segment visible in the same view. We can use the following similarity measurement

$$s_{ic}(i,j) = -\frac{\text{med}_v\{\max_{t_v \in [i,j]_v} g_v(t_v)\}}{2\sigma_{ic}^2},$$

where the inner term $\max_{t_v \in [i,j]_v} g_v(t_v)$ finds the maximum contour response along the projected line segment $[i,j]_v$ in view $v$, the outer term $\text{med}_v\{\cdot\}$ tries to seek the median contour response in all possible views, and $\sigma_{ic}$ is the variance of the median contour responses of all line segments. The response $g_v(\mathbf{u})$ is calculated from an edge map obtained by the similar orientation filter bank used in [8, 11].

(a) Strokes (b) Trimap (c) 3D projection

Figure 1. User assistance. (a) shows the strokes scribbled by the user. (b) shows the segmentation result in a trimap representation by our semi-supervised AP method. In (c), the 3D projections inside and outside the white-bounded region are assigned different hard labels, and are used to propagate the labels into the other joint points invisible in this view using our semi-supervised AP method.

**Patch histogram similarity** To ultilize texture similarity, we express each patch vector in term of multi-resolution histograms [2]. That is, for each joint point, we collect all its patches $\{P_1, \ldots, P_k\}$ remained after filtering, then build an average color histogram $h_0$. Without losing the spatial information, we further downsample the patches $t-1$ times and compute several normalized color histograms $h_1, \ldots, h_{t-1}$. Hence, a joint point now corresponds to a vector of histograms $h = [h_0, h_1, \ldots, h_{t-1}]$.

In this way, for any two joint points $i$ and $j$, with the histogram representations $h^i$ and $h^j$, their patch similarity is defined as

$$s_t(i,j) = -d(h^i, h^j) = -\frac{1}{t}\sum_{k=0}^{t-1} d(h_k^i, h_k^j),$$

where $d(\cdot,\cdot)$ is the dissimilarity measures for histograms. Here, we choose the Kullback-Leibler divergence.

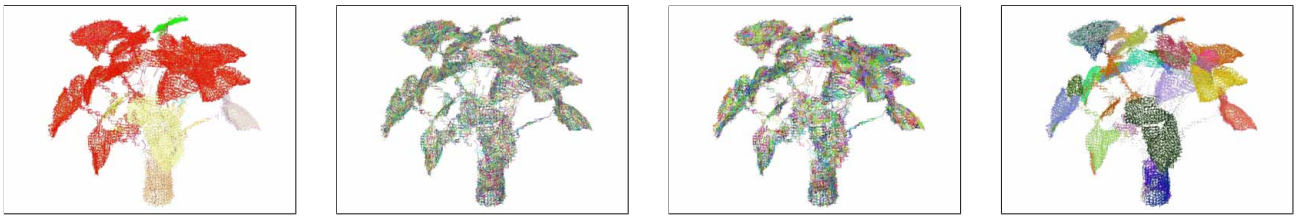
Finally, we are able to perform a simple addition of the similarities to define the joint similarity to be

$$s(i,j) = s_3(i,j) + s_c(i,j) + s_{ic}(i,j) + s_t(i,j).$$

## 4. Learning

The concept of segmentation is obviously subjective. Hence, some user assistant information will greatly improve the segmentation. In recent years, interactive 2D image matting [7] are very successful, and a semantic segmentation was induced from a training example in [12]. Here, we use a similar way to allow the user conceptually group different objects in some 2D images. To specify an object, the user marks a few lines on the images by dragging the mouse cursor while pushing down a button. An example of our user-interface is shown in Fig. 1(a), where different objects are marked by strokes with different colors.

We can segment 2D images using the semi-supervised contraction method, which will be discussed in Subsec. 5.3.



(a) Connected components (b) Level 1 (c) Level 5 (d) Level 17

Figure 2. Demonstration of hierarchical sparse affinity propagation. (a) shows the connected components on the initial $k$-NN graph. (b) shows the first-level result of hierarchical sparse affinity propagation. (c) shows the fifth-level result. (d) shows the final result.

We want to make use of these strokes and segmentation information to help segment the other views. This is more practical since we always have about 30 views and the user may not want to draw strokes on every view. Here, we make the assumption that all the surfaces in the scene are Lambertian. Under this assumption, the appearance models of all objects are roughly the same in all views. Hence, for the joint points with visible projections on this segmented 2D image, we directly set their labels the same as their projection's labels respectively, which can be obtained from segmented images. To handle the ambiguity of the projections near the boundary, such as the projections in the white area in Fig. 1(c), we regard the joint points corresponding to them as unlabeled joint points.

## 5. Optimization

Affinity propagation [1] (AP) is a recently-developed clustering algorithm, which clusters data points based on the similarities. It aims to find several exemplars such that the sum of the similarities between the data points and the corresponding exemplars is maximized. In this section, we propose two variants of the original algorithm to jointly segment the joint points. 1) A hierarchical sparse affinity propagation is proposed for automatic clustering. 2) A semi-supervised graph-based affinity propagation is proposed to refine the clustering result interactively.

### 5.1. Affinity propagation

In this subsection, we review the affinity propagation algorithm described in [1]. There are two kinds of messages communicated between data points, i.e. *responsibility* and *availability*, and each takes a different kind of competition into account.

To begin with, the availabilities are initialized to zero: $a(i,k) = 0$. The *responsibility* $r\,(i,k)$, sent from data point $i$ to candidate exemplar point $k$, reflects the accumulated evidence for how well-suited point $k$ is to serve as the exemplar for point $i$, taking into account other potential exemplars for point $i$. The responsibilities are computed as

$$r(i,k) \leftarrow s(i,k) - \max_{k' \neq k}\{a(i,k') + s(i,k')\}.$$

For $k = i$, the *self-responsibility* $r(k,k)$ reflects accumulated evidence that point $k$ is an exemplar, based on its input preference tempered by how ill-suited it is to be assigned to another exemplar.

The *availability* $a\,(i,k)$, sent from the candidate exemplar point $k$ to point $i$, reflects the accumulated evidence for how appropriate it would be for point $i$ to choose point $k$ as its exemplar, taking into account the support from other points that point $k$ should be an exemplar. Whereas the above responsibility update lets all candidate exemplars compete for ownership of a data point, the following availability update gathers evidence from data points as to whether each candidate exemplar would make a good exemplar:

$$a(i,k) \leftarrow \min\{0, r(k,k) + \sum\nolimits_{i' \notin \{i,k\}} \max\{0, r(i',k)\}\}.$$

The *self-availability* $a(k,k)$ is updated differently:

$$a(k,k) \leftarrow \sum\nolimits_{i' \neq k} \max\{0, r(i',k)\}.$$

This message reflects accumulated evidence that point $k$ is an exemplar, based on the positive responsibilities sent to candidate exemplar $k$ from other points.

After the convergence, availabilities and responsibilities are combined to identify exemplars. For point $i$, its corresponding exemplar is obtained as

$$k^* = \arg\max_k\{a(i,k) + r(i,k)\}. \tag{1}$$

This means to either identify point $i$ as an exemplar if $k^* = i$, or identify data point $k^*$ that is the exemplar for point $i$.

### 5.2. Hierarchical sparse affinity propagation

Affinity propagation on a sparse graph, called sparse affinity propagation, is more efficient as pointed in [1]. The implementation is similar to the description in Subsec. 5.1 except that the responsibilities and availabilities are only updated on the connected edges. Then sparse affinity propagation runs in $O(T|\mathcal{E}|)$ time with $T$ the number of the iterations and $|\mathcal{E}|$ the number of the edges. In our sparse graph, the time complexity is $O(Tn)$ since $|\mathcal{E}| = O(n)$.

We observed, however, according to the original sparse implementation in [1], the number of the data points that

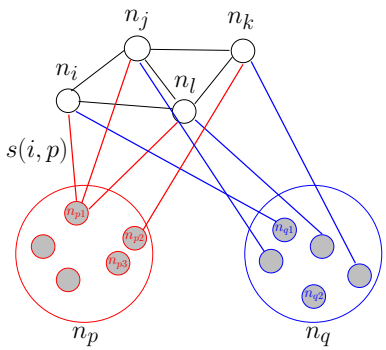



Figure 3. Illustration of semi-supervised contraction. $n_p$ and $n_q$ are candidate exemplars. The data points, $n_i, n_j, n_l, n_k$, are internally connected if they are neighbors, and are directly connected with the two candidate exemplars. $n_p$ or $n_q$ attracts competitively the data points in the semi-supervised AP algorithm.

have the same exemplar $i$ is at most $degree(i)$, where $degree(i)$ is the number of nodes connecting $i$. This is because point $i$, the exemplar for point $j$, must directly connect point $j$ according to Eqn. 1 and the number of the points that connect point $i$ is $degree(i)$. This will result in unexpectedly too many fragments as shown Fig. 2(b).

To handle this problem, we propose a hierarchical sparse affinity propagation method. After obtaining the exemplars on the original sparse graph, we run again sparse affinity propagation on the exemplars by constructing a sparse graph on the exemplars and connecting the exemplars $c_i$ and $c_j$ if the point with its exemplar $c_i$ is connected with at least one point whose exemplar is $c_j$. Then we can rerun sparse affinity propagation on the new exemplars until obtaining satisfactory results. Compared with the spectral clustering approach in [11], the hierarchical sparse affinity propagation is more efficient, running in $O(TLn)$ with $T$ the number of the iterations and $L$ the number of the hierarchies, and more effective. One example is shown in Fig. 2.

### 5.3. Semi-supervised contraction

The original affinity propagation is an unsupervised clustering method. To utilize the partially labeled nodes, an efficient and effective semi-supervised affinity propagation method is proposed.

First, we group the nodes $n_{p1}, n_{p2}, \cdots$ that have the same known label, and contract these nodes into a single new node $n_p$. Similarly, $n_{q1}, n_{q2}, \cdots$ are contracted into a single node $n_q$. A toy example is shown in Fig. 3. And we set the preferences of the contracted nodes to zero, i.e. $s(p,p) = s(q,q) = 0$ (this is equivalent that the exponential similarity is 1.).

Then, we update the new edges $\bar{\mathcal{E}}$ based on the original edges $\mathcal{E}$ and consider the similarities between all the remaining nodes $n_i \notin n_p \cup n_q$ and the contracted nodes $n_p, n_q$. We connect $n_i \notin n_p \cup n_q$ and $n_p, n_q$, and cut all the edges between the nodes in $n_p, n_q$. We set the similarities on the connected edges $\bar{\mathcal{E}}$ as follows.

1. For the similarity on $(n_i, n_j) \in \bar{\mathcal{E}}$ if $n_i \notin n_p \cup n_q$, $n_j \notin n_p \cup n_q$ and $(n_i, n_j) \in \mathcal{E}$, we just copy the similarity from the original weighted graph.

2. Considering the similarity on $(n_i, n_t) \in \bar{\mathcal{E}}$ if $p_t \in n_t$ and at least one $p_t$ such that $(n_i, p_t) \in \mathcal{E}$, $n_t \in \{n_p, n_q\}$, we set it as the largest similarity between node $n_t$ and any node $p_t \in n_t$ as $s(i,t) = \max_{p_t \in n_t} s(i, p_t)$.

3. For edge $(n_i, n_t) \in \bar{\mathcal{E}}$ if there is no point $p_t \in n_t$ such that $(n_i, p_t) \in \mathcal{E}$), we use the distance of the shortest path between $n_i$ and $n_t$ to estimate their similarity $s(i,t) = \max_{path_{i,t}} \sum_{(j,k) \in path_{i,t}} s(j,k)$.

Finally, when the algorithm converged, availabilities and responsibilities are combined to identify exemplars. For point $i$, its corresponding label is obtained as

$$k^* = \arg\max_{k \in \{p,q\}} \{a(i,k) + r(i,k)\}. \qquad (2)$$

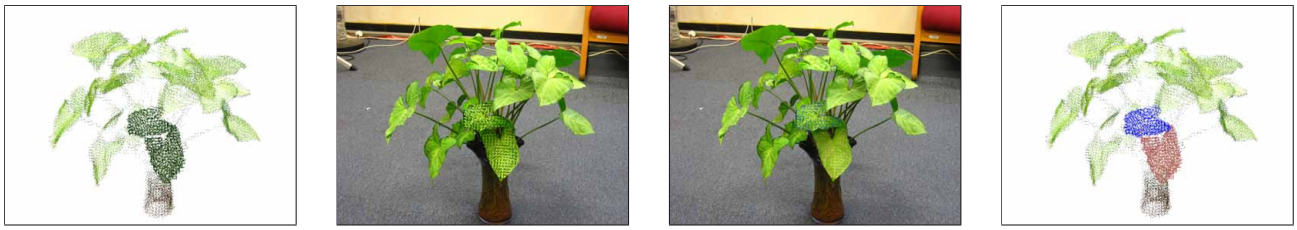
Here, we perform the exemplar assignment only from the labeled point set to obtain semi-supervised contraction, which is slightly different from Eqn. 1. One demonstration is shown in Fig. 4. Note that the algorithm can be easily generalized to more than two exemplars.

The semi-supervised affinity propagation propagates the message on the sparse graph by setting only the labeled nodes as candidate exemplars. It can converge in $O(Tn)$ time. Related discrete algorithms, such as iterated conditional mode, graph cuts, belief propagation, tree-reweighted message passing can also solve this problem [3] in more time compleixty, and other relaxation algorithms, such as label propagation [19], may not obtain good performance.

## 6. Experiment Results

Before performing our approach, we first run the connected component algorithm to extract connected components. One example is shown in Fig. 2(a). In addition, before patch filtering, we run the bilateral filtering method to smooth all 2D images while keeping the edges.

In Fig. 5, we first draw four strokes in one view shown in Fig. 5(a) to indicate that the scene consists of four major components: the tree, the desk, the ground and the wall. Then we learn the appearance models for each of the four components, respectively, and run the semi-supervised contraction on the 2D images to obtain 2D coarse segmentation and 3D segmentation as shown in Fig. 5(b) by checking their projections. Thirdly, we run our hierarchical sparse affinity propagation and semi-supervised affinity propagation to obtain the grouping results on each component, and the final 3D segmentation and 2D segmentation results are



(a) Selected group (b) 2D projections (c) Projection segmentation (d) 3D segmentation

Figure 4. Demonstration of semi-supervised contraction. (a) shows the selected cluster to be split. (b) shows the 2D projections visible on one view. (c) shows the separation of the visible 2D projections assisted by the user. (d) shows the clustering result on the selected group.

shown in Figs. 5(c) and 5(d) respectively. And in Fig. 6 shows another example using the similar process. In all our experiments, without code optimization, the hierarchical sparse affinity propagation takes one to three minutes at lower levels and becomes real-time at higher levels, while the semi-supervised affinity propagation always provides results on the fly, which is suitable for user interaction.

As an application mentioned above, 3D modeling can benefit from satisfactory multiple view segmentation. A modeling example is shown in Fig. 7. After we perform the proposed approach to obtain both 3D and 2D segmentation as shown in Figs. 7(a) and 7(c), we can build the 3D surface and appearance models using the similar technique in [10]. The rendering result is shown in Fig. 7(d).

## 7. Conclusion

Given both 2D images and 3D points reconstructed from those images, we proposed a joint segmentation approach to simultaneously segment 2D images and cluster 3D points. Efficient and effective hierarchical sparse and semi-supervised affinity propagation algorithms make the joint segmentation more practical. The results have demonstrated the powerfulness. Future work includes further study of the affects of different affinities.

## Acknowledgements

The work is supported by Hong Kong RGC projects 619005, 619006 and 619107, and NSFC/RGC Joint Grant N-HKUST602/05.

## References

[1] B. J. Frey and D. Dueck. Clustering by Passing Messages Between Data Points. *Science*, 315:972–976, February 2007.

[2] E. Hadjidemetriou, M. Grossberg, and S. Nayar. Multiresolution Histograms and Their Use for Texture Classification. In *3rd International Workshop on Texture Analysis and Synthesis*, Oct 2003.

[3] V. Kolmogorov. Convergent Tree-Reweighted Message Passing for Energy Minimization. In *AISTATS*, 2005.

[4] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother. Bi-Layer Segmentation of Binocular Stereo Video. In *CVPR (2)*, pages 407–414, 2005.

[5] V. Kolmogorov and R. Zabih. What Energy Functions Can Be Minimized via Graph Cuts? In *ECCV (3)*, pages 65–81, 2002.

[6] M. Lhuillier and L. Quan. A Quasi-Dense Approach to Surface Reconstruction from Uncalibrated Images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(3):418–433, 2005.

[7] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy Snapping. In *Proceedings of ACM SIGGRAPH*, pages 303–308, 2004.

[8] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and Texture Analysis for Image Segmentation. *International Journal of Computer Vision*, 43(1):7–27, 2001.

[9] I. Patras, E. Hendriks, and R. Lagendijk. Video Segmentation by MAP Labeling of Watershed Segments. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(3):326–332, 2001.

[10] L. Quan, P. Tan, G. Zeng, L. Yuan, J. Wang, and S. B. Kang. Image-based Plant Modeling. *ACM Trans. Graph.*, 25(3):599–604, 2006.

[11] L. Quan, J. Wang, P. Tan, and L. Yuan. Image-based Modeling by Joint Segmentation. *International Journal of Computer Vision*, To Appear, 2007.

[12] Y. Schnitman, Y. Caspi, D. Cohen-Or, and D. Lischinski. Inducing Semantic Segmentation from an Example. In *ACCV (2)*, pages 373–384, 2006.

[13] J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.

[14] P. Tan, G. Zeng, J. Wang, S. B. Kang, and L. Quan. Image-based Tree Modeling. *ACM Trans. Graph.*, 2007.

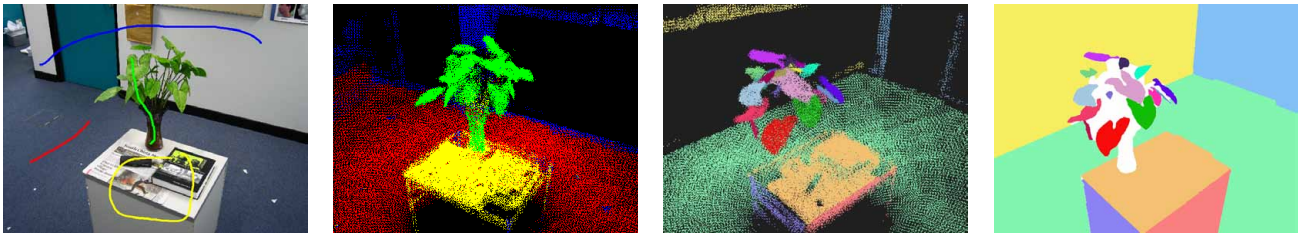
[15] Z. Tu and S. C. Zhu. Image Segmentation by Data-Driven Markov Chain Monte Carlo. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):657–673, 2002.

[16] J. Wang and E. Adelson. Representing Moving Images with Layers. *IEEE Transactions on Image Processing*, 3(5):625–638, 1994.

[17] J. Wills, S. Agarwal, and S. Belongie. What Went Where. In *CVPR (1)*, pages 37–44, 2003.

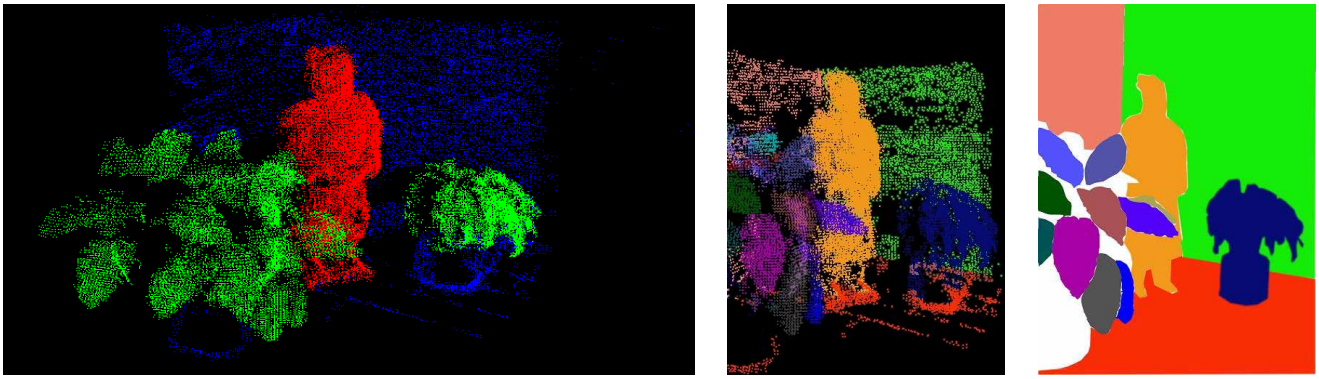
[18] J. Xiao and M. Shah. Motion Layer Extraction in the Presence of Occlusion Using Graph Cut. In *CVPR (2)*, pages 972–979, 2004.

[19] X. Zhu and Z. Ghahramani. Learning from Labeled and Unlabeled Data with Label Propagation. Technical Report tech report CMU-CALD-02-107, 2002.



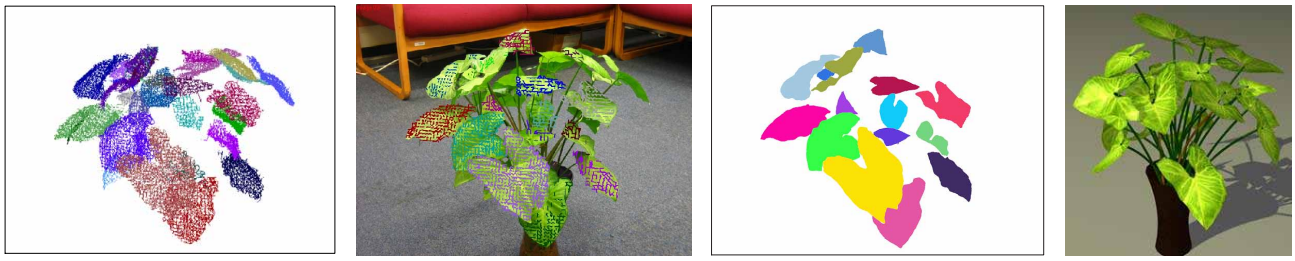
(a) Strokes (b) 3D coarse segmentation (c) Final 3D segmentation (d) Final 2D segmentation

Figure 5. Segmentation results for the office scene. (a) shows the user assistance in one view to indicate the four components of the scene, and the 3D segmentation result is shown in (b) by utilizing the user assistance and our semi-supervised contraction algorithm. (c) shows the final 3D segmentation result using our hierarchical sparse and semi-supervised affinity propagation. (d) shows the corresponding 2D segmentation result.



(a) Coarse 3D segmentation (b) Final 3D segmentation (c) Final 2D segmentation

Figure 6. Segmentation results for the terra-cotta warriors scene. (a) shows the initial result using user assistance. (b) shows the final 3D segmentation result using our hierarchical sparse and semi-supervised affinity propagation. (c) shows the corresponding 2D segmentation result.



(a) 3D groups (b) Group projection (c) 2D leaves segmentation (d) Rendering result

Figure 7. Segmentation results for the Nephthytis scene. To be clear, this example only shows the segmentation results on the leaves. (a) shows the grouping result on 3D space, (b) shows the projections of the groups, (c) shows the image segmentation result, and in addition (d) shows the 3D modeling example by using the techniques in [10] based on our multiple view segmentation result.