Transductive Segmentation of Textured Meshes

Anne-Laure Chauve, Jean-Philippe Pons, Jean-Yves Audibert, Renaud Keriven

Transductive Segmentation of Textured Meshes

Anne-Laure Chauve, Jean-Philippe Pons, Jean-Yves Audibert, and Renaud Keriven

IMAGINE, ENPC/CSTB/LIGM, Université Paris-Est, France

Abstract. This paper addresses the problem of segmenting a textured mesh into objects or object classes, consistently with user-supplied seeds. We view this task as transductive learning and use the flexibility of kernel-based weights to incorporate a various number of diverse features. Our method combines a Laplacian graph regularizer that enforces spatial coherence in label propagation and an SVM classifier that ensures dissemination of the seeds characteristics. Our interactive framework allows to easily specify classes seeds with sketches drawn on the mesh and potentially refine the segmentation. We obtain qualitatively good segmentations on several architectural scenes and show the applicability of our method to outliers removing.

1 Introduction

The generalization of digital cameras, the increase in computational power brought by graphical processors and the recent progress in multi-view reconstruction algorithms allow to create numerous and costless textured 3D models from digital photographs. In this work, we address the problem of segmenting a textured mesh into objects or object classes. The segmentation is an essential step of scene analysis, and can be used for semantic enrichment of architectural scenes, reverse engineering, subsequent recognition of known rigid objects... A meaningful mesh decomposition enables to retrieve objects of interest or remove undesired parts (see Fig.4). This problem raises several interesting challenges: the variability of scenes and object types to handle; the subjectivity of a meaningful segmentation, which depends on the application; the simultaneous classification and segmentation involved by object detection. We took up theses challenges by designing a flexible and interactive framework that conducts collective classification of the mesh.

We propose an easy-to-use, graphical tool to provide labeled training seeds by drawing sketches on the mesh, select appropriate features among the set of available ones, compute the segmentation via our transductive learning algorithm based on Support Vector Machine classification and Laplacian graph regularization, visualize the resulting segmented mesh and refine the training sketches to rerun the segmentation if necessary.

Our work is related to three main research topics: mesh segmentation, collective point cloud classification and transductive image segmentation. Even if

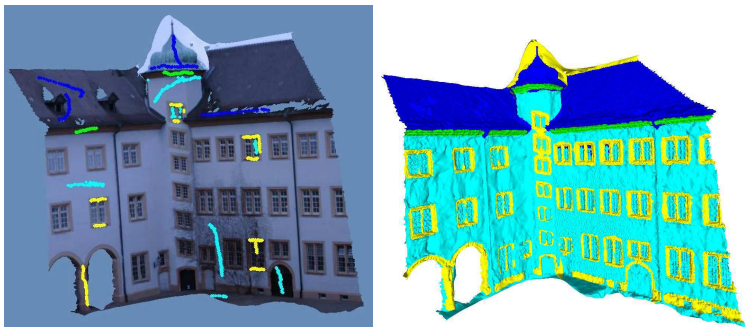

Fig. 1. Segmentation of the mesh into four classes: roof, wall, windows edges, cornice. Left: the input textured mesh with user supplied sketches. Right: the resulting segmentation using our algorithm.

we consider meshes in this paper, the type of data and the segmentation technique turn out to be quite different from the ones considered in the literature. Our issues are actually closer to a series of works on point cloud segmentation, however these are not directly applicable to our meshes whose connectivity and texture are of crucial importance. Our algorithm is rather based on a transductive segmentation method that was developed at first for 2D images.

The problem of mesh segmentation has become an important issue in various computer graphics applications, like parameterization and texture mapping, metamorphosis, 3D shape retrieval or modeling by example. Several segmentation algorithms for mesh partitioning have been compared in [1]. These algorithms deal with non-textured meshes of a single object, that are usually high-quality meshes coming from CAD models or dense scans – thus very different from the image-based meshes of entire scenes we process. These algorithms fall into two main categories. The first one gathers geometric approaches, where the mesh is segmented into patches fitted with simple mathematical surfaces. The typical application of these methods is reverse engineering of CAD models. The second one is rather semantic-oriented: the aim is to decompose meshes of "natural" objects (e.g., a body model) into "meaningful" pieces (e.g., the head, two arms, two legs and the torso). However, these semantic approaches do not involve any learning-based or classification procedure, and do not consider the similarities between different, non connected parts. Mesh decomposition is usually a preprocessing step, and should thus be able to handle various types of input models and of target applications. In order to deal with varying application-dependent requirements and with the subjectivity of a meaningful segmentation, [2] proposed an interactive framework similar to ours: the user draws sketches on the mesh that provide seeds for the algorithm. Nevertheless, as mentioned above and unlike [2], we process textured meshes: this greatly reduces the required amount of user interaction (see for example Fig.1).

A series of works [3–6] has been carried out on the problems of segmenting scan data into objects or objects classes using Markov networks. This problem

differs from the one we address mainly on the type of data to process: textured meshes possess supplementary attributes like color or mesh connectivity that we exploit, while 3D point clouds are far denser, and more precise, thus targeting distinct applications and requiring different processes. However these works share several characteristics with ours. One is the use of collective classification through graph-based methods: instead of classifying each point or facet independently, the problem is thought of as a global classification task and adjacency relationships are used to enforce spatial contiguity of the labels on the graph. Our graph Laplacian transduction performs such a collective classification. Besides, in both problems, the ability to handle various type of scenes as well as a certain variability inside a given class of objects is crucial: this is achieved by taking advantage of various kinds of features. Thus, the Markov random field segmentation algorithm of [3] has been tested successfully on both outdoor and indoor scenes, for real-world and synthetic scan datasets. In our framework, textured meshes possess both geometric and photometric attributes that should be chosen according to the scene type and jointly exploited. We combine these features in kernel weights of a graph Laplacian regularizer; the inherent modularity of kernel methods hence provides the desired flexibility.

Recently, several works have addressed the image segmentation problem in an interactive framework: a set of seed pixels representative of each region to be segmented is specified by the user, and the segmentation of the entire image is performed consistently with the seeds. The existing algorithms rely on computing weighted geodesic distances [7], graph cuts with discontinuity penalization [8], graph cuts with Gaussian mixture modelling of the segmented regions [9], random walks on a graph and its relation to electrical resistance in a circuit [10, 11] and transductive learning [12]. Due to its simplicity and its effective results obtained in the Microsoft GrabCut benchmark, the latter approach is adopted in our mesh segmentation algorithm. The segmentation results are visualized in our interactive framework, allowing for subsequent refinement of the query.

Outline. The rest of the paper is organized as follows. Section 2 presents the Laplacian regularizer on a graph to perform transduction and introduces the energy we minimize in the sequel; section 3 details our algorithm for the segmentation of textured meshes, and section 4 presents our experimental results. We conclude in section 5 with a brief discussion.

2 Graph Laplacian Transduction

2.1 Transductive Inference

The problem we are concerned with is a supervised classification task: given a set of labeled examples (called *training set*), we want to infer the labels of new input points (called *test set*). More specifically, we consider an input space $\mathcal{X}$ and an output space $\mathcal{Y}$ (typically $\mathcal{Y} = \{0, \dots, K\}$ for a classification problem); given a set of input-output couples $\{(x_1, y_1), \dots, (x_n, y_n)\}$, we want to determine the label y of a new point x.

There are two different approaches for this problem, the *inductive* and *transductive* settings. Inductive inference is a two-step process: one tries first to learn a function $f : \mathcal{X} \to \mathcal{Y}$ to map the entire input space to the output space, and then, for every point x of the test set, predicts $y = f(x)$ as its label. In contrast, in transductive inference, the test inputs are known beforehand. Thus, no general input-output mapping is inferred, but only the labels of the test points are predicted. This approach follows Vapnik's principle: "when solving a problem of interest, do not solve a more general problem as an intermediate step", and takes advantage of the unlabeled data to get an idea of the input space distribution. Indeed, transduction relies on this second principle, referred to as the *smoothness assumption*: "outputs vary a lot only on input regions having low density", or, in other words, "the decision boundary should lie in a low-density region".

Transductive inference can be compared to semi-supervised learning (SSL), where the training set is made up of both labeled and unlabeled data points. Like transduction, SSL utilizes the unlabeled points to infer the input distribution. However, in contrast to transduction, the test points are not (necessarily) known beforehand and semi-supervised algorithms can handle unseen data, thus being part of an inductive framework.

A large number of algorithms that have been proposed in the last few years for transductive inference relies on graph-based methods (see [13], [14] and references within), where nodes represent data points and edges encode similarities between them. The graph structure is used to propagate information from the known labels to the unlabeled points. We use the Laplacian graph regularizer of [12] with an unnormalized kernel (case $s = 2, \lambda = 0$): the labeling of the facets is carried out as the minimization of a quadratic cost function derived from the graph. The work [15] studies several cost functions for regularization on a graph and explains the links with label propagation algorithms.

2.2 Graph Laplacian Regularization and Energy Minimization

In the first place, we consider a binary classification problem, where the two classes have respectively 0 and 1 as labels (see 2.3 for generalization to the multi-class problem); 0 is the background class and 1 is the object class. Instead of directly predicting the labels of the test points ($y \in \{0, 1\}$), we consider a real-valued output space ($y \in \mathbb{R}$) and assign each point to the class $\mathbf{1}_{y \geq 1/2}$.

Graph Laplacian The geometry of the data is represented by a graph $G = (V, E)$ where the nodes $V = \{X_1, \ldots, X_n\}$ correspond to the input points coming from both the p labeled instances $\{X_1, \ldots, X_p\}$ of the training set, and the $n - p$ unlabeled data $\{X_{p+1}, \ldots, X_n\}$ of the test set, and the edges E represent similarities between them, in the form of a *weight* matrix W (of size $n \times n$). The coefficients of this matrix (also called *affinity* or *adjacency* matrix) must satisfy:

- $W_{ij} = 0$ if X_i and X_j are not "neighbours" (*i.e.* are not connected by an edge),

– $W_{ij} \geq 0$,
– $W_{ij} = W_{ji}$.

The meaning of the neighbourhood needs to be specified and depends on the nature of the data (e.g., facet adjacency on a mesh, symmetrized k-nearest neighbour for a point cloud, etc.). We write the weights in the form $W_{ij} = k(X_i, X_j)$ where $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a symmetric positive function giving the similarity between two input points. A typical example is the Gaussian kernel $k(x, x') = \exp\left(-\frac{d(x,x')^2}{2\sigma^2}\right)$ with d a distance on the input space $\mathcal{X}$ and σ a positive parameter.

Let D be the diagonal matrix $D_{ii} = \sum_j W_{ij}$. The matrix $L = D - W$ is called the *unnormalized graph Laplacian*. It is a discrete analogue of the Laplace-Beltrami operator on a Riemannian manifold (see (3)).

Energy design The goal is to find a labeling $Y = (y_1, \ldots, y_n)$ of the graph that is consistent with both the labeled training points and the geometry of the entire data (represented by the graph structure).

We partition the vector Y and the matrices W, D and L according to their labeled and unlabeled parts :

$$Y = \begin{pmatrix} Y_\ell \\ Y_u \end{pmatrix} \quad W = \begin{pmatrix} W_{\ell\ell} & W_{\ell u} \\ W_{u\ell} & W_{uu} \end{pmatrix} \quad D = \begin{pmatrix} D_{\ell\ell} & 0 \\ 0 & D_{uu} \end{pmatrix} \quad L = \begin{pmatrix} L_{\ell\ell} & L_{\ell u} \\ L_{u\ell} & L_{uu} \end{pmatrix}.$$

The initial labels of the training points are denoted $Y_\ell^0 = (y_1^0, \ldots, y_p^0)$.

We want to find the optimal labeling $\hat{Y}$ minimizing the following cost criterion, built up three different terms :

$$\hat{Y} = \underset{Y \in \mathbb{R}^n}{\arg\min} \underbrace{c \sum_{i=1}^{p} (y_i - y_i^0)^2}_{(\alpha)} + \underbrace{\frac{1}{2} \sum_{i,j=1}^{n} W_{ij}(y_i - y_j)^2}_{(\beta)} + \underbrace{\lambda \sum_{i=p+1}^{n} (y_i - s_i)^2}_{(\gamma)}, \quad (1)$$

where the scores s_i in (γ) are defined hereafter.

(α) Consistency with the initial labeling: $\sum_{i=1}^{p} (y_i - y_i^0)^2 = \|Y_\ell - Y_l^0\|^2$.

The parameter $c \in [0, +\infty]$ expresses the confidence assigned to the training outputs (it could be different for each point, $(c_i)_{i=1,\ldots,p}$).

(β) Consistency with the geometry of the data: This term penalizes rapid changes in $\hat{Y}$ between points that are close on the graph (as given by the similarity matrix W). It enforces the smoothness assumption along the graph.

$$\begin{aligned} \frac{1}{2} \sum_{i,j=1}^{n} W_{ij}(y_i - y_j)^2 &= \frac{1}{2}\Big(2 \sum_{i=1}^{n} y_i^2 \sum_{j=1}^{n} W_{ij} - 2 \sum_{i,j=1}^{n} W_{ij} y_i y_j\Big) \\ &= Y^\top (D - W) Y = Y^\top L Y \,. \end{aligned}$$

The latter term can be interpreted as a graph Laplacian regularizer in the following sense. Assume that the inputs are real vectors admitting a distribution with density p (with respect to the Lebesgue measure). Let φ be a smooth function such that $y_i = \varphi(X_i)$. From [12], the quantity $Y^T LY$ is a discrete approximation, up to a constant multiplicative factor, of the integral

$$\int \|(\nabla\varphi)(x)\|^2 p^2(x)dx\,, \tag{2}$$

which is equal to

$$\int \varphi(x)(\Delta\varphi)(x)p^2(x)dx\,, \tag{3}$$

for Δ a (weighted) Laplace-Beltrami operator. From (2), we see that the regularization term $Y^T LY$ will be small only when the labels vary in low density regions of the input space, thus fulfilling the principle stated in Section 2.1.

(γ) Consistency with some additional knowledge: $\displaystyle\sum_{i=p+1}^{n} (y_i - s_i)^2 = \|Y_u - S_u^0\|^2\,.$
This terms allows to incorporate either some prior information or the output of another algorithm in the form of a score s_i, measuring for each test point i the "likelihood" that it belongs to the object class. This term can be seen as an initializing score. We note S_u^0 the $(n-p)$-vector $(s_{p+1}, \ldots, s_n)$.

The optimization problem (1) can hence be expressed in the following matrix form:

$$\hat{Y} = \underset{Y\in\mathbb{R}^n}{\arg\min}\ c\|Y_\ell - Y_\ell^0\|^2 + Y^\top LY + \lambda\|Y_u - S_u^0\|^2\,. \tag{4}$$

Sparse linear system giving the predicted labels Since we assume that the user-supplied seeds are trustworthy, we constrain the labels on the labeled data ($\hat{Y}_\ell = Y_\ell^0$) and thus consider an infinite regularization coefficient $c = +\infty$. Hence the minimization is carried over the labeling Y_u of the test points, and the optimization problem rewrites

$$\begin{aligned}
\hat{Y}_u &= \underset{Y_u\in\mathbb{R}^{n-p}}{\arg\min}\ Y^\top LY + \lambda\|Y_u - S_u^0\|^2 \\
&= \underset{Y_u\in\mathbb{R}^{n-p}}{\arg\min}\ {Y_\ell^0}^\top L_{\ell\ell} Y_\ell^0 + {Y_u}^\top L_{u\ell} Y_\ell^0 + {Y_\ell^0}^\top L_{\ell u} Y_u + {Y_u}^\top L_{uu} Y_u \\
&\qquad\qquad + \lambda\left({Y_u}^\top Y_u - {Y_u}^\top S_u^0 - {S_u^0}^\top Y_u + {S_u^0}^\top S_u^0\right) \\
&= \underset{Y_u\in\mathbb{R}^{n-p}}{\arg\min}\ 2Y_u^\top\left(L_{u\ell} Y_\ell^0 - \lambda S_u^0\right) + Y_u^\top\left(L_{uu} + \lambda I\right) Y_u\,.
\end{aligned}$$

In order to minimize the cost criterion, we compute its derivative with respect to Y_u. Let $A = L_{u\ell} Y_\ell^0 - \lambda S_u^0$ and $B = L_{uu} + \lambda I$. The matrix B is symmetric positive definite matrix when $\lambda > 0$, since L_{uu} is symmetric positive semi-definite:

$${Y_u}^\top L_{uu} Y_u = \frac{1}{2}\sum_{i,j=p+1}^{n} W_{ij}(y_i - y_j)^2 \geq 0\,.$$

Thus the function $f : Y_u \mapsto 2Y_u^\top A + Y_u^\top B Y_u$ is strictly convex and admits a unique minimum where its derivative equals 0:

$$\frac{\partial f(Y_u)}{\partial Y_u} = 2A + 2BY_u = 0 \quad \Longleftrightarrow \quad BY_u = -A\,.$$

Hence, the optimal labeling $\hat{Y}_u$ is the solution of the sparse linear system

$$(L_{uu} + \lambda I)\,\hat{Y}_u = \lambda S_u^0 - L_{u\ell} Y_\ell^0\,. \tag{5}$$

2.3 Multi-class Segmentation

The extension of the previous algorithm to the multi-class case is straightforward using a one-versus-all approach. If there are d different classes, we resolve the linear system (5) for each class k against all other classes as background, thus having as initial label for a training point i, $Y_\ell^{0,k}(i) = 1$ if i is labeled k and $Y_\ell^{0,k}(i) = 0$ otherwise. We obtain d output vectors $\hat{Y}_u^k$, and assign the test point j to the class $\underset{k=1,\ldots,d}{\arg\max}\, \hat{Y}_u^k(j)$.

3 Segmentation of Textured Meshes

3.1 Our Algorithm

We work on textured meshes of a 3D scene built from a set of calibrated images: we use the multi-view stereovision algorithm from [16–18] to reconstruct a 3D model of the scene and the multi-band blending algorithm from [19] to compute a texture atlas with minimal color discontinuities or blurring. The meshes presented in our experiments have been reconstructed from datasets provided by C. Strecha *et al.* [20] (castle-P19, castle-P30 and Herz-Jesu-25).

We want to classify the facets of the mesh given seed facets for each class. The seeds are provided by the user who draws sketches on the mesh. Thus we consider a graph G with a node for every facet of the mesh and an edge between any two adjacent facets. The modularity of our kernel method allows to chose various edges weights, depending on the scene type. We use a Gaussian kernel with a distance between facets constructed from one or several features characterizing the facets (see section 3.2 for more details and examples of such features).

The training points are the user-supplied seeds and we use an SVM classification on these training points as additional knowledge on the test points (see section 3.3). The linear system (5) of the transductive classification is sparse due to the sparseness of the facet adjacency relationship on the mesh. We solve it with a conjugate gradient algorithm (we use the IML++ implementation of [21]). For each facet, we obtain a score for each class: the classification is given by the argmax of these scores (see section 2.3) but they carry much more information that can be exploited.

The interactive framework of our method allows to add supplementary seeds depending on misclassified facets and rerun the algorithm in order to improve the segmentation result or add new classes (see Fig.4).

3.2 Features and Kernels

We compare two different facets through a set of attributes extracted from the mesh. We describe each facet by a feature vector of length m. These features are chosen according to the scene type and the discriminant characteristics of such scenes. A major element of this algorithm is the ability to take advantage of several different kinds of features, mixing photometric and geometric informations or any other available attribute (see Fig.2).

Here are some examples of features that we used in the results of Section 4 or that could be used in other experiments. On the one hand, we use photometric features like the mean color of the facet or components of the mean color (for instance, without luminance to account for changes in illumination between cameras). On the other hand, we use geometric features like the normal of the facet (or solely its vertical component), the position of the center of the facet (for instance, height of the center), or the discrete curvature of the mesh. We could also also evaluate less local features that would consequently be smoother and more robust by averaging the previous features on a neighbourhood of the facet.

We do not compare globally the descriptor vectors of the facets, but feature by feature. Then we combine the component-wise distances in the kernel weight by multiplying the Gaussian kernels associated with each feature: if there are n_f different features (each feature being a vector of variable length) and we note $(f_i^1, \ldots, f_i^{n_f})$ and $(f_j^1, \ldots, f_j^{n_f})$ the feature vectors of facets X_i and X_j,

$$W_{ij} = k(X_i, X_j) = \prod_{k=1}^{n_f} \exp\left(-\frac{\|f_i^k - f_j^k\|^2}{2\sigma_k^2}\right) = \exp\left(-\sum_{k=1}^{n_f} \frac{\|f_i^k - f_j^k\|^2}{2\sigma_k^2}\right).$$

The kernel weight is thus parameterized by the n_f values $(\sigma_1, \ldots, \sigma_{n_f})$ which determine the trade-off between the various features.

3.3 Additional Knowledge on the Test Points

We train Support Vector Machines with the user-supplied training points in order to provide initialization information on the test points. We obtain a score s_i^k for each facet i and each class k with a one-versus-all SVM algorithm. We use the LibSVM implementation of the C-SV Classification (cf. [22]).

The incorporation of external knowledge in addition to the graph Laplacian regularization is essential in order to detect several objects of the same class which form several connected components. Indeed, each seed can generate at most one connected component in the segmentation. Besides, the initialization with the SVM scores accelerate the label propagation on the mesh and the convergence of the iterative resolution compared to the transductive segmentation based solely on graph Laplacian.

We can consider either the same kernel as for the graph Laplacian weights or a different one (different features, kernel type, or parameters), if we want to exploit distinct properties of the mesh.We learn the parameters of the SVM

(parameters of the kernel plus parameter C of the SVM) in a cross-validation process. Hence, the tradeoff between the various features combined in the kernel is learned automatically. Moreover, we can employ these selected parameters in the Laplacian kernel for transduction (if we use the same kernel).

The Support Vector Machine approach is a classification method by itself and provides a comparison basis for our algorithm (just as the transductive segmentation without the SVM initialization). However it does not enforce any spatial contiguity on the mesh, and it produces a noisy result (see Fig.3).

Note that we learn this term directly on the scene of interest, but we could incorporate as well a prior learned from other, previously segmented scenes, instead or in addition to this one. In this case, however, the algorithm is not anymore a purely transductive one.

3.4 Computational Complexity and Time

The computational cost of the algorithm can be split up into the cost of the SVM initialization and the cost of the transductive segmentation on the graph. Since in our framework the number of training points p is negligible compared to the total number of points n, we consider the complexity of the algorithm with respect to n only. In the SVM algorithm, the cost of the test prevails over the cost of the train (which is between $\mathcal{O}(p^2)$ and $\mathcal{O}(p^3)$), and its complexity is $\mathcal{O}(n.p) = \mathcal{O}(n)$. In the energy minimization of the transduction, the main cost comes from solving the sparse linear system (5); this can be done in $\mathcal{O}(k.m)$ where k is the number of iterations of the conjugate gradient and m is the number of non-zero entries in the matrix, which is equal to $4n$. We bound the number of iterations to limit the computational time, thus having a total computational complexity in $\mathcal{O}(n)$ (at the expense of precision on the convergence).

In practice, the computational time of the transduction prevails over the one of SVM. The whole segmentation process takes between fifteen seconds and five minutes on a Xeon 2.33 GHz, for meshes with 50,000 to 1,500,000 facets.

4 Experimental Results

Figure 1 in the introduction shows a first example of segmentation into four different classes using the mean color of a facet and the altitude of its center as features. The results are qualitatively good, and mostly agree with perceptual boundaries. Note that every window is detected while only few where initially labeled, illustrating the ability of our algorithm to detect several objects of the same class forming several connected components, thanks to the initialization of the transduction with the classification results of the SVM.

Figure 2 illustrates the importance of combining several different features in order to obtain a relevant segmentation. Indeed the use of mean color or altitude alone produces erroneous results (even if the latter seems less wrong, it is even more naive, and performs a simple thresholding on the altitude), while their combination gives a pertinent result. Hence, the set of features is chosen

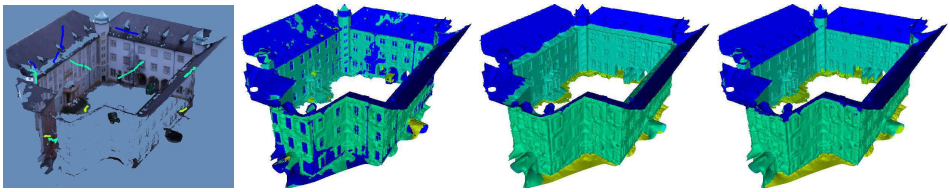

Fig. 2. Combining different features improves the segmentation. 3 classes: roof, wall, ground. From left to right: input textured mesh with seeds; segmentation using mean color; segmentation using altitude; segmentation using both mean color and altitude.

according to the scene of interest: in Fig.3, the segmentation of the top row is performed using color only, while in the bottom row, the color, the curvedness and the vertical normal of the facet are combined.

Figure 3 compares the classification of the SVM and the final result after transduction: as mentioned in section 3.3, the results of the SVM alone are noisy and do not exploit the connectivity of the mesh.

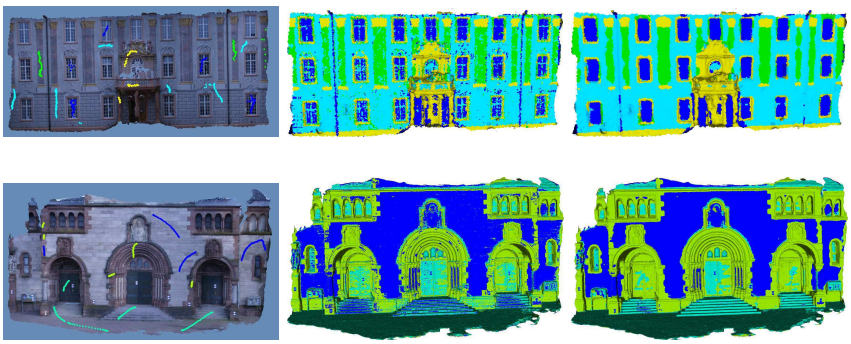

Fig. 3. Comparison of classification results using using SVM alone or SVM plus transductive segmentation. Left: textured mesh with selected facets. Middle: classification obtained with SVM; the labels are noisy and do not provide a decomposition of the mesh. Right: classification obtained with SVM initialization plus transduction.

Our algorithm can be used to remove outliers in a scene. In the original mesh of the castle of Figure 4, obtained by multi-view reconstruction, a portion of the sky has been reconstructed running on from the roof. It can be easily remove in our interactive framework, using only two classes, one for the outliers (sky facets), one for the inliers (castle facets), and color and altitude as features. The first segmentation *(middle column)* is not entirely satisfying (some portions of sky remain), so we add a few labeled points and rerun the algorithm, then obtaining a good segmentation *(right column)*.

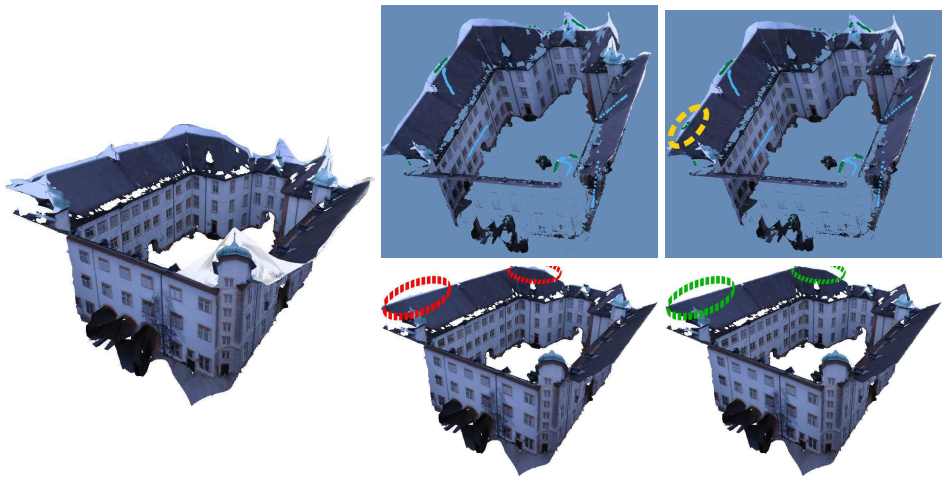

Fig. 4. Removing outliers. Left: original textured mesh of the castle with portions of "sky" mistakenly reconstructed by stereo. Middle: removing sky facets using selected facets. Right: refinement of the previous segmentation by adding sky facets to the selection (*yellow outline*). Top row: selected facets (*green: sky outliers, blue: castle inliers*), bottom row: textured meshes.

5 Conclusion

We have presented an efficient procedure for the segmentation of textured meshes: we designed a sketch-based interactive framework which produces a meaningful segmentation according to the user's aims, thanks to a possible refinement of the training sketches. Our experiments demonstrate that we can take advantage of geometric and photometric features at the same time, combining a various number of appropriate features in our kernel. The graph Laplacian regularizer enforces the spatial contiguity of the labels on the mesh, producing robust decompositions of the mesh for subsequent applications while the SVM initialization allows to detect non-connected, similar objects. Future work will focus on the development of supplementary features and kernels, in order to apply this algorithm to a larger range of scenes as well as to other types of data like point clouds.

References

1. Attene, M., Katz, S., Mortara, M., Patanè, G., Spagnuolo, M., Tal, A.: Mesh Segmentation - A Comparative Study. In: SMI. (2006)
2. Wu, Pan, Yang, Ma: A sketch-based interactive framework for real-time mesh segmentation. In: CGI. (2007)
3. Anguelov, D., Taskar, B., Chatalbashev, V., Koller, D., Gupta, D., Heitz, G., Ng, A.Y.: Discriminative Learning of Markov Random Fields for Segmentation of 3D Scan Data. In: CVPR. (2005)

4. Triebel, R., Kersting, K., Burgard, W.: Robust 3D Scan Point Classification using Associative Markov Networks. In: ICRA. (2006)
5. Triebel, R., Schmidt, R., Martínez Mozos, O., Burgard, W.: Instance-based AMN Classification for Improved Object Recognition in 2D and 3D Laser Raneg Data. In: IJCAI. (2007)
6. Munoz, D., Vandapel, N., Hebert, M.: Directional Associative Markov Network for 3-D Point Cloud Classification. In: 3DPVT. (2008)
7. Bai, X., Sapiro, G.: A Geodesic Framework for Fast Interactive Image and Video Segmentation and Matting. In: ICCV. (2007)
8. Boykov, Y.Y., Jolly, M.P.: Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images. In: ICCV. (2001)
9. Blake, A., Rother, C., Brown, M., Perez, P., Torr, P.H.S.: Interactive Image Segmentation Using an Adaptive GMMRF Model. In: ECCV. (2004)
10. Grady, L.: Random Walks for Image Segmentation. PAMI **28** (2006) 1768–1783
11. Kim, T.H., Lee, K.M., Lee, S.U.: Generative Image Segmentation Using Random Walks with Restart. In: ECCV. (2008)
12. Duchenne, O., Audibert, J.Y., Keriven, R., Ponce, J., Ségonne, F.: Segmentation by transduction. In: CVPR. (2008)
13. Belkin, M., Niyogi, P.: Semi-Supervised Learning on Riemannian Manifolds. Machine Learning **56** (2004) 209–239
14. Chapelle, O., Schölkopf, B., Zien, A., eds.: Semi-Supervised Learning. MIT Press (2006)
15. Bengio, Y., Delalleau, O., Roux, N.L.: Label Propagation and Quadratic Criterion. In Chapelle, Schlkopf, Zien, eds.: Semi-supervised Learning. MIT Press (2006) 193–216
16. Labatut, P., Pons, J.P., Keriven, R.: Efficient Multi-View Reconstruction of Large-Scale Scenes using Interest Points, Delaunay Triangulation and Graph Cuts. In: ICCV. (2007)
17. Pons, J.P., Keriven, R., Faugeras, O.: Multi-view Stereo Reconstruction and Scene Flow Estimation with a Global Image-Based Matching Score. IJCV **72** (2007) 179–193
18. Vu, H., Keriven, R., Labatut, P., Pons, J.P.: Towards high-resolution large-scale multi-view stereo. In: CVPR. (2009)
19. Allène, C., Pons, J.P., Keriven, R.: Seamless Image-Based Texture Atlases using Multi-band Blending. In: ICPR. (2008)
20. Strecha, C., von Hansen, W., Van Gool, L.J., Fua, P., Thoennessen, U.: On Benchmarking Camera Calibration and Multi-view Stereo for High Resolution Imagery. In: CVPR. (2008)
21. Dongarra, J., Lumsdaine, A., Pozo, R., Remington, K.: A Sparse Matrix Library in C++ for High Performance Architectures. In: Proc. of the Second Object Oriented Numerics Conference. (1992)
22. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm (2001)