Texture Superpixel Clustering from Patch-based Nearest Neighbor Matching

Rémi Giraud
Signal and Image group
Bordeaux INP, Univ. Bordeaux, CNRS, IMS, UMR 5218
F-33405 Talence, France
remi.giraud@ims-bordeaux.fr

Yannick Berthoumieu
Signal and Image group
Bordeaux INP, Univ. Bordeaux, CNRS, IMS, UMR 5218
F-33405 Talence, France
yannick.berthoumieu@ims-bordeaux.fr

***Abstract*—Superpixels are widely used in computer vision applications. Nevertheless, decomposition methods may still fail to efficiently cluster image pixels according to their local texture. In this paper, we propose a new Nearest Neighbor-based Superpixel Clustering (NNSC) method to generate texture-aware superpixels in a limited computational time compared to previous approaches. We introduce a new clustering framework using patch-based nearest neighbor matching, while most existing methods are based on a pixel-wise K-means clustering. Therefore, we directly group pixels in the patch space enabling to capture texture information. We demonstrate the efficiency of our method with favorable comparison in terms of segmentation performances on both standard color and texture datasets. We also show the computational efficiency of NNSC compared to recent texture-aware superpixel methods.**

***Index Terms*—Superpixels, Nearest Neighbor, Texture**

I. Introduction

The constant increase of image data may highly impact the computational cost of computer vision pipelines. Among the approaches used to reduce the processing load, dimension reduction and multi-resolution methods have been widely used over the past years. In this context, decompositions into superpixels appear to be very interesting, since the created regions tend to respect the boundaries of the image objects. The relations between these irregular regions at different resolution levels can still be inferred [1], and superpixel neighborhoods can be used as for standard regular patch-wise and multi-resolution processing [2]. Therefore, many superpixel-based methods have been proposed in the literature, for different image processing and analysis applications, *e.g.*, semantic segmentation [3], tracking with optical flow [4], depth estimation [5], and color [6] or style transfer [7].

Since their introduction and popularization with [8], the majority of superpixel methods decompose the image into regions approximately containing the same number of pixels with homogeneous colors. To compute this clustering, most methods such as [8]–[15], consider a trade-off distance between spatial and color spaces at the pixel scale. The spatial distance enables to provide a relatively regular decomposition of the image domain, while the color distance associates pixels to a superpixel with the same average color. Most state-of-the-art methods only use the pixel spatial and color features in their clustering model, since information at the pixel scale may be sufficient to detect the object boundaries in a natural color image. Consequently, recent works such as [16], [17] have highlighted the non robustness of pixel-wise state-of-the-art methods to noise or texture for instance. All methods relying on pixel-wise information may indeed highly fail at grouping textures and may provide very inconsistent decompositions [17]. Even recent methods using advanced feature spaces [13], [18] or additional information such as features on the path to the superpixel barycenter [12], [15], [19] do not explicitly capture texture patterns and fail to detect texture changes.

In the recent method TASP [17], a straightforward extension of the SLIC framework [8] is proposed to compute texture-aware superpixels. Patch comparisons are performed within the superpixel to provide a texture term in the clustering model. Nevertheless, such method presents an important computational complexity. The SLIC framework [8] begin based on a K-means clustering, each superpixel iteratively computes its distance to all pixels in a restricted area. In TASP [17], the texture term must be computed for each superpixel at each iteration, by a nearest neighbor search performed for all pixels in this area, leading to an important computational burden, *i.e.*, more than 60s for images of 321×481 pixels. Hence, it appears necessary to propose a more efficient approach in terms of complexity, also able to accurately cluster textures.

Contributions: In this paper, we propose a new Nearest Neighbor-based Superpixel Clustering (NNSC) method to generate accurate and texture-aware superpixels. We introduce a new clustering framework using patch-based nearest neighbor matching, while most existing methods are based on a K-means clustering. Hence, we directly group pixels in the patch space while previous methods such as [17], combine both approaches at the expense of an important computational load. We also propose a new method to merge several decomposition estimations obtained from different nearest neighbor searches.

In the following, we first show the interest of considering patches for texture clustering, and the limitations of their use in a K-means-based clustering algorithm [17]. Then, we present our new decomposition method relying on a patch-based nearest neighbor clustering, with much lower complexity. Finally, we study our method parameters and compare its segmentation performances to the ones of the state-of-the-art methods on both natural color and texture datasets.

978-9-0827-9703-9/19/$31.00

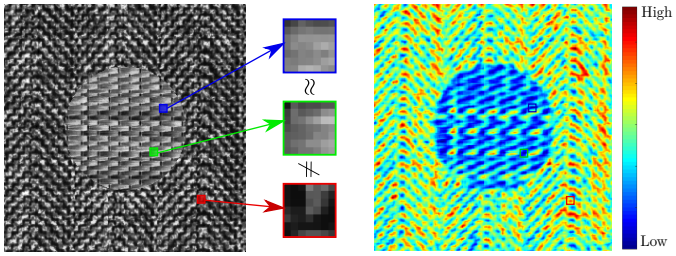


Fig. 1. Interest of patches for texture clustering. Selection of a reference patch of size 7×7 pixels in blue (left). A similar (green) and distant (red) patches are represented. Map of l-2 distance (1) between the reference blue patch and all image patches (right), demonstrating the patch ability to capture regions with the same texture.

II. Texture Superpixels using Patches

In this section, we first demonstrate the ability of patches to easily cluster image textures. Then, we present the standard pixel-wise K-means-based clustering algorithm [8] and the limitations of its extension using patches to generate texture-aware superpixels in [17].

A. Texture Clustering using Patches

Patches enable to capture the neighborhood of each image pixel. Non-local patch-based approaches, that have first become popular for texture synthesis [20] and denoising applications [21], use this structure to find similar patterns in the same or other images. The distance between fixed size patches enables to reflect both the similarity in terms of intensity and texture patterns. This distance d_P between two patches $P(p_i)$ and $P(p_k)$ describing the neighborhood of two pixels p_i and p_k, is generally computed with a l-2 norm such that:

$$d_P(p_i, p_k) = \frac{1}{n}\|P(p_i) - P(p_k)\|_2, \tag{1}$$

with n the patch size. In Figure 1, we illustrate the ability of patches to cluster image textures by computing distance (1) between a reference patch and all other image patches.

In the context of texture-aware superpixel clustering, the texture homogeneity between a pixel and a superpixel is not easy to measure since a pixel neighborhood must be compared to a superpixel having a variable size. Texture classification approaches could necessitate prior information on the image type, or additional parameter settings to be consistent with the pixel-wise color information that must also be taken into account in the clustering model [17]. Moreover, such approaches can be computationally costly. Therefore, using patches appears to be an interesting solution but requires a selection strategy to determine which patches to compare.

B. Texture Superpixels from K-means-based Framework

The recent TASP method [17] proposes to generate texture-aware superpixels using the K-means-based framework of [8], which is very popular due to its simplicity of use and understanding. The image is first split into regular blocks of size $s\times s$, depending of the input number of desired superpixels. Superpixels are then sequentially processed, and try to gather neighboring pixels in a restricted area of size $(2s+1)\times(2s+1)$. The clustering distance between a pixel and a superpixel is composed of a spatial and color distance. The pixel features are compared to the average features over all pixels in the superpixel. At the end of each iteration, pixels are associated to the superpixel providing the lowest distance.

The TASP method [17] adds a texture homogeneity term to the distance of [8]. It uses fixed size patches as descriptors to easily capture texture patterns while staying in the same feature space as the color distance between pixels and superpixels. Figure 1 shows that patch distances may be high even within the same texture area. Therefore, comparing a pixel neighborhood described by a patch to a reference one, for instance at the superpixel barycenter would not guarantee a relevant texture measure. Hence, [17] performs a patch-based nearest neighbor (NN) search to find similar patches in the superpixel. Similar patches then implies texture homogeneity, and favor the association of the pixel to the superpixel.

Limitations: With such approach, the NN search must be performed for all pixels in the $(2s+1)\times(2s+1)$ pixels area for each superpixel at each iteration, leading to overlapping pixels and repetition of the NN matching process. In the K-means-based clustering, a pixel is indeed approximately considered by 4 superpixels at each iteration. Therefore, TASP complexity depends on the number of image pixels $|I|$, number of K-means iterations N_K, and number of NN search iterations N such that $C_{\text{TASP}} = \mathcal{O}(|I|\times 4\times N_K \times N)$.

These limitations motivate the introduction of our new clustering framework, significantly reducing this complexity while preserving the ability to generate texture-aware superpixels.

III. Texture Superpixels from Patch-based Nearest Neighbor Matching

In this section, we first introduce our new clustering framework directly based on NN matching. Then, we present in detail the algorithm used to perform the search of similar patches. Finally, we propose a method to merge several superpixel decompositions obtained from different NN matching.

A. Nearest Neighbor Superpixel Clustering Framework

1) Clustering Algorithm: The proposed NNSC method directly clusters pixels using a patch-based NN matching process, that we prove to be necessary to provide texture-aware superpixels. The search is sequentially performed for all image pixels, to iteratively refine the initial superpixel grid decomposition. Therefore, it differs from the standard K-means-based framework that sequentially processes superpixels, leading the same pixel to be considered several times at the same iteration.

The NNSC decomposition process to obtain a label map $\mathcal{L}$ is illustrated in Figure 2. At a given iteration, the label of the superpixel containing the patch correspondence is assigned to the considered pixel position for next iteration. The complexity of NNSC with its clustering framework directly based on NN matching reduces to $C_{\text{NNSC}} = \mathcal{O}(|I|\times N)$. Note that the search for similar patches can be performed by any NN method, and we present the proposed search strategy in section III-B.

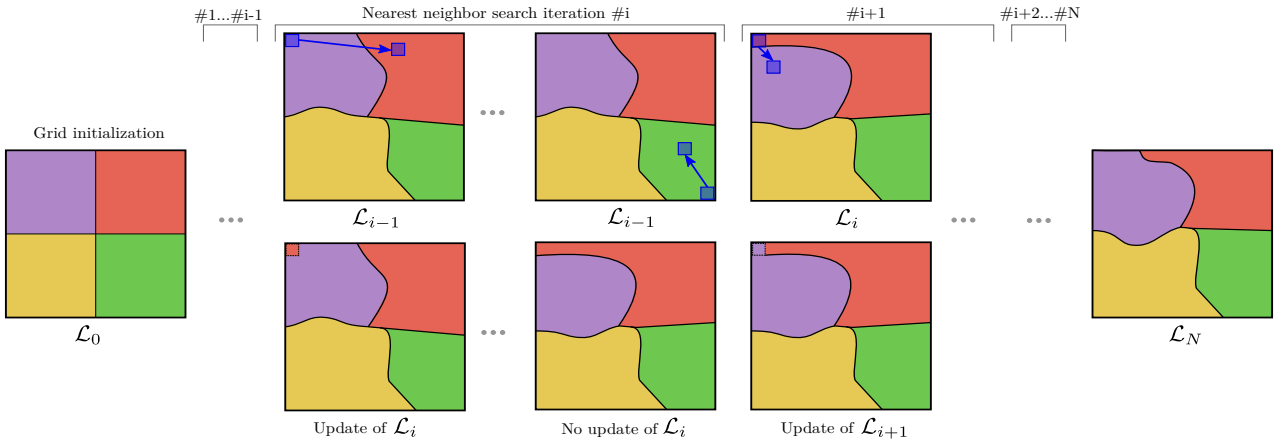


Fig. 2. Illustration of the NNSC clustering framework based on NN matching for a number of 4 superpixels. Superpixels are first decomposed into a regular grid. At each iteration $\#i$, pixels in blue are sequentially considered, and a correspondence is found within the image (lines), using patch-based distances (3). If a pixel gets associated to a different label, the i-th label map $\mathcal{L}_i$ is updated. After N iterations, the final label map $\mathcal{L}_N$ is obtained.

2) Patch-based Clustering Distance: To capture both the similarity in terms of intensity and texture patterns, patch intensities in the feature space (*e.g.*, colors in CIELab color space) are considered in the patch-based distance d_P computed between a pixel p_i of patch $P(p_i)$, and a patch $P(p_k)$ at position $p_k \in S_k$ such that:

$$d_P(p_i, p_k) = \frac{1}{n}\|P(p_i) - P(p_k)\|_2 + \frac{m_k^2}{s^2}\,\Gamma(p_k, X_{S_k}), \quad (2)$$

with m_k the regularity parameter, automatically set for each superpixel S_k [17], and Γ, a spatial weighting function defined such that $\Gamma(p_k, X_{S_k}) = 2s^2(1 - \exp(-\|p_k - X_{S_k}\|_2^2/s^2))$, favoring the search near to the superpixel barycenter X_k, preventing a superpixel S_k to cluster different textures.

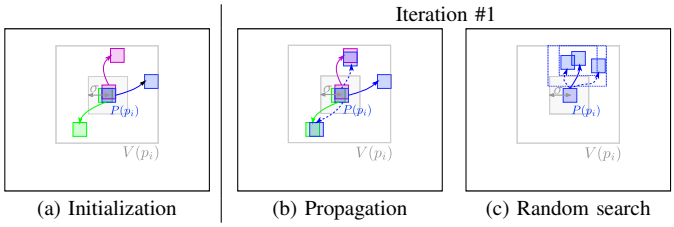
Finally, the global patch-based clustering distance D considers the patch distance term d_P (2), but also the standard color d_c and spatial d_s distances at the pixel scale [8]. These terms respectively enable to adapt superpixel borders to object contours and to ensure the shape regularity of superpixels. Hence, patch correspondences are computed according to:

$$D(p_i, p_k) = d_P(p_i, p_k) + d_c(p_i, S_k) + d_s(p_i, S_k)\frac{m_k^2}{s^2}. \quad (3)$$

B. Nearest Neighbor Search using PatchMatch

Since computing exact NN would be too costly, we choose to use the approximate NN search algorithm PatchMatch (PM) [22]. PM was initially proposed to provide for each patch of an image A, a correspondence in an image B. The algorithm starts from random correspondences and iteratively refines the patch associations using fast propagation of good matches from adjacent neighbors, and random tests. We adapt this algorithm to our context, *i.e.*, finding similar patches within the same image and into a restricted area around each patch.

First, to ensure the regularity of the decomposition, we limit the search to a $(2s+1)\times(2s+1)$ pixels area around the pixel position. For a pixel p_i, this area is denoted $V(p_i)$ in Figure 3, which illustrates the algorithm steps for a given patch $P(p_i)$. Naturally, to avoid to match the same patch $P(p_i)$, a σ-neighborhood is defined where to prevent the selection of patches. Random associations are first computed in these restricted areas (Figure 3(a)) after the grid initialization. Then, the propagation step considers the correspondences of the recently processed adjacent patches to lead $P(p_i)$ to new potential correspondences (Figure 3(b)). Finally, random selections are performed in areas of reducing size around the best current correspondence (Figure 3(c)). This adaptation of PM finds similar patches in the same image while spatially constraining the search area to ensure superpixel regularity.



(a) Initialization (b) Propagation (c) Random search

Fig. 3. Illustration of the PM algorithm adapted to the search of approximate NN within the same image. The processing is described for a blue patch. (a) Random initialization for the blue patch and two of its adjacent neighbors within a constrained window and outside a σ-neighborhood. (b) The propagation tests the shifted correspondences of recently processed adjacent patches (dotted lines). (c) The random search performs random tests within an iteratively reducing window around the current best match.

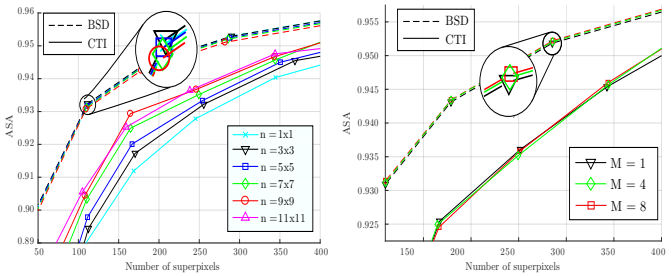
C. Aggregation of Multiple Clustering Estimation

PM being partly random, several clustering estimations can be computed, and averaged to improve the performances, as in [23]. These independent estimations can be easily launched in parallel using multi-threading implementation. Variations between estimations being reduced, the aggregation of M multiple label maps $\mathcal{L}_N^i$ can be performed as follows:

$$\mathcal{L}_{final}(p_i) = \underset{l \in \{labels\}}{\operatorname{argmax}} \sum_{i=1}^{M} \delta_{\mathcal{L}_N^i(p_i),l}\ , \quad (4)$$

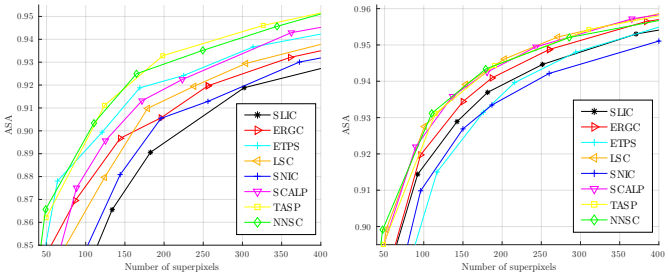
where $\delta_{i,j}$ equals 1 when $i = j$, and 0 otherwise.

Finally, as in [8], a post-processing step ensuring superpixel connectivity is performed on the final label map $\mathcal{L}_{final}$.



(a) (b)

Fig. 4. Influence of (a) the patch size (2) and (b) the number of aggregated label maps estimated from different NN searches (4) on ASA.



(a) (b)

Fig. 5. Comparison of NNSC segmentation performances measured with ASA, to the ones of the state-of-the-art methods on texture CTI (a) and natural color BSD (b) datasets.

IV. Evaluation of Performances

A. Validation Framework

1) Dataset: To evaluate the segmentation performances, we consider a standard composite texture image (CTI) dataset [24], which is composed of 10 grayscale images containing up to 16 different textures[1]. High performances on these images demonstrate the ability to detect texture changes. We also report the performances for the standard natural color Berkeley Segmentation Dataset (BSD) [25], which contains 200 test images of size 321×481 pixels.

2) Compared methods: NNSC performances are compared to the ones of the recent state-of-the-art methods SLIC [8], ERGC [19], ETPS [11], LSC [13], SNIC [14], SCALP [15], and TASP [17], used with parameters recommended by the authors. Performances are measured with the standard Achievable Segmentation Accuracy (ASA) [9] that evaluates the accuracy of superpixels according to a ground truth segmentation.

3) Parameters: In NNSC default settings, patches of size $n = 7\times7$ pixels are selected outside a $\sigma = 3$ neighborhood. $M = 4$ label maps are aggregated, and the number of iterations is set to 8. Features and parameters in (3) are computed as in [17]. These parameters are empirically set, and results in section IV-C are obtained using the same settings. Finally, note that the random sequence of PM is controlled to provide the same decomposition for the same image and parameters.

B. Influence of Parameters

1) Patch Size: The influence of the patch size (2) on the performances is shown in Figure 4(a). On the CTI dataset, large patches enable to efficiently capture textures, while on the BSD dataset patches larger than 3×3 do not provide more information, object contours being mainly detected by color changes. In NNSC default settings, a patch size of 7×7 is chosen as a good trade-off between accuracy and computational time. Nevertheless, parameters could be manually optimized.

2) Number of Clustering Estimations: The influence of the number M of aggregated label maps from different NN searches (4) is shown in Figure 4(b). Aggregating several estimates enables to improve the segmentation performances by smoothing the decision at superpixel boundaries. A reduce number of $M = 4$ label maps is chosen in the following.

[1]Dataset available at: http://rgiraud.vvv.enseirb-matmeca.fr/nnsc/

C. Comparison to the State-of-the-Art Methods

1) Segmentation Performances: Performances are reported for several superpixel scales in Figure 5. NNSC obtains performances similar to TASP [17] on the CTI dataset Figure 5(a), showing its capacity to produce texture-aware superpixels. while performing as well or better than the best compared methods on the BSD Figure 5(b). Note that these results are obtained using the same parameters.

NNSC is also visually compared to the most recent state-of-the-art approaches in Figure 6. On the natural color image, NNSC provides relevant superpixels that accurately detect structures, *e.g.*, the tree or the bear's arm. On the complex composite texture image, NNSC provides more accurate segmentation, with much less fuzzy superpixel shapes.

2) Computational Complexity: NNSC presents a significantly reduced complexity compared to the TASP texture-aware superpixel approach [17], whose complexity depends on the number of image pixels $|I|$, number of K-means iterations N_K, and number of NN search iterations N such that, $C_{\text{TASP}} = \mathcal{O}(|I|\times4\times N_K\times N)$, while $C_{\text{NNSC}} = \mathcal{O}(|I|\times N)$, since it is directly based on a NN clustering framework.

NNSC takes around 2s in its default settings, while TASP requires in average 60s to decompose a BSD image of 321×481 pixels on a linux computer with 4 cores at 1.90GHz and 16GB of RAM. With costly patch-based distances to handle textures, and without advanced code optimizations, NNSC achieves computational times similar to the ones of accurate methods such as [9], [15]. Finally, NNSC could reach real-time performances since several works have proposed such PM implementations using GPU architectures [26].

V. Conclusion

In this work, we propose a new superpixel method considering information at the patch scale to cluster pixels having similar local texture properties. The proposed approach iteratively clusters pixels using a locally constrained patch-based nearest neighbor matching. This way, it significantly reduces the complexity of existing texture-aware approaches, while preserving the accuracy of segmentation. Future works will focus on the extension of the proposed method to 3D supervoxel decomposition, with real-time processing, for applications such as object tracking on video.

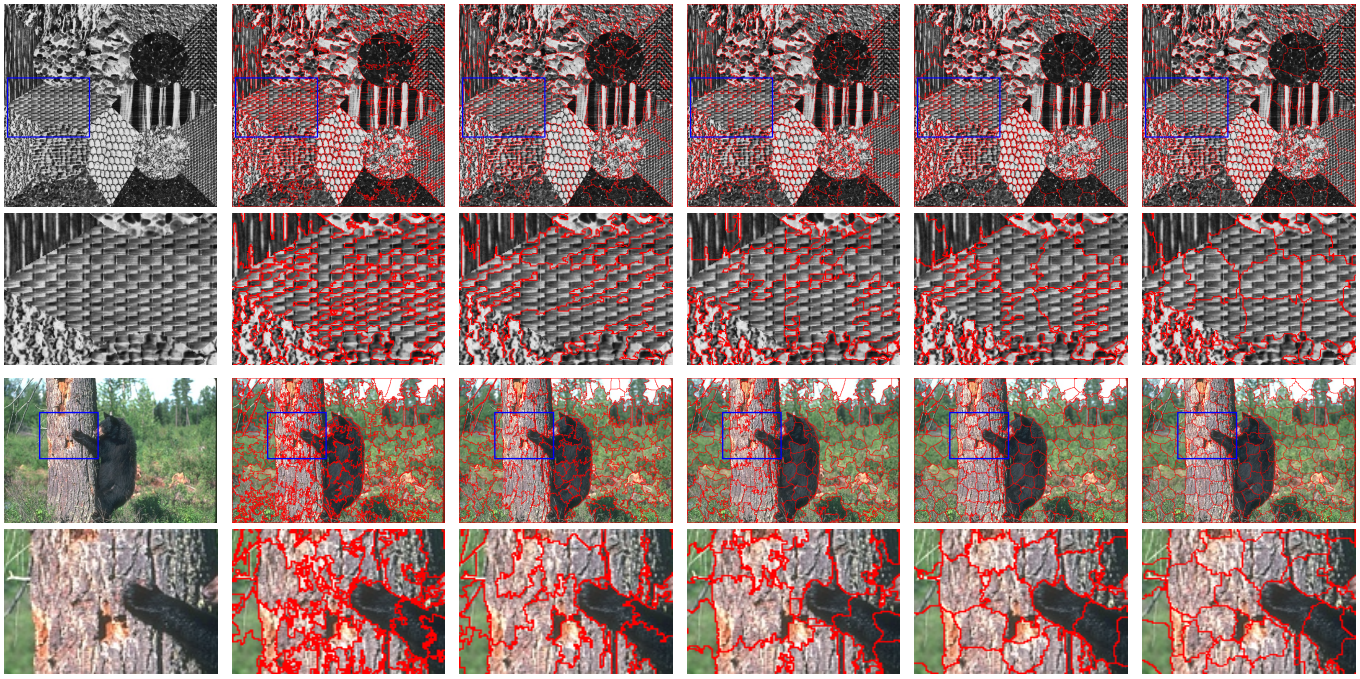


Fig. 6. Visual comparison of the proposed NNSC method to the most recent state-of-the-art approaches on a CTI (top) and BSD example (bottom) for 200 superpixels. NNSC provides as accurate or better decompositions with much less fuzzy superpixels on both texture composite and natural color images.

REFERENCES

[1] K. Nakamura, and B.-W. Hong, "Hierarchical image segmentation via recursive superpixel with adaptive regularity," in *International Conference on Computer Vision*, vol. 26, 2017.

[2] R. Giraud, V.-T. Ta, A. Bugeau, P. Coupé, and N. Papadakis, "SuperPatchMatch: an algorithm for robust correspondences using superpixel patches," in *Trans. on Image Processing*, vol. 26, pp. 4068–4078, 2017.

[3] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich, "Feedforward semantic segmentation with zoom-out features," in *Int. Conf. on Computer Vision and Pattern Recognition*, pp. 3376–3385, 2015.

[4] M. Menze, and A. Geiger, "Object scene flow for autonomous vehicles," in *Int. Conf. on Computer Vision and Pattern Recognition*, pp. 3061–3070, 2015.

[5] C. L. Zitnick, and S. B. Kang, "Stereo for image-based rendering using image over-segmentation," in *International Journal of Computer Vision*, pp. 49–65, 2007.

[6] J. Rabin, and N. Papadakis, "Non-convex relaxation of optimal transport for color transfer," in *Int. Conf. on Neural Information Processing Systems*, 2014.

[7] J. Liu, W. Yang, X. Sun, and W. Zeng, "Photo stylistic brush: robust style transfer via superpixel-based bipartite graph," in *Trans. on Multimedia*, vol. 20, pp. 1724–1737, 2018.

[8] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *Trans. on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 2274–2282, 2012.

[9] M. Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa, "Entropy rate superpixel segmentation," in *Int. Conf. on Computer Vision and Pattern Recognition*, pp. 2097–2104, 2011.

[10] M. Van den Bergh, X. Boix, G. Roig, B. de Capitani, and L. Van Gool, "SEEDS: superpixels extracted via energy-driven sampling," in *European Conference on Computer Vision*, pp. 13–26, 2012.

[11] J. Yao, M. Boben, S. Fidler, and R. Urtasun, "Real-time coarse-to-fine topologically preserving segmentation," in *Int. Conf. on Computer Vision and Pattern Recognition*, pp. 2947-2955, 2015.

[12] N. Zhang, and L. Zhang, "SSGD: superpixels using the shortest gradient distance," in *International Conference on Image Processing*, pp. 3869–3873, 2017.

[13] J. Chen, Z. Li, and B. Huang, "Linear spectral clustering superpixel," in *Trans. on Image Processing*, vol. 26, pp. 3317–3330, 2017.

[14] R. Achanta, and S. Süsstrunk, "Superpixels and polygons using simple non-iterative clustering," in *Int. Conf. on Computer Vision and Pattern Recognition*, pp. 4895–4904, 2017.

[15] R. Giraud, V.-T. Ta, and N. Papadakis, "Robust superpixels using color and contour features along linear path," in *Computer Vision and Image Understanding*, vol. 170, pp. 1–13, 2018.

[16] D. Stutz, A. Hermans, and B. Leibe, "Superpixels: an evaluation of the state-of-the-art," in *Computer Vision and Image Understanding*, vol. 166, pp. 1–27, 2018.

[17] R. Giraud, V.-T. Ta, N. Papadakis, and Y. Berthoumieu, "Texture-aware superpixel segmentation," *International Conference on Image Processing*, 2019.

[18] Y.-J. Liu, C.-C. Yu, M.-J. Yu, and Y. He, "Manifold SLIC: a fast method to compute content-sensitive superpixels," in *Int. Conf. on Computer Vision and Pattern Recognition*, pp. 651–659, 2016.

[19] P. Buyssens, I. Gardin, S. Ruan, and A. Elmoataz, "Eikonal-based region growing for efficient clustering," in *Image and Vision Computing*, vol. 32, pp. 1045–1054, 2014.

[20] A. Efros, and T. Leung, "Texture synthesis by non-parametric sampling," in *International Conference on Computer Vision*, vol. 2, pp. 1033–1038, 1999.

[21] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *Int. Conf. on Computer Vision and Pattern Recognition*, vol. 2, pp. 60-65, 2005.

[22] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "PatchMatch: a randomized correspondence algorithm for structural image editing," in *Trans. on Graphics*, vol. 28, 2009.

[23] R. Giraud, V.-T. Ta, N. Papadakis, J. V. Manjón, L. Collins, and P. Coupé, "An optimized PatchMatch for multi-scale and multi-feature label fusion," in *Neuroimage*, vol. 124, pp. 770–782, 2016.

[24] T. Randen, and J. H. Husoy, "Filtering for texture classification: a comparative study," in *Trans. on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 291–310, 1999.

[25] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A Database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *International Conference on Computer Vision*, vol. 2, pp. 416–423, 2001.

[26] H. Nover, S. Achar, and D. Goldman, "ESPReSSo: efficient slanted PatchMatch for real-time spacetime stereo," in *Int. Conf. on 3D Vision*, pp. 578–586, 2018.