sensors

MDPI

Article

Supervoxel Segmentation with Voxel-Related Gaussian Mixture Model

Zhihua Ban *, Zhong Chen * and Jianguo Liu

National Key Laboratory of Science and Technology on Multi-spectral Information Processing, School of Automation, Huazhong University of Science and Technology, Wuhan 430074, China; jgliu@ieee.org

* Correspondence: zhihua_ban@hust.edu.cn (Z.B.); henpacked@163.com (Z.C.); Tel.: +86-139-7132-8968 (Z.C.)

Received: 30 October 2017; Accepted: 2 January 2018; Published: 5 January 2018

Abstract: Extended from superpixel segmentation by adding an additional constraint on temporal consistency, supervoxel segmentation is to partition video frames into atomic segments. In this work, we propose a novel scheme for supervoxel segmentation to address the problem of new and moving objects, where the segmentation is performed on every two consecutive frames and thus each internal frame has two valid superpixel segmentations. This scheme provides coarse-grained parallel ability, and subsequent algorithms can validate their result using two segmentations that will further improve robustness. To implement this scheme, a voxel-related Gaussian mixture model (GMM) is proposed, in which each supervoxel is assumed to be distributed in a local region and represented by two Gaussian distributions that share the same color parameters to capture temporal consistency. Our algorithm has a lower complexity with respect to frame size than the traditional GMM. According to our experiments, it also outperforms the state-of-the-art in accuracy.

Keywords: superpixel; supervoxel; video segmentation; Gaussian mixture model; expectation–maximization

1. Introduction

Superpixel segmentation is to partition a still image into atomic segments of similar size and adhering to object boundaries, namely superpixels [1–4]. In recent decades, superpixel segmentation has been found to be a very useful preprocessing step in many computer vision tasks (e.g., object detection [5–7], image segmentation [8–10], visual saliency [11], and noise estimation [12]). This is mainly because superpixels improve the computational efficiency and robustness of subsequent applications by reducing the number of inputs and removing a large amount of redundant information.

Because of the effectiveness of superpixel segmentation, the idea of partitioning data points into homogeneous atomic clusters has been extended into video analysis by adding a constraint on temporal consistency [13,14]. The new atomic cluster in video is called the *supervoxel*, as the video analog to the superpixel in a still image [15]. In addition to the constraints inherited from superpixel segmentation (i.e., adhering to object boundaries and having similar size), the new temporal constraint—namely spatiotemporal coherence—requires that a supervoxel belong to the same object over time. Superpixel and supervoxel are used interchangeably in the following text because a supervoxel in a single frame is a superpixel. Existing works solve the supervoxel problem by either stacking video frames together as 3D volumetric data and performing segmentation by treating the time axis as an additional spatial dimension (e.g., [1,16]), or tracking or propagating the initial superpixel segmentation from the first frame through inferring temporal correspondence in successive frames (e.g., [17–19]). Methods falling into the first category cluster video pixels in 3D Euclidean space with color information added to each point. This strategy may result in supervoxels only preserving temporal consistency in very few neighboring frames. When more frames are considered, the same object in the video can be easily separated into different supervoxels, even if the video is completely stacked by the same single still

image. These kinds of methods seem to be more suitable for real 3D volumetric data (e.g., 3D electron microscope (EM) images), and not appropriate for video.

More methods explore the second strategy to improve the temporal consistency of supervoxels. Generally, those methods extract the superpixels of the current frame by using the immutable segmentation of the previous frames [13,17,20]. For instance, the method using partially absorbing random walks (PARW) [17] initializes superpixels for the current frame using the seeds of the previous frame and generates new superpixels based on the current frame and the next frame. Because the segmentations of the previous frames are immutable, the seeds that were used to initialize superpixels of the current frame may change to a different object due to occlusions, and thus the temporal consistency is easily lost (Figure 1a illustrates this problem). As shown in Figure 1a, temporal superpixels (TSP) [20] is able to detect "dead" and "new" superpixels to deal with object occlusion. However, their model overacts when dealing with moving objects, as shown in Figure 1b. Moreover, most of the existing supervoxel algorithms rely on optical flows [18,19,21], which strongly influence their temporal consistency and execution speed.

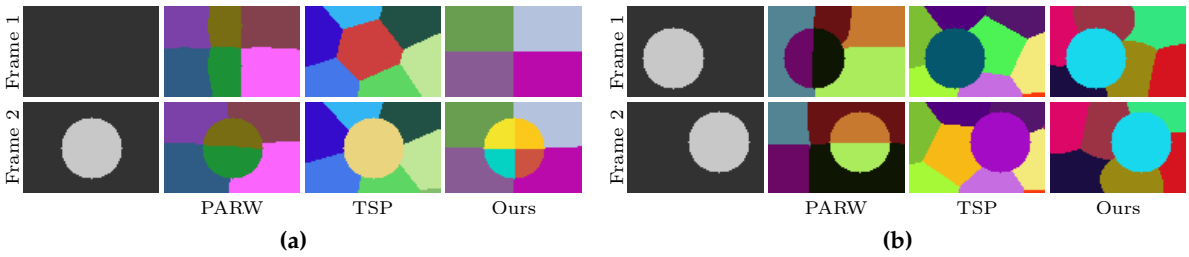

Figure 1. Two toy examples of supervoxel segmentation on (**a**) video with new object and (**b**) video with moving object. The first row are synthetic video frames. The results of two representative state-of-the-art algorithms—PARW [17] and TSP [20]—are plotted in the second and third rows. Our results are shown in the last row. Pixels with the same supervoxel label are painted using the same color, best viewed in color.

Aiming to improve the temporal consistency of supervoxel segmentation when video is with object occlusion and moving objects, we propose an alternative scheme that takes only two adjacent frames into account at a time and produces two valid segmentations for each internal frame. Except for the original colors and spatial locations, our method does not rely on any precomputed information (e.g. optical flows). This scheme has two major benefits: (a) it provides a coarse-grained parallelism that every two frames can be segmented in parallel; (b) each internal frame having two segmentations gives the subsequent applications an opportunity to validate their results. A possible usage of our supervoxels is depicted in Figure 2, where the propagation of a segmentation is not performed at the superpixel level but at the object level.

The traditional Gaussian mixture model (GMM)—a weighted sum of Gaussian functions—has been widely applied to the problem of classification [22,23]. However, it cannot be directly applied to supervoxel segmentation because its computational complexity is relatively high and it does not encode the constraint on segment size. GMM has been explored for superpixel segmentation and achieved good segmentation accuracy in our previous work [24]; however, temporal consistency was not considered. In this work, we extend the model of [24] to supervoxel segmentation and propose a voxel-related GMM to tackle temporal consistency. Inherited from [24], we use constant weights and subsets of all the Gaussian functions in the sums to ensure that the produced segments have similar sizes and that the algorithm has linear complexity. In our new model, the size of the subsets is controllable so that we can tune it to track objects with different moving speed. For instance, a large size is suitable for videos containing fast-moving objects (see Section 3.4 for more details). Each supervoxel is composed of two superpixels on two consecutive frames, and each superpixel is represented by a Gaussian distribution. To ensure temporal consistency, we use the same color parameters (i.e., color

mean vector and color covariance matrix) for the two distributions of each supervoxel. This is mainly because the same object in two consecutive frames tends to be similar in color. Experiments conducted on a well-known dataset show that the proposed algorithm is superior to the state-of-the-art in terms of accuracy.

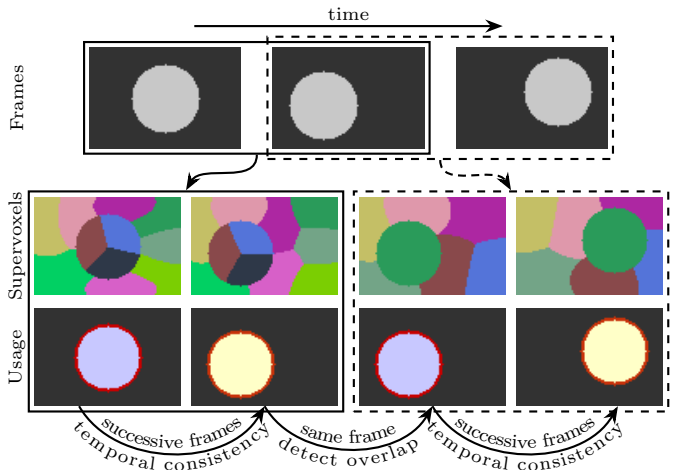

Figure 2. An illustration of a possible usage of the proposed method. The first row is three successive frames. In the second row, the first two superpixel segmentations are extracted using only the first two frames. Similarly, the last two superpixel segmentations are extracted using only the last two frames. In this example, the second frame has two superpixel segmentations. The third row explains a possible usage in foreground segmentation or object tracking, best viewed in color.

The rest of this paper is organized as follows: some related works are introduced in Section 2. Section 3 presents our supervoxel method. Experiments are conducted in Section 4. Finally, we conclude our work in Section 5.

2. Related Works

Using superpixels as basic elements for image analysis and processing was first introduced by Ren and Malik [25]. After the algorithms proposed in [1,16,26], the following works of extracting superpixels for video data began to take temporal consistency into account. In this section, we will review some representative algorithms that are related to supervoxel segmentation.

The hierarchical graph-based method (GBH) [26] constructs a 3D graph using all the video frames and applies Felzenszwalb and Huttenlocher's algorithm [27] to iteratively merge voxels using a hierarchical scheme. Instead of building a regular grid graph base on a 26-neighborhood in 3D spatial temporal space, the edges between frames are built in such a way that each voxel is connected to its nine neighbors along the backward flow vector. The flow vectors are also used in the clustering process. This method uses dense optical flow to ensure temporal consistency. However, errors from the precomputed flows may shift to the segmentation process. Additionally, GBH cannot generate supervoxels with similar size, and the number of supervoxels cannot be directly controlled. Since GBH requires all frames to be loaded into memory (in which case it may fail for a longer video), Ref. [28] gives an implementation of GBH to make it have a streaming capability by using a Markov assumption.

Simple linear iterative clustering (SLIC) [1] uses a modified k-means to group pixels based on their spatial location in a still image. In each iteration of SLIC, the search space of the current superpixel is limited to a square region whose center is the spatial center of the current superpixel and whose size is proportional to the desired size of each superpixel. To extend the modified k-means to video data, voxels in frames are seen as points in 3D Euclidean space, and thus the search space becomes a cube. Seeds are regularly distributed among the fake 3D volumetric data, and as a result supervoxels

are temporally consistent only over a short range of frames. In SLIC, small connected regions are merged into neighboring supervoxels in a 3D 10-neighborhood. However, a moving object captured in different frames may not be connected in the 10-neighborhood. Therefore, the merging step may cause a negative effect on the segmentation accuracy. Similar to SLIC, the method besed on graph cuts (GC) [16] also stacks all frames together as a fake 3D volume. GC extracts supervoxels by partitioning graphs in an energy minimization framework optimized using graph cuts. However, GC cannot guarantee temporal consistency for a long range of frames.

Spatiotemporal closure (SC) [18] starts by extracting superpixels in the first frame using the original method of TurboPixels [3]. The seeds for the superpixel segmentation of the next frame are projected along the weighted flow vectors from the segmented seeds of the current frame. This method also relies on precomputed optical flow and is not self-contained. An incorrect flow vector may easily produce a supervoxel that covers multiple objects. Following works like Ref. [13], PARW [17] use a similar method to move seeds from the superpixel segmentation of the current frame to the next adjacent frame. Although TSP [20] does not move seeds with the aid of optical flow, seeds are still moved from an immutable superpixel segmentation, and the seeds may be further evolved to a new object. Superpixels Extracted via Energy-Driven Sampling (SEEDS) [29,30] was first designed for superpixel extraction, and is extended to video data in video SEEDS (vSEEDS)[19]. Instead of moving superpixel representatives from previous segmentations, vSEEDS propagates the rough block-level segmentation of each frame into the next frame. However, vSEEDS still shares the same drawback with the seeds moving methods when dealing with new objects.

Overall, existing supervoxel algorithms generally require motion information to aid the segmentation. However, this kind of information is not originally equipped with the frames, and needs to be computed by additional algorithms. The error in motion vectors may result in erroneous results in supervoxels. Algorithms that evolve previous superpixel segmentation to new frames usually make the previous segmentation immutable. Because the previous labels may shift to a new object, these methods often fail to handle new objects or moving objects.

3. The Method

In the proposed method, the supervoxel problem is simplified as supervoxel segmentation on two adjacent frames (see Figure 2 for an illustration). Each voxel is represented by a five-dimensional vector, in which the time property is not involved. Inspired by Ref. [13], the time property of each voxel is modeled in an implicit fashion such that data points are organized in subspaces: one color subspace and two spatial subspaces (each frame has one spatial subspace). Each supervoxel is composed of two superpixels, each of which is associated to a Gaussian distribution with unknown parameters. To estimate the parameters, voxels are assumed to be observed from a mixture of Gaussian distributions. Based on maximum likelihood estimation, the unknown parameters are estimated using the expectation–maximization (EM) method. Once the values of the parameters are obtained, each voxel's supervoxel label is determined to be the one that has the maximum posterior probability.

3.1. Problem Formulation

For a given sequence of frames, we use z_i^t to denote pixel i in frame t, with $i \in \mathcal{I}$ and $t \in T$, where $\mathcal{I}$ is the pixel index set and T is the frame set. The symbol $|\cdot|$ denotes the number of elements in a given set; e.g., $|\mathcal{I}|$ is the number of pixels in each frame, and $|T|$ is the number of frames. The width and height of the frames are denoted by W and H, respectively. Hence, we have $|\mathcal{I}| = W \cdot H$.

The desired size of each superpixel in a given frame is specified by $v_x \cdot v_y$, where v_x and v_y are the number of pixels along width and height, respectively. Usually, the values of v_x and v_y are given by users. If we use values with a large difference, the shape of the generated superpixels in each frame will tend to be a narrow rectangle (see Figure 3). Although people can assign different values to them, it is encouraged to assign the same value, or at least two different values with a very small difference, unless the narrow shape is useful for a special purpose.

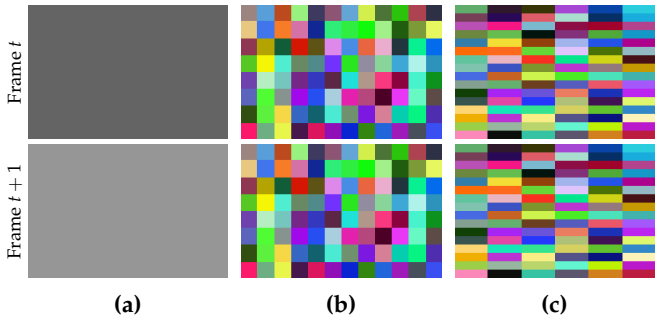

Figure 3. Supervoxel segmentation with different values for v_x and v_y. (**a**) two successive synthetic frames with constant colors; (**b**) supervoxels with $v_x = v_y$; (**c**) supervoxels with $v_x = 4v_y$; In both (**b**) and (**c**), voxels with the same color form a supervoxel. Although (**b**) and (**c**) use different v_x and v_y, the number of the generated supervoxels are the same, but the shapes are very different.

Supervoxel segmentation is to assign each voxel a unique label. Voxels with the same label form a supervoxel. All the possible supervoxel labels form a supervoxel set $\mathcal{K} = \{0, 1, \ldots, K-1\}$, where $K = |\mathcal{K}|$ is the number of supervoxels, which is computed in Equation (1):

$$n_x = \frac{W}{v_x}, n_y = \frac{H}{v_y}, K = n_x \cdot n_y, \tag{1}$$

where n_x and n_y are the desired numbers of superpixels along width and height for a single frame. For simplicity, we assume $W \bmod v_x = 0$ and $H \bmod v_y = 0$.

In this work, the supervoxel segmentation procedure is performed on every two frames. Therefore, a supervoxel is only valid on two successive frames. For instance, if some voxels in frame t and frame $t+1$ share the same supervoxel label, they are a subset of the same supervoxel. However, if the voxels are in frame t and frame $t+3$, they are not in the same supervoxel. In order to provide cues for some subsequent applications (e.g., object tracking), frame t—where $t \neq 0$ and $t \neq |T|-1$—will be used two times. In the first time, the segmentation is performed on frame $t-1$ and frame t. In the second time, the same procedure of segmentation is performed on frames t and $t+1$. By doing this, frame t will have two valid superpixel segmentations. The detected regions in frame t can be propagated by finding the overlapping superpixels in the second segmentation (see Figure 2 for an illustration). Because the same methods are used to segment any two frames, the supervoxel problem becomes finding K supervoxels for frame t and $t+1$ such that the generated supervoxels are similar in size, adhere to object boundaries well, and are temporally consistent. Therefore, only frame t and frame $t+1$ will be considered in Section 3.2.

3.2. The Model

To distinguish variables between two frames, a symbol with a hat at its top indicates that the symbol is related to frame $t+1$. Each voxel in frame t is represented by a five-dimensional vector including spatial location (x_i, y_i) and three CIELAB color components—lightness u_i and two color components a_i and b_i. This can be expressed by

$$z_i = (x_i, y_i, u_i, a_i, b_i)^T, \tag{2}$$

in which the superscript T indicates vector transpose. Similarly, voxel i in frame $t+1$ is represented using

$$\hat{z}_i = (\hat{x}_i, \hat{y}_i, \hat{u}_i, \hat{a}_i, \hat{b}_i)^T. \tag{3}$$

For two given frames t and $t+1$, voxels are assumed to be distributed according to mixtures of Gaussian distributions in which each Gaussian distribution corresponds to a superpixel. Gaussian

function $g(\cdot;\cdot)$ is defined in Equation (4), where the semicolon is used to separate variables and parameters:

$$g(z;\mu,\Sigma) = \frac{1}{\sqrt[D]{2\pi}\sqrt{\det(\Sigma)}} \exp\left\{(z-\mu)^T \Sigma^{-1} (z-\mu)\right\}, \tag{4}$$

in which z is a D-dimensional column vector, μ and Σ are mean vector and covariance matrix, respectively.

Generally, if voxels in different frames have similar colors and have small spatial distances, they generally belong to the same object. This is particularly true in the videos with moving objects. To incorporate this notion into our model, we use different parameters for spatial information but the same parameters for color information for the same supervoxel in the definition of voxel density functions $d(\cdot)$ and $\hat{d}(\cdot)$, as shown in Equation (5). This is the key point to ensure temporal consistency:

$$d(z_i) = \sum_{k\in\mathcal{K}_i} P_k^i \cdot g(z_i;\mu_k,\Sigma_k), \quad \hat{d}(\hat{z}_i) = \sum_{k\in\mathcal{K}_i} P_k^i \cdot g(\hat{z}_i;\hat{\mu}_k,\hat{\Sigma}_k), \tag{5}$$

in which

$$\mu_k = (\mu_k^s, \mu_k^c)^T, \ \hat{\mu}_k = (\hat{\mu}_k^s, \mu_k^c)^T, \ \Sigma_k = \begin{bmatrix} \Sigma_k^s & 0 \\ 0 & \Sigma_k^c \end{bmatrix}, \ \hat{\Sigma}_k = \begin{bmatrix} \hat{\Sigma}_k^s & 0 \\ 0 & \Sigma_k^c \end{bmatrix}, \tag{6}$$

where μ_k^s and Σ_k^s are spatial mean vectors and spatial covariance matrices for frame t with $k \in \mathcal{K}$. Their parallel notations $\hat{\mu}_k^s$ and $\hat{\Sigma}_k^s$ are for frame $t+1$. The color mean vectors μ_k^c and color covariance matrices Σ_k^c are for both frame t and frame $t+1$. Supervoxel k can be characterized by the parameters μ_k^s, $\hat{\mu}_k^s$, Σ_k^s, $\hat{\Sigma}_k^s$, μ_k^c, and Σ_k^c, and thus each supervoxel corresponds to two Gaussian distributions. Accounting for the locality of supervoxels, $\mathcal{K}_i$ in Equation (5) is a subset of $\mathcal{K}$, and its elements are related to the spatial location of voxel i. The definition of $\mathcal{K}_i$ will be discussed later. Instead of defining P_k^i as a variable just like existing Gaussian mixture models, P_k^i is defined as a constant $1/|\mathcal{K}_i|$ here to make the generated supervoxels similar in size.

Once two successive frames t and $t+1$ are given, parameters in the Gaussian densities can be inferred and the label of each voxel, l_i for frame t and $\hat{l}_i$ for frame $t+1$, will be determined by the following equations:

$$l_i = \arg_{k\in\mathcal{K}_i} \max \Pr(k \mid z_i), \quad \hat{l}_i = \arg_{k\in\mathcal{K}_i} \max \Pr(k \mid \hat{z}_i), \tag{7}$$

in which $\Pr(k \mid z_i)$ is the probability of assigning voxel i in frame t to supervoxel k given the observation z_i. $\Pr(k \mid \hat{z}_i)$ has a similar meaning. By applying Bayes rule, the posterior probabilities in Equation (7) can be expressed by

$$\Pr(k \mid z_i) = \frac{g(z_i;\mu_k,\Sigma_k)\, P_k^i}{p(z_i)}, \quad \Pr(k \mid \hat{z}_i) = \frac{g(\hat{z}_i;\hat{\mu}_k,\hat{\Sigma}_k)\, P_k^i}{p(\hat{z}_i)}. \tag{8}$$

Based on Equations (7) and (8), l_i and $\hat{l}_i$ can be computed by the equivalent equations, as shown below:

$$l_i = \arg_{k\in\mathcal{K}_i} \max g(z_i;\mu_k,\Sigma_k), \quad \hat{l}_i = \arg_{k\in\mathcal{K}_i} \max g(\hat{z}_i;\hat{\mu}_k,\hat{\Sigma}_k). \tag{9}$$

3.3. Estimating Parameters of Gaussian Distributions

Given two frames t and $t+1$, we use the method of maximum likelihood to estimate the unknown parameters in the Gaussian distributions. Since the proposed density functions for voxels may be not identical because the elements in $\mathcal{K}_i$ may be different for different $i \in \mathcal{I}$ (see Section 3.4 for details), updating formulas for traditional GMM cannot be simply copied to our new model. Therefore, we will derive the updating formulas for the proposed model in this section by applying the classical expectation–maximization (EM) method to iteratively improve the log-likelihood.

As the voxels in frames t and $t+1$ are assumed to be distributed independently, the log-likelihood $\tilde{\mathcal{L}}(\theta)$ for the two frames can be written out as follows:

$$\tilde{\mathcal{L}}(\theta) = \sum_{i\in\mathcal{I}} \left\{ \log\left(d(z_i)\right) + \log\left(\hat{d}(\hat{z}_i)\right) \right\} \tag{10}$$

$$= \sum_{i\in\mathcal{I}} \left\{ \log\left(\frac{1}{|\mathcal{K}_i|^2}\right) + \log\left(\sum_{k\in\mathcal{K}_i} g(z_i;\theta_k)\right) + \log\left(\sum_{k\in\mathcal{K}_i} g(\hat{z}_i;\hat{\theta}_k)\right) \right\}, \tag{11}$$

where θ is a vector of all the unknown parameters composed of μ_k^s, $\hat{\mu}_k^s$, Σ_k^s, $\hat{\Sigma}_k^s$, μ_k^c, and Σ_k^c with $k \in \mathcal{K}$. For each supervoxel k, $\theta_k = (\mu_k, \Sigma_k)$ and $\hat{\theta}_k = (\hat{\mu}_k, \hat{\Sigma}_k)$. Because the number of elements in supervoxel set $\mathcal{K}_i$ is constant (see Section 3.4), the value of the parameter θ that maximizes $\tilde{\mathcal{L}}(\theta)$ is equal to the value that maximizes the following function $\mathcal{L}(\theta)$:

$$\mathcal{L}(\theta) = \sum_{i\in\mathcal{I}} \left\{ \log\left(\sum_{k\in\mathcal{K}_i} g(z_i;\theta_k)\right) + \log\left(\sum_{k\in\mathcal{K}_i} g(\hat{z}_i;\hat{\theta}_k)\right) \right\}. \tag{12}$$

It is difficult to find the optimal value for θ by maximizing $\mathcal{L}(\theta)$ directly. We insert new variables R_k^i and $\hat{R}_k^i$ into Equation (12) such that

$$\sum_{k\in\mathcal{K}_i} R_k^i = 1\,,\ R_k^i \geq 0\,,\ \sum_{k\in\mathcal{K}_i} \hat{R}_k^i = 1\,,\ \hat{R}_k^i \geq 0\,,\ i \in \mathcal{I}\,,\ k \in \mathcal{K}_i\,. \tag{13}$$

Then, Equation (12) will become Equation (14). By applying Jensen's inequality, the EM method is to alternatively find $R = \{R_k^i, \hat{R}_k^i | i \in \mathcal{I}, k \in \mathcal{K}_i\}$ satisfying the equality of the inequality in Equation (15) with parameters in θ being known (expectation step or E-step), and find parameters in θ that maximize $Q(R,\theta)$, which is defined in Equation (15) using the obtained R (maximization step or M-step):

$$\mathcal{L}(\theta) = \sum_{i\in\mathcal{I}} \left\{ \log\left(\sum_{k\in\mathcal{K}_i} R_k^i \frac{g(z_i;\theta_k)}{R_k^i}\right) + \log\left(\sum_{k\in\mathcal{K}_i} \hat{R}_k^i \frac{g(\hat{z}_i;\hat{\theta}_k)}{\hat{R}_k^i}\right) \right\} \tag{14}$$

$$\geq \sum_{i\in\mathcal{I}} \left\{ \sum_{k\in\mathcal{K}_i} R_k^i \log\left(\frac{g(z_i;\theta_k)}{R_k^i}\right) + \sum_{k\in\mathcal{K}_i} \hat{R}_k^i \log\left(\frac{g(\hat{z}_i;\hat{\theta}_k)}{\hat{R}_k^i}\right) \right\} \doteq Q(R,\theta)\,. \tag{15}$$

E-step: According to the theory of Jensen's inequality, equality holds if and only if

$$\frac{g(z_i;\theta_k)}{R_k^i} \doteq \alpha_i\,, \text{ and } \frac{g(\hat{z}_i;\hat{\theta}_k)}{\hat{R}_k^i} \doteq \hat{\alpha}_i \tag{16}$$

are constant. With the constraints in Equation (13), formulas to update R can be derived by eliminating the temporal variables α_i and $\hat{\alpha}_i$ in Equation (16), as shown below:

$$R_k^i = \frac{g(z_i;\theta_k)}{\sum_{k\in\mathcal{K}_i} g(z_i;\theta_k)}\,,\ \hat{R}_k^i = \frac{g(\hat{z}_i;\hat{\theta}_k)}{\sum_{k\in\mathcal{K}_i} g(\hat{z}_i;\hat{\theta}_k)}\,, \tag{17}$$

where $\mathcal{K}_i$ is defined in Section 3.4.

M-step: To find the parameters θ that maximize $Q(R,\theta)$, we first get the partial derivatives of $Q(R,\theta)$ with respect to different components of θ, as shown in Equations (18)–(22), and then set them to zero to get the optimal θ, which is shown in Equations (24)–(27):

$$\frac{\partial Q(R,\theta)}{\partial \mu_k^s} = \sum_{i\in\mathcal{I}_k} \left\{ R_k^i (\Sigma_k^s)^{-1} (z_i^s - \mu_k^s) \right\}, \quad \frac{\partial Q(R,\theta)}{\partial \hat{\mu}_k^s} = \sum_{i\in\mathcal{I}_k} \left\{ \hat{R}_k^i (\hat{\Sigma}_k^s)^{-1} (\hat{z}_i^s - \hat{\mu}_k^s) \right\}, \tag{18}$$

$$\frac{\partial Q(R,\theta)}{\partial \Sigma_k^s} = \sum_{i \in \mathcal{I}_k} \frac{R_k^i}{2} \left\{ (\Sigma_k^s)^{-1}(z_i^s - \mu_k^s)(z_i^s - \mu_k^s)^T(\Sigma_k^s)^{-1} - (\Sigma_k^s)^{-1} \right\}, \tag{19}$$

$$\frac{\partial Q(R,\theta)}{\partial \hat{\Sigma}_k^s} = \sum_{i \in \mathcal{I}_k} \frac{\hat{R}_k^i}{2} \left\{ (\hat{\Sigma}_k^s)^{-1}(\hat{z}_i^s - \hat{\mu}_k^s)(\hat{z}_i^s - \hat{\mu}_k^s)^T(\hat{\Sigma}_k^s)^{-1} - (\hat{\Sigma}_k^s)^{-1} \right\}, \tag{20}$$

$$\frac{\partial Q(R,\theta)}{\partial \mu_k^c} = \sum_{i \in \mathcal{I}_k} \left\{ R_k^i(\Sigma_k^c)^{-1}(z_i^c - \mu_k^c) + \hat{R}_k^i(\Sigma_k^c)^{-1}(\hat{z}_i^c - \mu_k^c) \right\}, \tag{21}$$

$$\begin{aligned} \frac{\partial Q(R,\theta)}{\partial \Sigma_k^c} = \sum_{i \in \mathcal{I}_k} \Bigg\{ & \frac{R_k^i}{2} \left\{ (\Sigma_k^c)^{-1}(z_i^c - \mu_k^c)(z_i^c - \mu_k^c)^T(\Sigma_k^c)^{-1} - (\Sigma_k^c)^{-1} \right\} + \\ & \frac{\hat{R}_k^i}{2} \left\{ (\Sigma_k^c)^{-1}(\hat{z}_i^c - \mu_k^c)(\hat{z}_i^c - \mu_k^c)^T(\Sigma_k^c)^{-1} - (\Sigma_k^c)^{-1} \right\} \Bigg\}, \end{aligned} \tag{22}$$

where $z_i^s = (x_i, y_i)^T$ and $\hat{z}_i^s = (\hat{x}_i, \hat{y}_i)^T$ are spatial vectors of voxel i in frame t and $t+1$, respectively. Similarly, $z_i^c = (u_i, a_i, b_i)^T$ and $\hat{z}_i^c = (\hat{u}_i, \hat{a}_i, \hat{b}_i)^T$ are color vectors. For each supervoxel k, $\mathcal{I}_k$ is a voxel set that supervoxel k may cover, and is deduced from $\mathcal{K}_i$ as shown in Equation (23):

$$\mathcal{I}_k = \{ i \,|\, i \in \mathcal{I}, k \in \mathcal{K}_i \}, \tag{23}$$

$$\mu_k^s = \frac{\sum_{i \in \mathcal{I}_k} R_k^i z_i^s}{\sum_{i \in \mathcal{I}_k} R_k^i}, \quad \Sigma_k^s = \frac{\sum_{i \in \mathcal{I}_k} R_k^i (z_i^s - \mu_k^s)(z_i^s - \mu_k^s)^T}{\sum_{i \in \mathcal{I}_k} R_k^i}, \tag{24}$$

$$\hat{\mu}_k^s = \frac{\sum_{i \in \mathcal{I}_k} \hat{R}_k^i \hat{z}_i^s}{\sum_{i \in \mathcal{I}_k} \hat{R}_k^i}, \quad \hat{\Sigma}_k^s = \frac{\sum_{i \in \mathcal{I}_k} \hat{R}_k^i (\hat{z}_i^s - \hat{\mu}_k^s)(\hat{z}_i^s - \hat{\mu}_k^s)^T}{\sum_{i \in \mathcal{I}_k} \hat{R}_k^i}, \tag{25}$$

$$\mu_k^c = \frac{\left(R_k^i z_i^c + \hat{R}_k^i \hat{z}_i^c \right)}{\sum_{i \in \mathcal{I}_k} \left(R_k^i + \hat{R}_k^i \right)}, \tag{26}$$

$$\Sigma_k^c = \frac{R_k^i (z_i^c - \mu_k^c)(z_i^c - \mu_k^c)^T + \hat{R}_k^i (\hat{z}_i^c - \mu_k^c)(\hat{z}_i^c - \mu_k^c)^T}{\sum_{i \in \mathcal{I}_k} \left(R_k^i + \hat{R}_k^i \right)}. \tag{27}$$

Usually, the EM method starts by feeding it with a guess of the parameters in θ. Then, R and θ will be alternatively updated using the formulas mentioned in E-step and M-step. However, there is a risk that the covariance matrices may become singular and we may fail to obtain their inverse matrices. For example, when the voxels in $\mathcal{I}_k$ have the same constant color, during the iteration of EM Σ_k^c will become zero matrix, which is obviously singular. To avoid this trouble, one can perturb each z_i^c and each $\hat{z}_i^c$ with a small random vector before the EM iterations. This trick may succeed in most cases, but it may fail when an object in a frame is very narrow (e.g., a straight line), in which case certain covariance matrices Σ_k^s or $\hat{\Sigma}_k^s$ may become singular. In order to prevent the covariance matrices from being singular, we first obtain their eigenvalues and impose a lower bound to the eigenvalues to reproduce the covariance matrices. When an eigenvalue is less than the specified lower bound, we assign the eigenvalue to that lower bound. We use λ_s and λ_c to denote the lower bound of spatial eigenvalues and color eigenvalues, respectively. Experimentally, we have found that $\lambda_s = 2$ and $\lambda_c = 8$ is appropriate to outperform the state-of-the-art algorithms. We will use this setting for the remaining text.

In theory, the iteration of the EM method will not stop until the parameters in θ converge. However, EM needs hundreds of iterations to reach the condition of convergence, resulting in a low computational efficiency. In practice, aiming to reduce run-time, we use a fixed number of iterations $M = 20$, which is sufficient in practice for generating supervoxels with state-of-the-art accuracy.

3.4. Defining $\mathcal{K}_i$ and Initializing θ

The definition of supervoxel subsets $\mathcal{K}_i$ of $\mathcal{K}$ will be discussed in this section using mainly the notations mentioned in Section 3.1. After the value of v_x and v_y are assigned, we define $n_x \cdot n_y$ rectangle regions called *anchor regions*, with each anchor region corresponding to a supervoxel. As illustrated in Figure 4, for each frame, an anchor region contains $v_x \cdot v_y$ voxels and all the anchor regions are regularly placed on every frame. For a given supervoxel k, all voxels in its anchor region are assigned an *anchor label* $h_i = k$. Then, $\mathcal{K}_i$ is defined using the anchor label of voxel i using the following equations:

$$h_i^x \doteq h_i \bmod n_x\,,\ h_i^y \doteq \lfloor h_i / n_x \rfloor\,, \tag{28}$$

$$\mathcal{K}_i \doteq \left\{ k_y + k_x \cdot n_x \,\middle|\, \begin{array}{l} k_x \in \{h_i^x - \eta_x, \ldots, h_i^x + \eta_x\}, 0 \le k_x \le n_x - 1 \\ k_y \in \{h_i^y - \eta_y, \ldots, h_i^y + \eta_y\}, 0 \le k_y \le n_y - 1 \end{array} \right\}, \tag{29}$$

where η_x and η_y are parameters used to control the number of supervoxels from which each voxel i may be generated. Clearly, at least one of the two parameters must be greater than or equal to 1.

Recall the definition of $\mathcal{I}_k$ in Equation (23) of Section 3.3. Elements of $\mathcal{I}_k$ are in turn determined by $\mathcal{K}_i$. The voxel set $\mathcal{I}_k$ is called the k-th supervoxel's *overlap region*, into which voxels in supervoxel k may spread. With the help of Figure 4, it is easy to conclude that an overlap region $\mathcal{I}_k$ of a supervoxel k is the region whose center is the anchor region of the supervoxel k and whose width and height can be divided evenly by v_x and v_y respectively, and can be expressed by η_x and η_y using $(2\eta_x + 1)v_x$ and $(2\eta_y + 1)v_y$. There are some exceptions, however. When an anchor region k is at the boundary of a frame, the size of $\mathcal{I}_k$ may be less than $(2\eta_x + 1)v_x \cdot (2\eta_y + 1)v_y$, but is at least $(\eta_x + 1)v_x \cdot (\eta_y + 1)v_y$, which is the case in which the anchor region is at one of the four corners of the frame. This conclusion can be used to deduce the computational complexity of our algorithm. As discussed in Section 3.5, the computational complexity can be affected by η_x and η_y. A large value for $\eta_x \cdot \eta_y$ will increase the run-time of the algorithm. Meanwhile, a large value for $\eta_x \cdot \eta_y$ indicates that a supervoxel has a large overlap region, which may result in a better performance in temporal consistency. By default, we use $\eta_x = \eta_y = 2$ for our experiments.

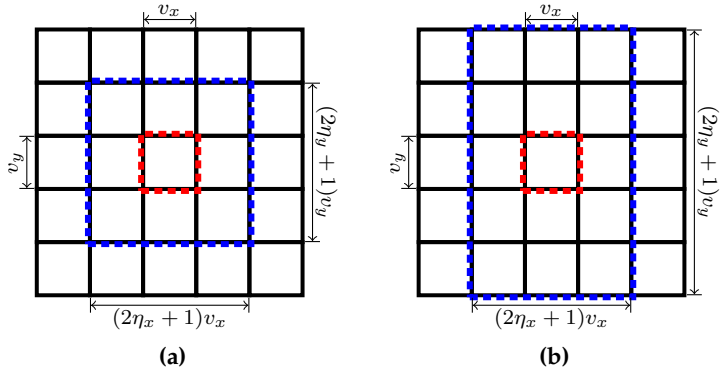

Figure 4. Illustration of anchor region and overlap region with two different settings for η_x and η_y: (**a**) $\eta_x = \eta_y = 1$, (**b**) $\eta_x = 1$, $\eta_y = 2$. In this example of both (**a**) and (**b**), $n_x = n_y = 5$ and the 25 anchor regions are marked with black rectangles. The region within the blue rectangle is the overlap region $\mathcal{I}_k$ of supervoxel $k = 13$ whose anchor region is highlighted by a red rectangle.

In our model, a Gaussian distribution represents a superpixel in a single frame, and two Gaussian distributions with the same color parameters (color mean vector and color covariance matrix) but with different spatial parameters represent a single supervoxel. As we expect that the generated superpixels are regularly distributed on each frame, it is straightforward to initialize μ_k^s and $\hat{\mu}_k^s$ using the center of the k-th anchor region. Since a color mean vector μ_k^c varies according to the color information of two

frames (see Equation (26)) during the EM iterations, the μ_k^c is initialized by the mean color of the two voxels at the center of the k-th anchor region of each frame.

For each supervoxel k, the corresponding covariance matrices serve as normalizers for the squares of the Euclidean distances (refer to Equations (4) and (9)). To initialize each of the covariance matrices, the idea is to assign their diagonals with the same value, which can be interpreted as a distance within which two voxels tend to be in the same supervoxel. We have found that it is sufficient to initialize color covariance matrices with a color distance $\sigma_c = 10$ (see Equation (30)) and a small perturbation for σ_c affects the result less. As we hope each superpixel for a single frame will have the same size $v_x \cdot v_y$, the spatial covariance matrices can be initialized using Equation (30):

$$\Sigma_k^c = \begin{bmatrix} (\sigma_c)^2 & 0 & 0 \\ 0 & (\sigma_c)^2 & 0 \\ 0 & 0 & (\sigma_c)^2 \end{bmatrix}, \Sigma_k^s = \hat{\Sigma}_k^s = \begin{bmatrix} (v_x)^2 & 0 \\ 0 & (v_y)^2 \end{bmatrix}. \tag{30}$$

3.5. Computational Complexity

With the discussion above, for any two successive frames, the proposed algorithm can be summarized in Algorithm 1. The proposed algorithm is composed of three major procedures, initializing θ (line 1 to line 3), updating R (line 5 to line 9), and updating θ (line 10 to line 13). It is obvious that the initialization of θ needs a computational cost of $\mathcal{O}(|\mathcal{K}|) = \mathcal{O}(K)$, where K is the number of desired supervoxels in two frames and is originally defined in Equation (1).

Algorithm 1 The proposed supervoxel algorithm.

Input: v_x and v_y, two successive frames.
Output: l_i and $\hat{l}_i$, $i \in \mathcal{I}$.

1: **for all** $k \in \mathcal{K}$ **do**
2: Initialize μ_k^s, $\hat{\mu}_k^s$, Σ_k^s, $\hat{\Sigma}_k^s$, μ_k^c, Σ_k^c (refer to Section 3.4).
3: **end for**
4: **for** $m = 1$ to M **do** {refer to Section 3.3 for the value of M}
5: **for all** $i \in \mathcal{I}$ **do**
6: **for all** $k \in \mathcal{K}_i$ **do** {refer to Section 3.4 for $\mathcal{K}_i$}
7: Update R_k^i and $\hat{R}_k^i$ using Equation (17).
8: **end for**
9: **end for**
10: **for all** $k \in \mathcal{K}$ **do**
11: Update μ_k^s, $\hat{\mu}_k^s$, and μ_k^c using Equations (24)–(26).
12: Update Σ_k^s, $\hat{\Sigma}_k^s$ and Σ_k^c using Equations (24)–(27).
13: **end for**
14: **end for**
15: **for all** $i \in \mathcal{I}$ **do**
16: l_i and $\hat{l}_i$ are determined by Equation (9).
17: **end for**

According to Equations (28) and (29), the number of elements in $\mathcal{K}_i$ satisfies the following inequality:

$$(\eta_x + 1) \cdot (\eta_y + 1) \leq |\mathcal{K}_i| \leq (2\eta_x + 1) \cdot (2\eta_y + 1). \tag{31}$$

For each voxel i, updating R_k^i or $\hat{R}_k^i$, $k \in \mathcal{K}_i$ needs time $\mathcal{O}(|\mathcal{K}_i|)$. For all the elements in R, we therefore have a computational complexity $\mathcal{O}(\eta_x \cdot \eta_y \cdot |\mathcal{I}|)$. Based on Equations (24)–(27), for a given supervoxel k, updating the parameters in θ_k or $\hat{\theta}_k$ needs a time of $\mathcal{O}(|\mathcal{I}_k|)$. By the conclusions about the size of $\mathcal{I}_k$ in Section 3.4, we know that

$$(\eta_x + 1)(\eta_y + 1)v_x v_y \leq |\mathcal{I}_k| \leq (2\eta_x + 1)(2\eta_y + 1)v_x v_y. \tag{32}$$

Therefore, the computational complexity for updating θ is

$$\mathcal{O}(\eta_x \eta_y v_x v_y |\mathcal{K}|) = \mathcal{O}(\eta_x \eta_y v_x v_y n_x n_y) = \mathcal{O}(\eta_x \eta_y |\mathcal{I}|). \tag{33}$$

Because $|\mathcal{K}| \ll |\mathcal{I}|$ and we use constant values for η_x and η_y, the computational complexity of our algorithm is $\mathcal{O}(|\mathcal{I}|)$.

When the input video has more than two frames, every internal frame will have two segmentation results: one generated with its previous frame and another generated with its next frame (refer to Figure 2 for a visual illustration). For the entire video sequence, the complexity is $\mathcal{O}((|T| - 1) \cdot |\mathcal{I}|) = \mathcal{O}(|T| \cdot |\mathcal{I}|)$, where $|T|$ is the number of frames in the input video and has been mentioned in Section 3.1.

4. Experiments

Our method has been designed to produce supervoxels of similar size. To evaluate the performance of our algorithm, it is reasonable to compare the proposed method with algorithms that are also designed to generate supervoxels of similar size. We compared our method with four of these kinds of algorithms, including video SLIC [1] (vSLIC), PARW [17], vSEEDS [19], and TSP [20], whose source codes are publicly available at their respective research websites. We used the default parameters provided by their authors for all the compared methods. Comparisons of some early methods that oversegment video data without considering the property of similar size can be found in the work of [15].

4.1. Quantitative Comparisons

We conducted experiments on the Chen dataset [31] and adopted five metrics to evaluate the quality of the supervoxels generated by different algorithms. This dataset contains eight video sequences, and every frame has a ground truth label. Each metric is compared as a function of the average number of superpixels per frame.

Three of the five metrics are borrowed from still image segmentation, and they are 2D boundary recall (2D BR), 2D under-segmentation error (2D UE), and 2D achievable segmentation accuracy (2D ASA). For a single frame with ground truth labels, 2D BR measures the proportion of ground truth boundaries that fall within two pixels of the superpixel boundaries [3]. As shown in Figure 5a, our method and vSEEDS [19] worked equally well in terms of boundary recall when a relatively large number of superpixels are generated. However, the superiority of our method becomes obvious with the decrease of the number of superpixels.

2D UE, shown in Figure 5b, is another important measure of boundary adherence. For a single frame, given a region g_j from the ground truth segmentation and the set of superpixels required to cover it, $\{s_k | s_k \cap g_j \neq \phi\}$, where ϕ denotes an empty set, 2D UE measures how many pixels from s_k are not in the region g_j. Given that $|\cdot|$ is the number of elements in a given set, G is the set of ground truth segments, and M is the minimum number of pixels in s_k overlapping g_j, 2D UE can be expressed as

$$\text{2D UE} = \frac{1}{|\mathcal{I}|} \left\{ \sum_{g_j \in G} \left(\sum_{\left\{k \,\middle|\, |s_k \cap g_j| > M\right\}} |s_k| \right) - |\mathcal{I}| \right\}. \tag{34}$$

It is generally accepted to set M to five percent of $|s_k|$ to account for ambiguities in the ground truth segmentations. Superpixels that do not tightly adhere to the ground truth indicate high 2D UE. Clearly, our method had the minimum under-segmentation error, as shown in Figure 5b.

If we assign every superpixel with the label of a ground truth segment that covers the greatest number of pixels of the corresponding superpixel, 2D ASA measures how much segmentation accuracy we can achieve or how many pixels are correctly segmented, as shown in Equation (35):

$$\text{2D ASA} = \frac{1}{|\mathcal{I}|}\sum_{s_k}\max\left\{|s_k \cap g_j| \,\middle|\, g_j \in G\right\}. \tag{35}$$

Superpixels with high segmentation accuracy will result in a high value of 2D ASA. As shown in Figure 5c, our method achieved the best 2D ASA.

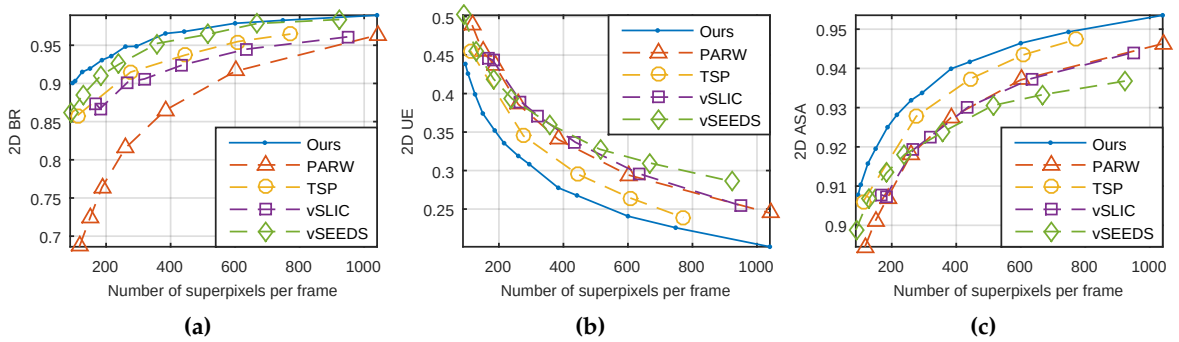

Figure 5. Results of 2D metrics. (**a**) 2D BR; (**b**) 2D UE; (**c**) 2D ASA.

The other two metrics—namely 3D UE and 3D ASA—were used to evaluate temporal consistency by performing the formulas of 2D UE and 2D ASA on every two consecutive frames. Similarly, low 3D UE and high 3D ASA indicate better performance. As shown in Figure 6, our method presented the best temporal consistency.

To compare computational efficiency, we test the selected algorithms on a 4-core Intel CPU at 3.3 GHz. As shown in Figure 7, although our algorithm did not show the best performance in terms of run-time, it is still worth noting that we achieved better results than PARW and TSP and had extremely similar performance to vSEEDS. vSLIC presented the best run-time. However, vSLIC is not a real supervoxel algorithm because it treats video as a fake 3D volume and temporal consistency is not real considered in vSLIC. In addition, Figure 7 experimentally confirms that our method is of linear complexity with respect to frame size.

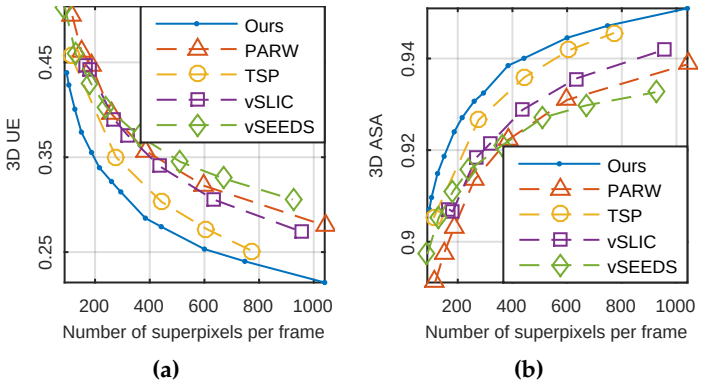

Figure 6. Results of 3D metrics. (**a**) 3D UE; (**b**) 3D ASA.

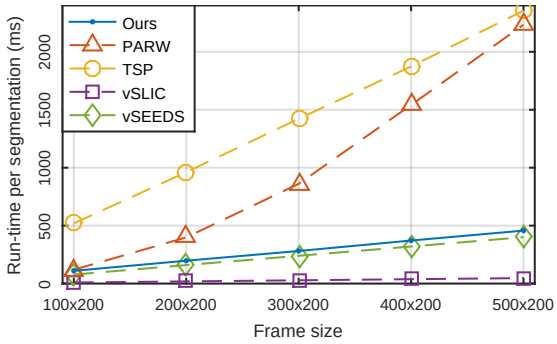

Figure 7. Comparison of run-time.

4.2. Qualitative Comparisons

As displayed in Figure 8, we selected four frames from the Chen dataset to compare the segmentation results of four algorithms. Note that vSLIC is not included because it does not consider temporal consistency for real and cannot segment videos with very few frames [1] (e.g., two frames). Our algorithm correctly assigned a new label for the new appearing object, as shown in the fourth row of Figure 8d. Although TSP also correctly detected the new object, this algorithm is easy to overact (certain moving objects are assigned with new labels in Figure 8b). If an object has similar colors in two different frames, our method is able to track it. For instance, most of the supervoxels of our method preserved temporal consistency, as shown in the third last row of Figure 8d, in which the three people and the soccer ball of moving in different directions are similar in color between the two frames. vSEEDS succeeded in segmenting the moving soccer ball, as shown in Figure 8c. However, supervoxels of vSEEDS tend to be considerably dissimilar in size.

Although supervoxel algorithms are not real visual tracking algorithms, supervoxels may be useful in visual tracking. For example, instead of tracking a rectangle window, a tracking algorithm can track the superpixels to either save computing time or improve robustness. In this case, the temporal consistency that supervoxel algorithms attempt to capture becomes an important property that can boost the performance of such tracking algorithms. To compare the performance in terms of temporal consistency when many frames are involved, we manually select some superpixels that cover the same region in one frame and track them using temporal consistency. As shown in Figures 9–11, our method presented the best results in tracking the three different kinds of objects.

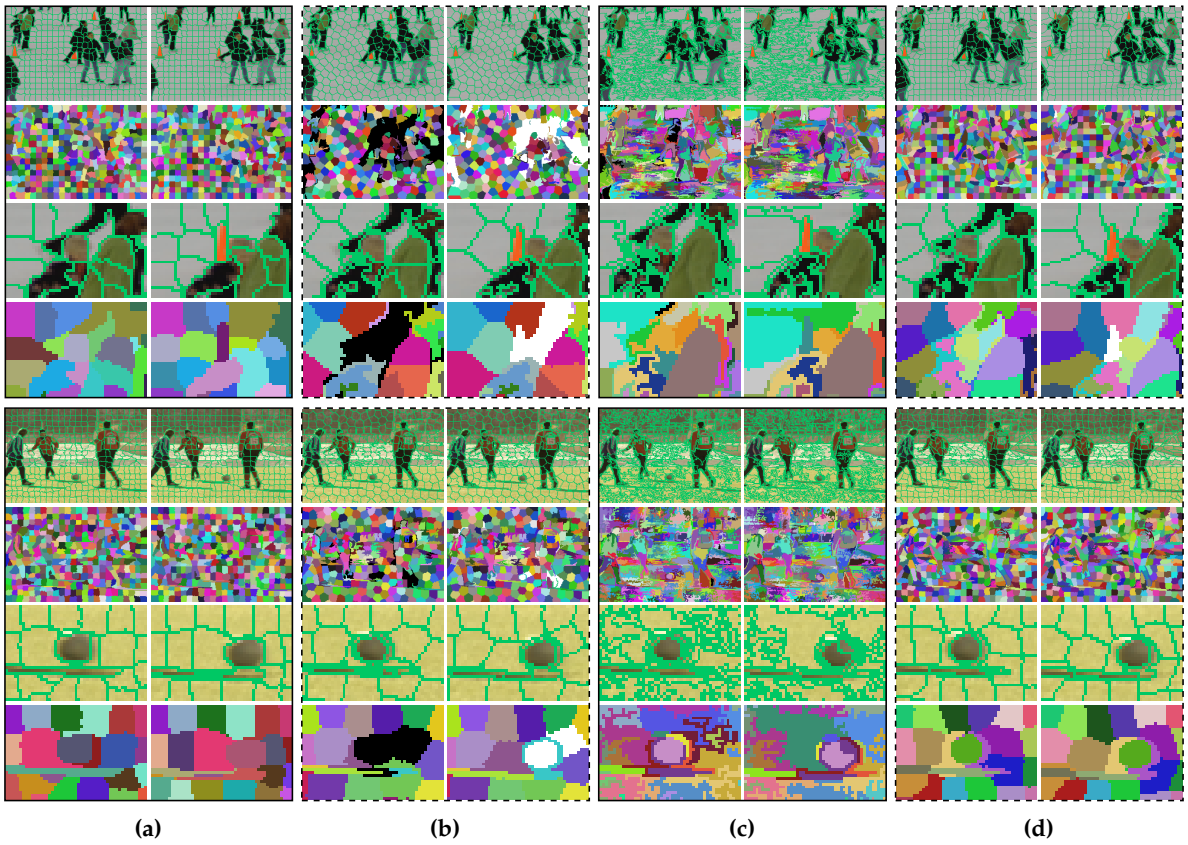

Figure 8. Examples of supervoxel segmentation. (**a**) PARW [17]; (**b**) TSP [20]; (**c**) vSEEDS [19]; (**d**) ours. The algorithms extract approximately the same number of supervoxels. Superpixel boundaries are plotted in the first, third, fifth, and seventh rows. In the remaining rows, we paint voxels of the same supervoxel using the same color. Disappearing and appearing superpixels are painted using black in the first frames and white in the second frames, respectively. The third, fourth, seventh, and eighth rows zoom in on regions of the first, second, fifth, and sixth rows, respectively.

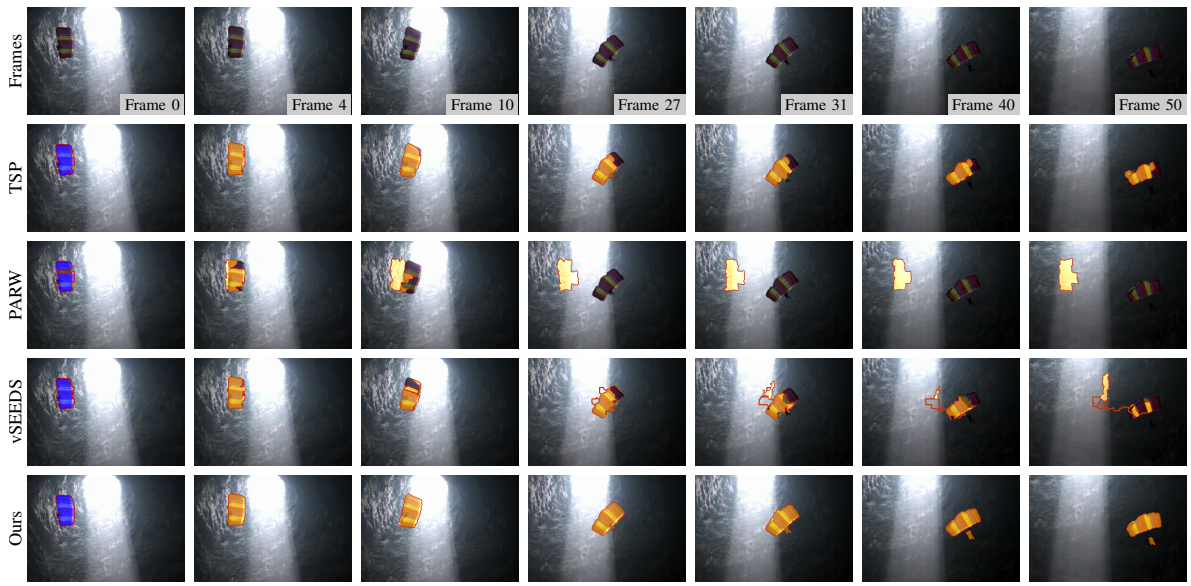

Figure 9. Segmentation results of a moving target that becomes two separated parts during the moving.

Figure 10. Segmentation results of a moving target that is partially covered by another object during the moving.

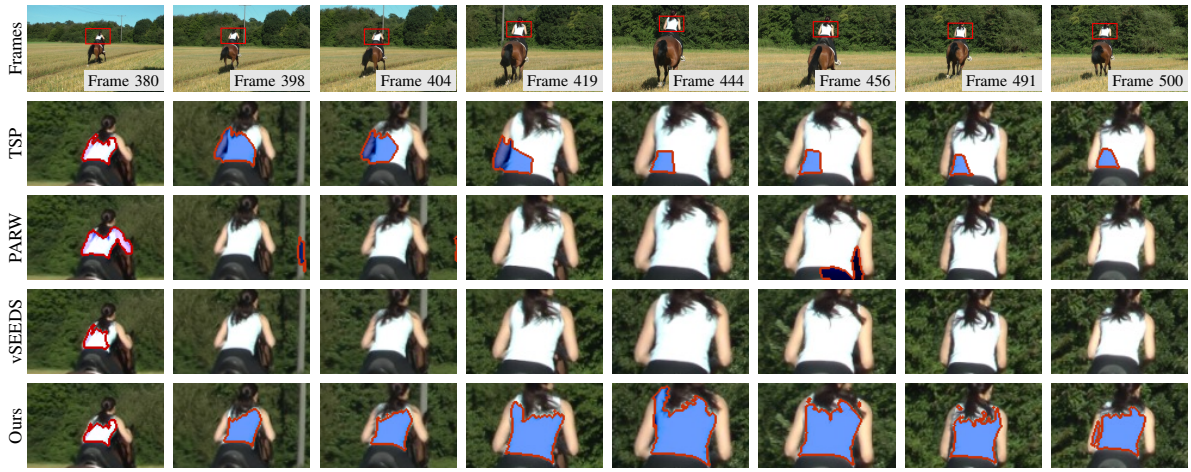

Figure 11. Segmentation results of a moving target of varying size. Only the regions of interest defined by red rectangles in the first row are plotted in the last three rows. This sequence is from [32,33].

5. Conclusions

A temporal superpixel algorithm was developed based on our novel voxel-related Gaussian mixture model (GMM). Instead of producing immutable superpixels for each frame, we proposed a new scheme for supervoxel segmentation. In this scheme, every two adjacent frames are independently segmented into superpixels and so that every internal frame has two valid superpixel segmentations, which provides our algorithm with a coarse-grained parallel ability and allows subsequent applications to adjust their results on each internal frame to further improve robustness.

In the voxel-related GMM, a supervoxel is represented by two Gaussian distributions, each of which models a superpixel in one frame. To guarantee temporal consistency, the two Gaussian distributions of a supervoxel share the same color parameters. Every superpixel is assumed to be distributed in a local region, resulting in an algorithm with lower complexity than the traditional GMM. According to our experiments, the proposed method outperforms the state-of-the-art algorithms in terms of segmentation accuracy while possessing a competitive computing performance.

As a contribution to open source society, our test code will be publicly available at https://github.com/ahban.

Author Contributions: Zhihua Ban conceived, designed the experiments and wrote the paper; Zhong Chen and Jianguo Liu contributed analysis tools.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Achanta, R.; Shaji, A.; Smith, K.; Lucchi, A.; Fua, P.; Süsstrunk, S. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 2274–2282.
2. Chen, J.; Li, Z.; Huang, B. Linear Spectral Clustering Superpixel. *IEEE Trans. Image Process.* **2017**, *26*, 3317–3330.
3. Levinshtein, A.; Stere, A.; Kutulakos, K.N.; Fleet, D.J.; Dickinson, S.J.; Siddiqi, K. TurboPixels: Fast Superpixels Using Geometric Flows. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *31*, 2290–2297.
4. Ban, Z.; Liu, J.; Fouriaux, J. GLSC: LSC superpixels at over 130 FPS. *J. Real-Time Image Process.* **2016**, 1–12, doi:10.1007/s11554-016-0652-5.
5. Xie, Y.; Chen, K.; Lin, J. An Automatic Localization Algorithm for Ultrasound Breast Tumors Based on Human Visual Mechanism. *Sensors* **2017**, *17*, 1101.
6. Zhang, Q.; Liu, Y.; Zhu, S.; Han, J. Salient object detection based on super-pixel clustering and unified low-rank representation. *Comput. Vis. Image Underst.* **2017**, *161*, 51–64.
7. Zhang, Q.; Lin, J.; Tao, Y.; Li, W.; Shi, Y. Salient object detection via color and texture cues. *Neurocomputing* **2017**, *243*, 35–48.
8. Van De Sande, K.E.A.; Uijlings, J.R.R.; Gevers, T.; Smeulders, A.W.M. Segmentation as selective search for object recognition. In Proceedings of the 2011 International Conference on Computer Vision (ICCV), Barcelona, Spain, 6–13 November 2011; pp. 1879–1886.
9. Zhong, Z.; Lei, M.; Cao, D.; Fan, J.; Li, S. Class-specific object proposals re-ranking for object detection in automatic driving. *Neurocomputing* **2017**, *242*, 187–194.
10. Liu, J.; Tang, Z.; Cui, Y.; Wu, G. Local Competition-Based Superpixel Segmentation Algorithm in Remote Sensing. *Sensors* **2017**, *17*, 1364.
11. Qiu, W.; Gao, X.; Han, B. A superpixel-based CRF saliency detection approach. *Neurocomputing* **2017**, *244*, 19–32.
12. Fu, P.; Li, C.; Cai, W.; Sun, Q. A spatially cohesive superpixel model for image noise level estimation. *Neurocomputing* **2017**, *266*, 420–432.
13. Reso, M.; Jachalsky, J.; Rosenhahn, B.; Ostermann, J. Temporally Consistent Superpixels. In Proceedings of the 2013 IEEE International Conference on Computer Vision (ICCV), Sydney, Australia, 1–8 December 2013; pp. 385–392.
14. Liang, Y.; Dong, X.; Shen, J. Supervoxel using random walks. In Proceedings of the 2014 7th International Congress on Image and Signal Processing, Dalian, China, 14–16 October 2014; pp. 120–124.
15. Xu, C.; Corso, J.J. Evaluation of super-voxel methods for early video processing. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, 16–21 June 2012; pp. 1202–1209.
16. Veksler, O.; Boykov, Y.; Mehrani, P. Superpixels and Supervoxels in an Energy Optimization Framework. In Proceedings of the 11th European Conference on Computer Vision (ECCV), Crete, Greece, 5–11 September 2010; pp. 211–224.
17. Liang, Y.; Shen, J.; Dong, X.; Sun, H.; Li, X. Video Supervoxels Using Partially Absorbing Random Walks. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *26*, 928–938.
18. Levinshtein, A.; Sminchisescu, C.; Dickinson, S. Spatiotemporal Closure. In Proceedings of the 10th Asian Conference on Computer Vision (ACCV), Queenstown, New Zealand, 8–12 November 2011; pp. 369–382.
19. Bergh, M.V.D.; Roig, G.; Boix, X.; Manen, S.; Gool, L.V. Online Video SEEDS for Temporal Window Objectness. In Proceedings of the 2013 IEEE International Conference on Computer Vision (ICCV), Sydney, Australia, 1–8 December 2013; pp. 377–384. .
20. Chang, J.; Wei, D.; Fisher, J.W., III. A Video Representation Using Temporal Superpixels. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Portland, OR, USA, 23–28 June 2013; pp. 2051–2058.
21. Horn, B.K.; Schunck, B.G. Determining optical flow. *Artif. Intell.* **1981**, *17*, 185–203.

22. Reynolds, D. Gaussian mixture models. In *Encyclopedia of Biometrics*; Li, S.Z., Jain, A.K., Eds.; Springer: Boston, MA, USA, 2015; pp. 827–832.
23. Ma, J.; Zhao, J.; Ma, Y.; Tian, J. Non-rigid visible and infrared face registration via regularized Gaussian fields criterion. *Pattern Recognit.* **2015**, *48*, 772–784.
24. Ban, Z.; Liu, J.; Cao, L. Superpixel Segmentation Using Gaussian Mixture Model. *arXiv* **2016**, arXiv:1612.08792.
25. Ren, X.; Malik, J. Learning a classification model for segmentation. In Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV), Nice, France, 13–16 October 2003; pp. 10–17.
26. Grundmann, M.; Kwatra, V.; Han, M.; Essa, I. Efficient hierarchical graph-based video segmentation. In Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, CA, USA, 13–18 June 2010; pp. 2141–2148.
27. Felzenszwalb, P.F.; Huttenlocher, D.P. Efficient Graph-Based Image Segmentation. *Int. J. Comput. Vis.* **2004**, *59*, 167–181.
28. Xu, C.; Xiong, C.; Corso, J.J. Streaming Hierarchical Video Segmentation. In Proceedings of the 12th European Conference on Computer Vision (ECCV), Florence, Italy, 7–13 October 2012; pp. 626–639.
29. Van den Bergh, M.; Boix, X.; Roig, G.; de Capitani, B.; Van Gool, L. SEEDS: Superpixels Extracted via Energy-Driven Sampling. In Proceedings of the 12th European Conference on Computer Vision (ECCV), Florence, Italy, 7–13 October 2012; pp. 13–26.
30. Van den Bergh, M.; Boix, X.; Roig, G.; Van Gool, L. SEEDS: Superpixels Extracted via Energy-Driven Sampling. *Int. J. Comput. Vis.* **2015**, *111*, 298–314.
31. Chen, A.Y.C.; Corso, J.J. Propagating multi-class pixel labels throughout video frames. In Proceedings of the 2010 Western New York Image Processing Workshop, Rochester, NY, USA, 5 November 2010; pp. 14–17.
32. Ochs, P.; Malik, J.; Brox, T. Segmentation of Moving Objects by Long Term Video Analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *36*, 1187–1200.
33. Brox, T.; Malik, J. Object Segmentation by Long Term Analysis of Point Trajectories. In Proceedings of the 11th European Conference on Computer Vision (ECCV), Crete, Greece, 5–11 September 2010; pp. 282–295.