1. **Literature Review**

The first paper about tensor voting was published in 1996 by Guy and Medioni. After the initial publication, Medioni and Lee together published a few more papers to extend this technique. Tensor voting relates to Gestalt’s Principle of Proximity. The key idea is that humans group objects based on proximity, similarity, continuity, closure, fate, and form.

Tensor voting utilizes this idea and applies segmentation without requiring prior knowledge. It collects information around its neighbors, then decides if the target pixel is foreground or background. Tensor voting is often used after edge detection as a refinement procedure, such as noise remove or edge strengthen. My procedure is inspired by two papers. One is by Maggiori, Manterola, and Fresno. The authors explained the tensor voting tehcnique indetails. The other one is by Loss, Bebis, and Parvin. These authors explained the technique sightly different and proposed new voting mechanism. *Loss el at.*, tested on synthetic images, instead of real lab images. Their result came out nicely, but it could not justify the usefulness of the technique since the synthetic images too simple and clean.

The original Tensor Voting method was published by Medioni and Guy in 1997. Tensor Voting can be applied to both 2 dimensional and 3 dimensional data. In this paper, we only consider its 2D implementation to tackle the image segmentation problem. Tensor Voting becomes a very popular technique for edge detection because of its robustness towards to noise. Tensor Voting is usually implemented after a line detection due to two reasons. First, Tensor Voting is computational expensive. We can use an edge detector to identify potential edges, then let Tensor Voting to identify the true edges. However, this may reduce the robustness because the input for Tensor Voting may contain a lot of noise. Second, Tensor Voting can only work on the objects that are salient, which means white foreground and dark background. So, the outcome from an edge detector perfectly fits to this requirement. The following paragraph is the brief review of Tensor Voting procedure.



However, Loss proposed to apply Tensor Voting directly on images without applying an edge detector first. However, their procedures are not very convincing for two reasons. First, Loss applied simple threshold method to pre-label some pixels to be background. The formula is I xy / I white. If the value is above an arbitrary value, they label it as foreground; otherwise, it is considered as background. Second, Loss used synthetic images, instead of real lab images. The problem is that their sythetic images look the same as the images that have been applied with an edge detector. Therefore, their claim is not valid. However, I am inspired by their procedure because I see the potential of using Tensor Voting as an object segmentation technique.

Many works were built on this technique, and many variants were proposed to speed up Tensor Voting’s computation. My proposed method have three main difference to the original work:

1. I propose no decay function because decay function does not contribute
2. Replace SV + BV with a special designed kernel and projection function. All neighbour around the targeted pixel are included.
3. Only target pixel only take the neighbour pixels who are close to itself. This helps to converge and eliminate the side effect of blurring.

There are three main contributions to the field. First, we fully utilize matrix manipulation tricks to speed up the algorithm. Second, our design simplified the algorithm without compromising the accuracy. Third, we are the first attempt to apply Tensor Voting to a segmentation problem.

1. **Proposed Method:**

This section has 4 components. The first two components (3.1 and 3.2) are providing insights what terms in the original tensor voting algorithm can be modified. The last two components (3.3 and 3.4) are addressing the issues that the algorithm encounters and how the issues can be solved.

* 1. **Replace Decay Function**

No authors have mentioned in their papers how far they allowed Tensor Voting to collect information from its neighbors. The decay function behaves as weights, which controls how much information the voters can contribute. The further the voters are, the less value they can send to the targeted tensors. So, the question is how the decay function is controlling the contributions and if it is optimal to the algorithm. σ is the only variable that requires an arbitrary decision. I use a 7x7 matrix with two different σ values -- 1 and 18, to illustrate what the decay function looks like. The reason I choose 18 is that this is a suggested value by Medioni.

For σ = 1, we can clearly see that this decay function is useless. Only the two horizontal voters contribute a little bit to the targeted tensor.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **2.2688e-10** |  |  |  |  |  | **2.2688e-10** |
| **4.5280e-07** | **5.1723e-05** |  |  |  | **5.1723e-05** | **4.5280e-07** |
| **3.1918e-05** | **0.0046** | **0.0848** |  | **0.0848** | **0.0046** | **3.1918e-05** |
| **1.2341e-04** | **0.0183** | **0.3679** | **Voter** | **0.3679** | **0.0183** | **1.2341e-04** |
| **3.1918e-05** | **0.0046** | **0.0848** |  | **0.0848** | **0.0046** | **3.1918e-05** |
| **4.5280e-07** | **5.1723e-05** |  |  |  | **5.1723e-05** | **4.5280e-07** |
| **2.2688e-10** |  |  |  |  |  | **2.2688e-10** |

For σ = 18, the values are boosted dramatically. We can see that there are very minor differences among the values. Therefore, the weights control becomes insignificant since the values are so closed to each other.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **0.9137** |  |  |  |  |  | **0.9137** |
| **0.9384** | **0.9236** |  |  |  | **0.9236** | **0.9384** |
| **0.9610** | **0.9532** | **0.8159** |  | **0.8159** | **0.9532** | **0.9610** |
| **0.9726** | **0.9877** | **0.9969** | **Voter** | **0.9969** | **0.9877** | **0.9726** |
| **0.9610** | **0.9532** | **0.8159** |  | **0.8159** | **0.9532** | **0.9610** |
| **0.9384** | **0.9236** |  |  |  | **0.9236** | **0.9384** |
| **0.9137** |  |  |  |  |  | **0.9137** |

I also tried with σ = 30. The values in the cells are not changing much. For example, the ones at the furthest corners are near to 0.96. No values exceed 1 in the matrix due to the decay function design. Based on the illustration, we can tell that decay function is not a very significant term to the algorithm. Therefore, I propose to either remove the decay function or use a Gaussian filter as a weight control term.

**3.2 Special Kernel and Projection Matrix**

A new tensor is the sum of SV and BV. As we introduced in the Section 2, BV is the integration of SV over θ. However, this integration cannot be applied directly on SV because both of the complex structure of the decay function. One simple solution is to drop the decay function in the integration since I have proved that the decay function is not playing an important role for Tensor Voting. As a result, I only apply integration to the project matrix.

**Projection Matrix**

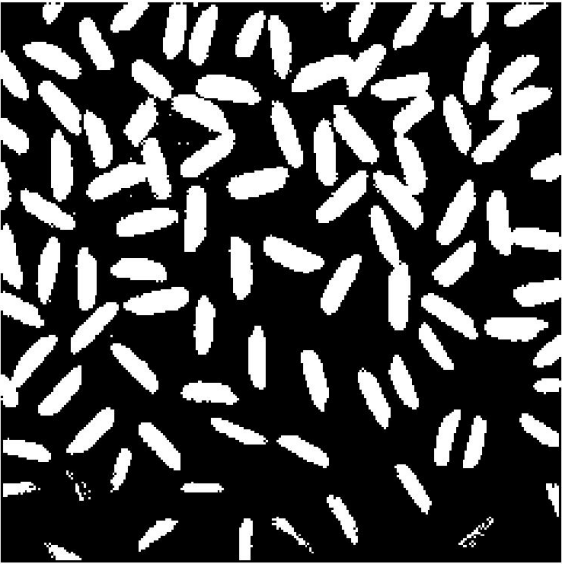
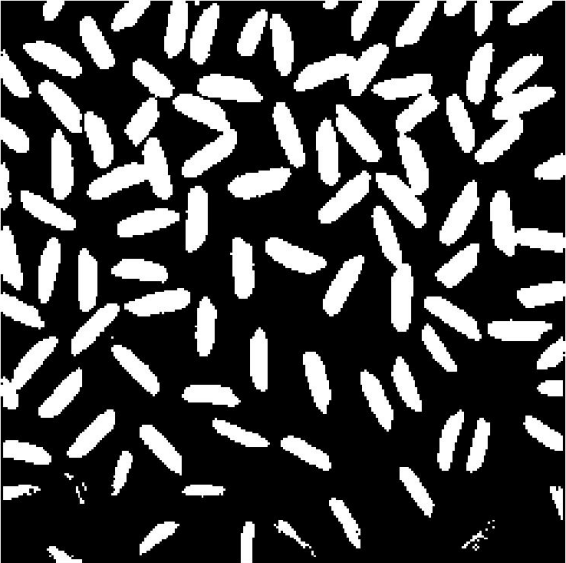
|  |  |
| --- | --- |
| sin(2θ)^2 | - sin(2θ) \* cos(2θ) |
| - sin(2θ) \* cos(2θ) | cos(2θ)^2 |

|  |  |
| --- | --- |
| θ/2 - sin(4θ)/8 + sin(2θ)^2 | (cos(2θ)^2) / 4 - sin(2θ) \* cos(2θ) |
| (cos(2θ)^2) / 4 - sin(2θ) \* cos(2θ) | θ/2 + sin(4θ)/8 + cos(2θ)^2 |

**Integrate Over θ + SV Projection Matrix**

To be clear, the θ refers to radian, not degree. The following images are generated with different projection matrix -- 1) SV only, and 2) SV + BV. The result shows that SV + BV method is more robust than SV-only projection, but it suffers lower recall under the same threshold point. I will not implement other proposed methods that could approximate the BV component. The main idea in this experiment is to convey that BV may not introduce new information because it only utilizes the known information from SV. The vertical neighbors are important too, and they should not be approximated or inferred.

SV Only SV + BV

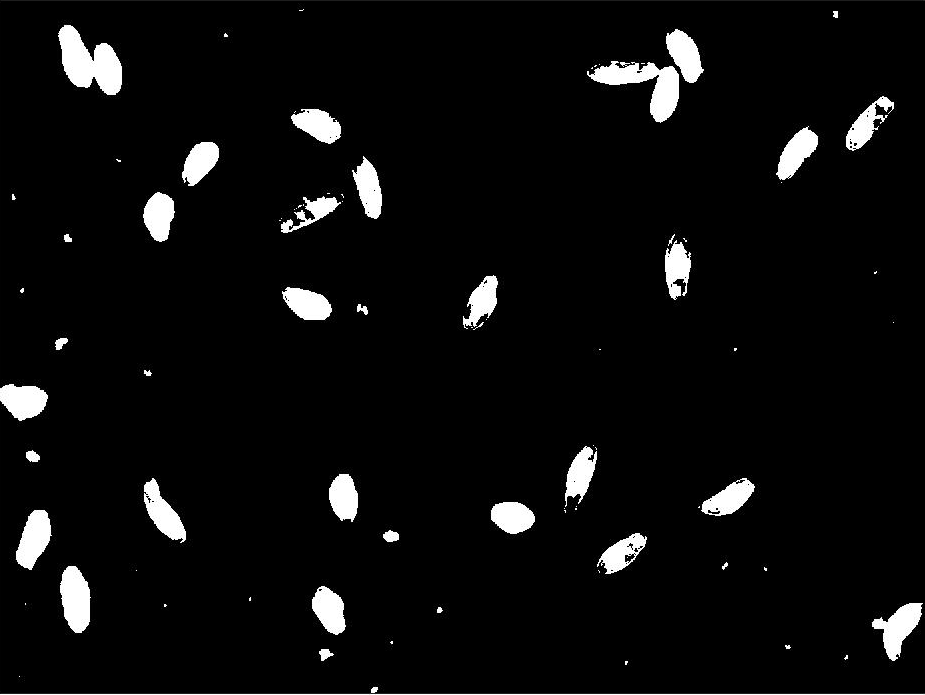
 

The kernel should take in information from all neighbor tensors. This may raise a concern to many readers because Medioni claimed that the effective range is only between

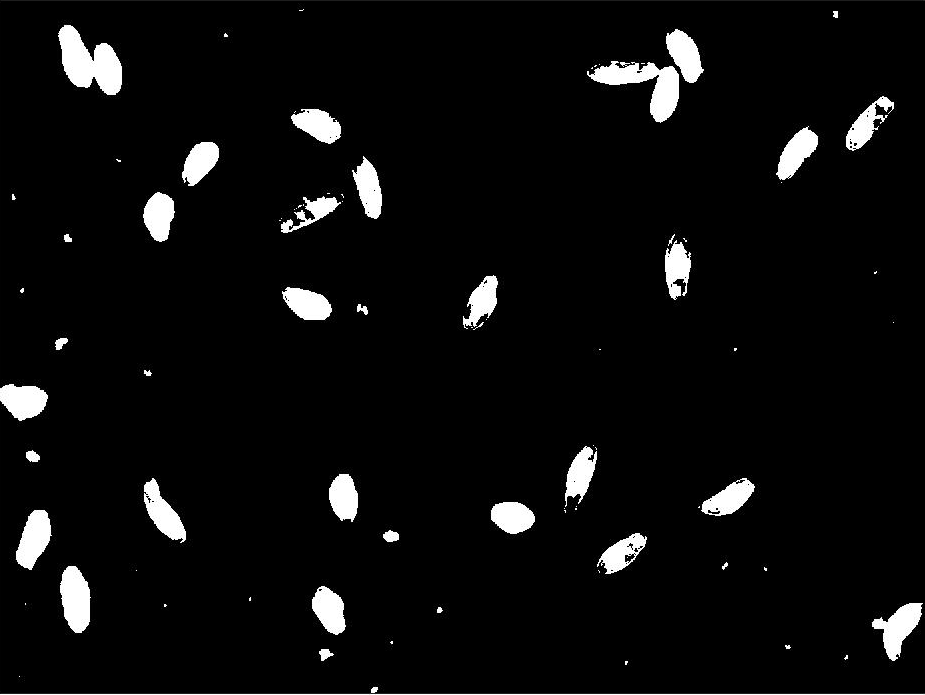
-π/4 and π/4. However, the I would suggest to a different the projection matrix is just a way to encode neighbors’ information.

I used a real lab image to compare the performance between my proposed kernel and the original version. The first row is

SV + BV Sin + Cos

SV + BV Sin + Cos

3.3Variance 3 -- Selective Neighbour

Tensor Voting can be applied iteratively. In each iteration, the output values that are lower than an arbitrary threshold value will be labeled as background. Although it seems that Tensor Voting is performing removing noise, it is actually blurring the pixel values. To illustrate the idea, I iterate 10 times over the rice image without removing any pixel value. I normalized the output between 0 and 255 in order to generate an image, instead of applying a thresholding value to generate a salience map. The second image is the outcome after 10 iterations. The third image is when I apply a simple thresholding value. More iterations only leads to a blurry outcome. Therefore, I propose a selective voting kernel.

The idea is to align the targeted pixel to agree with its majority neighbors, instead of taking nearly same amount of information from all its neighbor, which causes blurring. For example, if I decide to apply a 3x3 filter, there are