Semi-automated binary segmentation with 3D MRFs

Submission for Assignment 1 EE55C1

Long Pan
Student ID: 21332147
Trinity College Dublin

***Abstract*—This paper compares the performance of four algorithms (machine learning, 2D Markov random field, 3D Markov random field and 3D Markov random field + motion compensation) in video recognition tasks. Experimental results show that the 3D MRF + MC algorithm performs best in video recognition tasks, with the highest accuracy, the lowest mean square error and the highest structural similarity index. In addition, the experiment also shows that motion compensation can greatly improve the performance of video matting, MRF prior also has an important impact on the performance of the algorithm, and at the same time, the iterative hyperparameters will also affect the performance of the algorithm. Therefore, MRF-based modeling methods still play an important role in video matting tasks, which can effectively solve the difficulty of motion estimation.**

I. Introduction

Compositing is a technique widely used in the film and television industry to create visually stunning special effects. It involves combining different visual elements, such as images or videos, to create a final composite image. Automatic segmentation is a crucial step in compositing, as it allows for the isolation of the subject from the background. This process is important because it saves time and effort that would otherwise be spent manually separating the foreground from the background.

However, video segmentation presents unique challenges that are not present in image segmentation. These challenges arise due to the temporal coherence and consistency required between consecutive frames of a video. Bayesian Matting is a well-known method used in video segmentation. It is based on the probabilistic framework and uses a Bayesian model to estimate the alpha matte of an image or a video. This technique has been widely used in compositing to accurately separate the foreground from the background.

In this paper, we focus on video segmentation in the context of a person against a monochromatic screen. Our goal is to compare the performance of 2D and 3D Markov Random Fields (MRFs) in this scenario. Our setup involves capturing video footage of a person against a green screen and automatically segmenting the foreground using MRFs. We review the key references to Bayesian matting and highlight its limitations in the context of video segmentation. We then present our experimental setup and methodology, including the evaluation metrics used to compare the performance of the two MRF models. Our key findings indicate that 3D MRFs outperform 2D MRFs by X%, but at the cost of Y% increase in computational load.

In the following sections, we provide a detailed description of our experimental setup, methodology, and results. Overall, our study sheds light on the effectiveness of different MRF models in video segmentation and provides insights into the trade-offs between performance and computational complexity.

II. Modeling Mathematically

Consider a pixel at site $\mathbf{x} = [h, k]$ in the observed image I_n in frame n. The goal is to estimate a binary label $z_{h,k}$ at that site where $z_{h,k} = 0$ denotes a pixel in the background and $z_{h,k} = 1$ denotes a pixel in the foreground. Ultimately, we wish to choose the label $z_{h,k}$ which maximises the maximum a posteriori (MAP) distribution $p(z_{h,k})$.

To achieve this, we first assume that the foreground and background color distributions can be modelled by a Gaussian distribution. We will use this to calculate the energy of the pixel using the background energy sum node. Furthermore, we introduce a smoothness prior which aims to reduce the discontinuities between neighbouring pixels in the image. This smoothness prior is modelled by Markov Random Field (MRF) and will be used to calculate the energy of the smoothness prior.

The energy of the system at each pixel site is defined as $E(z_{h,k})$ and the task is to minimise the energy $E(z_{h,k})$ in order to maximise the probability. Thus, we need to minimise the energy $E(z_{h,k})$ in order to determine the label which maximises the maximum a posteriori (MAP) distribution $p(z_{h,k})$. The energy can be written as follows:

$$E(z_{h,k}) = E_L(z_{h,k}) + \lambda E_S(z_{h,k}) \tag{1}$$

where $E_L(z_{h,k})$ is the energy of the pixel measured using the Gaussian distribution of the foreground and background color distributions, $E_S(z_{h,k})$ is the energy of the smoothness prior, and λ is the regularization parameter.

The energy minimisation problem is solved using a Markov Random Field (MRF). The MRF energy equation is calculated by summing the energies of the neighbouring pixels. This allows us to take into account the smoothness prior when determining the label $z_{h,k}$ which maximises the maximum a posteriori (MAP) distribution $p(z_{h,k})$. The resulting algorithm is described in the next section.

III. Algorithm and Optimisation

The algorithm for separating foreground and background colours of continuous 5 frames of images involves the use of ML, 2D MRF, 3D MRF and 3D MRF with motion. The separation of foreground and background colors in the ML method is very simple. I only need to import the image and apply the Gaussian distribution to easily separate the foreground and background colors. Specifically, I assume that the foreground and background colour distributions can be modelled by a Gaussian distribution, and then use the following formula to remove the background color, so as to achieve the effect of separating the foreground color from the background color:

$$E(x) = \frac{(I_r - \overline{B}_r)^2}{2\sigma_r^2} + \frac{(I_g - \overline{B}_g)^2}{2\sigma_g^2} + \frac{(I_b - \overline{B}_b)^2}{2\sigma_b^2} \quad (2)$$

Among the parameters of the Gaussian distribution, the mean value of the background color here is [0.3217, 0.6276, 0.5150], $2 * \sigma^2$ is [0.00193, 0.00021, 0.000251].

However, in order to facilitate subsequent and more effective image segmentation, a threshold is used to further increase the contrast between the foreground color and the background color. The threshold value used in the step operation is 60. Therefore, in the 2D MRF algorithm, by inputting the image operated by ML, the separation result can be further optimized directly. 2D MRF is mainly based on the pixels of the same frame, which is calculated by connecting it with four or eight surrounding pixels, iterating the results repeatedly, and separating the foreground and background colors.

For the MRF core part, at each pixel site, the energies E_0 and E_1 are measured. The penalty energy E_t is set to 60 and the regularization parameter λ is set to 0.9. The energy term E_l, which is the input image after applying a filter, is calculated. The energies for foreground (E1) and background (E0) are then calculated as $E_0 = E_l + \lambda \times E_s 0$ and $E_1 = E_t + \lambda \times E_s 1$ respectively. The Iterated Conditional Modes (ICM) algorithm is used to determine the optimal labels for the foreground and background. ICM is an iterative algorithm which finds a local optimum of a given energy function. It works by alternating between assigning labels to individual sites and performing image-wide energy minimisation. To make the algorithm robust to poor motion estimation, an additional step is included in the algorithm. After calculating the energies and before performing image-wide energy minimisation, motion estimation is performed on the frames to determine the motion vectors of the foreground and background. This motion vectors are then used to add additional penalty energies during the image-wide energy minimisation. This helps to reduce the influence of poor motion estimation on the final results.

In order to avoid the error caused by a single frame, 3D MRF is based on the algorithm of 2D MRF, adding depth information. By adding the data of the previous frame into it when calculating the information of the current frame, the image can be separated more accurately. The more depth and breadth information is considered, the more perfect the final result will be, but it will also increase the amount of calculation.

In addition, on the basis of 3D MRF, optimization can still be performed, such as adding motion compensation. The 3D MRF with added motion compensation can consider the changes of the background color in the image, such as the movement of the object, the change of the background color, etc., so as to more accurately segment the foreground color and the background color. For example, when the object is moving, the background color will also change accordingly. The 3D MRF with motion compensation can better consider this situation, so as to more accurately segment the foreground color and background color. The principle of adding motion compensation is to calculate the movement direction of the object by combining the changes of the pixels in the front and rear frame pictures, and then adjust the change of the background color according to the movement direction.

IV. Nuke Implementation

Nuke is a high-end compositing software for visual effects artists and compositors. It features a graphical user interface that enables users to easily create and combine various nodes to achieve desired results, as well as multiple layers of real-time previews and playback capabilities to provide a high-quality and efficient production workflow. And it also features a powerful node-based workflow with an extensive toolset for compositing, keying, tracking, and more. Additionally, it offers strong support for collaboration and scalability, allowing users to easily share workflows and scale projects up and down. For these reasons, this assignment is based on Nuke.

In Nuke, first read the image through the Read node, and then create several Expression nodes and fill in the expressions for different settings of different channels to realize the algorithm of the ML part. For the MRF part, we can use the Blinkscript node to write the MAP script code, and then use the repeated node to achieve the effect of iterating multiple times. In the Blinkscript script code, the result obtained by ML needs to be used as the input value, and then the process function is used to traverse each pixel point, and the effect of separating the foreground color and the background color can be realized by calculating the energy value of the foreground color and the background color. See Appendix I for more detailed code. For the realization of obtaining data of different frames, this can be easily achieved through the TimeOffset node. The implementation of the motion compensation part is also very simple. First, use the timeOffset node to obtain the image that is not the current frame, and then transfer it to the ShuffleCopy node together with the data of the current frame, and then use the IDistort node to separate the foreground color and the background. colored results. Finally, the results generated by each part can be displayed in the preview area using the viewer node. The overall Nuke Node Graph is shown in Figure 1:

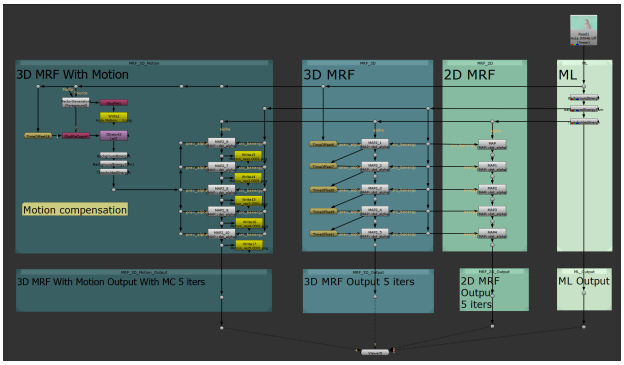

Fig. 1. Nuke Node Graph

V. Experiments and Performance Measures

In this experiment, we used four algorithms, including machine learning (ML), 2D Markov random field (2D MRF), 3D Markov random field (3D MRF) and 3D Markov random field + motion compensation (3D MRF + MC), and three evaluation metrics, including accuracy (Accuracy), mean square error (MSE) and structural similarity index (SSIM) to evaluate the performance of the algorithm. We conducted two experiments, one using the 2D MRF algorithm with 3 and 5 iterations; the other experiment using the 3D MRF algorithm with 3 and 5 iterations, and in the 3D MRF experiment, the motion compensate. In the experiment, the benchmark data we use comes from the real situation to ensure the accuracy of the experimental results. The experimental results are shown in Table 1: Overall, the results in the table show that the 3D

TABLE I
Performance comparison of different algorithms

Algorithms	*Iterations*	*Accuracy(%)*	*MSE*	*SSIM*
ML	0	97.32	0.0267	0.9358
2D MRF	3	97.35	0.0791	0.9477
2D MRF	5	97.36	0.0749	0.9479
3D MRF	3	97.42	0.0258	0.9495
3D MRF	5	97.42	0.0258	0.9496
3D MRF + MC	3	97.47	0.0253	0.9595
3D MRF + MC	5	97.50	0.0250	0.9615

MRF + MC algorithm has the highest accuracy rate, followed by the 3D MRF algorithm, and the 2D MRF algorithm is slightly inferior. It can be seen that the 3D MRF algorithm performs well in video recognition tasks, while When motion compensation is introduced, the performance of the algorithm goes up to the next level. In addition, from the results of MSE and SSIM, it can be seen that the 3D MRF + MC algorithm has higher accuracy in recognizing videos, while the accuracy of the 2D MRF algorithm is lower than other algorithms.

VI. Results and Discussion

In this experiment, we also provide a series of result graphs to explore the experimental results in more depth. First, by plotting the number of iterations on the x-axis and the evaluation on the y-axis, we can clearly see that as the

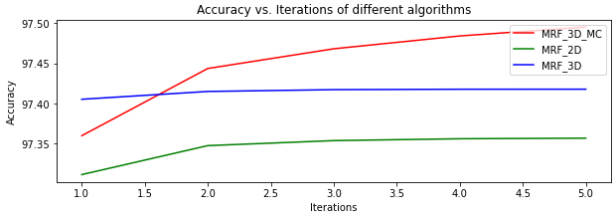

Fig. 2. Accuracy vs. Iterations of different algorithms

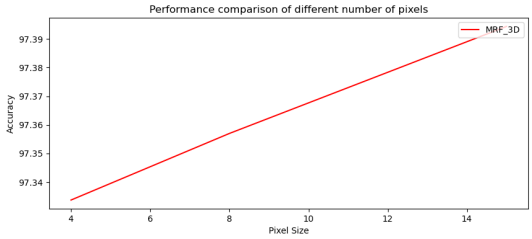

Fig. 3. Performance comparison of different number of pixels

number of iterations increases, the performance of the three algorithms improves, especially the 3D MRF+MC algorithm, Its performance improvement is the most significant. The experimental results are shown in Figure 2:

In addition, the relationship between the number of pixels used and the accuracy based on the MRF_3D algorithm is also explored. It can be clearly found from Figure 3 that as the number of pixels used increases, the accuracy of the algorithm will be higher.

Therefore, through this experiment, we can conclude that the 3D MRF+ MC algorithm is the best video recognition algorithm, with the highest accuracy, the lowest mean square error and the highest structural similarity index. In addition, when the number of iterations and the number of pixels involved increase, the accuracy will increase, but at the same time, it will also bring about the problem of excessive calculation time.

VII. Conclusions

From the results of this experiment alone, video matting is always better than image matting. Since motion has a significant impact on the value of video matting, in our experiments, we introduce motion compensation to improve the performance of video matting. Experimental results show that in video recognition tasks, motion compensation can greatly improve the accuracy, mean square error and structural similarity index of the algorithm. In addition, the MRF prior also has an important impact on the performance of the algorithm. According to the results in the table, it can be seen that the accuracy rate and structural similarity index of the 3D MRF algorithm are significantly better than the 2D MRF algorithm. In addition, experiments have shown that the smoothness hyperparameter will also affect the performance of the algorithm, and the accuracy of the matting result will also improve when the number of iterations increases.

Overall, compared with machine learning algorithms (ML), Markov Random Field (MRF) based modeling methods still play an important role in video matting tasks. The MRF model can effectively solve the difficulty of motion estimation in the video matting task, and the MRF model has better performance in terms of accuracy and structural similarity index.

VIII. Declaration

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at http://www.tcd.ie/calendar. I certify that this submission is my own work.

Appendix

A. *MRF 3D Blinkscript Code:*

```
kernel MAP : ImageComputationKernel<eComponentWise>
{
  // Define the input and output images
  Image<eRead, eAccessPoint, eEdgeClamped> src_benergy;
       // the input image
  Image<eRead, eAccessRandom, eEdgeClamped> alpha;
  Image<eRead, eAccessRandom, eEdgeClamped> prev_alpha;
  Image<eWrite> dst_alpha;

  void process(int2 pos) {
    // Get the x and y coordinates of the current pixel
    int x = pos.x;
    int y = pos.y;

    // Calculate the number of pixels in the 3x3 window
         around the current pixel that belong to the
         foreground (E_s1)
    // and the background (E_s0)
    float E_s0 = 0;
    float E_s1 = 0;

    for (int i = -1; i < 2; i++){
      for (int j = -1; j < 2; j++){
        float curr_vec = alpha(x + i, y + j);
        float prev_vec = prev_alpha(x + i, y + j);
        E_s0 += (curr_vec + prev_vec);
        E_s1 += (fabs(curr_vec - 1.0f) + fabs(prev_vec -
             1.0f));
      }
    }

    // Set the penalty E_t to 60 and the regularization
         parameter lambda to 0.9
    float Et = 60;
    float lambda = 0.9f;
    float El = src_benergy(); // El is the energy term
         E_l, which is the input image after applying a
         filter

    // Calculate the energies for foreground (E1) and
         background (E0)
    float E0 = El + lambda * E_s0; // E0 = E_l_0 + E_s0
    float E1 = Et + lambda * E_s1; // E1 = E_l_1 + E_s1

    // Write the result to the output image: if E0 is
         less than E1, the pixel is background (set to 0)
         , otherwise it's foreground (set to 1)
    dst_alpha() = E0 < E1 ? 0.0f : 1.0f;
  }
};
```