BLOCK BASED VIDEO ALIGNMENT WITH LINEAR TIME AND SPACE COMPLEXITY

Armin Kappeler[⋆] *Michael Iliadis*[⋆] *Haohong Wang*[†] *Aggelos K. Katsaggelos*[⋆]

[⋆] Northwestern University, Departement of EECS, Evanston, IL 60208, USA
[†]TCL Research America, San Jose, CA, 95134, USA

ABSTRACT

Video retrieval and video copy detection are well studied problems. The goal is to find the matching video in a database from a given query video. Typically, these query videos are short and aligning the query video is of secondary importance. Short sequences can be aligned using dynamic time warping. But, since time and memory usage increases quadratically with the length of the sequences, such process is not suitable for the alignment of two full length movies. A typical feature film is between 70 and 210 minutes long. Our goal is to find an accurate frame-by-frame alignment of a full length original film and a copy that has inserted and deleted sequences (e.g., commercial breaks or censorship), as well as differences in quality, format and framerate. We propose a fast, robust and memory efficient video sequence alignment algorithm which has linear space and time complexity.

Index Terms— Video Alignment, Sequence Alignment, Keyframe Extraction, Dynamic Time Warping, A*

1. INTRODUCTION

Dynamic Time Warping (DTW) [1] is used to find the best possible alignment of two similar sequences. It has been widely used in many tasks e.g., DNA sequence alignment [2] or disparity estimation [3]. DTW can be depicted as a path finding problem (see Figure 2), where we try to find the optimal path from the lower left corner to the upper right corner in a matrix. The time and space complexity of DTW is given as $O(NM)$, where N and M are the lengths of the sequences or the size of the matrix. The Sakoe-Chiba band [4] and the Itakura Parallelogram band [5] address this issue by limiting the search space, but they are based on assumptions that do not necessary hold for video alignment. An extremely fast version of DTW is introduced in [6]; however this method uses a sliding window approach which is not suitable for large insertions and deletions. Another algorithm is introduced in [7], which uses a hierarchical approach to approximate DTW. A tradeoff is a possible loss of accuracy. Our proposed video sequence alignment algorithm (VSA) is based on the A* path finding algorithm. A* reduces the alignment time without any loss of accuracy [8]. We extend the A* algorithm by dividing the matrix (Figure 2) into Blocks and adding an abstraction level on top of the matrix. The idea is similar to the idea of [7], but without any loss of accuracy. Our A* extension (Block A*) further reduces runtime and memory usage compared to the standard A* to a high extend.

In order to increase accuracy, we add an additional keyframe matching step, which creates a coarse alignment by using only keyframes instead of every single frame. We modify the standard DTW algorithm for the keyframe alignment to take into account the discontinuity of keyframes. The modification is similar to combinational subsequence matching (CSM) proposed in [9]. Whereas CSM finds an approximation to the alignment by conversion into a linear programming problem, we propose a parameter-less dynamic programming approach. Video copy detection and video retrieval are related to our problem and have been studied thoroughly [10, 11, 12]. However, frame-by-frame alignment is usually of secondary importance. Furthermore they are usually evaluated on short clips such as trecvid [13] and not designed to align two long videos.

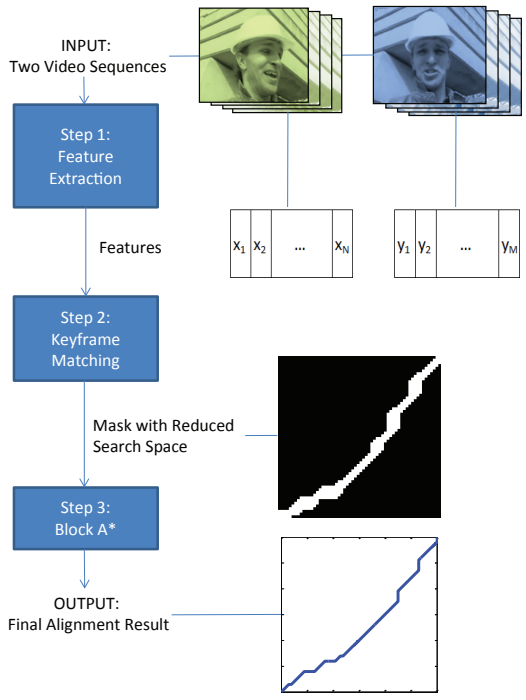

Fig. 1. Proposed Algorithm

The main contributions of this paper are the keyframe matching using path extension and a parameter-free DDTW for the keyframe matching, which outperforms the standard DTW in terms of accuracy while maintaining the same space and time complexity, and the Block A* approach, which significantly reduces the time and space requirements of the video alignment algorithm. To the best of our knowledge, A* has not been used before to speed up sequence alignment. The paper is organised as follows. In Section 2 we briefly introduce related work and in Section 3 we explain our proposed algorithm. Sections 3 contains our results and experimental evaluation and Section 4 concludes the paper.

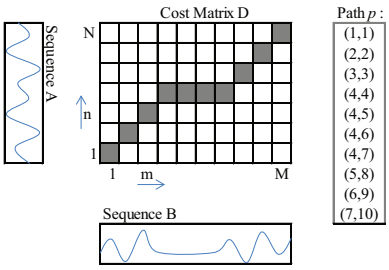

Fig. 2. Cost Matrix of Two Sequences

2. RELATED WORK

2.1. Dynamic Time Warping (DTW)

We denote two time series as $X = [x_1, x_2, ...x_N] \in \mathbb{R}^{l \times N}$ and $Y = [y_1, y_2, ...y_M] \in \mathbb{R}^{l \times M}$, where N and M are the lengths of the time series and l the feature dimension. In the video alignment problem, N and M are the number of frames and x_n and y_m are the feature vectors of each frame. For the alignment of the sequences X and Y, we create a cost matrix D with the dimensions $N \times M$. We will refer to the matrix elements as nodes. Each node $d(n, m)$ in D is calculated as a similarity measure of the frames x_n and y_m and represents the *cost* of aligning the two frames. In order to align the two sequences, we try to find a path p from the lower left corner to the upper right corner of the matrix (see Figure 2) that minimizes the sum of the nodes along the path.

$$DTW(X, Y) = min \sum_{(n,m) \in p} d(n, m) \tag{1}$$

The movement steps are restricted to positive horizontal, vertical or diagonal movements to neighboring nodes. To find the best possible path, we use dynamic programming. We define a new matrix G, whose elements $g(n, m)$ contain the total cost of the optimal path from the start node to the node (n, m). We calculate each node of G with the following recursive formulation:

$$g(n, m) = d(n, m) + min \begin{cases} g(n-1, m) \\ g(n-1, m-1) \\ g(n, m-1) \end{cases} \tag{2}$$

The value of the top right node of the matrix G corresponds to the total alignment cost. For further details about the DTW algorithm, we refer to [1] and [14].

2.1.1. Adaption for Video Alignment (DTW_W)

We adjusted Equation 2 to make it more suitable for our task and refer to it as DTW_W. Diagonal moves in the path correspond to a match in the sequences, whereas vertical or horizontal moves correspond to an insertion or a deletion of frames, respectively. The value $d(n, m)$ is not meaningful for non-diagonal movements, because we do not care about the similarity of mismatching frames. We therefore replace $d(n, m)$ in equation (2) by a constant penalty W,

$$g(n, m) = min \begin{cases} g(n-1, m) + W \\ g(n-1, m-1) + d(n, m) \\ g(n, m-1) + W \end{cases} \tag{3}$$

The cost to move from $(n-1, m-1)$ to (n, m) via node $(n-1, m)$ is given by $W + W$. The cost of a diagonal move in case of a match is given by $d(n, m)$. In order to guarantee that the inequality $2W > d(n, m)$ always holds in case of a match, we can define a lower bound for W

$$W_{min} = d_{max}/2, \tag{4}$$

where d_{max} is the maximal tolerable value for $d(n, m)$ to be considered a match. In practice we can simply set $W = W_{min}$.

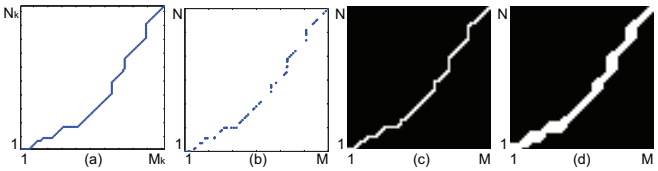

Fig. 3. Path estimation with Keyframes: (a) DDTW on keyframes, (b) mapping of keypoints from the keyframe sequence cost matrix to the full sequence cost matrix, (c) connect keypoints to complete the path estimate (d) extend line with morphological operation *dilation*

3. PROPOSED ALGORITHM FOR VIDEO ALIGNMENT

Figure 1 shows an overview of the proposed algorithm. It is divided into feature extraction, keyframe matching and Block A*. We will not describe the feature extraction, as it is outside the scope of this paper. Our method can be applied with any type of feature as an input. Instead we focus on the performance of the alignment algorithm. The following subsections describe the keyframe matching and Block A* algorithm in details.

3.1. Keyframe Matching

The goal of the keyframe matching step is to create an estimate of the final alignment path in order to reduce the search space for the refinement of the alignment path to a relatively narrow band. The keyframe matching consists of three steps: keyframe extraction, Discontinuous Dynamic Time Warping (DDTW) and path extension. For the keyframe extraction, we implemented an algorithm based on the shot detection method described in [15]. The details of the other two step are described in the following sections.

3.1.1. Discontinuous Dynamic Time Warping (DDTW).

We define two sequences $X_k = [x_{k1} \dots x_{kn} \dots x_{kN_k}] \in \mathbb{R}^{l \times N_k}$, and $Y_k = [y_{k1} \dots y_{km} \dots y_{kM_k}] \in \mathbb{R}^{l \times M_k}$, where x_{kn} and y_{km} are the keyframes from X and Y, and N_k and M_k are the number of keyframes. For the keyframe alignment, we do not enforce movements to neighboring nodes anymore. Instead we require that every node along the shorter dimension is assigned exactly to one node along the longer dimension. The proposed DDTW turned out to be far more robust and accurate than the standard version, and it is parameter-free and more tolerant towards mismatched keyframes. For $M_k \leq N_k$, the recursive formula is described as

$$g(n, m) = \begin{cases} d(n, m) & m = 1 \\ d(n, m) + \min_{i=1..n}\{g(i, m-1)\} & otherwise \end{cases} \tag{5}$$

Directly implemented, this formulation has complexity of $O(M_k N_k^2)$. But we can express equation (5) as

$$g(n, m) \tag{6}$$
$$= \begin{cases} d(n, m) & m = 1 \\ d(n, m) + \min \begin{cases} g(n-1, m) - d(n-1, m) \\ g(n, m-1) \end{cases} & otherwise \end{cases}$$

Equation 6 reduces the complexity to $O(M_k N_k)$.

3.1.2. Path Extension

To create a path estimate, we project the nodes from the keyframe alignment path (x_n, y_m) (Figure 3a) back to the coordinates (n, m) in the full matrix with dimension $N \times M$ (Figure 3b). Then we create a path estimation by connecting all the projected nodes (Figure 3c). As a last step, we extend the path with the morphological operation *dilation* (Figure 3d). The size of the structure element which is used for *opening* depends on the direction of the line. We calculate the scaling factor $s_{i-1,i}$ for the line between keypoint $i-1$ and i of the structure element with the equation

$$s_{i-1,i} = C \cdot \left(\frac{\mathbf{v}_{i-1,i}}{||\mathbf{v}_{i-1,i}||} \cdot \frac{\mathbf{u}}{||\mathbf{u}||} \right)^{-2}, \tag{7}$$

where C is a parameter which accomodades the uncertainty in the path estimation and $\mathbf{v}_{i-1,i}$ is the vector from keypoint $i-1$ to i. We use the result shown in Figure 3d as a mask for the Block A* algorithm described in Section 3.2. Specifically, the Block A* algorithm will only consider the masked region as a valid region for the path, which is limited by the path length and the size of the structure element. This step reduces the time- and memory complexity of the Block A* algorithm to $O(M+N)$.

3.2. Block A* Algorithm

3.2.1. Standard A*

The A* algorithm [8] is an extension of Djikstra's path finding algorithm [16]. The A* algorithm finds the path from the start node to the goal node by using a *best-first* approach. It follows the path with the lowest estimated alignment cost (f-score), while it stores potential alternative paths in a priority queue, which is known as the *open set*.

Beginning at the start node, in every iteration it selects the node with the next best f-score, and calculates the f-score of its neighbors. All new calculated f-scores are stored in the *open set* while the f-score of the just selected node is deleted from the *open set*.

The f-score $f(n,m)$ is calculated as

$$f(n,m) = g(n,m) + h(n,m), \tag{8}$$

where $g(n,m)$ is the cost from the start to the current position calculated with equation (3) and $h(n,m)$ is the heuristic cost, an estimate of the cost for the residual path from the current position to the goal (see Figure 4). A* is admissible or optimal, as long as the heuristic cost function is admissible [17]. An admissible heuristic means, that it has to be an underestimate of the actual alignment cost.

3.2.2. Heuristic Cost Function h.

In order for the Block A* algorithm to find the path with the minimal cost, we need a heuristic cost function (h-score) that is admissible. The lowest possible cost for a diagonal movement is 0. The cost for a single nondiagonal movement step is W (from Equation 3). Following that, we define the heuristic cost as

$$h(n,m) = abs((N-n) - (M-m)) \cdot W \tag{9}$$

where N and M are the sequence lengths.

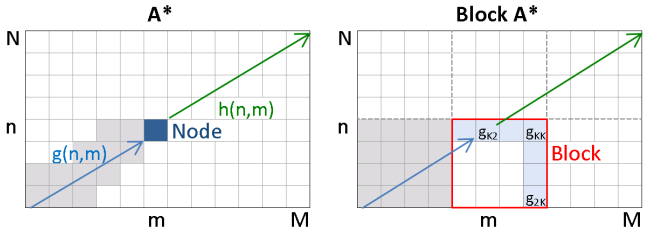

Fig. 4. Comparison of A* and Block A* algorithm. Left: The gray shaded nodes were already evaluated, the blue node (n,m) is the current node. Right: The gray shaded block on the left is already evaluated. The framed block is the current block. All the blue shaded nodes are considered for the f-score calculation

3.2.3. Block A* Extension

The Block A* algorithm is derived from the standard A* algorithm [8]. Instead of dealing with single nodes, we group them into blocks of $K \times K$ nodes and apply the A* algorithm directly to the blocks. Instead of calculating the f-score for each node, we define the $\hat{f}$-score, which is equivalent to an f-score for the whole block.

3.2.4. Calculation of Block $\hat{f}$-score

For a selected block, we first calculate the $g(m,n)$ values for every node within that block with DTW_W. Then we calculate the $\hat{f}$-score as the minimum f-score from the nodes of the top and right row (see blue nodes in Figure 4)

$$\hat{f} = min \begin{cases} f(K,i), & i = 1...K-1 \\ f(K,K) & \\ f(j,K), & j = 1...K-1 \end{cases} \tag{10}$$

where i is the row index of the node in the right column, j the column index of the node in the top row of the current block and K is the block size. Combining equations 8 and 10, we can express $\hat{f}$ as

$$\hat{f} = min \begin{cases} g(K,i) + h(K,i), & i = 1...K-1 \\ g(K,K) + h(K,K) & \\ g(j,K) + h(j,K), & j = 1...K-1 \end{cases} \tag{11}$$

While maintaining the same accuracy, the block A* architecture reduces the maximal length of the open list priority queue[1] by a factor K^2 and allows the calculation of independent blocks in parallel, which results in a significant speed boost.

4. EXPERIMENTAL EVALUATION

In this Section we present the details of the experimental setup to evaluate the performance of the Video Alignment algorithm. For all experiments in Section 4, we used the same features. We downsampled the frames to a size of 8×4 pixels and then normalized the pixels with min-max normalization. Stacked into a vector, this results in a feature size of $8 \times 4 \times 3$ values per frame. For the alignment cost / feature similarity we used the L_1 distance between the feature vectors.

[1] Although priority queue implementations are generally very efficient, they become slow when they grow very large

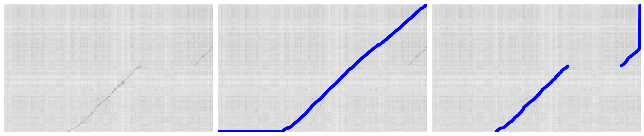

Fig. 5. **Keyframe alignment path:** The left image shows the cost matrix D of the keyframes. The two images on the right show the alignment paths found by DTW and DDTW.

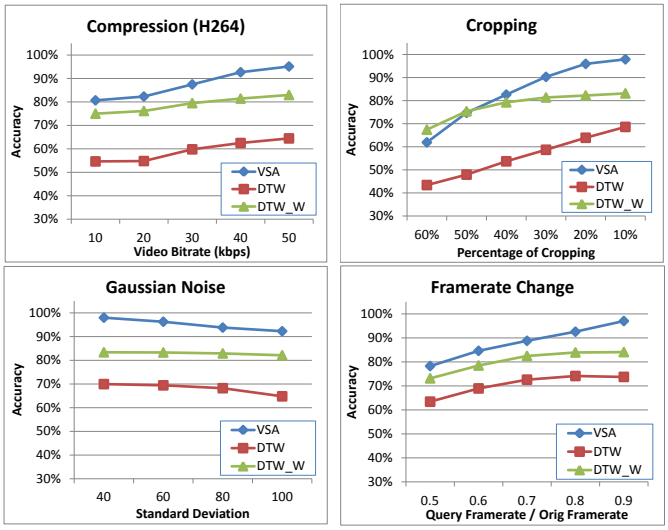

Fig. 6. **Accuracy comparison** DTW described in equation (2), DTW_W: described in equation (3),VSA: proposed Video Sequence Alignment Algorithm

4.1. Accuracy of Discontinuous Dynamic Time Warping (DDTW)

In the following experiment, we demonstrate on an example the advantages of the DDTW over the standard DTW. Figure 5 shows the cost matrix D of the keyframes of two video sequences. Dark colors resemble high similarity between the frames. Two dark line segments, separated by a large gap, represent two matching parts in video sequences. DTW fails to recognize the second segment because it assumes a continuous path whereas DDTW correctly identifies both segments.

4.2. Accuracy

For this experiment, we used 60 video segments of 9 minutes each generated from youtube videos from different genres. Then we re-encoded them with a different codec and deleted and inserted short sequences between 3 and 6 minutes at different locations. From the resulting videos we generated 20 copies with different levels of cropping, Gaussian noise, compression rate and framerate. Our dataset for the accuracy tests consists of 60 original videos and 1200 query videos.

We used our proposed VSA algorithm to align the query videos to the original videos from our dataset. We compared the result to the DTW and the DTW_W algorithms. The accuracy is calculated as the number of correctly matched frames divided by the total number of frames in the query video. We allow a tolerance of ± 3 frames (120ms). The averaged accuracy results for different levels of

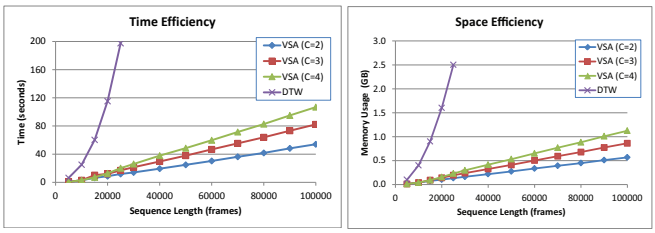

Fig. 7. **Efficiency:** Execution time and memory usage of VSA for different mask sizes C in comparison with DTW

compression, Gaussian noise, cropping and framerate differences are shown in the four graphs in Figure 6. The figures show that our proposed algorithm is robust against many attacks and significantly outperforms both, the DTW and the DTW_W algorithms except for the test case where 50% and 60% of the image is cropped. The higher accuracy values are due to the DDTW and the keyframe matching step, because the Block A* algorithm alone would produce the same result as the DTW_W algorithm, as they are both optimal.

4.3. Time and Space Efficiency

In the experiments in Figure 7, we used two different versions of the movie "The Warring States". The original has a framerate of 25 fps and a resolution of 704×528 pixels, the query has 24 fps and 768×432 pixels and a different beginning. We extract clips with the same length N from both videos. Then we increase the length of both clips simultaneously until we reach 100,000. We run the algorithms on our test machine with an Intel Xeon CPU @3.3GHz Processor and 8GB RAM. We tested three different mask sizes for VSA. The mask size is dependent on the parameter C in Equation (7). The values for C correspond to an average mask diameter of 800, 1200 and 1600 nodes. The time and memory complexity of DTW and DTW_W is $O(N^2)$. The curves for DTW and DTW_W are identical, hence only DTW is shown in Figure 7. The statistics for DTW ends at N=25,000, because longer sequences exceeded the available memory. We can clearly see however the N^2 complexity in time and memory usage in the DTW graph. We show in Figure 7 that the execution time as well as the memory usage of the proposed VSA algorithm increases linearly with $O(N)$, which is what we claimed in Section 3.1. The time used for the keyframe matching can be neglected. For example, the alignment of two 1 hour videos took 91 seconds from which only 3 seconds were used for the keyframe matching.

5. CONCLUSION

In this paper we introduced a robust Video Sequence Alignment algorithm which has linear time and space complexity. We also showed that the accuracy of the video alignment can be improved by adding a keyframe matching step prior to the frame-by-frame alignment. A discontinuous, parameter-free dynamic time warping (DDTW) method for the keyframe alignment led to significant improvements in accuracy. We furthermore reduced the time and memory requirements while maintaining the accuracy of the alignment process by introducing the Block A* algorithm.

6. REFERENCES

[1] J B Kruskall and M Liberman, *The Symmetric Time Warping Algorithm: From continuous to discrete*, Addison, 1983.

[2] S B Needleman and C D Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins.," *Journal of molecular biology*, vol. 48, no. 3, pp. 443–53, Mar. 1970.

[3] CJ Tsai and AK Katsaggelos, "Dense disparity estimation with a divide-and-conquer disparity space image technique," *Multimedia, IEEE Transactions on*, vol. 1, no. 1, pp. 18–29, 1999.

[4] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 26, no. 1, pp. 43–49, Feb. 1978.

[5] F Itakura, "Minimum prediction residual principle applied to speech recognition," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 23, no. 1, pp. 67–72, 1975.

[6] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 262–270.

[7] Stan Salvador and Philip Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol. 11, no. 5, 2007.

[8] PE Hart, NJ Nilsson, and B Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, 1968.

[9] Maodi Hu, Yunhong Wang, and James J Little, "Combinational subsequence matching for human identification from general actions," in *Computer Vision–ACCV 2012*, pp. 453–464. Springer, 2013.

[10] Jérôme Revaud, Matthijs Douze, Cordelia Schmid, and Hervé Jégou, "Event retrieval in large video collections with circulant temporal encoding," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 2459–2466.

[11] Ja-Hwung Su, Yu-Ting Huang, Hsin-Ho Yeh, and Vincent S. Tseng, "Effective content-based video retrieval using pattern-indexing and matching techniques," *Expert Systems with Applications*, vol. 37, no. 7, pp. 5068–5085, July 2010.

[12] Matthijs Douze, H Jégou, C Schmid, and P Pérez, "Compact video description for copy detection with precise temporal alignment," *Computer VisionECCV 2010*, pp. 522–535, 2010.

[13] George Awad, Paul Over, and Wessel Kraaij, "Content-based video copy detection benchmarking at trecvid," *ACM Trans. Inf. Syst.*, vol. 32, no. 3, pp. 14:1–14:40, July 2014.

[14] Lawrence Rabiner and Biing-Hwang Juang, "Fundamentals of speech recognition," *Prentice hall*, 1993.

[15] HongJiang Zhang, Atreyi Kankanhalli, and Stephen W. Smoliar, "Automatic partitioning of full-motion video," *Multimedia Systems*, vol. 1, no. 1, pp. 10–28, Jan. 1993.

[16] EW Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, , no. l 959, pp. 269–271, 1959.

[17] Stuart Jonathan Russel, Peter Norvig, John F Canny, J M Malik, and Douglas D, *Artificial intelligence: a modern approach*, Prentice hall Englewood Cliffs, 1995.