DEMOCRATIC AND POPULAR REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research
University of Mohamed Khider - BISKRA
Faculty of Exact Sciences, Natural Sciences and Life
Computer Science Department

Order Number: IVA12/M2/2019

REPORT

PRESENTED TO OBTAIN THE ACADEMIC MASTER DIPLOMA IN

COMPUTER SCIENCE

OPTION: ARTIFICIAL LIFE AND IMAGE

Vehicle Registration Plate Recognition

By:

Ganibardi Mohamed Salah

Defended the 11/07/2019, in front of the jury composed of:

Belaiche Hamza	MAA	President
Mokhtari Bilal	Phd	Supervisor
Mouaki Benani Nawel	MAA	Examiner

MOHAMED KHEIDHER UNIVERSITY

Abstract

Exact Sciences and Sciences of Nature and Life Faculty
Computer Science Department

Master of Computer Graphics

Vehicle Registration Plate Recognition

by Mohamed Salah GANIBARDI

Objects recognition and reading characters are a natural and easy process for humans which can tell easily if an object is a home or a car, if a letter is a ‘C’ not a ‘G’ or is a ‘O’ not a ‘Q’, also can distinguish between a letter and a number like the letter ‘O’ and the number ‘0’, or the letter ‘I’ and the number ‘1’, what makes this process so easy is the ability of human to learn. Is this process easy for a computer? Learning is part of human nature, and the hippocampus of brain is where this happens as one keep assimilating new information on any subject, the other is trying to skill itself in. For a computer is a difference story, how we can make a computer learn?

Keywords: object recognition, optical character recognition, real-time.

Contents

List of Figures

List of Tables

Introduction

Nowadays, the majority of the developing countries suffer from the traffic congestion issues, and use a huge amount of efforts to solve such issues. Because the number of vehicles are increasing in a way that will not be controllable in the future, which makes finding a stolen car or one's that violates traffic laws hard. Vehicle Number Plate Recognition system will be a good start for resolving not only the previous two problems, but we can use it in a wide varieties of places, such as controlling gates, traffic monitoring etc. However, in order to create such system, it will be needed to construct such algorithm which will be able to identify the location of number plate of the vehicle in the frame, then extract the characters from it, and then recognize them. In this paper we are going to review our number plate recognition application based on deep learning which have several operations that can helps for resolving the problems listed before.

The memory is organized as follows. In the first chapter, we give basis notions in the field of image processing and recognition. In the second chapter, we present neural networks and deep learning architecture. In the third chapter, we give in details the used Faster R-CNN architecture for detecting and recognizing objects. In the last chapter, we gives the conception and the implementation of our used system.

Chapter 1

Background

1.1 Introduction

In our topic we are interested in Alphanumeric Characters Recognition (OCR), which is a process that allows recognizing different alphanumeric characters from a scanned image. It is a key research area of pattern recognition, and it has been widely applied in automatic number-plate recognition check, street signs, invoice reading, voice reading of pdf documents, library call number recognition, and so on.

In this chapter, we will describe in details the field of alphanumeric characters recognition, and its use for number-plate recognition.

1.2 Optical Character Recognition

1.2.1 Definition

Optical Character Recognition usually called OCR, is the ability of the computer to detect and extract a printed or handwritten text characters inside digital images, or photograph might contain a street sign or traffic sign, etc, by examining the text and translating the characters into code that can be used for data processing.

1.2.2 Optical Character Recognition use cases

Recently, Optical Character Recognition engines have been developed for various application areas including [1] :

- License plates recognizing
- Scanning printed documents into editable versions
- Process cheques without human involvement in banking
- Process paperwork in health care
- Archiving historic documents, like newspapers, magazines, or phone books into searchable formats
- Read text from digital images
- Archiving signed legal documents into an electronic versions

- Etc...

1.2.3 OCR processing steps

1.2.3.1 Preprocessing

The first and the common step in the OCR processing is called preprocessing. In this phase there is a series of operations performed to the input digital image, it convert the image into a black and white image, were the dark areas are identified as characters that need to be recognized and white areas are identified as background. Other operation as filtering and adjusting illumination can also be applied to the input image [2].

1.2.3.2 Shape matching and pattern recognition

The variety of different fonts and ways of writing a single character makes the process of OCR hard to solve. The most common way to recognize a character is to apply a machine learning algorithm on a data set of examples in order to detect patterns which allow characterizing each class with a signature which distinguish it from other classes. We can resume this on the following concepts.

1.2.3.2.1 Feature extraction

The success of OCR depends on the stability and accuracy of extracted features. The quality of the extracted features will directly affect the recognition result. According to the unified entropy theory of pattern recognition, we must ensure that the extracted features contain sufficient information to get a higher recognition rate. It consists in extracting some features from the shape of the objects in the images. These features are a kind of numerical values that can be easily used to calculate the distance between shapes. For example, possible features may be the length and the height of the object. In the literature, a variety of methods have been developed for extracting features from images, as well as characters images [Huang, Zhihu, 2010, Freeman, 1995, Dalal, Navneet,2005, Rao, N,2016].

1.2.3.2.2 Shape matching

Shape matching is the process of measuring the similarity between two given images. To do so, we calculate a distance between features extracted from images. Note that small distance between images means that they are similar and they belongs to the same class. For example, distances between different handwritten of the digits 9 must be small as possible.

Actually, the results of calculating a shape feature for an image provides one dimensional vector of values. The distance between two shapes A and B or images described each one by a vector of features is simply the Euclidean distance norm between their respective vectors v_1 and v_2.
The distance for two vector v_1 and v_2 is than given by :

$$d(A,B) = d(v_1,v_2) = \sqrt{\sum_{i=0}^{n-1}(v_i^A - v_i^B)^2}$$

Were n is the size of the vectors, and v_i^A and v_i^B are respectively the ith elements in the vectors v_1 and v_2

1.3 Machine learning

Machine Learning is a sub-area of artificial intelligence, it enables it systems to recognize patterns on the basis of existing algorithms and data sets and to develop adequate solution concepts . It is a king of algorithms which can build a model that make prediction for regression, classification tasks based only on given known data (features), and similarity measurement (distance). For example, we can find a model that predicts the price of an house based on known prices of others houses according to given features (such as area, number of pieces, etc). The required algorithms and data must be fed into the systems in advance and the respective analysis rules for the recognition of patterns in the data stock must be defined. Once these two steps have been completed, the system can perform the following tasks by Machine Learning :

- Finding, extracting and summarizing relevant data
- Making predictions based on the analysis data
- Calculating probabilities for specific results
- Adapting to certain developments autonomously
- Optimizing processes based on recognized patterns

The basic premise of machine learning is to build algorithms that can receive input data and use statistical analysis [Vapnik, Vladimir,1999] to predict an output while updating outputs as new data becomes available. In the literature, they are a lot of existing learning algorithms such as K-nearest neighbors, k-means, Support Vector Machine (SVM) [Laura Auria, 2008], neural network [Zurada, Jacek,1992], deep learning [LeCun, Yann,2015]. In this work, we will use Deep learning based algorithms.

1.4 Segmentation

1.4.1 Definition

Segmentation is one of the most important processes of image processing. Image segmentation technique is used to partition an image into meaningful parts having similar features and properties, every pixel in an image is allocated to one of a number of these features and properties.

Pixels belonging in the same region must have similar characteristics based on neighbors pixels (or close to each other), and those of different region must be different. Segmentation can be used for simplification, making image more easily analyzable, and selecting regions of interest. Thus, in our work, we use this process to select the number-plate from an image.

1.4.2 Segmentation approaches

Segmentation methods can be categorized into two types based on properties of image.

1.4.2.1 Discontinuity detection

This approach worked by partition an image, based on changes in gray-level using three types of detection [3] :

- Point detection : The detection of isolated points in an image by using a mask. we can say that a point has been detected at the location on which the mask is centered, if :

  $$|R| > T$$

  where T is the threshold and

  $$R = -(x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9) + 8 * x_5$$

  x_i is an image pixel [3].

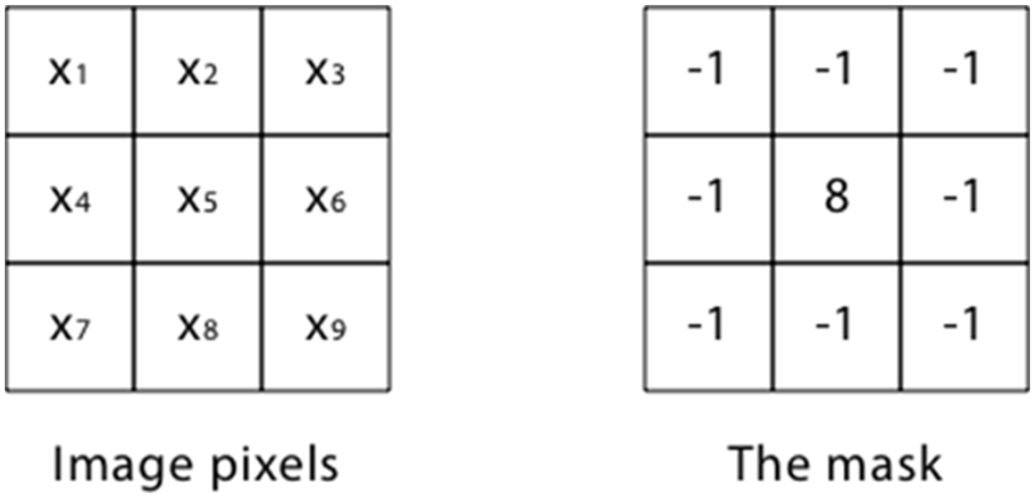


FIGURE 1.1: Point detection mask.

The gray level of an isolated point will be quite different from the gray level of its neighbors.

- Line detection : Detection of lines in an image using the following four masks R1, R2, R3 and R4 (see Figure1.2). If the first mask (see Figure1.1) were moved around the whole image, it would respond better to only one line oriented horizontally. With constant background, the maximum response would result when the line passed through the middle row of the mask. R1, R2, R3 and R4 run along the same image. Then at a certain point, we can say that the mask which has the maximum response, will make the line into its direction, i.e. if R2> R1, R3, R4 then the line has the direction 45° [3].

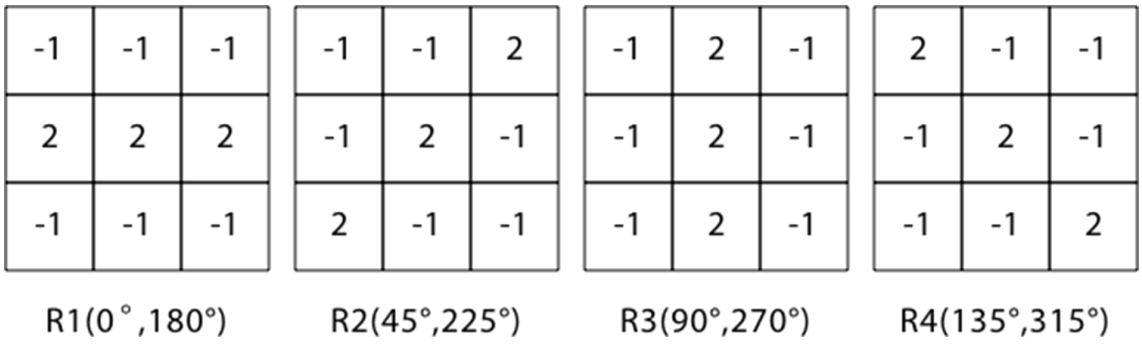


FIGURE 1.2: Line detection masks.

- Edge Detection : Edge detection methods transform an image into edge image benefits from the changes of grey tones in the image [4].

FIGURE 1.3: Edge detection example.

1.4.2.2 Similarity detection

In this methods, the image is segmented based on similarity, using several techniques, like thresholding techniques, region growing techniques and region splitting and merging, to divide the image into regions having similar set of pixels. The clustering techniques as K-means also use this methodology. The image is so divided into different clusters containing each one similar features based on a given criteria [3].

1.4.3 Segmentation for OCR

In order to achieve the Optical Character Recognition goal, we need first to use the segmentation process to extract words from the image. This is done in two different steps: line segmentation, and segmenting words.

1.4.3.1 Line segmentation

It consists of slicing a page of text or a zone of interest into its different lines, We can identify a line of text by looking for rows of white pixels with rows of black pixels in between [3] (see Figure 1.4).

FIGURE 1.4: Line segmentation example.

1.4.3.2 Segmenting words and characters

This step consist in isolating each word from another and separate the various letters of a word, by using the horizontal white space between words, which is called “interword space” to separate them, and using interletter space (horizontal white space between letters) to separate letters [3] (see Figure 1.5).

FIGURE 1.5: Segmenting words and characters.

1.5 Vehicle Plate Recognition

1.5.1 Definition

Vehicle Plate Recognition have been one of the most interesting path in the history of video surveillance, it uses Optical Character Recognition to make a computer capable of detecting and extracting text characters inside digital images contains a vehicle plate, of the purposes like traffic safety enforcement, automatic toll text collection, and vehicle parking system. Vehicle Plate Recognition can use existing road cameras or specified ones. In follows, different kind of approaches used for vehicle plate recognition.

1.6 Conclusion

In this chapter, we presented the concept of Optical Character Recognition, and its possible application for vehicle plate recognition. One of most successful strategy used in this fields is the uses of deep learning algorithms. Therefore, we will describe this kind of methods in the next chapter, and gives a detailed description of the algorithm we used.

Chapter 2

Neural Network

2.1 Introduction

Machine learning has widely used in the field of image recognition, and it allowed to achieve a spectacular results. In our work, we will use one of powerfull machine leaning techniques which is called Artificial Neural Networks. In this chapter, we will describe the basis of this concept.

2.2 Artificial Neural Network

An Artificial Neural Network is a computational model that is designed to model the way in which the brain performs a particular task or function of interest. Neural networks employ a massive interconnection of simple computing cells referred to as "neurons" inspired by the natural neurons to achieve the best performance possible [5]. Artificial Neural Network acts like the brain in two way:

- Knowledge is acquired by the network through learning process.
- Inter neuron connection strengths known as synaptic weights are used to store the knowledge.

An artificial neuron is a computational model inspired in the natural neurons. Natural neurons receive signals through synapses located on the dendrites or membrane of the neuron. When the signals received are strong enough (surpass a certain threshold), the neuron is activated and emits a signal though the axon. This signal might be sent to another synapse, and might activate other neurons.

2.2.1 Biological neural network

A neuron is a switch with information input and output. Will be activated if there are enough stimuli of other neurons hitting the input. Then, at the output, a pulse is sent to other neurons. Any single physiological action perceived, such as pain or pleasure is the output response of a collective activity due to innumerable neurons participating in the decision making control procedures in the nervous system [6].

2.2.1.1 Neural networks components

Neurons are made up of a nerve cell composed by a cell body called soma, Each soma has many dendrites and one axon. Axon signal can split hundreds of times, however,

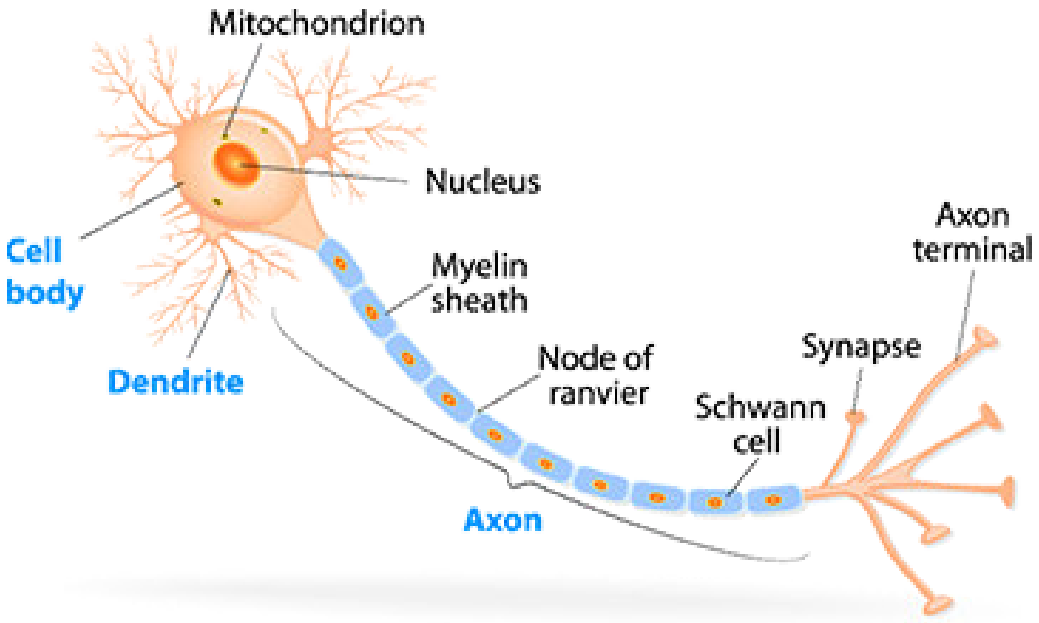


FIGURE 2.1: Biological neuron diagram [7].

Dendrites are thin structures that arise from the main cell body. Axons are nerve fibers with a special cellular extension that comes from the cell body.
Natural neurons contain hundreds of inputs. Dendrites are part of it. Synapse is characterized by effectiveness, called synaptic weight. Neuron output is formed in a following way : signals on dendrites are multiplied by corresponding synaptic weights, results are added and if they exceed threshold level on the result is applied a transfer function of neuron. Only limitation of transfer function is that it must be limited and non-decreasing. Neuron output is routed to axon, which by its branches transfers result to dendrites. In this way, output from one layer of network is transferred to the next one [6].

2.2.2 The perceptron

2.2.2.1 Definition

Perceptron is a single layer neural network and a multi-layer perceptron is called Neural Networks. Perceptron is a linear classifier (binary) and it is used in supervised learning. It helps to classify the given input data. with a simple input output relationship which shows we are summing n number of inputs multiplied with their associated weights and then sending this input to a another function with a defined threshold. Normally with perceptrons, this is a Heaviside step function with a threshold value of 0.5. This function will give a real valued single value (0 or a 1), depending on the input.

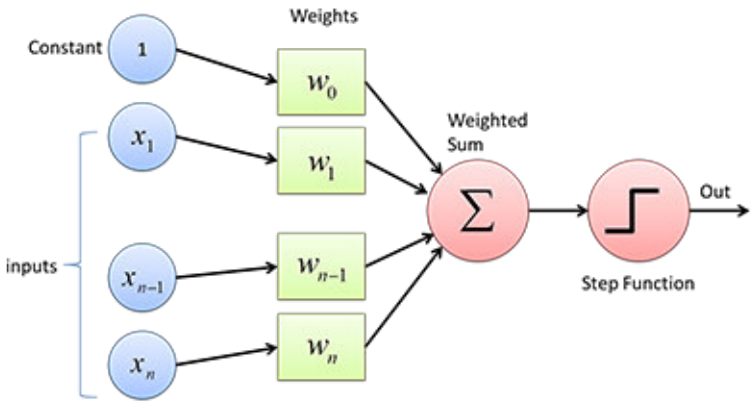


FIGURE 2.2: Perceptron.

2.2.2.2 The perceptron learning algorithm

The perceptron learning algorithm goal is to find the weights vector that can perfectly classify positive inputs and negative inputs in our data by changing them until all input records are all correctly [8].

2.2.2.3 Limitations of the perceptron

Linear models like the perceptron cannot represent some functions, can only learn to approximate the functions for linearly separable datasets. The linear classifiers that we have examined find a hyperplane that separates the positive classes from the negative classes, if no hyperplane exists that can separate the classes, the problem is not linearly separable. Where a multilayer perceptron could solve many nonlinear problems.

2.2.3 Multilayers neural network

This multi-layer network has different names : multi-layer perceptron (MLP), feed-forward neural network, artificial neural network (ANN), backprop network. It is a neural network with one input layer, one or more hidden layers, and one output layer. Each layer contains an artificial neurons or more. These artificial neuron of the multilayer perceptron is similar to its predecessor, the perceptron, but it adds flexibility in the type of activation layer we can use, wich is more generalized [5].

2.2.3.1 Artificial neuron input

The artificial neuron input are based on the weights on the input connections. These input passed to the activation function or they can be ignored by a 0.0 weight on an input connection.
The net input is produced by multiplying the weights on connections by activation, the total weighted input of the artificial neuron can be represented as

$$I_i = W_i.A_i$$

where W_i present the vector of the weights leading into neuron i and A_i is the vector of activation values for the inputs to neuron i. Build on this equation by accounting the bias term that is added per layer

$$I_i = W_i.A_i + b$$

For producing the output, we have to wrap this I_i with an activation function σ

$$a_i = \sigma(I_i)$$

The output value passed to the next layer for the neuron i is the activation value a_i, the output value passed through connections to other artificial neurons as an input value. In case of the activation function is the sigmoid function :

$$f(z) = \frac{1}{1+e^{-z}}$$

The range of the output will be between 0 and 1, which is the same output as the logistic regression function.

2.2.3.2 Weights

The weights are the coefficients that use to amplify or minimize an input signal, which represent the strength of the connection between neurons, and decides how much influence the input will have on the output.

2.2.4 Biases

A bias has it's own connection weight, what makes sure that even when all the inputs are none there's gonna be an activation in the neuron.

2.2.4.1 Activation function

It's used to introduce non-linearity to neural networks, it defines the status of node if it should be activated or not based on the weighted sum. The Sigmoid activation function is one of the most used activation function, it squashes values between a range 0 to 1.

2.2.4.2 Multilayer neural network architecture

We have artificial neurons arranged into groups called layers. Building on the layer concept, we see that the multilayer neural network has the following concept: A single input layer. One or many hidden layers, fully connected. A single output layer [8]. The neurons in each layer are all fully connected to all neurons in all adjacent layers. The neurons in each layer all use the same kind of activation function. For the input layer, the input is the raw vector input. The input to neurons of the other layers is the output of the previous layer's neurons. As data flows through the network in a feed-forward fashion, it is impacted by the connection weights and the activation function kind. Let's now take a look at the specifics of each layer type.

2.2.4.3 Input layer

The input layer is responsible to receive the initial data for the neural network then duplicate it to it multiple outputs. The number of neurons in an input layer is the same number as the input feature.

2.2.4.4 Hidden layer

Hidden layer have neurons, it's the intermediate between input and output layer which apply different transformations to the input data, there are one or more layers. To encode the learned information extracted from the raw training data, hidden layers use the weight values.

2.2.4.5 Output layer

Output layer is the last layer in the network, the active nodes of the output layer combine and modify the received data from the last hidden layer data to produce the result for given inputs [8].

2.2.4.6 Connections

The connections between neurons in one layer to other neurons in other layer or the same layer. It has it weight value. The Goal of the training is to update this weight value to decrease the loss(error).

2.3 Deep Learning

Deep learning is a neural network with more than two layers. A computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers [9].

2.4 History of deep learning application

The path of deep learning was very rich, in the late 80s and early 90s, deep learning advances in modeling sequential data with recurrent neural networks, as time went on, the research community created better artificial neuron variants over the course of the late 90s. During the 2000s, researchers and industry applications began to progressively apply these advances in products like Google Translate, Amazon Echo, Self-driving cars.

2.5 Deep learning applications

Deep learning excels at identifying patterns in unstructured data, which most people know as media such as images, sound, video and text. Below is a list of sample use cases we've run across :

- TIMIT phoneme recognition (Graves et al., ICASSP 2013).
- Optical character recognition (Breuel et al., ICDAR 2013).
- Language identification (Gonzalez-Dominguez et al., Google, Interspeech 2014).
- Text-to-speech synthesis (Fan et al., Microsoft, Interspeech 2014).
- Prosody contour prediction (Fernandez et al., IBM, Interspeech 2014).
- Large vocabulary speech recognition (Sak et al., Google, Interspeech 2014).
- Medium vocabulary speech recognition (Geiger et al., Interspeech 2014). 2014).
- English-to-French translation (Sutskever et al., Google, NIPS 2014).
- Audio onset detection (Marchi et al., ICASSP 2014).
- Social signal classification (Brueckner Schulter, ICASSP 2014).
- Arabic handwriting recognition (Bluche et al., DAS 2014). 2014).
- Image caption generation (Vinyals et al., Google, 2014).
- Video-to-textual description (Donahue et al., 2014).
- Syntactic parsing for natural language processing (Vinyals et al., Google, 2014).
- Photo-real talking heads (Soong and Wang, Microsoft, 2014).

2.6 Deep learning architecture

Deep learning architecture can be divide in three layers, the input layer, multiple hidden layers and output layer, where the depth of DL architecture is defined by the number of hidden layers.
Depending on the type of hidden layers used, different non-linear functions can be learned [10].
The next two equations (2.1), (2.2) represent a simple Artificial Neural Network, where the equation (2.1) (hidden layer) represents the non-linearity observed within the data, the activation function determines the non-linearity characteristics. The equation (2.2) (output layer) produce prediction from the previous non-linearity characteristics .

$$A_1 = Activation(W_1 X + b_1) \tag{2.1}$$

$$y = W_2 A_1 + b_2 \tag{2.2}$$

2.6.1 Activation functions

The activation functions in the hidden layer, its goal is helping in mapping the non-linearity relationship between input and output. Most used activation functions in hidden layers are sigmoid and hyperbolic tangent function (e.g., tanh). There is no rule for applying specific activation functions, depending on the dataset we use, an activation function need to be evaluate [10].

2.6.2 Hidden layers

Each hidden layer complexity is defined by the number of parameters used to represent it, where the number of parameters is controlled by hyper-parameters, which are the number of hidden units and the parameter of the L2 regularisation.
The number of hidden units indicates the number of parameters (i.e. weights) in each layer and L2 regularisation reduces the magnitude of these parameters to avoid over-fitting [I. Goodfellow, Y. Bengio, A. Courville, Deep Learn. (2016).]. Tuning hyper-parameters is important to obtain a good model fit [10].

2.6.3 Output layer

Typically, the output consist of an identity activation (also referred to as linear activation) for regression problems, what makes negative predictions possible. In case where data used demands are non-negative response values, identity activation is not suitable. Instead, the rectified linear activation function can be used [10].

2.6.4 Parameters

In deep networks, we still have a parameter vector representing the connection in the network model we're trying to optimize. The biggest change in deep networks with respect to parameters is how the layers are connected in the different architectures [10].

2.6.5 Loss functions

The loss functions quantify the agreement between the predicted output (or label) and the ground truth output. The loss functions is used to determine the penalty for an incorrect classification of an input vector. The following are examples of some of the loss functions:

- Hinge loss
- Squared loss
- Logistic loss
- Negative log likelihood

2.6.6 Optimization methods

Training a model in machine learning involves finding the best set of values for the parameter vector of the model. We can think of machine learning as an optimization problem in which we minimize the loss function with respect to the parameters of our prediction function (based on our model) [11].

2.6.7 Hyperparameters

Hyperparameters are free to be chosen by the user that might affect performance. Hyperparameters fall into several categories [10] :

- Layer size
- Magnitude (momentum, learning rate)
- Regularization (dropout, drop connect, L1, L2)
- Activations (and activation function families)
- Weight initialization strategy
- Loss functions

2.7 Conclusion

In this chapter, we have presented Deep Learning which uses uses the concept of artificial neural networks that is based from the biological neural networks, what makes humans able to learn faster, with such inspiration it is possible to make a computer able to learn from images and recognize its contents.

Chapter 3

Faster Region-based Convolutional Neural Network

3.1 Introduction

The process of recognizing objects in an image pass through two steps: detection objects, and classify them. Detection consists in drawing a bounding box around the object, and the recognition consists in associating the detected object to a category. The called Convolution Neural Networks (CNN) is one of the most technique widely used to detect and recognize objects. However, this kind of methods have some drawbacks, especially because they are a fully connected layers, and need a huge number of fixed regions which are trained in a considerable time. A new kind of convolution networks such as Faster Region-based Convolution Neural Network (R-CNN) [12], Efficient and Accurate Scene Text detector (EAST), and You Only Look Once (YOLO) [13] are proposed to resolve these issues. In our work we opted to use Faster R-CNN, we are going to describe in this chapter.

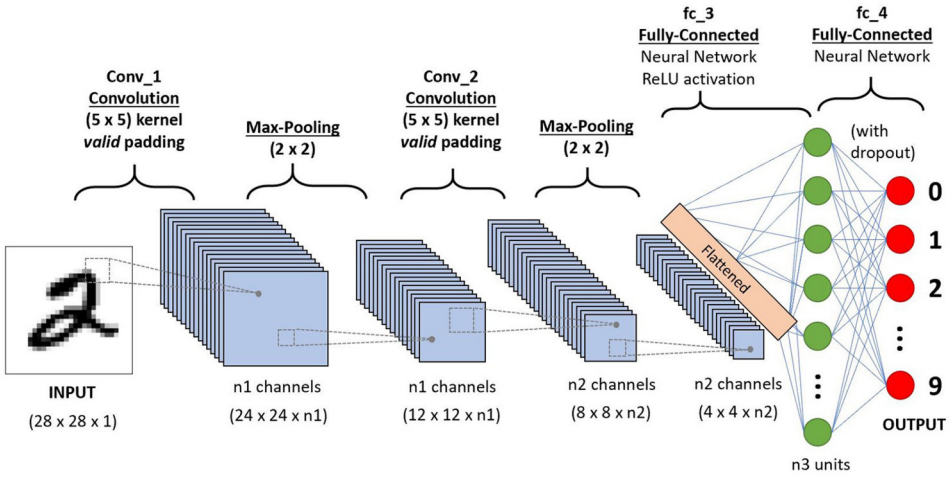


FIGURE 3.1: An example of CNN architecture

3.2 Convolution Neural network

In order to describe R-CNN, we have first to introduce the CNN architecture. CNN [14] is a deep learning network, which combines neural networks concept and convolutions. These convolutions are a hidden layers, which can be seen as a filter (also called kernel, or mask) applied to the input image.

The overall architecture of the Convolutional Neural Network (CNN) includes an input layer, multiple alternating convolution and max-pooling layers, one fully-connected layer and one classification layer [15].

3.2.1 Feature extraction

Convolutional layers transform data from the input layer using the previous layer patch. each layer will compute a dot product between the region of the neurons in the input layer and the weights to which they are locally connected in the output layer [15]. Actually, Convolutional layers takes input, applies a convolution kernel, and gives us a feature map as output.

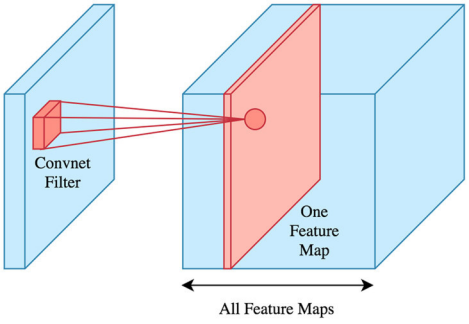


FIGURE 3.2: Feature maps [16].

The convolution operation goal is convolution neural network feature detection. Have as input a raw data, or a feature map output from another convolution.

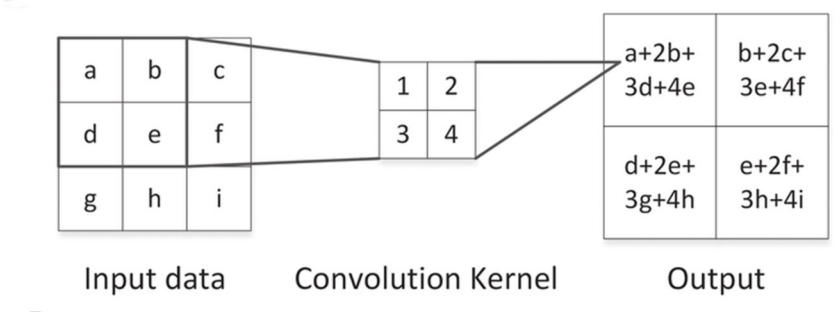


FIGURE 3.3: An example of convolution operation

The kernel slid across the input data to produce the output data. While sliding, the kernel is multiplied by the input data values within its bounds, creating a single entry in the

output feature map. In practice the output is large if the feature we're looking for is detected in the input. We commonly refer to the sets of weights in a convolutional layer as a filter (or kernel). This filter is convolved with the input and the result is a feature map (or activation map) [15].

Convolutional layers perform transformations on the input data volume that are a function of the activations in the input volume and the parameters (weights and biases of the neurons). The activation map for each filter is stacked together along the depth dimension to construct the 3D output volume. Convolutional layers have parameters for the layer and additional hyperparameters . Gradient descent is used to train the parameters in this layer such that the class scores are consistent with the labels in the training set. The major components of convolutional layers are :

- Filters
- Activation maps
- Parameter sharing
- Layer-specific hyperparameters
- Learned filters and renders
- Rectified Linear Unit (ReLU) activation functions.

3.2.2 Pooling

Pooling layers are commonly inserted between successive convolutional layers. Its goal is to reduce the spatial size of the representation. Pooling layers reduce the data representation progressively over the network and help control overfitting.
The Pooling Layer operates independently on every depth slice of the input, it uses the MAX operation to resize the input data spatially.
One of the most used pooling layer form is with a 2x2 filter size, taking the largest of four numbers in the filter area.
Pooling layers perform downsampling operations along the spatial dimension of the input data using filters (kernels).
This means that if the input is a 32x32 image, the output image would have smaller dimensions (e.g., 16x16).
The most common setup for a pooling layer is to apply 2x2 filters with a stride of 2. This will downsample each depth slice in the input volume by a factor of two on the spatial dimensions (width and height). This downsampling operation will result in 75 of the activations being discarded. Pooling layers do not have parameters for the layer but do have additional hyperparameters. This layer does not involve parameters, because It is not common to use zero-padding for pooling layers.

3.2.3 Fully connected layers

Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset. See the Neural Network section of the notes for more information.

3.3 Faster Region-based Convolutional Neural Network

Faster Region-based Convolutional Neural Network or Faster R-CNN is an approved version of CNN, which is widely used for object detection. It takes an image as input and outputs bounding boxes around objects of interest with class labels. In our work, we will use it to detect registration plate in an image.

Befor talking about Faster R-CNN, we have to understand the original of it, which is the R-CNN. it main idea is to use search selective method to find the regions of interests and passe them to ConvNent, R-CNN tries to find out the areas that have possibility to be an object combining the similar pixels and textures into boxes. From search selective it uses 2000 proposed boxes. These boxes will passe to the pre-trained CNN model, where regression between the predicted bounding boxes and the ground-truth bounding boxes are computed.

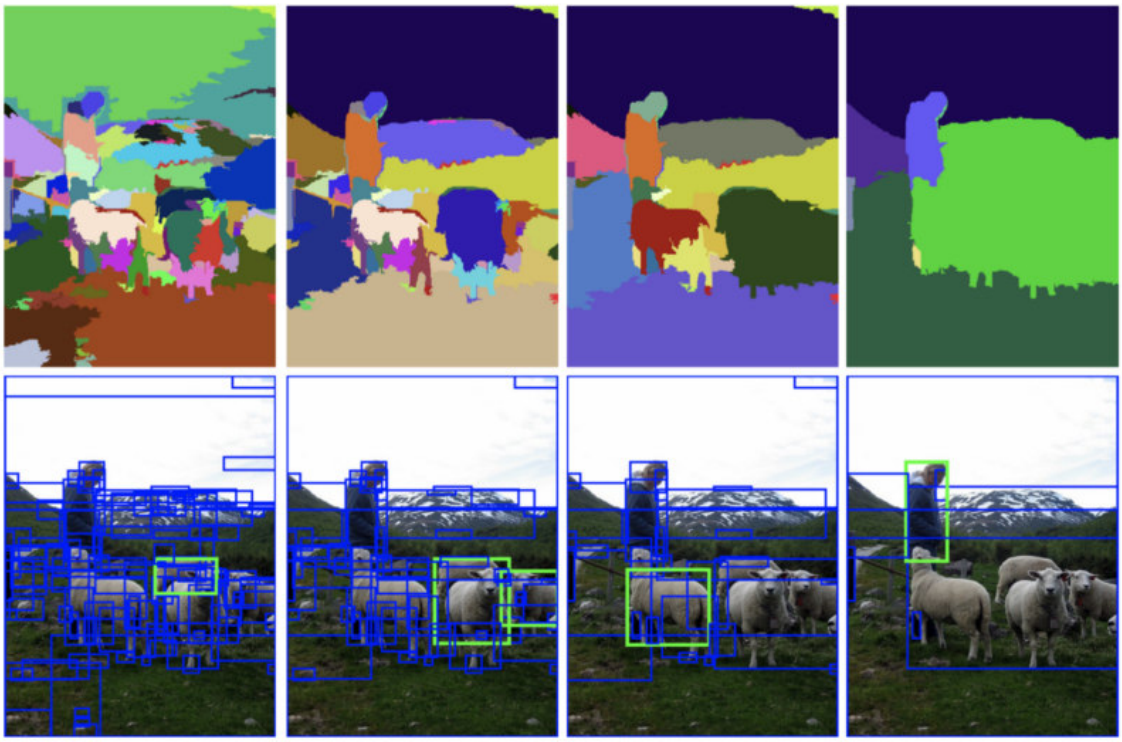


FIGURE 3.4: An example of search selective [17].

The next step is Fast R-CNN. What makes Fast R-CNN better than the original R-CNN is that Fast R-CNN passes only the original image to the pre-trained CNN, instead of

applying 2000 times CNN to proposed boxes. In this case the search selective algorithm is computed base on the output feature map, after that, it ensure the standard and pre-defined output size by using ROI pooling layer. Then it passes the valid outputs to a fully connected layer.
Two output vectors are used to predict the observed object with a softmax classifier and adapt bounding box localisations with a linear regressor.
Faster R-CNN moves forward than Fast R-CNN by replacing search selective process by Region Proposal Network (RPN), which acts as an attention network that identifies the regions the classifier should consider. It became the state of the art in object detection when it was released in 2015. It takes an image as input and runs it through 2 modules: the first uses a RPN that proposes object regions and the second is a Fast R-CNN object detector that classifies the region proposals.

3.3.1 Faster R-CNN architecture

Faster R-CNN has several moving parts, which makes it, starting from an image to obtain a list of bounding boxes, a label assigned to each bounding box and a probability for each label and bounding box [18].

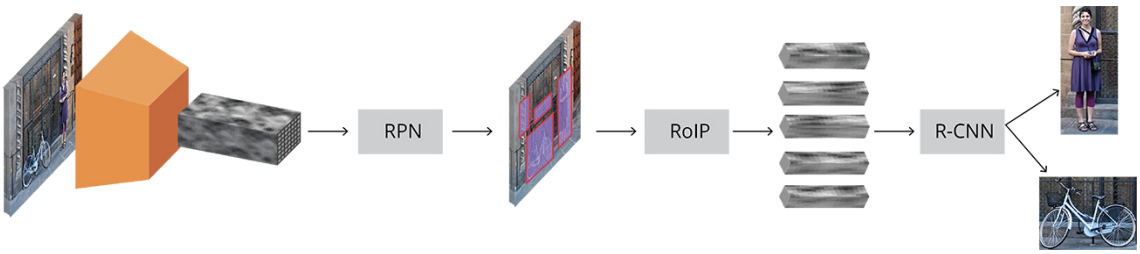


FIGURE 3.5: Complete Faster R-CNN architecture [19]

A 3 dimensions arrays (Height x Width x Depth) represents the input images. We use this as a feature extractor after are passing these arrays (tensors) through a pre-trained CNN up until an intermediate layer, ending up with a convolutional feature map [18].
The next step is Region Proposal Network (RPN). RPN is used to find up to a predefined number of regions (bounding boxes) using the features that the CNN computed.
The variable-length list of bounding boxes are generated by using anchors, where fixed sized reference bounding boxes are placed uniformly throughout the original image.
Every anchor have two problems to solve :

- Does it contain a relevant object?
- How we can adjust it to fit correctly the relevant object?

After having the list of possible relevant objects with their locations, and by using the features extracted by the CNN and the bounding boxes with relevant objects, we can apply Region of Interest (RoI) Pooling and extract those features which would correspond to the relevant objects into a new array [18].
In the end, the R-CNN module uses that array to classify the content in the bounding box, and adjusts it coordinates to fits better the object.

3.3.2 Base network

Acctualy, the first step of the Faster R-CNN is using a pre-trained CNN for classification and using the output of an intermediate layer. It's important to understand how and why it works. So, we choose the standard VGG-16 as an example [19].

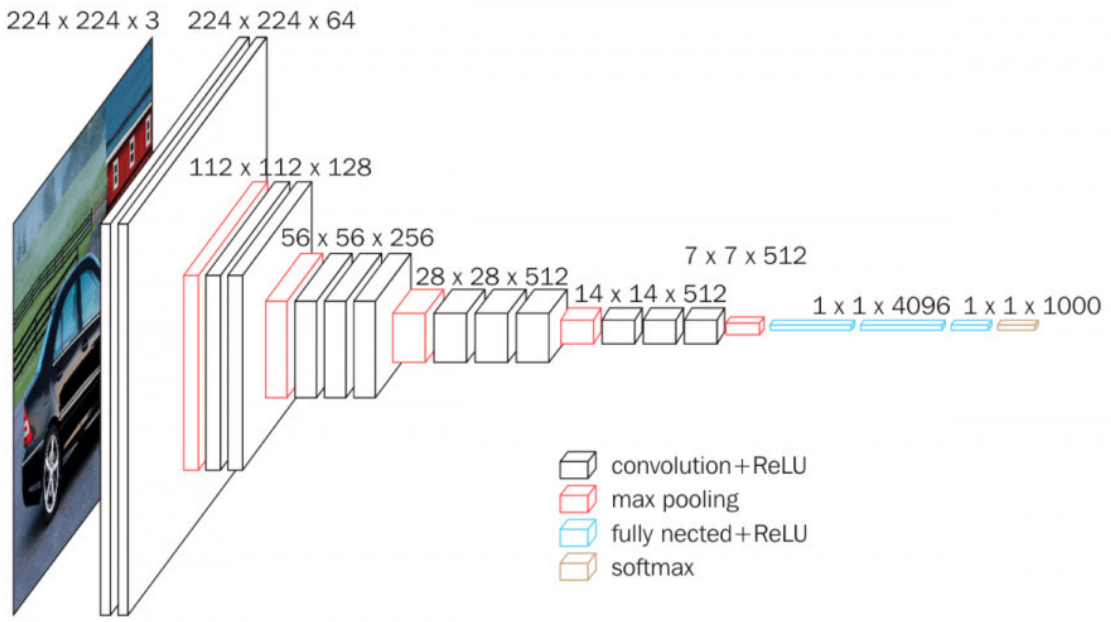


FIGURE 3.6: Base network VGG-16 architecture [19].

Each convolutional layer creates abstractions based on the output of the previous layer. The first layers are using for learn edges, and the second are using to find patterns in edges in order to activate for more complex shapes.
Like a results we have a convolutional feature map which has encoded all the information for the image while maintaining the location of the things. This convolutional feature map has spatial dimensions much smaller than the original image, but greater depth, the results of applying of the pooling between convolutional layers. The depth increases based on the number of filters the convolutional layer learns [19].

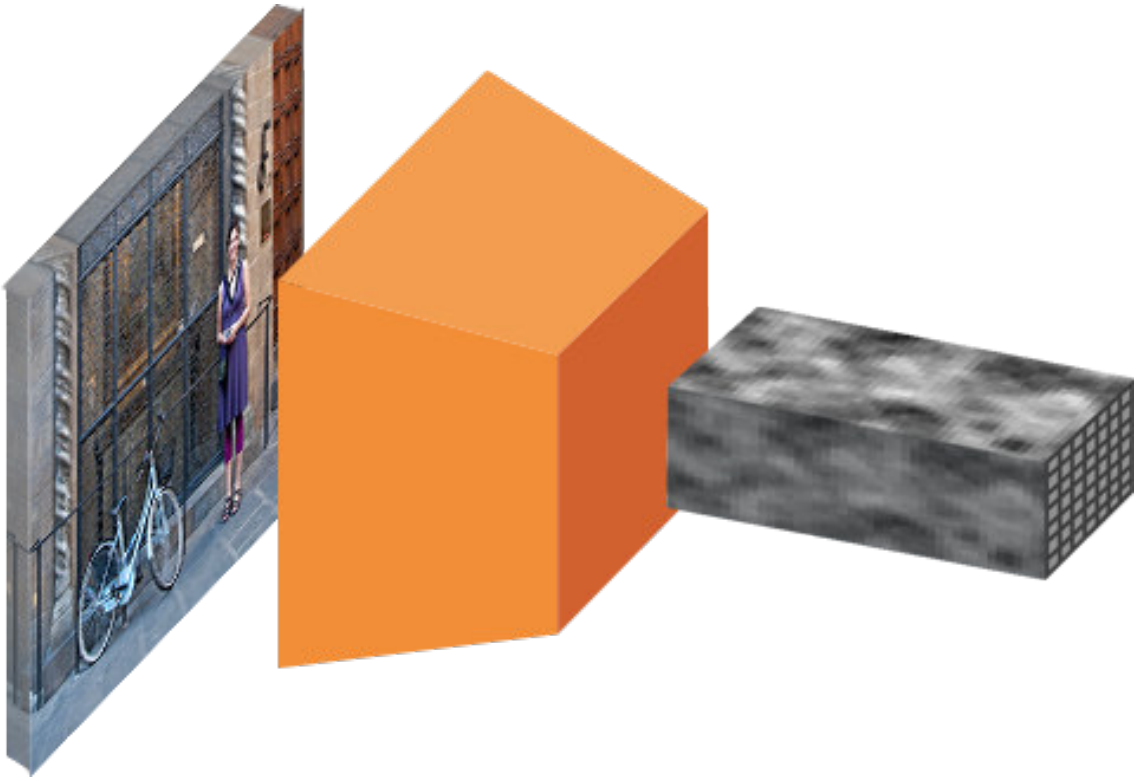


FIGURE 3.7: Image to convolutional feature map.

By today's standards VGG would not be considered very deep, ResNet architectures have mostly replaced VGG as a base network for extracting features.
ResNet is bigger than VGG, it has more capacity to actually learn what is needed. Also, ResNet uses residual connections and batch normalization wich makes it easy to train deep models.

3.3.3 Region Proposal Network

The RPN uses a convolutional layer with 512 channels and 3x3 kernel size, then, two convolutional layers using a 1x1 kernel with a number of channels depends on the number of anchors per point.

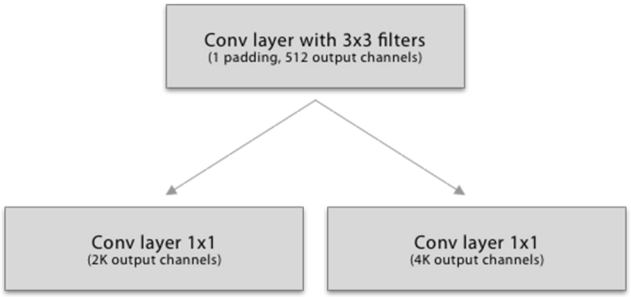


FIGURE 3.8: Convolutional implementation of an RPN architecture, where k is the number of anchors.

It produces as outputs a set of good proposals for objects based on it inputs which are the previous anchors, for each anchor it have 2 outputs :

- The probability of an anchor is an object.
- The bounding box regression.

In classification layer, we have two output predictions per anchor :

- The score of not being an object.
- The score of being an object.

For the adjustment of bounding box layer (Regression layer), we have 4 output predictions which we will apply to the anchors to get the final proposals : $\Delta_{x_{center}}, \Delta_{y_{center}}, \Delta_{width}, \Delta_{height}$.

3.3.3.1 Anchors

Anchors are fixed bounding boxes that are placed throughout the image with different sizes and ratios that are going to be used for reference when first predicting object locations, they are defined based on the convolutional feature map, the final anchors reference the original image. A set of anchors will be created for each of the points in conv width x conv height to end up with a bunch of anchors separated by r pixels. In the case of VGG, r = 16.

FIGURE 3.9: Anchor centers throught the original image [19].

3.3.3.2 Training, target and loss functions

Now, we know that RPN have two type of predictions, which are the binary classification and the bounding box regression adjustment. So for training the RPN, we need to categorize all the anchors into two types :

- Foreground, which are the anchors that overlap a ground-truth object with an Intersection over Union (IoU) bigger than 0.5.
- Background, the anchors that don't overlap any ground truth object or have less than 0.1 IoU with ground-truth objects.

After that, we sample those anchors randomly to form a mini batch of size 256.
To calculate the classification loss, all the selected anchors are needed, the classification loss is calculated using binary cross entropy. For the regression loss needs only the foreground mini batch anchors to be calculated. Now for calculating the targets for the regression, we use the foreground anchor and the closest ground truth object and calculate the correct Δ (Delta) needed to transform the anchor into the object [18].

3.3.3.3 Post processing

- Non-maximum suppression comes to solve the duplicate proposals by taking the list of proposals sorted by score, iterategs over the sorted list, then discarding

proposals with an IoU bigger than predefined threshold with a proposal that has a higher score, the commonly used value is 0.6.

- Proposal selection, After applying NMS, we keep the top N proposals sorted by score. In the paper N = 2000 is used, but it is possible to lower that number to as little as 50 and still get quite good results.

3.3.4 Region of Interest Pooling

The output of RPN is a bunch of bounding boxes with no class assigned to them. So we have classify these object proposal into categories.
Faster R-CNN reuse existing convolutional feature map in this process, by extracting fixed-sized feature maps for each proposal using region of interest pooling, wich is needed for the R-CNN to classify them into a fixed number of classes [20].

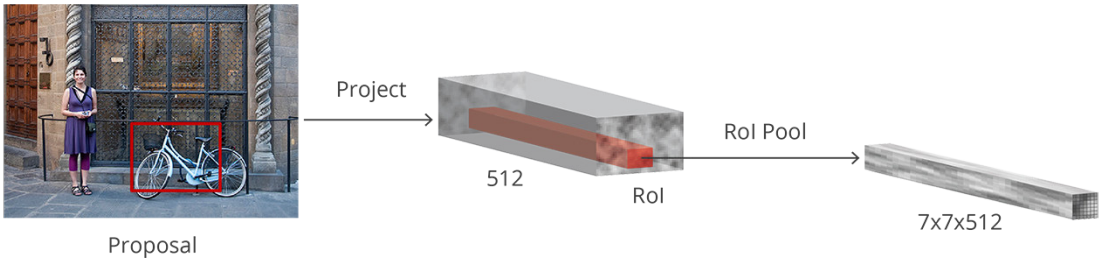


FIGURE 3.10: Region of Interest Pooling [19].

3.3.5 Region-based Convolutional Neural Network

The final step of Faster R-CNN's pipeline is Region-based convolutional neural network (R-CNN)
In this step we need to use the extracted features from the previous layer for classification. R-CNN tries to mimic the final stages of classification CNNs where a fully-connected layer is used to output a score for each possible object class.
R-CNN has two different goals:

Classify proposals into one of the classes, plus a background class (for removing bad proposals).Better adjust the bounding box for the proposal according to the predicted class.

R-CNN takes the feature map for each proposal, flattens it and uses two fully-connected layers of size 4096 with ReLU activation.
Then, R-CNN uses two different fully-connected layers for each different object :

- Classify The fist layer with N+1 units, where N is the total number of classes and 1 is for the background class.
- The second layer with 4N units for each of the N possible classes, which are regression prediction, $\Delta_{center_x}, \Delta_{center_y}, \Delta_{width}, \Delta_{height}$.

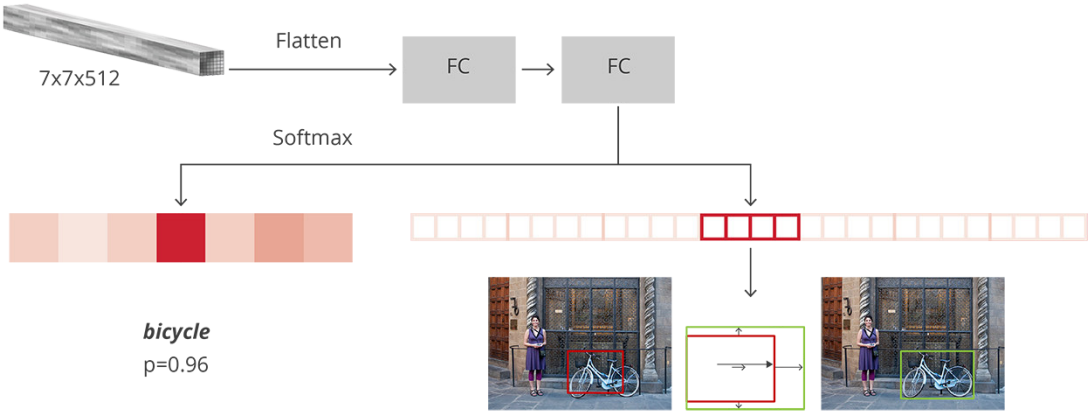


FIGURE 3.11: R-CNN architecture [21].

3.4 Conclusion

In this chapter we described Faster R-CNN, which it proved it capability to solve complex computer vision problems, which make is a canonical model for deep learning-based object detection. Faster R-CNN is state of the art object detection networks depend on region proposal algorithms to hypothesize object locations. In this work, we introduce a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals.

Chapter 4

Implementations

4.1 Introduction

This chapter gives the general design of our application. We gives first the description and the objectives of the project, and than the used materials and softwares. Finally, we present the achieved results and discussion.

4.2 Project description and objectives

Our objective of this project is to implement a vehicle registration plate recognition system, which has accurate results in character recognition for a single image, in real-time, and searching for existed license plate using cameras.

4.3 Concept

The proposed algorithm has been implemented for recognition of registration plate characters after the processing of captured image and plate detection.

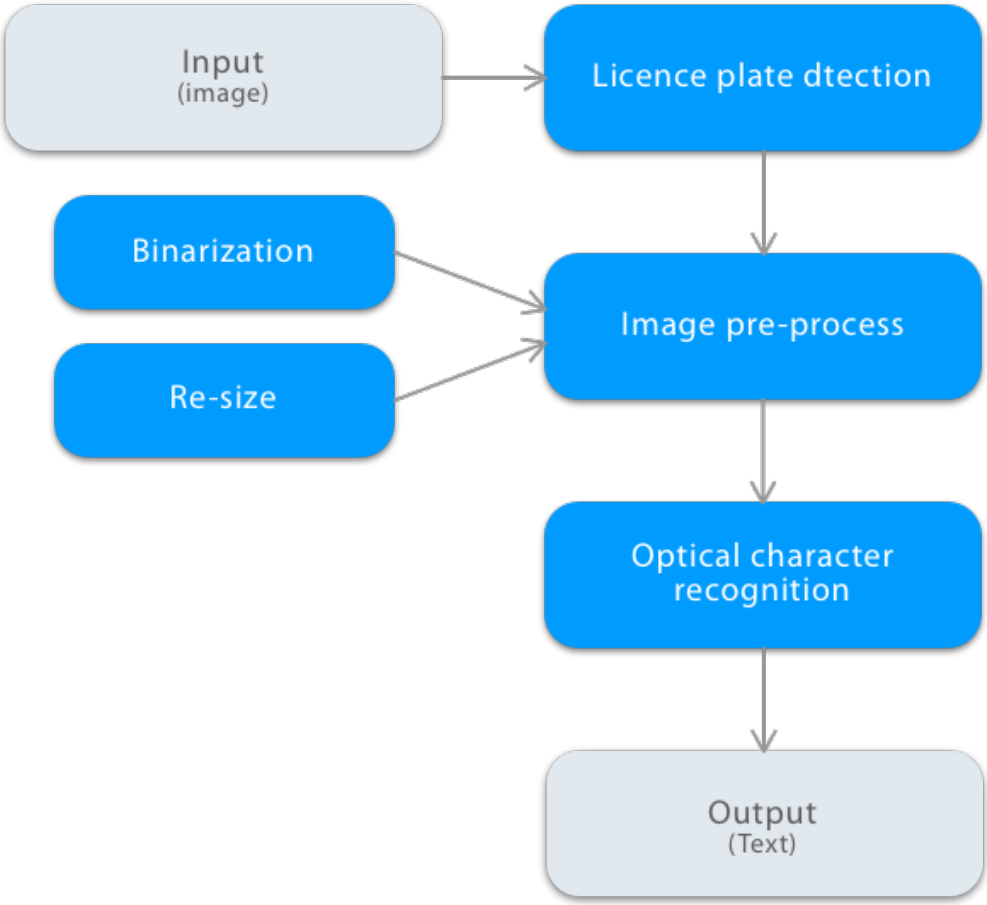


FIGURE 4.1: General design.

4.3.1 Licence plate detection

This step is for locating the number plate in the captured image. Once the number plate is located in the image, we save a new image contains only the number plate. The goal of this step is to make sure that our system will not recognize any characters in the input image, but only those of the licence plate.

As we said earlier, we use Faster R-CNN for licence plate detection. Faster R-CNN takes an image as input and runs it through 2 modules : the first module uses a RPN that proposes object regions and the second is a Fast R-CNN object detector that classifies the region proposals

After getting the output feature map from the pre-trained model which is VGG-16, our input image has 800x600x3 dimensions (Width,Height,RGB), the output feature map would be 50x37x256 dimensions.

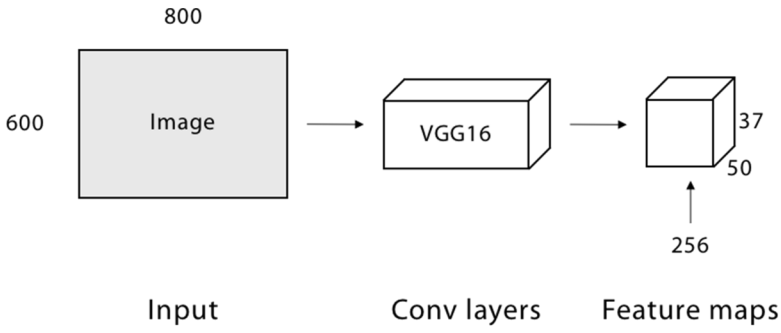


FIGURE 4.2: First step of FasterR-CNN.

Each point in 50x37 is considered as an anchor. We need to define specific ratios and sizes for each anchor (1:1, 1:2, 2:1) for three ratios and 128^2, 256^2, 512^2 for three sizes in the original image.

After that, the RPN is connected to a Conv layer with 3x3 filters, with 1 padding and 512 output channels. The output is connected to two 1x1 convolutional layer for determine if the box is an object or not and for the box-regression

In our case, each anchor has 9 boxes (3x3), in total, we have 16650 boxes (50x37x9) in the original image. We choose only 256 of these boxes as a mini batch, a 128 of them are positive which represents foregrounds and the other 128 are negative which represents backgrounds. To avoid overlapping (duplicate proposals), we applied non-maximum suppression. Now, we are done with RPN.

In the second stage of Faster R-CNN we will use a ROI pooling for the proposed regions. The output is 7x7x512. Then, we flatten this layer with some fully connected layers. The final step is a softmax function for classification and linear regression to fix the boxes location.

Finally we will use the bounding box coordinates (the coordinates of the box that bound the object detected) to crop the input image if an object is detected. The output is in image that contains only a licence plate.

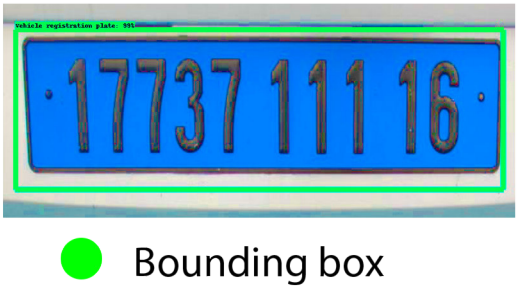


FIGURE 4.3: An example of bounding box

4.3.1.1 Pre-process image

For reason to have more accurate results some image process proposed

4.3.1.1.1 Binarization

High quality binarized image give us more accuracy in character recognition as compared original image because noise is present in the original image, so we used a binirization method to convert a licence plate image from grey scale (0 to 256 gray levels) in to black and white image (0 or 1) (see Figure 4.4).

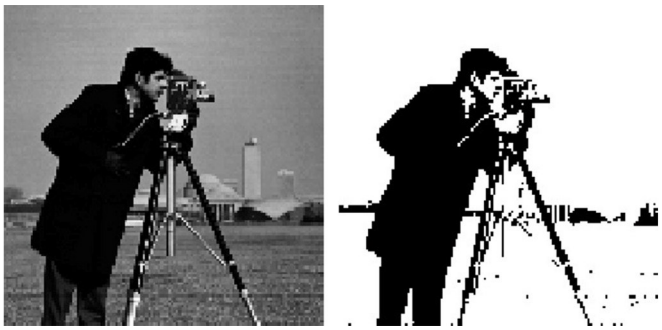


FIGURE 4.4: An example of binarization

The method of binarization we choose is Otsu's thresholding method. Otsu Method used for automatic binarization level decision, based on the shape of the histogram, involving iterating through all the possible threshold values, and calculating a measure of spread for the pixel levels each side of the threshold [22].
Otsu's thresholding method divide in 4 steps :

- In the first step, we separate the pixels into two clusters (q1,q2) according to the threshold.

$$q1(t) = \sum_{i=1}^{t} p(i)$$

$$q2(t) = \sum_{i=t+1}^{I} p(i)$$

where p represents the image histogram.

- In the second step, we will find the mean of each cluster.

$$m1(t) = \sum_{j=1}^{t} \frac{ip(i)}{q1(t)}$$

$$m2(t) = \sum_{j=t+1}^{I} \frac{ip(i)}{q2(t)}$$

- In the third we will calculate the individual class variance.

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - m1(t)]^2 \frac{p(i)}{q1(t)}$$

$$\sigma_2^2(t) = \sum_{i=t+1}^{I} [i - m2(t)]^2 \frac{p(i)}{q2(t)}$$

- In the fourth step we will square the difference between the means.

$$\sigma_b^2(t) = \sigma^2 - \sigma_w^2(t) = q1(t)[1 - q1(t)][m1(t) - m2(t)]^2$$

This expression can safely be maximized and the solution is t that is maximizing $\sigma_b^2(t)$.

4.3.1.1.2 Re-size

Some licence plate have too many characters, what make space between characters too small, which make the characters recognition process difficult. To solve this problem we add some width to the image that contain the registration plate by changing fx from 1 to 1.5, where fx is the width of the image, in goal to give more space between characters and making characters recognition more accurate (see Figure 4.5).

Original image Re-sized image

FIGURE 4.5: An example of image re-size.

4.3.1.2 Optical character recognition

In this step, we will extract the text form the previous image we worked on using Tesseract OCR (see Figure 4.6).

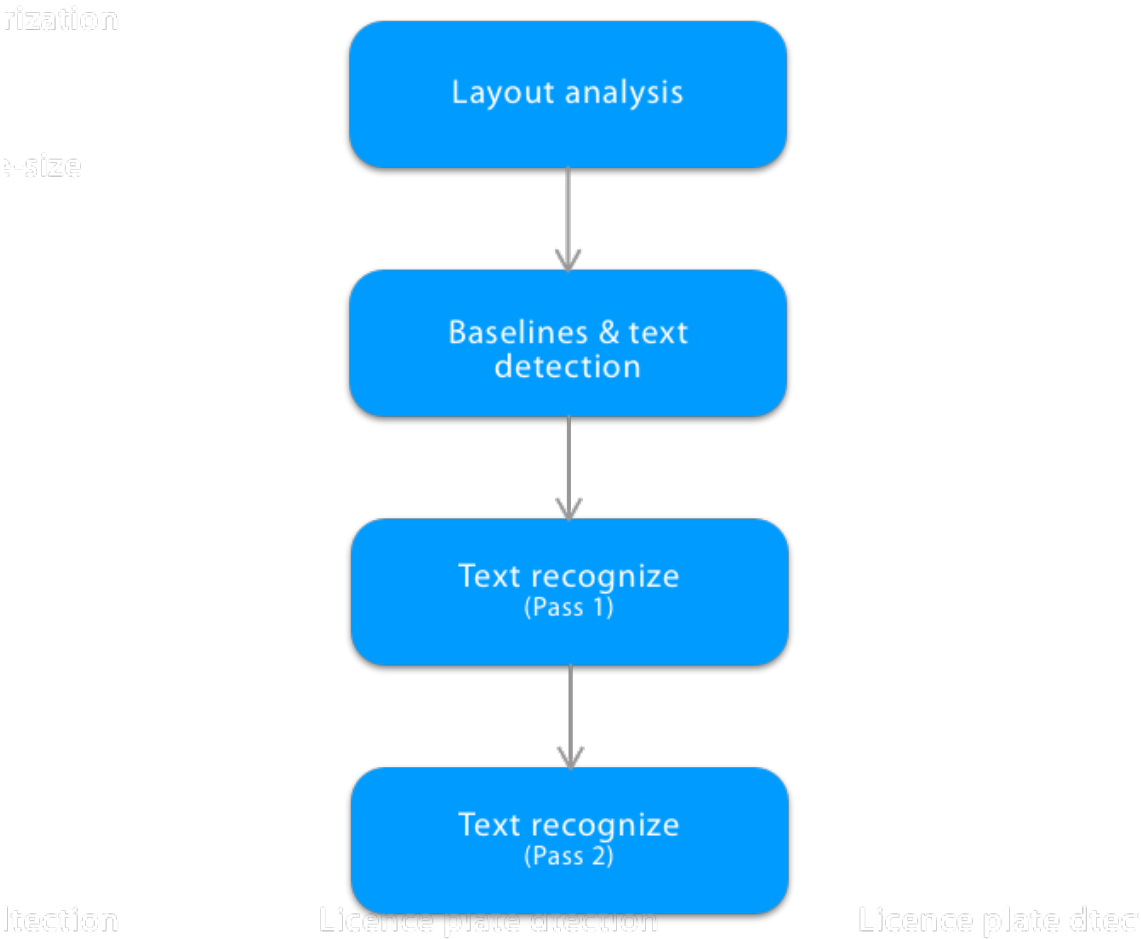


FIGURE 4.6: Architecture of our OCR engine.

4.3.1.2.1 Layout analysis

In this first step, we will divide the input image (the output of the previous step) into two types of areas, areas which contains text, and areas without text, using the following steps [2] :

- Vertical lines detection, using the morphological processing from Leptonica. where these elements are removed from the input image before passing the cleaned image to connected component analysis.
- The candidate tab-stop connected components that look like they may be at the edge of a text region are found and then grouped into tab-stop lines.
- Scanning connected components from left to right and top to bottom.

4.3.1.2.2 Baseline and text detection

Line Finding We used a line finding algorithm from Tesseract. The key parts of this algorithm are blob filtering and line construction. It has the following steps :

- Removing the drop caps vertically touching characters by the application of a percentile height filter.
- The median height approximates the text size in the region, so it is safe to filter out blobs that are smaller than some fraction of the median height, being most likely punctuation, diacritical marks and noise.
- The filtered blobs are more likely to fit a model of non-overlapping, parallel, but sloping lines. Sorting and processing the blobs by x-coordinate makes it possible to assign blobs to a unique text line, while tracking the slope across the page, with greatly reduced danger of assigning to an incorrect text line in the presence of skew. Once the filtered blobs have been assigned to lines, a least median of squares fit [4] is used to estimate the baselines, and the filtered-out blobs are fitted back into the appropriate lines.
- In the last step of the line creation process, the merges blobs that overlap by at least half horizontally, putting diacritical marks together with the correct base and correctly associating parts of some broken characters.

Baseline Fitting : After Finding the lines, we use a quadratic spline to fit the baseline by partitioning the blobs into groups with a reasonably continuous displacement for the original straight baseline (see Figure 4.7).

FIGURE 4.7: An example of baseline detection

Fixed pitch detection and chopping : To determine whether the text lines are fixed pitch, Tesseract test them. Where it finds fixed pitch text, Tesseract chops the text into characters using the pitch, and disables the chopper and associator for the word recognition step (see Figure 4.8).

FIGURE 4.8: An example of fixed pitch

4.3.1.2.3 Classification

The classification proceeds as a two-step process :

- First step involves a class pruner that creates a shortlist of character classes that the unknown might match.
- The classes shortlisted in step one are taken further to the next step, where the actual similarity is calculated from the feature bit vectors. Each prototype character class is represented by a logical sum-of-product expression with each term called a configuration.

4.3.1.2.4 Training

We used a pre-trained data, which the classifier was trained in total on more than 60k samples, using different fonts and attributes (normal, italic, bold and bold italic), for the size we use the same size for all characters.

4.4 The graphical user interface

The application allows the users to choose between 3 operations. Searching for vehicle, streaming road cameras or extracting licence plate from an image.

We provide a simple interface for the user, starting from the home page wich allow user to navigate between the application operations (see Figure 4.9).

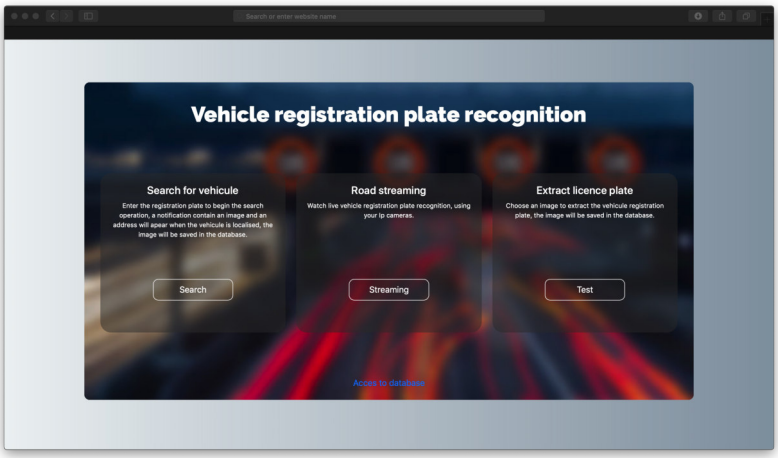


FIGURE 4.9: Home page.

Jumping up to the search for vehicle page, in which the user can type the licence plate number to begin the searching operation (see Figure 4.10).

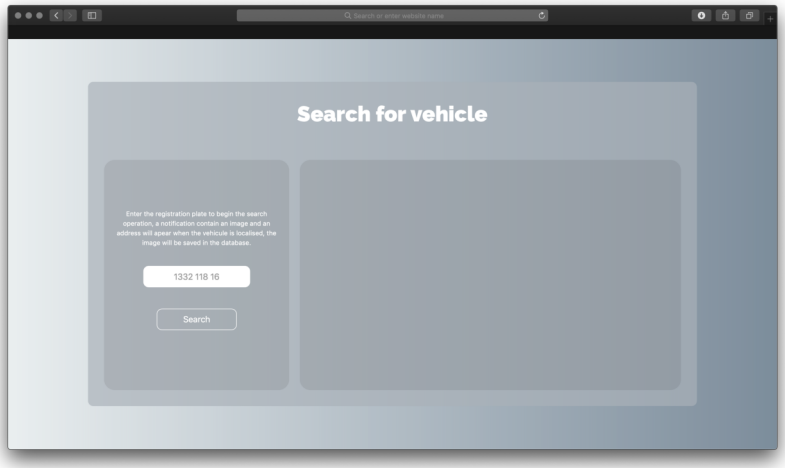


FIGURE 4.10: Search for vehicle page.

The next figure (see Figure 4.11) represent test page, which give the user the possibility of testing the applications on chosen images.

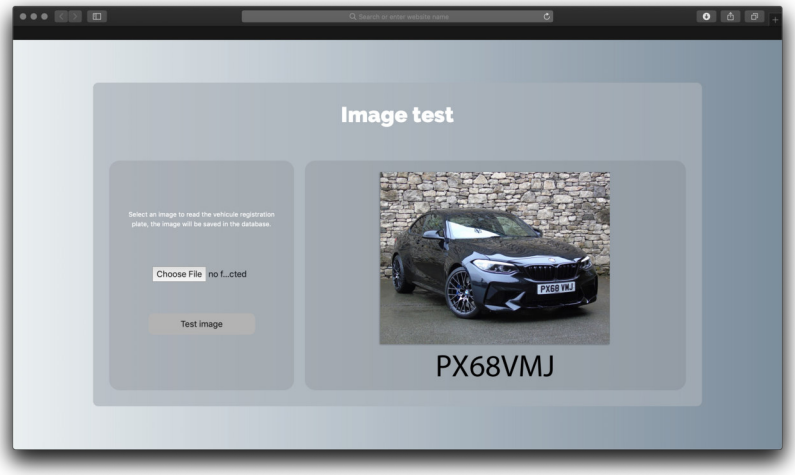


FIGURE 4.11: Extracting licence plate from an image.

The forth page (see Figure 4.12) represent road streaming page, which shows all connected cameras and allow the user to choose a one these camera to start a real time number plate recognition using that camera.

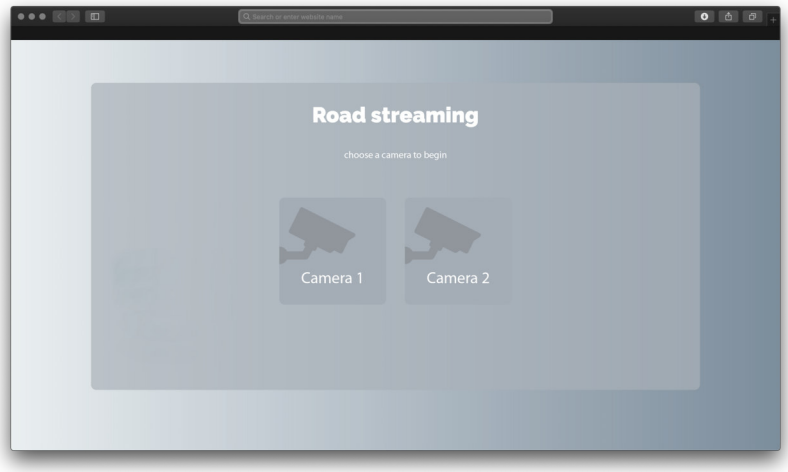


FIGURE 4.12: Road streaming page.

4.4.1 Used Softwares and Materials

4.4.1.1 Materials

MacBook Pro (Retina, 13-inch, mid 2014).
Processor : 2.6 GHz Intel Core i5.
Memory : 8 GB 1600 MHz DDR3.
Graphics : Intel Iris 1536 MB.

4.4.1.2 Softwares

Operating System : macOS Mojave version 10.14.2.
Programming language : Python.
API : Tensorflow, Keras, Tesseract, Leptonica.

4.5 Limitations

- Our application have three types of use. One of them is real time licence plate recognition, which use a real time object detection and optical characters recognition, due to hardware limitation, we will have a low frame per second.
- Licence plate have a lot of types, due to countries or states regulations. Several models have additional text, which makes our application not capable of separating between the licence plate number and the extra text.

4.6 Conclusion

In this chapter, we did an implementation of an Vehicle Registration Plate Recognition application, using the original Faster R-CNN with our custom dataset for the detection of registration plate. For characters recognition we use a pre-trained data from Tesseract. After few tests, we found that the image pre-process (Otsu's thresholding method and the image re-size) make a huge deference in results.

Conclusion

In this project, we have presented the foundation of an Vehicle Registration Plate Recognition application using deep learning approaches in two part, for licence plate detection and for characters recognition. We use also in this project two image process, which are image binarization and image re-sizing, these process helps a lot in characters recognition, what makes our application more accurate. Our proposed solution works in general cases, where the distance from camera to the vehicle is reasonable and weather conditions are good. For the future work, we propose to use a Graphics Processor Unit (GPU) instead of only the Central Processing Unit (CPU), what makes our application run smoothly and give us more frames per second. As well as the image pre-process we can add a de-skew algorithm which helps in case where the number plate are skewed.

References

[1] Prashant Manoharrao Kakde Raamesh Gowri Raghavan and Sukant Khurana. *Applications of OCR You Haven't Thought Of*. Accessed: 2019-04-13. 2018. URL: https://medium.com/swlh/applications-of-ocr-you-havent-thought-of-69a6a559874b.

[2] Arindam Chaudhuri et al. "Optical character recognition systems". In: *Optical Character Recognition Systems for Different Languages with Soft Computing*. Springer, 2017, pp. 9–41.

[3] Salem Saleh Al-Amri, NV Kalyankar, and SD Khamitkar. "Image segmentation by using edge detection". In: *International journal on computer science and engineering* 2.3 (2010), pp. 804–807.

[4] Poonam Dhankhar and Neha Sahu. "A review and research of edge detection techniques for image segmentation". In: *International Journal of Computer Science and Mobile Computing* 2.7 (2013), pp. 86–92.

[5] Jason Yosinski et al. "Understanding neural networks through deep visualization". In: *arXiv preprint arXiv:1506.06579* (2015).

[6] Josh Patterson and Adam Gibson. *Deep learning: A practitioner's approach*. " O'Reilly Media, Inc.", 2017.

[7] *Introduction to Neural Networks*. Accessed: 2019-04-02. 2016. URL: https://www.neuraldump.net/2016/03/introduction-to-neural-networks/.

[8] Imad A Basheer and Maha Hajmeer. "Artificial neural networks: fundamentals, computing, design, and application". In: *Journal of microbiological methods* 43.1 (2000), pp. 3–31.

[9] Md Zahangir Alom et al. "A State-of-the-Art Survey on Deep Learning Theory and Architectures". In: *Electronics* 8.3 (2019), p. 292.

[10] Sundaravelpandian Singaravel, Johan Suykens, and Philipp Geyer. "Deep-learning neural-network architectures and methods: Using component-based models in building-design energy prediction". In: *Advanced Engineering Informatics* 38 (2018), pp. 81–90.

[11] Christian Szegedy, Dumitru Erhan, and Alexander Toshkov Toshev. *Object detection using deep neural networks*. US Patent 9,275,308. 2016.

[12] Ross Girshick Jian Sun Shaoqing Ren Kaiming He. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. Accessed: 2019-05-30. 2015. URL: https://arxiv.org/abs/1506.01497.

[13] Ross Girshick Ali Farhadi Joseph Redmon Santosh Divvala. *You Only Look Once: Unified, Real-Time Object Detection*. Accessed: 2019-06-05. 2016. URL: https://arxiv.org/abs/1506.02640.

[14] Geoffrey E. Hinton Alex Krizhevsky Ilya Sutskever. "ImageNet Classification with Deep Convolutional Neural Networks". In: 1.1 (2012).

[15] Rikiya Yamashita et al. "Convolutional neural networks: an overview". In: *Insights into imaging* 9.4 (2018), pp. 611–629.

[16] Arden Dertat. *Applied Deep Learning - Part 4: Convolutional Neural Networks*. Accessed: 2019-06-18. 2017. URL: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2.

[17] Jasper RR Uijlings et al. "Selective search for object recognition". In: *International journal of computer vision* 104.2 (2013), pp. 154–171.

[18] Jan Hosang et al. "What makes for effective detection proposals?" In: *IEEE transactions on pattern analysis and machine intelligence* 38.4 (2015), pp. 814–830.

[19] AKA Vierja Javier Rey. *Faster R-CNN: Down the rabbit hole of modern object detection*. Accessed: 2019-06-03. 2018. URL: https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/.

[20] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[21] Yinghan Xu. *Faster R-CNN (object detection) implemented by Keras for custom data from Google's Open Images Dataset V4*. Accessed: 2019-05-28. 2018. URL: https://towardsdatascience.com/faster-r-cnn-object-detection-implemented-by-keras-for-custom-data-from-googles-open-images-125f62b9141a.

[22] BATHINDA Puneet GRDIET. "Binarization Techniques used for Grey Scale Images". In: *International Journal of Computer Applications (0975 – 8887)* 71.1 (2013), pp. 3–11.

Dedication

i must express my very profound gratitude to my mother, my wife, my brothers and sisters, my best friends and specially, my deceased father *for providing me with unfailing support and continuous encouragement throughout my years of study.*
This accomplishment would not have been possible without them.
Thank you. . .

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Kheidher – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre : Iva 3/M2/2019

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Images et vie artificielle**

Simulation comportementale des agents virtuels autonomes dans un environnement 3D

Par :

LHADJ AMEL

Soutenu le 07 juillet 2019, devant le jury composé de :

Mr Zerari Abd El Moumén	**MCB**	Président
Mr Boucetta Mebarek	**MAA**	Rapporteur
Mme.Babahenini Djihenne	**MAA**	Examinateur

Remerciement

Je tiens à remercier d'abord notre dieu qui nous a donné la force et le courage pour continuer et nous avoir aidés et éclairer le chemin pour réaliser ce travail.

Mes remerciement ma gratitude et reconnaissance à mon encadreur Mr :

Boucetta Mebarek pour son soutien et son aide précieuse pour la réalisation de ce travail.

Mes remerciement vont également aux membres de jury, Mr **ZERARI Abd El Moumén** et Mm **BABAHENINI Djihanne** pour m'avoir honoré par leur participation à l'évaluation de ce travail.

Un grand merci à mes parents qui mon encouragé et aidés tout au long de mes études.

En fin, j'adresse mes remerciements à tous les enseignants qui ont contribués à ma formation durant toutes mes études.

LHADJ Amel

Résumé

La simulation comportementale des agents virtuels est un domaine de recherche très vaste, inclus dans nombreux domaines, telque le domaine de cinéma , jeux vidéo, architecture, modélisation urbaine et d' autres domaines. Peupler ces environnements virtuels avec des agents est une tâche difficile, et pose des véritables problèmes tels que le rendu en temps réel et la simulation des comportements. Une simulation pour de tels environnements est difficile à cause de compromis qui doit être assuré entre le réalisme du rendu et la simulation en temps réel. Notre objectif principal est de concevoir un système capable de simuler des comportements individuels et quelques comportements de groupe sur des agents virtuels. Nous présentons dans ce travail un modèle de planification de chemin des agents en temps réel. Notre travail consiste en l'utilisation de la méthode de représentation basé sur la triangulation de Delaunay. Et un algorithme de planification A* qui permet à l'agent de trouver un chemin vers son but. Nous avons aussi utilisé les forces de directions de model de Reynold. Pour simuler le comportement des individus et de groupe. Notre simulation est réalisée en utilisant **OGRE 3D** et **QT_CREATOR** pour construire l'interface graphique qui va montrer toutes les taches de notre système.

Mots clés : simulation comportementale, environnement virtuel, comportement, agent virtuel, navigation, planification.

Abstract

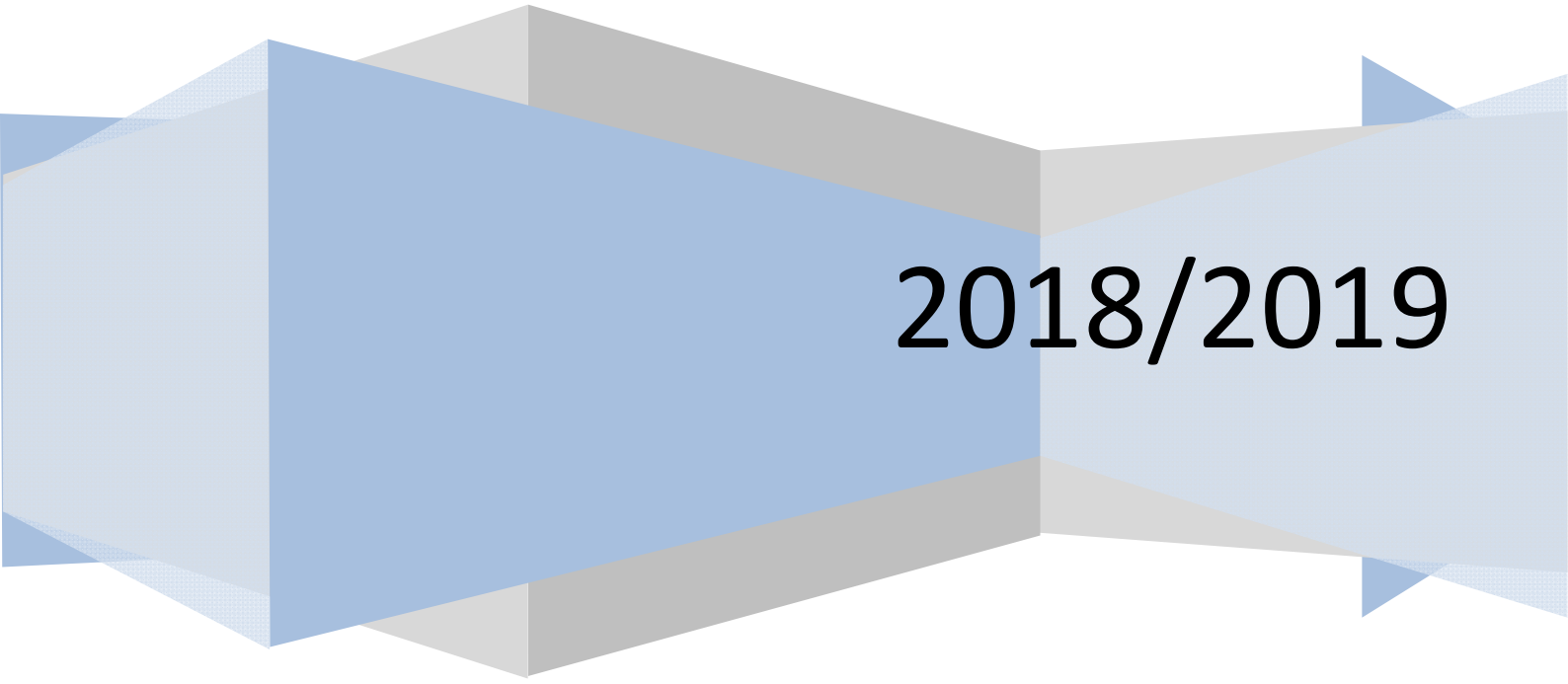
Behavioral simulation of virtual agents is a very broad area of research, encompassing many fields, such as the field of cinema, video games, architecture, urban modeling and other fields. Populating these virtual environments with agents is a difficult task, and poses real problems such as rendering in real time and simulating behaviors. A simulation for such environments is difficult because of the compromise that must be ensured between realistic rendering and real-time simulation. Our main goal is to design a system that can simulate individual behaviors and some group behaviors on virtual agents. In this work, we present a real-time agent path planning model. Our work consists in using Delaunay's method of representation based on triangulation. And an A * scheduling algorithm that allows the agent to find a path to his goal. We also used the Reynolds model directions forces. To simulate the behavior

of individuals and groups. Our simulation is performed using OGRE 3D and QT_CREATOR to build the GUI that will show all the tasks of our system.

Keywords: behavioral simulation, virtual environment, behavior, virtual agent, navigation.

Université Mohamed kheidher Biskra

Département de l'informatique



2018/2019

I. Introduction Générale

Les mondes virtuels ont rapidement pris le devant de la technologie, tant dans l'industrie que dans le monde universitaire, ils sont utilisés dans nombreux domaines allant de l'éducation, la sécurité, la conception urbaine, la création de contenu cinématographique et le divertissement interactif. Un aspect clé de peuplement des environnements virtuels est l'utilisation d'humains virtuels autonomes pour injecter de la vie dans ces derniers. Cela a conduit à mettre fortement l'accent sur l'amélioration de la crédibilité et de l'intelligence de ces agents. Ce peuplement comme il a des avantages il pose un défi qui est la simulation en temps réel des agents virtuels autonomes et fonctionnels dans des environnements complexes et dynamiques, pour des centaines, avoir des milliers d'agents. [1] .Les gens se sont rassemblés depuis longtemps collectivement pour observer, célébrer ou protester contre divers événements. Le collectif assemblées ou rassemblements appelés foules. Avec les ordinateurs, il devient possible non seulement d'observer des foules humaines dans le monde réel, mais aussi pour simuler divers phénomènes de différents domaine et du simuler des comportements collectif dans les environnements virtuels. Pour cela il faut parler sur la planification de chemin, ou cette dernière consiste à déterminer une trajectoire, en générale la meilleure, que doit suivre un agent virtuel pour aller d'un point à un autre dans un environnement bien présenté en terme de la topologie.

L'animation Est une technique qui sert à donner l'aspect dynamique à une entité dans un environnement 3D cette entité peut être un objet (table, armoire), primitive géométrique (point, sphère, polygone) ou bien un humain virtuel. L'animation réaliste du comportement d'agents virtuels imitant l'être humain et évoluant au sein de mondes virtuels en 3D est l'une des tâches les plus difficiles en infographie. [2]

La problématique :

La simulation dans l'environnement virtuel est totalement basé sur une bonne représentation et une bonne planification de chemin alors comment on va planifier le chemin dans notre environnement afin de faciliter le déplacement pour les agents ?

Dans ce mémoire on va s'intéresser à l'animation comportementale des humains virtuels autrement dis la simulation comportementale des entités virtuels dans l'environnement virtuel ces comportement sont simple à base de règles. On va aussi représenter l'environnement afin que les agents puissent se déplacer et faire des comportements.

En fin, la structure de ce mémoire est comme suit :

. Il y'aura quatre chapitres :

- Un Etat de l'art qui est sous forme de deux chapitres (chapitre1, chapitre2).
  - Le premier chapitre va parler sur l'animation comportementale.
  - Le deuxième chapitre va parler sur les environnements virtuels et les méthodes de Représentations qui peuvent bien représenter un environnement.
  - Le troisième chapitre va avoir une conception globale et détaillée du projet.
  - Le dernier chapitre va contenir les résultats Obtenus**.**

Chapitre 1

« L’animation comportementale »

1. Introduction

Dans ce chapitre, le sujet principale sera l'animation comportementale des agents virtuel, ou l'agent est un élément nécessaire pour faire une simulation comportementale, L'animation des comportements est un sujet très importants dans le secteur de recherche scientifique on le trouve presque dans différents domaine

2. Animation comportementale

L'animation comportementale sert à modéliser les comportements des entités qui peuplent le monde virtuel. Ces entités ont des caractéristiques communes comme la capacité de percevoir le monde et un certain niveau d'autonomie.

Toutes les approches proposées par l'animation comportementale ont comme objectif de sélectionner le meilleur comportement ou action à réaliser dans certaine situations.

L'animation comportementale est un type d'animation procédurale, qui est un type d'animation par ordinateur. Dans l'animation comportementale, l'agents est autonome détermine ses propres actions son aucune aide extérieur.

La boucle de l'animation comportementale

L'animation comportementale prend comme base la boucle **Perception-D´ecision-Action** : une entité perçoit l'environnement, décide de la prochaine action à exécuter et agit sur le monde.

3.1. Perception

Nécessite des modèles de perception, Détermine les réactions de l'agent autonome vis-à-vis de son environnement.

2.2. Décision

Cette phase consiste à construire l'autonomie de l'humain virtuel en lui apprenant des règles d'apprentissages qui va lui servir pour l'étape de l'action. Le comportement finale de l'entité est n'est pas prédéfinit.

2.3. Action

Agir sur l'environnement ou sur les autres acteurs il s'agit de faire un contrôle sur les actions de l'entité .

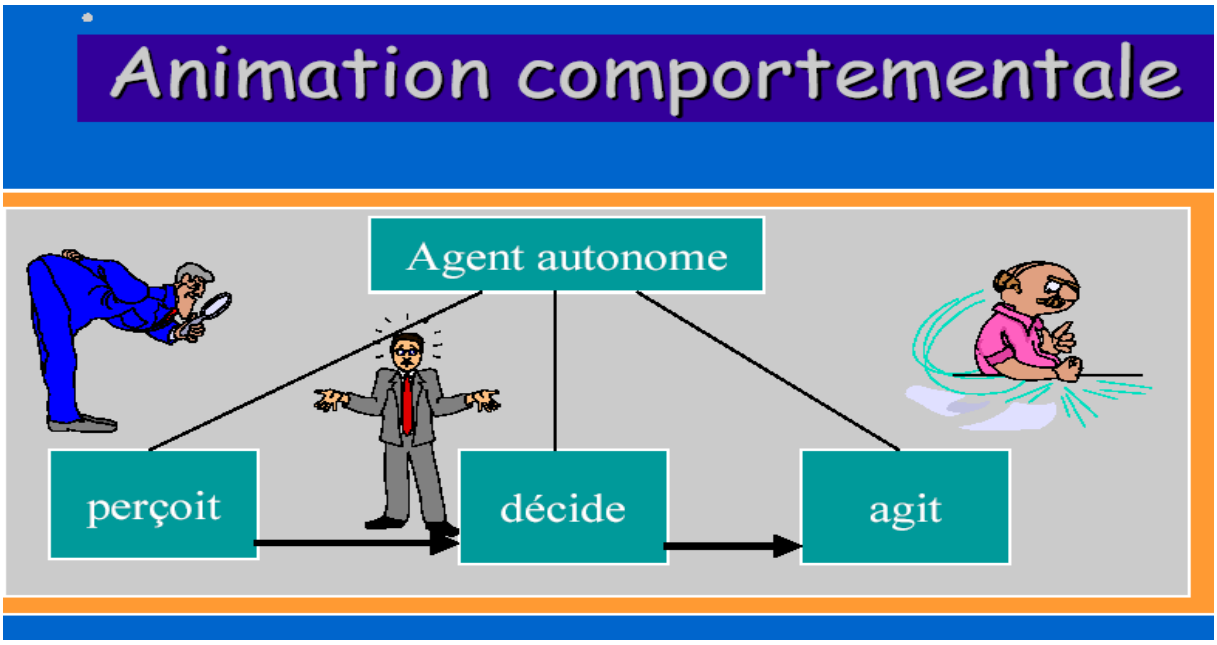


Figure 1.1 : la boucle de l'animation comportementale.

3. Agent virtuel

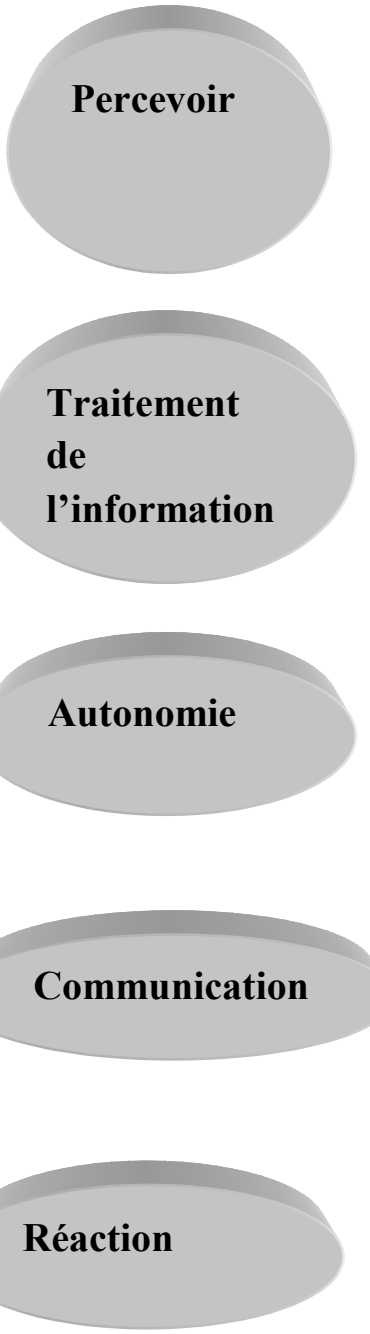
Est une entité dans la quelle on va appliquer des comportements pour qu'elle soit autonome. Ces comportement peut être simple (marcher, tourner, courir) ou bien complexe (éviter un ou des obstacles) ses obstacles peuvent être statiques ou dynamiques)ici on pourra parler aussi sur les comportements de Reynold (regroupement, séparation et alignement).

Le terme agent virtuel signifie une entité avec des buts et des intentions, qui apportent des changements dans le monde. Les agents virtuels sont capables de percevoir, de décider et d'agir par eux-mêmes dans un certain cadre qui leur est imposé.

Le terme agent autonome fait généralement référence à une entité qui fait ses propres choix quant à la manière d'agir dans son environnement sans aucune influence d'un dirigeant ou d'un plan global. *Les agents Virtuels sont des entités logicielles qui effectuent un ensemble d'opérations pour le compte d'un utilisateur ou d'un autre programme avec un certain degré d'indépendance ou d'autonomie. [3]

4. Caractéristique d'un Agent Virtuel

L'agent virtuel à des caractéristiques un ensemble de caractéristiques résumé comme suit :



Un agent virtuel a une capacité limitée à percevoir l'environnement. Il est logique qu'un être vivant qui respire doive avoir une conscience de son environnement.

Un agent virtuel traite l'information de son environnement et fait une action.

l'agent fait l'action sans aucune aide extérieur (programmeur).

L'interaction passe par un langage, compréhensible et la communication est réciproque entre les agents.

L'agent intelligent doit pouvoir observer son "environnement" et déterminer si une situation donnée se présente ou pas pour ensuite y réagir par l'action appropriée.

6. Types d'agents virtuels

6.1. Agents réactifs

Les agents réactifs sont connus par des comportements stimuli-action, c'est-à-dire, ses actions sont choisies et exécutées selon les événements perçus dans l'environnement, les

agents de ce type peuvent ne pas être intelligents de manière individuelle, mais produisent des comportements intelligents en collectivité. [5]

6.2 . Agents cognitifs

Les agents cognitifs sont capables de coopérer avec l'autre afin de résoudre un problème. Pour cela ils maintiennent une représentation commune de l'environnement dont ils font partie. Contrairement aux agents réactifs, les agents cognitifs sont plus autonomes et possèdent un niveau de connaissance plus élevé [5]

6.3 Agents hybrides

Il est de plus possible de combiner les deux approches précédentes pour obtenir une architecture hybride.

Dans une telle architecture, un agent est composé de modules qui gèrent indépendamment la partie réactive et la partie cognitive du comportement de l'agent. [6][7]

7. La simulation du comportement

Pour Avoir un réalisme parfait, il faut que l'environnement soit représenté d'une manière cohérente, en plus, Les agents virtuels doivent faire des taches telles que la planification de chemin et la navigation dans l'espace, pour cela, les agents doivent avoir des comportements réels. Un modèle de Reynolds est proposé pour faire ces comportements, ce modèle à modéliser plusieurs comportement mais sur des oiseaux.

Dans cette partie, on va parler sur la modélisation de comportement individuel pour les agents ou chaque agent va avoir son propre comportement puis, on parlera sur le comportement de groupes ou comportement émergents.

Le model de Reynold a utilisé trois comportement de base pour ensuite créer de nouveau comportements :

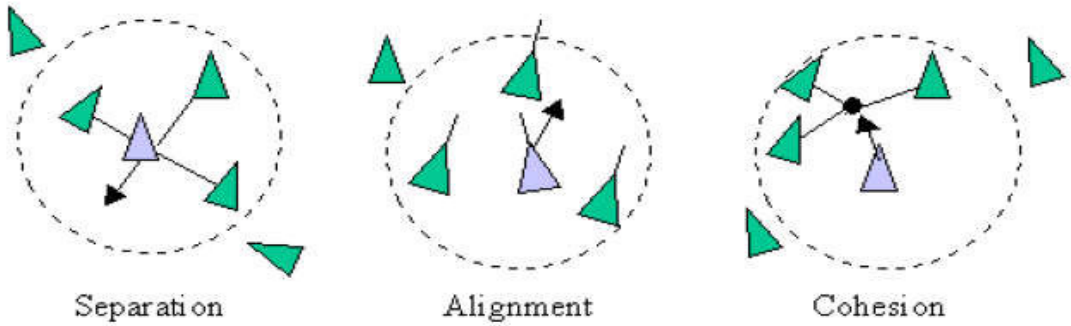


Figure1.2 : Les règles de base de modèle de Reynold.

7.1. Les comportements individuels

Les comportements individuels sont associés à chaque individu, ces comportements peuvent sont simples parmet ces comportement on peut citer : le suivie de chemin, l'évitement d'obstacle, la poursuite …..Ect).

Reynold a modélisé plusieurs comportement parmet eux :

7.1.1. La recherche (Seek)

Le comportement de la recherche ne reposent pas sur des stratégies complexes impliquant la planification de trajectoire ou des calculs globaux, mais utilisent plutôt des informations locales, telles que les forces des voisins. Cela les rend simples à comprendre et à mettre en œuvre, tout en permettant de produire des schémas de mouvement très complexes.

Le comportement de recherche implique deux forces: curent velocity qui représente la direction de l'agent et desired velocity qui représente la distance entre la position de l'agent est le but. Le vecteur desired velocity est une force qui guide le personnage vers sa cible en utilisant le chemin le plus court possible (ligne droite entre eux - auparavant, c'était la seule force agissant sur le personnage). La force de direction résulte de desired velocity soustraite de la vitesse actuelle et pousse également le personnage vers la cible. [8]

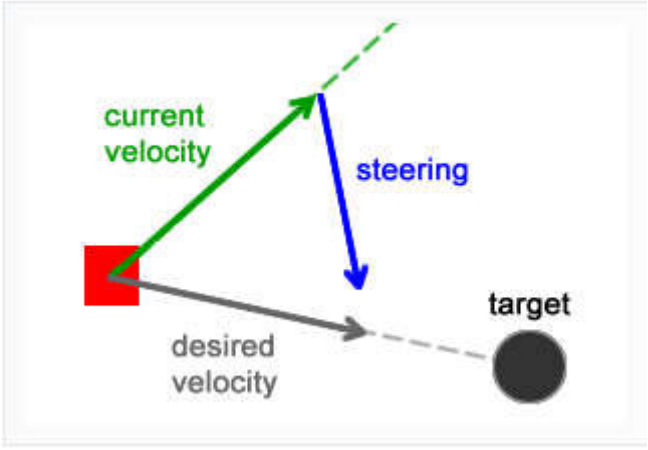


Figure 1.3 : Les forces qui influent sur le personnage lors de la recherche.[8]

7.1.2. Fuite et Arrivée (Flee and arrival)

- **Le comportement de fuite,** est un comportement qui oblige le personnage à s'éloigner du poursuivant. Le comportement de fuite utilise mêmes forces utilisées dans le comportement de la recherche, mais elles sont **opposés** pour obliger l'agent à fuir de la cible.[8]

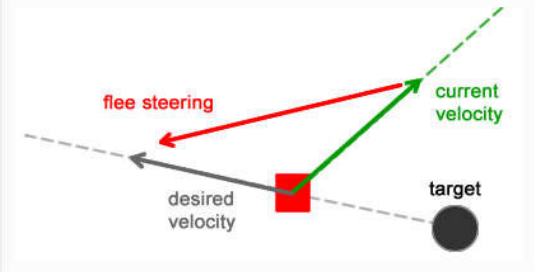


Figure 1.4 : Les forces qui influent sur le personnage lors de la fuite. .[8]

Ce nouveau vecteur "desire_velocity" est calculé en soustrayant la position du personnage de la position de la cible, ce qui produit un vecteur qui va de la cible au personnage .Les forces résultantes sont calculées presque de la même manière que dans le comportement de recherche. En comparant le vecteur de desired velocity du comportement de fuite avec celui du comportement de recherche, la relation suivante peut être établie.

```
flee_desired_velocity = -seek_desired_velocity
```

Une fois que la force de direction est calculée, elle doit être ajoutée au vecteur velocity du personnage. Étant donné que cette force éloigne le personnage de la cible, chaque frame

cessera de se déplacer vers la cible et commencera à s'éloigner de celle-ci, produisant un tracé de fuite (courbe orange dans la figure ci-dessous):

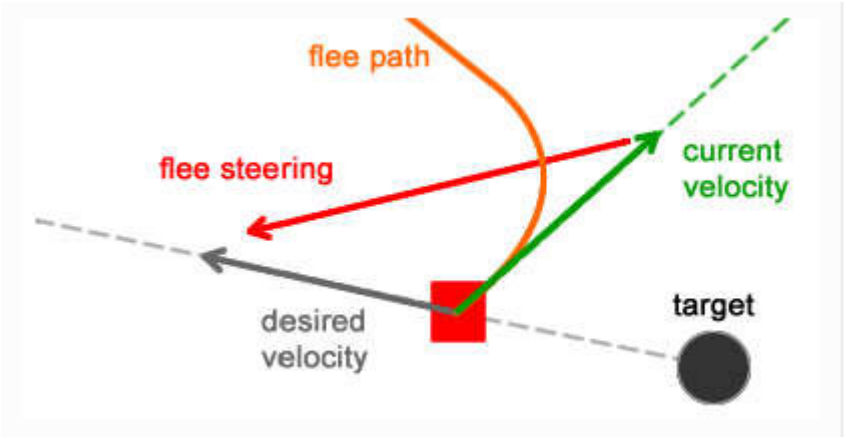


Figure 1.5 : le chemin résultant lors du comportement de la fuite .[8]

Le comportement d'arrivée empêche le personnage de se déplacer à travers la cible. Cela ralentit le personnage à l'approche de la destination, s'arrêtant éventuellement à la cible.

Le comportement est composé de deux phases :

- La première phase est celle où le personnage est éloigné de la cible et fonctionne exactement de la même manière que le comportement de recherche (se déplacer à toute vitesse vers la cible).
- La deuxième phase est celle où le personnage est proche de la cible, dans la "zone de ralentissement" (un cercle centré sur la position de la cible) [8]

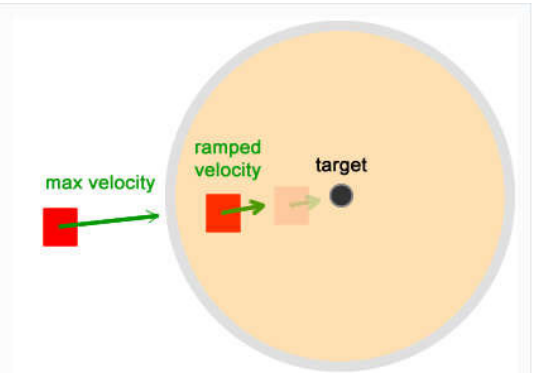


Figure 1.6 : Le comportement d'arrivé .[8]

Remarque : quand le personnage entre dans le cercle, il va commencer à réduire sa vitesse jusqu'elle soit nulle.

7.1.3. Evitement de collision (Avoid_collision)

L'idée de base de la prévention des collisions est de générer une force de direction permettant d'éviter les obstacles chaque fois que l'on est suffisamment proche pour bloquer le passage. Même si l'environnement présente plusieurs obstacles, ce comportement utilisera un à la fois pour calculer la force d'évitement. [8]

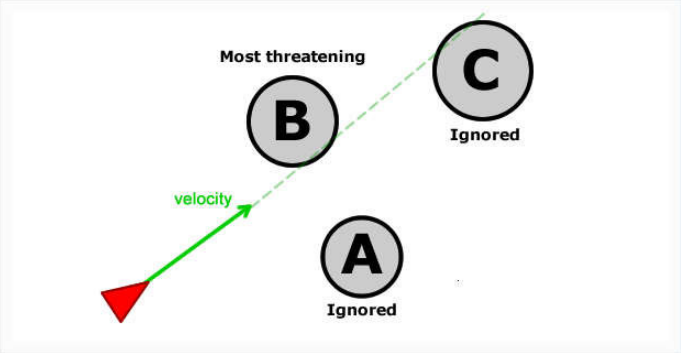


Figure 1.7 : Détection de collision.[8]

7.1.4. Suivie de chemin (Path_following)

Le suivi de chemin est un problème fréquent dans le développement de jeux. , ce comportement permet aux personnages de suivre une trajectoire prédéfinie constituée de points et de lignes. [8]

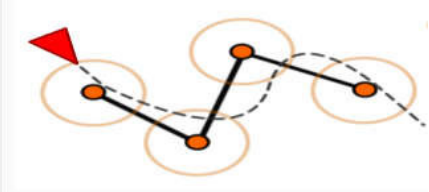


Figure 1.8 : Suivie de chemin .[8]

7.2. Comportement de groupe

Les comportement de groupe sont des comportements simple qui se fait par l'ensemble de groupe , la combinaison de ses comportement peuvent résulter un nombre de comportements complexe appelés comportements émergents

7.2.1. Le Suivie d'un chef

C'est un comportement de groupe qui sert à suivre un guide (leader) ce dernier va avoir son chemin et les autres agents de groupe doivent le suivre on appliquant une combinaison de comportement comme la recherche, l'évasion, la poursuite et la séparation pour éviter l'encombrement lorsque plusieurs personnages son âpres le chef du groupe. [9]

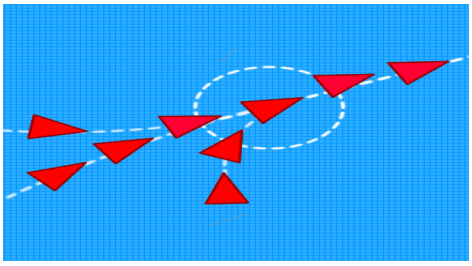


Figure1.9 : le comportement de suivie d'un chef.[9]

7.2.2. La Séparation

C'est un comportement qui nécessite une force entre les membres de groupes tel qu'ils doivent se séparer âpres quand la distance entre eux est très petite. [9]

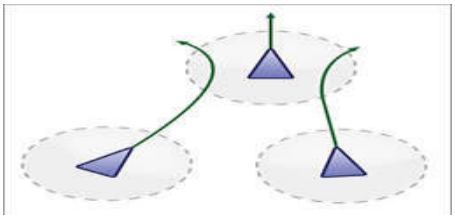


Figure1.10 : le comportement de séparation

7.2.3. L'alignement

Ce comportement donne aux membres de groupe la capacité de s'aligner l'un derrière l'autre, la direction de l'alignement se calcule en comptant la moyenne des directions des membres voisins. [9]

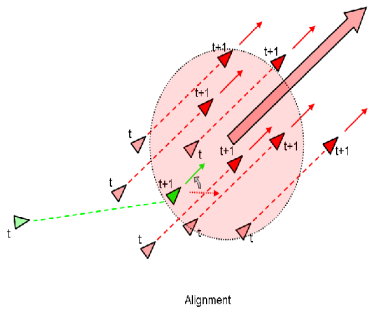


Figure 1.11 : comportement d'alignement

7.2.4. La cohésion

Est un comportement qui aide les membre de groupe a rester proche l'un de l'autre formant un groupe via une force de direction qui se calcule en déterminant la position moyenne de tous les membre voisins. [9]

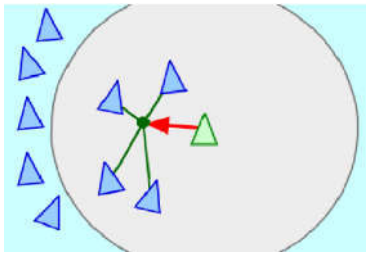


Figure1.12 : le comportement de cohésion .[9]

8. Conclusion

Dans ce chapitre on a parlé sur l'animation comportementale et on a vu aussi ses phases qui sont la base de toute simulation comportementale, on a parlé aussi sur l'agent virtuel et ses types ou ce dernier est très important pour la simulation de groupe et pour créer des environnements virtuels, nous avons parlés aussi de différents comportements.

Dans le chapitre suivant on va parler des environnements virtuels et les différents méthode de représentation qui aide a faire la planification de chemin.

Chapitre 02

« Les environnements virtuels »

1. Introduction

Dans ce chapitre on va parler sur la notion des environnements virtuel , ce dernier est un la base de toute planification de chemin , pour utilisé l'environnement virtuel , il faut qu'il soit bien représenté donc on va voir aussi un nombre de méthodes de représentations et un ensemble d'algorithmes de planification de chemin .

2. L'environnement Virtuel

L'environnement virtuel est la partie extérieure ou l'agent virtuel peut interagir avec**,** cet environnement est crée par ordinateur. En autre définition, Les environnements ou mondes virtuels sont des mondes artificiels qui n'existent que dans l'univers informatique. Des utilisateurs, représentés par leur avatar (apparence virtuelle choisie par l'utilisateur) peuvent y interagir un peu comme dans la réalité. La communication peut se faire de vive voix ou via le texte. Les utilisateurs sont aussi amenés à se promener dans différentes régions de ces mondes virtuels, et à y faire des découvertes.

3. Type d'environnements

Les environnements sont devisés en deux catégories, chaque catégorie à sa propre représentation est son type d'utilisation:

3.1. Les environnements statiques : sont des environnements qui ne subissent pas de modifications au cours de temps. Les environnements statiques se caractérisent par leur structure qui n'est pas amenée à évoluer au cours du temps. Cette propriété implique donc qu'il n'y a pas de nouvelles accessibilités ou obstructions créées pendant le déroulement de la simulation. Puisque la configuration de ces environnements ne change pas, il est donc possible d'effectuer des pré-calcules afin de proposer des structures de données performantes lors de requêtes de planification de chemin ultérieures.

3.2. Les environnements dynamiques : pour leur part présentent des caractéristiques qui évoluent au cours de temps. Nous appelons un environnement dynamique lorsque les positions occupées par certains obstacles peuvent changer dans le temps. Cela peut être dû à

des objets qui : modifient la position dans le temps, modifient la forme dans le temps où apparaissent ou disparaissent dans l'espace de travail.

4) Méthodes de représentation de l'environnement

Avant de faire la planification de chemin dans l'environnement pour les agents virtuels, il faut d'abord réaliser une représentation pour l'environnement afin que ce dernier soit exploité facilement et d'une manière cohérente. La planification de chemin dépend en grande partie de l'environnement dans lequel l'agent va évoluer.

La formulation des problèmes de planification de chemin va donc généralement reposer sur la représentation qui est faite de cet environnement, aussi appelée "espace de travail" ou workspace. Les différentes méthodes de planification proposées explorent une représentation duale de l'espace de travail afin d'identifier le chemin désiré.

Dans cette partie on verra plusieurs méthodes pour représenter notre environnement.

4.1. décomposition en cellules

La décomposition en cellules propose la division de l'espace de travail en cellules de formes prédéfinies. Ces cellules sont ensuite caractérisées selon leur appartenance à l'espace libre(C-Free) ou l'espace fermé(C-Obstacle) afin de construire une structure de données plus adaptée à la planification de chemin.

Ce type de décomposition se reparti en deux autres types le premier est la décomposition approchée et l'autre est la décomposition exacte.

4.1.1.Décomposition approchée en cellules

Les méthodes de décompositions approchées proposent l'utilisation d'une forme prédéfinie de cellules afin de caractériser l'environnement. Ces cellules représentent des sous-ensembles de l'espace des configurations.

Les cellules obstruées par une partie de(C-Obstacles) sont ensuite annotées et interdites alla navigation lors de la planification de chemin.

Le rôle de la planification va être d'identifier une séquence de cellules
Interconnectées et appartenant à (C-Free) permettant de relier la position courante de l'agent à son but.

a) **La grille régulière**

Cette représentation est composée de cellules carrées. Les obstacles de l'environnement sont ensuite projetés dans cette grille permettant de marquer les cellules non navigables.

Chaque cellule dans l'espace libre permet l'accès à ses 8 cellules voisines si celles-ci appartiennent également à l'espace libre.
L'exploration de la grille régulière peut également être améliorée en utilisant des heuristiques afin d'empêcher les changements de direction trop brusques de l'agent ce qui permet également de réduire l'espace de recherche. [10]

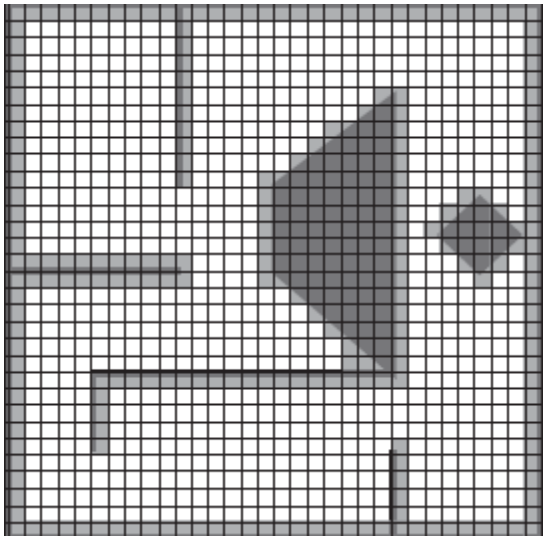


Figure 2.1 : représentation de l'espace de travail avec une grille régulière .[10]

b) Représentation hiérarchique

Cette représentation permet de représenter l'environnement avec plusieurs niveaux de détails.
L'utilisation d'une structure hiérarchique permet de réduire
Le nombre de cellules dans la grille en conservant de l'information détaillée la ou il y en a besoin.
Cette structuration hiérarchique est faite à l'aide d'un quadtree structurant les données des cellules. [10]

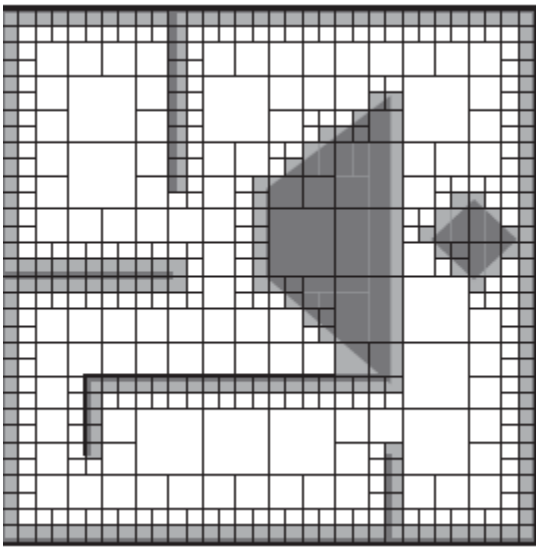


Figure 2.2 : représentations à l'aide de quadtree .[10]

L'utilisation d'un quadtree permet de réduire considérablement le nombre de cellules de la grille, et donc l'utilisation mémoire. Il est ainsi plus efficace de représenter de larges environnements ouverts en focalisant le raffinement sur les frontières entre C-Free et C-Obstacle.

c) Décomposition cylindrique

Cette représentation consiste à représenter l'environnement navigable a l'aide d'une décomposition en un ensemble de cylindres. L'utilisation de ces cylindres lui permet de créer une structure comprenant un nombre assez restreint
D'éléments et permettant la navigation de plusieurs milliers d'agents en temps réel.

Les axes les plus éloignes des obstacles et présentant donc la plus grande sécurité au niveau de la navigation, sont tout d'abord identifies.
Des cylindres, centres sur ces axes et dont le rayon correspond à la distance de l'obstacle le plus proche, sont définis le long de ces axes. Ces cylindres représentent donc dessous-espaces de C-Free ou la navigation est possible. En connectant ensuite les cylindres qui s'intersectés, des couloirs de navigation ou les entités peuvent naviguer librement sont crées. En identifiant les axes médians et l'éloignement maximum aux obstacles, cette méthode construit une représentation relativement proche d'un diagramme de Voronoi généralisé. [10]

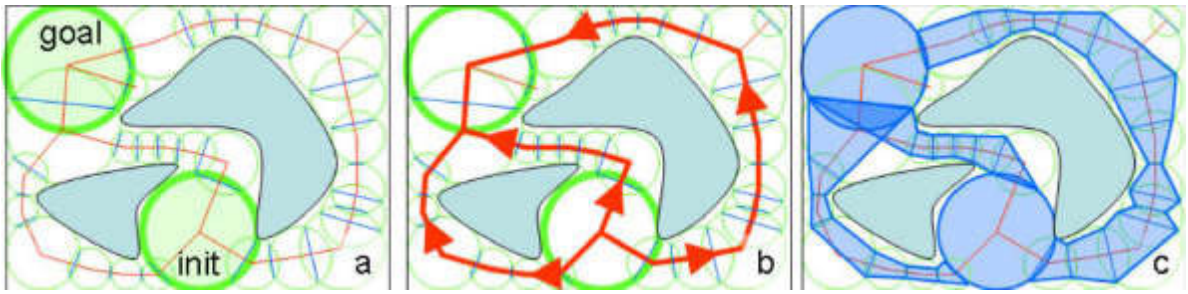


Figure2.3 : Représentation à l'aide des cylindres .[10]

4.1.2 Décomposition exactes en cellules

Les décompositions exactes permettent de capturer entièrement la représentation de l'espace libre à l'inverse des décompositions approchées.

a) La triangulation de Delaunay

Cette méthodes cherchent à caractériser précisément les espaces libres en utilisant les bordures des obstacles comme des contraintes formant les données d'entrées ,les méthodes basées sur les triangulations de Delaunay contraintes utilisent ainsi les frontières des obstacles de l'environnement afin de définir la triangulation l'espace libre est donc représenté par un ensemble de triangles dont les arêtes libres représentent des connexions entre zones navigables et les arêtes contraintes des obstacles.
La planification de chemin se ramène alors a identifier une séquence de triangles relies par des bordures franchissables.[10]

b) Représentation à l'aide de cellules

Bien que la majorité des représentations exactes permettent une représentation d'un environnement 2D, certaines approches se sont intéressées a gérer des environnements 3D a l'aide de décompositions exactes. Une approche utilisant une subdivision exacte en 3D incluant les contraintes de sol et de plafond permet par exemple d'ajouter des informations de hauteur à l'analyse de l'environnement. Cette subdivision, dite prismatique, permet ensuite de générer des cartes 2D interconnectées et d'augmenter automatiquement les cartes 2D d'information tridimensionnelle permettant entre autre de différencier les obstacles franchissables, tels que les marches, des obstacles infranchissables, tels que les murs .[30]

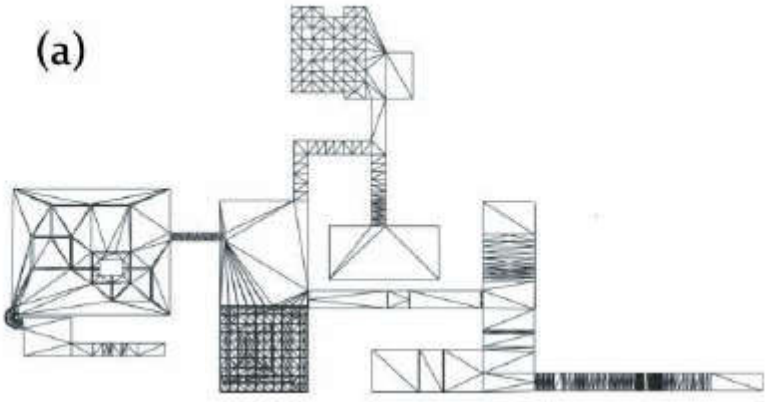


Figure2.4 : Représentation exacte à l'aide de cellules. .[10]

c) Décomposition en trapèzes

Le principe de cette méthode est de décomposer l'environnement en cellules trapézoïdales. Elle utilise un algorithme de balayage tel que les points délimitant la géométrie de l'environnement sont triés suivant l'axe des ordonnées pour que chaque segment ait pour origine le point sélectionné et pour extrémité l'intersection avec le bord d'un obstacle. Les segments générés (maximum 2 segments) ont pour origine le point sélectionné et pour extrémité la prochaine intersection avec le bord d'un polygone délimitant un obstacle.

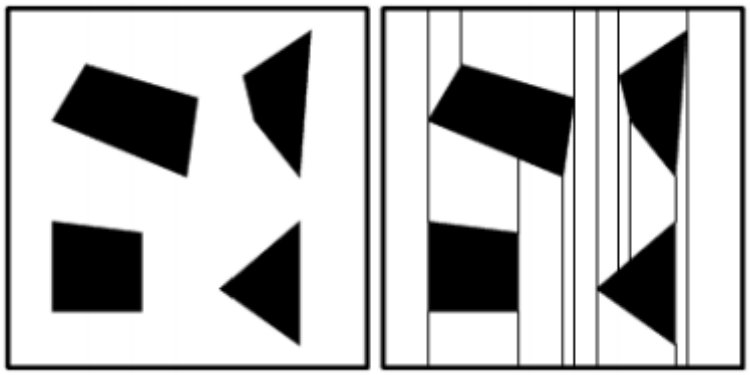


Figure2.5 : Représentation à l'aide de décomposition en trapèze .[10]

4.2 Cartes de cheminement

Les cartes de cheminement permettent de représenter un sous-ensemble de l'espace libre en déterminant des configurations libres de l'environnement et

En reliant ces configurations par des arcs de transition étant également contenus dans C-Free. En procédant ainsi les cartes de cheminement permettent une exploration beaucoup plus rapide de l'espace des configurations. Les différentes méthodes utilisant des cartes de cheminement vont donc s'intéresser principalement a déterminé des configurations libres de C-Free et a les connecter.

Les techniques utilisant les cartes de cheminement se décomposent en
Deux familles : les méthodes à requêtes multiples (multiple-Query) et les méthodes à requêtes simples (simple-Query). Dans le premier cas, une exploration de l'ensemble de l'environnement est effectuée a priori et réutilisée ensuite a de multiples reprises pour la planification de plusieurs chemins. Dans le second cas, l'environnement est explore seulement partiellement au moment de la demande de planification, chaque nouvelle requête entrainant une nouvelle exploration partielle de C-Free. [10]

4.2.1 Cartes de cheminement probabilistes

Les cartes de cheminement probabilistes, ou Probabiliste RoadMaps (PRM).
Les PRMs ont été introduites simultanément par Kavraki et Overmars avant de faire l'objet d'une étude commune de leur part. Pendant la phase de construction de la carte de cheminement, l'exploration de l'espace des configurations est faite de manière aléatoire en tirant des configurations au hasard. [10]

a)Cartes de routes probabilistes

La carte de route probabiliste est un algorithme de planification de mouvement en robotique, qui résout le problème de la détermination du chemin entre la configuration du robot et la planification tout en évitant les collisions.

Un exemple de PRM pouvant être explorée de différentes manières, l'idée de base derrière le PRM est d'examiner de plus près la configuration du robot, de tester son espace libre et d'utiliser un planificateur local pour tenter de connecter ces configurations à d'autres configurations proches.
Le planificateur PRM comprend deux phases: une phase de construction et une phase de requête. Lors de la phase de construction, Une PRM est construite, approximant les mouvements pouvant être effectués dans l'environnement. Tout d'abord, une configuration

aléatoire est créée. Ensuite, il est connecté à certains voisins, généralement des configurations et des connexions sont ajoutées au graphique jusqu'à ce que la feuille de route soit suffisamment dense. [10]

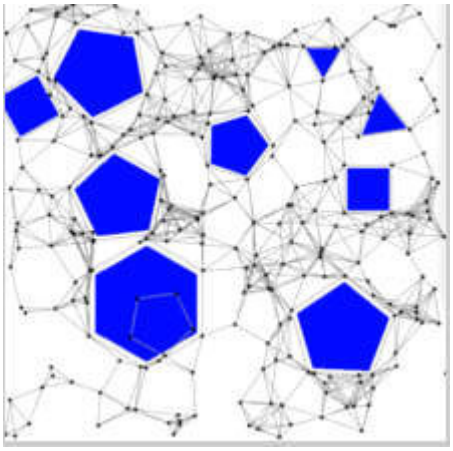


Figure2.6 : Représentation à l'aide de carte de routes probabilistes .[10]

Compte tenu de certaines conditions relativement faibles sur la forme de l'espace libre, le RMP est probablement complet, ce qui signifie que lorsque le nombre de points échantillonnés augmente sans limite, la probabilité que l'algorithme ne trouve pas de chemin, le cas échéant, approche de zéro. Le taux de convergence dépend de certaines propriétés de visibilité de l'espace libre, où la visibilité est déterminée par le planificateur local. En gros, si chaque point peut "voir" une grande partie de l'espace et si une grande partie de chaque sous-ensemble de l'espace peut "voir" une grande partie de son complément, le planificateur trouvera rapidement un chemin. [10]

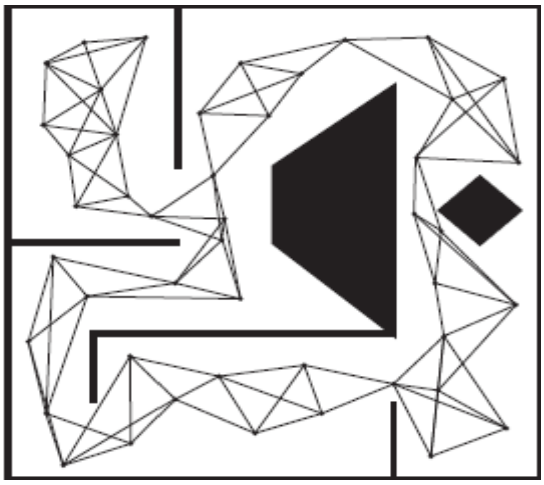


Figure2.7 : Autre représentation qui utilise la PRM .[10]

Il existe de nombreuses variantes de la méthode PRM de base, dont certaines assez sophistiquées, qui font varier la stratégie d'échantillonnage et la stratégie de connexion pour obtenir des performances plus rapides.

b) RRT (Rapidly-exploring random Tree)

L'algorithme RRT est un algorithme conçu pour rechercher efficacement des espaces non convexes de grande dimension en construisant de manière aléatoire une arborescence de remplissage d'espace. L'arbre est construit progressivement à partir d'échantillons prélevés de manière aléatoire dans l'espace de recherche et il est par nature contraint de se développer vers de vastes zones non recherchées du problème. Les RRT ont été développés par Steven M. LaValle et James J. Kuffner Jr. Ils gèrent facilement les problèmes d'obstacles et de contraintes différentielles (non holonomiques et cinodynamiques) et ont été largement utilisés dans la planification de mouvements des agents virtuels [10]

Un RRT développe une arborescence enracinée dans la configuration de départ en utilisant des échantillons aléatoires à partir de l'espace de recherche. Au fur et à mesure que chaque échantillon est dessiné, une connexion est tentée entre celui-ci et l'état le plus proche dans l'arbre. Si la connexion est réalisable (traverse entièrement l'espace libre et respecte toutes les contraintes), il en résulte l'ajout du nouvel état à l'arborescence. Avec un échantillonnage uniforme de l'espace de recherche, la probabilité d'élargir un état existant est proportionnelle à la taille de sa région de Voronoï. Comme les plus grandes régions de Voronoï appartiennent aux États situés à la frontière de la recherche, cela signifie que l'arbre s'étend de manière préférentielle vers les grandes zones non explorées.

Le but des RRT est d'explorer l'espace de configuration a l'aide d'un
Arbre de recherche. La racine de l'arbre d'exploration est fixée au niveau de
la configuration courante de l'entité et des échantillons sont ensuite tires aléatoirement dans l'environnement, entrainant la croissance de l'arbre d'exploration, en limitant l'expansion maximale de l'arbre a chaque pas d'exploration, la croissance de l'arbre est biaisée et tend a explorer en priorité les zones de l'environnement qui n'ont pas été explorées jusqu'alors, ce qui permet de générer rapidement un arbre possédant une bonne couverture de l'espace de recherche .[10]

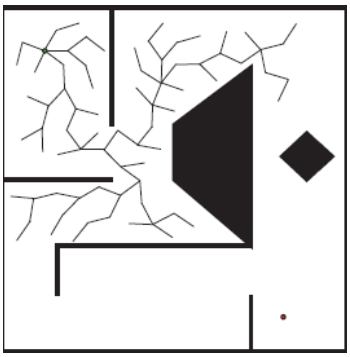


Figure2.8 : Représentation à l'aide des RRT

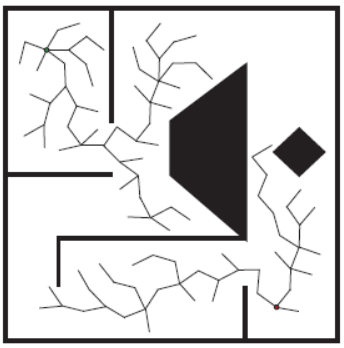


Figure 2.9 : Représentation à l'aide desRRT connect

Dans la Figure 2.8 La position courante de l'agent est le point en haut à gauche de l'environnement et la cible à atteindre le point en bas a droite.

Dans la figure 2.9 : Un arbre de recherche est relié à la position courante de l'agent tandis qu'un second est liée à la cible à atteindre. Les deux arbres de recherches explorent l'environnement de manière alternative.

4.2.2 Cartes de cheminement déterministes

a) Le graphe de visibilité

En planification de mouvement, un graphe de visibilité est un graphe d'emplacements invisibles, utilisé généralement pour un ensemble de points et d'obstacles dans le plan euclidien. Chaque nœud du graphe représente un emplacement de point et chaque arête représente une connexion visible entre eux. Autrement dit, si le segment de ligne connecte deux emplacements ne passent aucun obstacle, un bord est tracé entre eux dans le graphe. Lorsque l'ensemble des emplacements d'une ligne, cela peut être compris comme une série ordonnée. Les graphes de visibilité ont été étendus au domaine de l'analyse des séries chronologiques.

Dans un graphe de visibilité, les points correspondent aux coins des obstacles. Les sommets du graphe sont reliés par une arrête s'il est possible d'aller d'un point à l'autre en ligne droite, sans rencontrer d'obstacle. Cette technique a l'inconvénient de produire un grand graphe si l'environnement contient des espaces très ouverts.

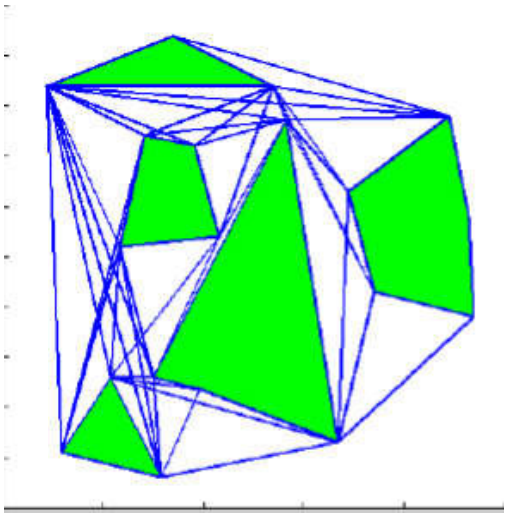


Figure2.10 : Représentation à l'aide de graphe de visibilité .[20]

b) Diagramme de Voronoï

En mathématiques, un diagramme de Voronoï est un découpage du plan en cellules à partir d'un ensemble discret de points appelés « germes ». Chaque cellule enferme un seul germe, et forme l'ensemble des points du plan plus proches de ce germe que de tous les autres. La cellule représente en quelque sorte la « zone d'influence » du germe.

Le diagramme doit son nom au mathématicien russe Gueorgui Voronoï (1868 - 1908), et est aussi appelé décomposition de Voronoï, partition de Voronoï, polygones de Voronoï, tessellation de Dirichlet ou polygones de Thiessen.

De manière plus générale, il représente une décomposition particulière d'un espace métrique déterminée par les distances à un ensemble discret d'objets de l'espace, en général un ensemble discret de points.[10]

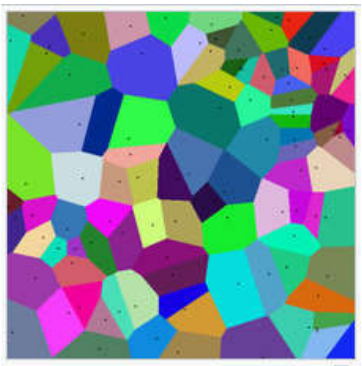


Figure 2.11 : Représentation à l'aide de diagramme de veronoi .

c) Diagramme de Veronois généralisé

Cette méthode se base sur une notion d'équidistance aux obstacles de l'environnement. Des sites sont évalués au sein de l'environnement, correspondant aux obstacles, dont les intersections vont former les points clefs. Les chemins générés maximisent la distance aux obstacles.

L'objectif est de partitionner l'espace grâce à des lignes équidistantes des obstacles les plus proches. Les chemins calculés à partir de cette représentation seront assez éloignés des obstacles, contrairement à ceux obtenus à partir du graphe de visibilité.

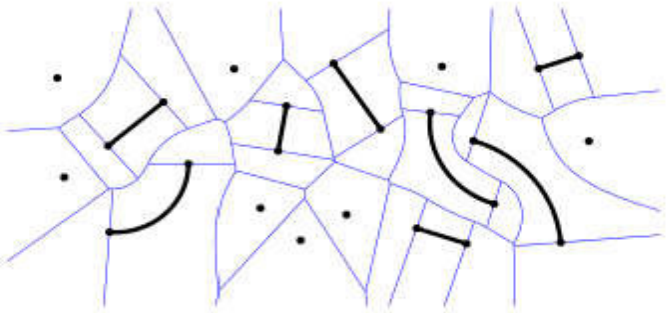


Figure 2.12 : Représentation à l'aide de diagramme de veronoi généralisé

4.3 Champs potentiels

Afin de résoudre directement le déplacement des personnes, une méthode utilisant des champs de potentiel peut également être appliquée à l'environnement de telle sorte que les obstacles aient une force de répulsion tandis que les buts une force d'attraction. Les notions de répulsion et d'attraction sont présentées plus en détail et illustrées dans la littérature [11]
Les champs potentiels génèrent un champ global pour l'ensemble du paysage où le gradient potentiel dépend de la présence d'obstacles et de la distance qui le sépare du but. La méthode pose certains problèmes, tels que des minima locaux où les agents pourraient rester bloqués et ne jamais atteindre l'objectif. Des champs de potentiel dynamiques ont été utilisés pour intégrer la navigation globale avec des obstacles et des personnes en mouvement, résolvant efficacement le mouvement de grandes foules sans qu'il soit nécessaire d'éviter explicitement les collisions [12]

5) LA planification de chemin

La planification de chemin est le fait que l'agent virtuel autonome trouve un chemin de son point de départ vers une destination prédéfinit .En tenant compte pour que ce chemin soit vide d'obstacle. La phase de la planification de chemin est importante dans la simulation de mondes virtuels interactifs peuplés d'humains virtuels autonomes. Les chemins calculés doivent prendre en compte différents critères d'optimalité tels que la longueur du chemin, le temps et l'énergie dépensés pour parcourir le chemin. L'efficacité du calcul revêt également une importance primordiale, car les environnements peuvent être extrêmement volumineux et complexes.

LA planification à deux majeurs types qui sont motionnée dans la figure si dessous

Figure 2.13 : les deux types de navigation

5.2. Algorithmes de planification de chemin

Il existe un nombre fameux d'algorithmes de planification qui facilite la planification de chemin et l'optimiser aussi par le choix de plus court chemin. Avant, pour trouver le plus court chemin possible dans un environnement les programmeurs utilise des algorithmes de la théorie de graphe théorie de graphe, ils utilisent des méthodes très classiques et simples à programmer, on va Siter quelques fameux algorithmes pour trouver le plus cour chemin :

5.2.1. Depth-First Search Algorithm

L'algorithme de recherche par profondeur utilise la pile (Last-In-First-Out) et son algorithme est récursif.

Cet algorithme approfondit l'espace de recherche à tout moment, lorsque cela est possible.

Il est simple à mettre en œuvre, mais le problème majeur de cet algorithme est qu'il nécessite

Grande puissance de calcul pour une petite augmentation de la taille de la carte [13]

5.2.2. Best-First Search Algorithm

L'algorithme de recherche Best-First utilise deux listes (liste ouverte et liste fermée) pour limiter les états qu'il utilise pour représenter l'environnement

A chaque étape, la recherche Best-First trie la file d'attente selon une fonction heuristique. Cet algorithme sélectionne simplement le nœud non visité avec la meilleure valeur heuristique à visiter ensuite. [13]

5.2.3. Hill-Climbing Search Algorithm

Cet algorithme de recherche développe l'état actuel de la recherche et évalue ses

Nœuds fils. C'est un algorithme itératif qui commence par une solution arbitraire à un problème, et fait alors un effort pour trouver une meilleure solution en modifiant progressivement un seul élément de la solution. Si le changement produit une meilleure solution, un changement progressif est à la nouvelle solution, en répétant jusqu'à ce qu'aucune amélioration supplémentaire ne puisse être localisée.

Cet algorithme trouve des solutions optimales aux problèmes convexes .Pour d'autres problèmes, il ne trouvera que des optima locaux (solutions qui ne peuvent pas être améliorées par des configurations voisines), qui ne constituent pas nécessairement la meilleure solution possible (l'optimum global) parmi toutes les solutions possibles (l'espace de recherche).

[13]

5.2.4. Algorithme de Dijkstra

L'algorithme de Dijkstra permet de résoudre un problème algorithmique : le plus court chemin. Ce problème a plusieurs variantes. La plus simple est la suivante : étant donné

un graphe non orientées, dont les arêtes sont munies de poids, et deux sommets de ce graphe, trouver un chemin entre les deux sommets dans le graphe, de poids minimum. L'algorithme de Dijkstra permet de résoudre un problème plus général : le graphe peut être orienté et l'on peut désigner un unique sommet, et demander d'avoir la liste des plus courts chemins pour tous les autres nœuds du graphe.

Nous maintenons deux ensembles, l'un contient les sommets inclus dans l'arborescence du chemin le plus court, l'autre ensemble comprend les sommets non encore inclus dans l'arborescence du chemin le plus court. À chaque étape de l'algorithme, nous trouvons un sommet qui se trouve dans l'autre ensemble (ensemble de pas encore inclus) et qui a une distance minimale par rapport à la source.

Les étapes détaillées utilisées dans l'algorithme de Dijkstra pour trouver le chemin le plus court d'un sommet à une source à tous les autres sommets du graphique donné.

Cet algorithme est résumé comme suit :

1) Créez un ensemble (ensemble d'arborescence de chemin le plus court) qui garde la trace des sommets inclus dans l'arborescence de chemin le plus court, c'est-à-dire dont la distance minimale par rapport à la source est calculée et finalisée. Au départ, cet ensemble est vide.

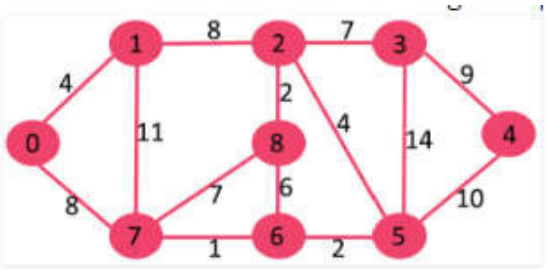
2) Attribuez une valeur de distance à tous les sommets du graphe en entrée. Initialisez toutes les valeurs de distance en tant que INFINITE. Attribuez la valeur de distance à 0 pour le sommet source afin qu'il soit sélectionné en premier.

3) Alors que la liste n'inclut pas tous les vertices

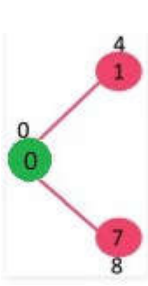
4) Choisissez un sommet u qui ne figure pas dans sptSet et qui a une distance minimale.

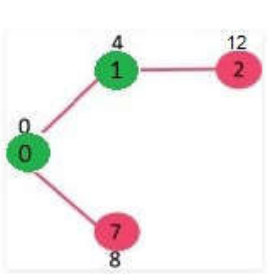
5) Incluez u dans la liste.

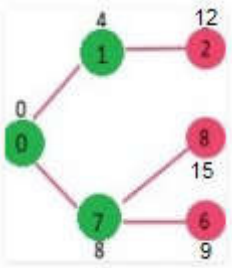
6) Met à jour la valeur de distance de tous les sommets adjacents de u. Pour mettre à jour les valeurs de distance, parcourez tous les sommets adjacents. Pour chaque sommet adjacent v, si la somme de la valeur de distance de u (à partir de la source) et du poids du bord u-v est inférieure à la valeur de distance de v, mettez à jour la valeur de distance de vecteur. [14]



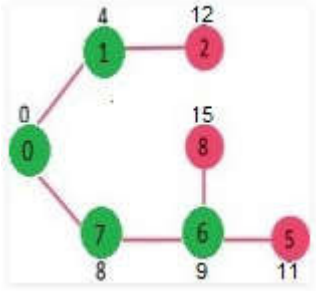
_1_

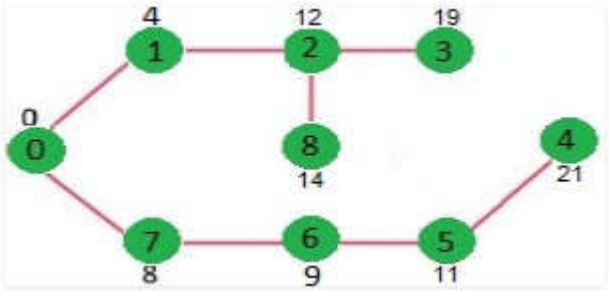






_2_ _3_ _4_





_5_ _6_

5.2.5. Algorithme A*

L'algorithme A* est une extension de l'algorithme de Dijkstra. Sa différence est qu'il va chercher à se diriger vers le nœud d'arrivée**,** à l'aide d'une heuristique. Une heuristique est une fonction qui, pour deux nœuds, renvoie une valeur. Cette fonction peut être quelconque mais doit satisfaire une contrainte : elle ne doit jamais surestimer la distance réelle.

L'algorithme de recherche **A*** est un algorithme de recherche de chemin dans un graphe entre un nœud initial et un nœud final tous deux donnés. De par sa simplicité il est souvent présenté comme exemple typique d'algorithme de planification . L'algorithme A* a été créé pour que la

première solution trouvée soit l'une des meilleures, c'est pourquoi il est célèbre dans des applications comme les jeux vidéo privilégiant la vitesse de calcul sur l'exactitude des résultats. Cet algorithme a été proposé pour la première fois par **Peter E. Hart , Nils John Nilsson** et **Bertram Raphael** en (1968).

L'algorithme se résume comme suit :

A* utilise deux listes, ces listes contiennent des points. Pour être plus général, on peut même dire que ces listes

Contiennent des nœuds d'un graphe représentant notre terrain.

La première liste, appelée *liste ouverte*, va contenir tous les nœuds étudiés. Dès que l'algorithme va se pencher sur

Un nœud du graphe, il passera dans la liste ouverte (sauf s'il y est déjà).

La seconde liste, appelée *liste fermée*, contiendra tous les nœuds qui, à un moment où à un autre, ont été considérés

Comme faisant partie du chemin solution. Avant de passer dans la liste fermée, un nœud doit d'abord passer dans

La liste ouverte, en effet, il doit d'abord être étudié avant d'être jugé comme bon.

On commence par le nœud de départ, c'est le nœud courant

• On regarde tous ses nœuds voisins

• si un nœud voisin est un obstacle, on l'oublie

• si un nœud voisin est déjà dans la liste fermée, on l'oublie

• si un nœud voisin est déjà dans la liste ouverte, on met à jour la liste ouverte si le nœud dans la liste ouverte

A une moins bonne qualité (et on n'oublie pas de mettre à jour son parent)

• sinon, on ajoute le nœud voisin dans la liste ouverte avec comme parent le nœud courant

• On cherche le meilleur nœud de toute la liste ouverte. Si la liste ouverte est vide, il n'y a pas de solution, fin de l'algorithme

• On le met dans la liste fermée et on le retire de la liste ouverte

• On réitère avec ce nœud comme nœud courant jusqu'à ce que le nœud courant soit le nœud de destination.

Voici une illustration d'une itération de l'algorithme. On souhaite aller du point orange au point bleu. Les nœuds voisins de la case orange sont les cases marquées en vert, ils passent en liste ouverte. Et de chacune d'elles on calcule les couts G et H pour aller à la case orange et

pour aller à la destination. J'ai choisi la distance à vol d'oiseau. Et la case qui aura le cout le plus faible sera la case verte du dessous, elle va donc passer en liste fermée. L'algorithme Va réitérer à partir de cette case.

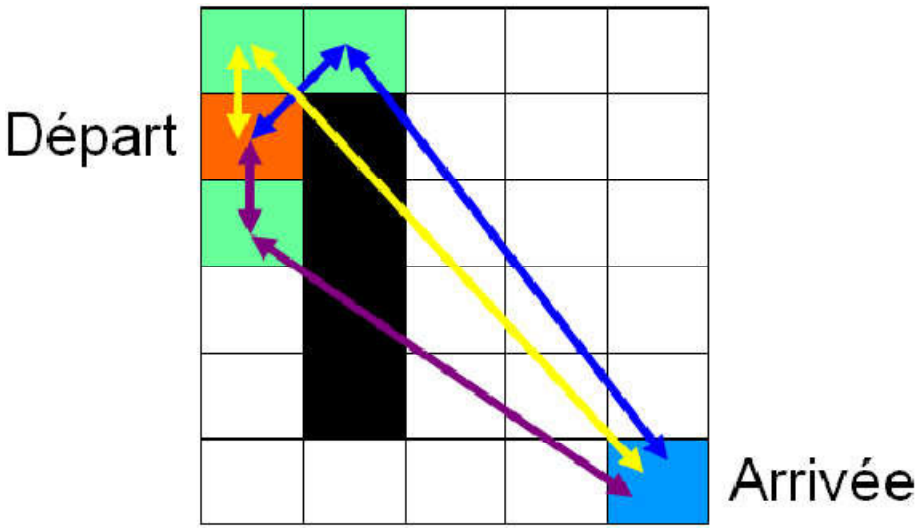


Figure 2.14 : l'algorithme A* sur une representation de grille reguliere

5.2.6.Evolution de l'algorithmes A*

L'algorithme A* a connu une evoulution durant le temps ,quelques algorithmes ont apparues apres A* qui sont une extension de ce fameux algorithmes parmet ces algorithmes on peut citer : IDA*, HPA* , A* herarchique, Theta*,Weighted A*, [15]

6) Les utilisations des environnements virtuels

Les environnements virtuels sont utilisés dons nombreux domaine telque

6.1 Les jeux vidéo

Ce domaine est parmet les premiers et les plus classés dans l'utilisation des environnements virtuels 3D et selon le type de jeux sera le type de d'environnement et sa nature, pour cela on va citer quelques types de jeux. [16]

a. les jeux de tirs subjectifs ou FPS (First-Person Shooters)

Le point de vue est celui de l'utilisateur qui est, en général, un personnage muni d'une arme. (Exemple : Quake, Unreal Tournament, Half-Life, ...). Dans ces jeux, le monde est visuellement très réaliste et relativement étendu. [16]

b. les jeux de combat

Ces jeux sont généralement jouables à deux et les deux personnages doivent s'affronter au cours d'un combat (Exemple : Soul Calibur, Tekken, Mortal kombat, ...). Le monde y est restreint et le réalisme est de moindre importance. [16]

c. les jeux de simulation de vie

Ce type de jeu n'a pas de but précis, il s'agit de gérer la vie d'un personnage ou d'une famille. (Exemple : Les Sims, SimCity). Le monde est restreint et son réalisme est important.

d. les jeux de sport

Jeux de simulation du sport (Exemple : FIFA, NBA) qui requièrent un environnement réaliste mais peu étendu.

6.2. Les Metavers

Sont des mondes virtuels 3D, dans lesquels les utilisateurs évoluent sous forme d'Avatars personnalisables, le contenu des Metavers est crée par les utilisateurs eux même ce qui mène à avoir plusieurs contenus sans cesse d'évolution.

La fonction principale d'un Metavers est de mettre en relation des utilisateurs réels du monde entier par leurs Avatars pour développer des activités éventuellement lucratives.

Le premier Metaver fut lancé en 1993 lors du Siggraph, le plus populaire à ce jour est « second_ life » qui a été lancé en 2003, Celui-ci totalisait 2 Millions de résidents en Décembre 2006 et en comptait 15 Millions en 2009. [16]

6.3. L'apprentissage en ligne et les jeux sérieux

L'utilisation des environnements n'est pas concentrée sur les jeux uniquement, ils sont utilisés aussi dans l'apprentissage par la simulation et l'éducation et ça c'est pas nouveau car dés 1973 ,les jeux éducatifs comme « the oregon trail » ce jeu proposait au jeune américains et canadiens de suivre la vie de pionniers en route vers l'oregon en incarnant un chef de convoi , et « Lemonada stand » qui à pour but d'initiait les élèves à la gestion d'un commerce en gérant un stand de limonade .

Ce type d'application connait un nouvel essor grâce à l'engouement pour les jeux sérieux, ces derniers ont connus plusieurs définitions selon les auteurs. [16]

Les jeux sérieux intéressent de plus en plus de domaine depuis le sucée de jeux de simulation d'entrainement militaire et de mission de combats « America 's Army » afin d'inciter les joueurs à s'engager dans l'armé américain, il a été lancé en 2002 et totalisé 17 Millions de téléchargement en 2004. [16]

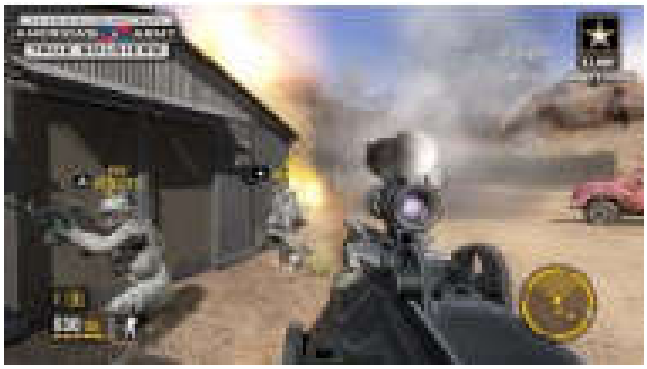


Figure 2.15: America's Army.

6.4. Le web geospaciale

Dans ce domaine, les environnements virtuels ont été l'élément principal, le terme geospaciale regroupe tous les services de cartographie 3D sur le web qui permet de visualiser les données géographiques dans un système d'information géographique.

Dans ce domaine, les environnements virtuels ont été l'élément principal, le terme geospaciale regroupe tous les services de cartographie 3D sur le web qui permet de visualiser les données géographiques dans un système d'information géographique. [16]

Ces services proposent aux utilisateurs d'ajouter de nouveaux modèles 3D mais aussi des annotations geolocalisées, ce qui crée une communauté au sein de cet environnement virtuel.

[16]

a-Bing Maps 3D b-Google Earth

FIGURE 2.15 : Vues de New York dans Bing Maps 3D et Google Earth

7. Conclusion

En concluant ce chapitre, cette partie a basé sur la notion de l'environnement virtuel ainsi que les différentes méthodes de la présentation avec plusieurs algorithmes de planification qui on été utilisé dans la planification de chemin comme Dijkstra et A* on a citer aussi les algorithmes qui ont apparus pour optimiser A* ,on peut dire maintenant que la représentation de l'environnement avec un choix précis de l'algorithme de navigation facilite le déplacement de l'agent virtuel et les groupes d'agents aussi ce qui va donner comme résultat un environnement navigable et cohérent. Dans le chapitre suivant on va parler de la conception de travail.

Chapitre 03 :

« La conception »

1) Introduction

La conception est une phase importante dans le processus de développement des applications, elle permet de décrire l'état interne du système et permet aussi de donner une vision globale pour ceux qui vont lire ou bien utiliser le système .Pour cela ,on va parler de l'architecture globale et détaillé du système qui est sous forme de l'union de plusieurs modules .

Dans ce travail, notre principal objectif est de simuler des comportements des entités virtuelles de manière que chaque entité virtuelle soit capable d'abord de planifier un chemin qui va lui permettre de déplacer au sein de son environnement en prenant en compte la topologie de l'environnement et ses obstacles et aussi d'avoir la capacité de suivre ce chemin tout on évitant les collisions avec les autres entités virtuelles.

Dans notre travail on va considérer l'environnement simple peuplé d'obstacles et constitué d'une seule zone navigable qui relie la destination avec le but, pour cela on va formuler la problématique on proposant une méthode pour élaborer la topologie de l'environnement, ensuite on va faire appel à un algorithme qui va nous aider à trouver un chemin optimal, ensuite on fera ce qu'il faut pour faire déplacer l'agent vers son but on respectant le réalisme de la simulation.

2) Objectif

Notre système a pour objectif de peupler un environnement d'agents virtuels autonomes, l'autonomie va permettre à chaque agent de planifier un chemin et naviguer dans ce dernier. Pour ce faire il faut prendre en considération la topologie de l'environnement, ce qui va nous conduire vers un défi dans le choix de la méthode de représentation de l'environnement car la base de toute algorithme de planification est de bien repartir les zones navigables et non navigables. La navigation a pour but de simuler le déplacement des entités virtuels, dans notre système ,la navigation nécessite la simulation de différents comportements , ces comportements vont subir des forces de direction afin qu'elles soit naturelles est pseudo similaires à celle de l'humain réel ,Les comportement qu'on va voir dans notre travail sont des comportements simples visent à changer la direction des entité selon des forces qu'on va

appliquer sur eux , donc on va faire la réalisation de certains comportements qui sont repartis en deux types , le premier type inclus les comportements individuels tel que : la recherche , la fuite , la poursuite , l'évasion , le suivie de chemin , ….etc. et le deuxième type qui va inclure les comportement simples de groupe comme : l'alignement, le suivie de chef, la cohésion et la séparation.

Pour atteindre les objectifs de notre travail on va suivre les étapes motionnées ci-dessous.

- Représentation de l'environnement à l'aide de la triangulation de Delaunay.
- Planification de chemin à l'aide de l'algorithme A*.
- Une navigation qui est un ensemble de comportement d'individus et de groupe.

3) La conception globale

La conception globale vise à représenter l'état du système en générale, dans cette partie on va motionner les module de notre système qui sont des étapes complémentaire pour avoir le résultat final.

La première étape est la représentation de l'environnement via la triangulation de Delaunay pour générer l'espace navigable et non navigable, puis une planification de chemin va s'élaborer en basant sur la triangulation de Delaunay via un algorithme A*âpres, on va faire une navigation pour l'entité virtuel en faisons les comportements individuels et de groupe, dans le schéma si dessous l'architecture de notre système est illustré.

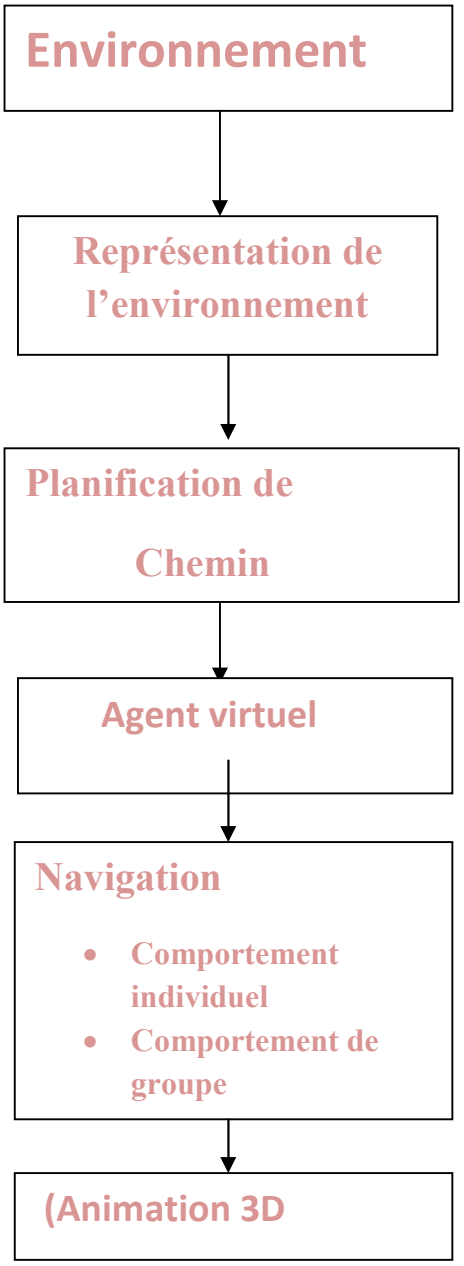


Figure 3.1: Schéma qui illustre la conception globale du système

4) La conception détaillée

La conception détaillé sert à montrer l'architecture détaillé du système en discutant les modules qui composent le système, notre travail est composé de quatre modules ,le premier module s'agit sur la représentation de l'environnement ,le deuxième est la planification de chemin , le troisième est le module de l'agent virtuel qui est caractérisé par son état physique et le quatrième qui est la navigation qui est basée sur la simulation de comportements individuels et de groupe, pour cela , dans cette partie on va parler et discuter chaque module de système en détail .

4.1.La représentation de l'environnement

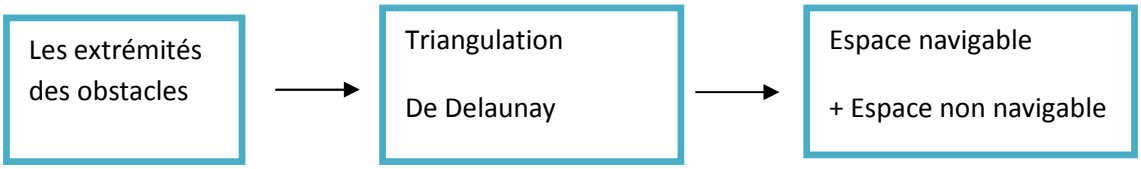


Figure3.2 : schéma illustratif pour la triangulation de Delaunay

- **Explication de triangulation de Delaunay**

La triangulation de Delaunay est parmi les représentations cohérentes qui est utiliser pour la représentation de l'environnement .Le mathématicien russe *Boris Delaunay* (1890 – 1980) a énoncé pour la première fois la définition de la triangulation qui porte son nom en 1934 dans son œuvre «Sur la sphère vide» [17]. La triangulation de *Delaunay* impose une contrainte : pour chaque triangle du maillage, son cercle circonscrit ne doit contenir aucun élément de l'ensemble des sommets comme le montre la figure suivante :

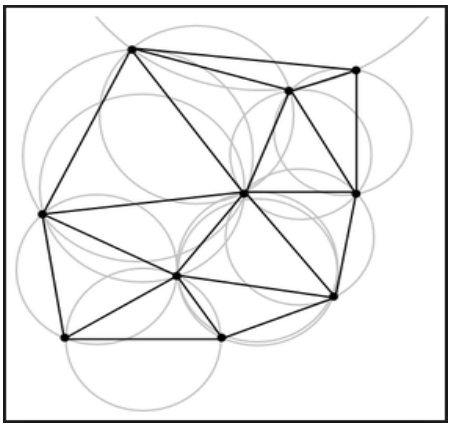


Figure3.3:représentation de la propriété de Delaunay.

Dans la triangulation de Delaunay, les points de passages sont les points d'intersection de l'ensemble de cercles qui passe par les extrémités, d'âpres Delaunay, chaque triangle est entouré par un cercle, l'intersection de ses cercles définissent les points de passages que l'agent va utiliser dans sa navigation [18], cette représentation respectes des conditions résumés en quelques lignes :

- Tous les cercles circonscrits des triangles de réseaux doivent être vides.
- La triangulation de Delaunay ne s'implémente pas sur un ensemble de points alignés.
- Elle peut s'implémenter sur des rectangles en utilisant chacune des deux diagonales pour respecter la triangulation de Delaunay.

4.2.LA planification de chemin

Pour que l'agent suive le plus court chemin vers sa cible cela a besoin d'un algorithme de planification afin d'avoir un chemin optimal, pour cela on a choisi l'algorithme A* comme méthode pour planifier le chemin.

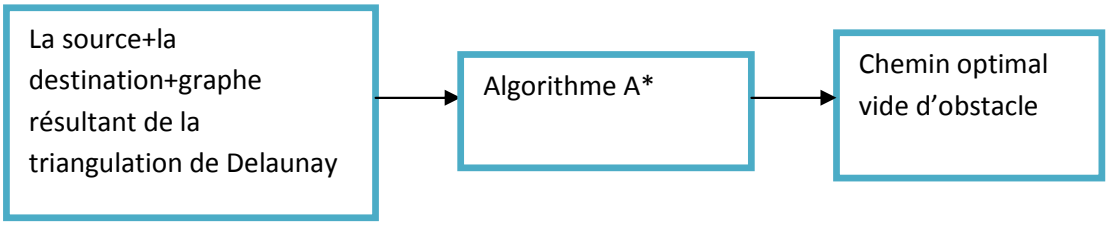


Figure3.4 schémas illustratif pour A*

- **L'algorithme A***

Est un Algorithme de planification qui aide a trouver le chemin optimal vide d'obstacles entre deux nœuds (source – destination).Il utilise deux listes une listes ouverte et une liste fermé et il utilise aussi une fonction de cout pour l'estimation de la distance restante entre la source et la destination.

- **Explication de l'algorithme**

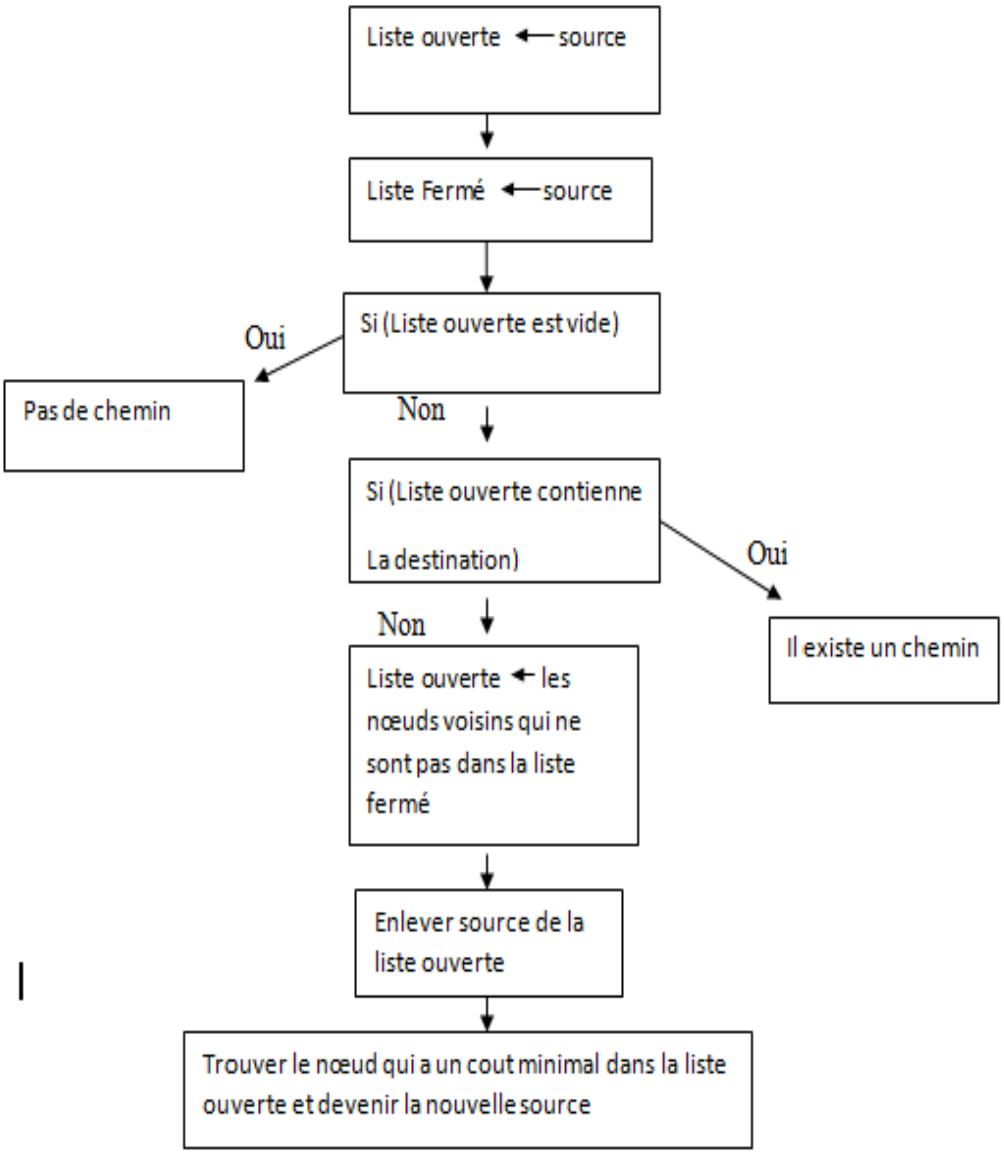


Figure3.5 : Organigramme explicatif pour expliquer A*

❖ **La Navigation :**

La navigation est la phase qui se fait après la fin de la planification de chemin, son but est de faire naviguer les entités virtuels dans l'environnement on faisant des comportement précis qui se basent sur la capacité physique de l'entité virtuel ,dans cette phase l'environnement est bien représenté et le chemin est planifié .Dans notre travail la navigation est basé sur les comportements réalisés par l'agent virtuel , c'est grâce à ces comportement et la capacité physique de l'agent que ce dernier peut se déplacer librement dans l'espace de travail pour que la navigation soit bien faite elle doit baser sur la perception pour aider à déterminer la prochaine action a accomplir .
Dans notre système la navigation est basée sur le comportement

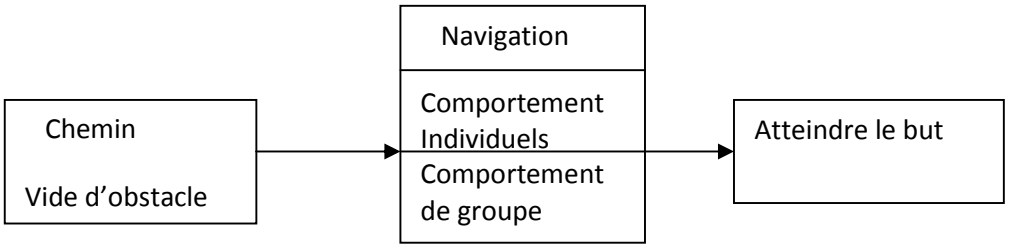


Figure3.6 : schémas qui illustre le processus de navigation

a) La perception

Cette opération est très importante dans quelques comportements comme la détection de collision et l'évitement d'obstacle. La perception se fait d'une manière mathématique par lancer de rayon.

- Le pseudo code de l'évitement

```
Début
        r.  Lancer rayon
          Si ( r .intersecte (Object)
            //évitement
           Changer de direction ;
           Sinon
    Avancement ;
    Fin
```

b) Les comportements individuels

Les comportements individuels sont proposés par **Craig Reynolds [9]** ces comportements simple sont réalisé par chaque entité on utilisant l'information locale

afin de simuler ce dernier, ces comportements sont réalisé par des forces sociale qui vont aider l'agent autonome à se déplacer d'une manière cohérente [9]

b.1) La recherche (Seek)

Ce comportement est très simple vise à déplacer l'agent vers un but en utilisant une force de direction, les paramètres utilisés dans ce comportement sont le vecteur de curent velocity et un autre vecteur appelé desired velocity. [9]

Explication :

- **Desired velocity = La position de but – la position de l'agent**
- Apres le calcul de ce vecteur on calcule maintenant le vecteur de la force de direction.
- Force = desired velocity – curent velocity.
- Normaliser le vecteur force puis le multiplier par une valeur
- La valeur résultante sera la nouvelle destination pour l'agent.
- En fin on aura ce chemin qui est en orange (chemin de recherche)

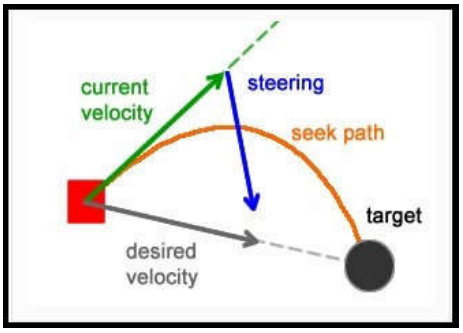


Figure3.7 : le chemin résultant du comportement de recherche [8]

b.2. La fuite (Flee)

Ce comportement et l'opposé du comportement de recherche la force de direction dans ce comportement va faire l'agent s'enfuir du but.[19]

Explication

La seule différence avec le Seek est la force

- Desired velocity = position de l'agent – La position de but.
- Force = desired velocity – curent velocity.

- Force(Fuite) = - Force (Recherche)

b.3. LA poursuite (Pursuit)

Ce comportement vise à faire suivre un agent ou un but dynamique la différence avec la recherche est que dans la recherche le but on connait sa position, mais en poursuite on ne connait pas la nouvelle destination de but ou de l'agent à suivre.

Explication

La poursuite est un processus qui consiste à suivre un but afin de l'attraper lorsque l'agent poursuit quelque chose, il doit anticiper ou estimer la position de but dans la prochaine seconde.

- **Pseudo code**

```
Poursuite (pos_agent à suivre)
Début
  Suivre le but avec une force de recherche
Fin
```

b.4. L'évasion(Evade)

Ce comportement produit une force de direction similaire à celle de la fuite mais reste que l'agent ne connait pas la prochaine position de but ou l'agent qu'on va éviter. Dans ce processus, âpres l'estimation de la position de but dans la prochaine seconde on l'évite. . [19]

- **Pseudo code**

```
Début

Dists = position –position de but ;

Max vélocité = Dist.normalize () ;

Mise à jour de position=dist /Max_vilocité ;

Fuite(Agent) ;

Fin
```

C.comportements de groupe :

Les comportements de groupe sont des comportements qui sont produit par de simples comportement, ces comportements sont dictent comportements émergent.

c.1.L'alignement

Les membres de groupe essaient toujours d'atteindre leurs position par rapport à la position de chef pas de force ici, les membres vont déplacer en fonction de la direction de chef.

- **Explication**
  - ❖ Calculer la moyenne de la direction de membre de groupe.
  - ❖ Deviser la somme par le nombre de groupe afin de garder la même direction pendant toute la trajectoire.

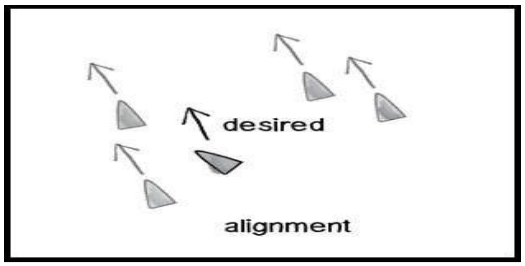


Figure3.8 : l'alignement dans le groupe

- **Pseudo code**

```
Début
Pour (groupe(i)) faire
Pour (membre j) faire
Si(le voisin n'est pas soit même) alors
Calculer la moyenne des directions () ;
Finsi
Deviser le vecteur de moyennes sur le nombre de membre de groupe
Finpour
Finpour
Fin
```

c.2 La cohésion

Ce comportement est fait pour pouvoir garder les membres de groupes formant un groupe, dans ce comportement au lieu d'ajouter la direction au vecteur de calcul, la position est ajoutée à la place. [19]

- **Pseudo code**

```
Début
Pour (chaque groupe) faire
Pour (chaque membre de groupe) faire
Si (voisin n'est pas soit même) alors
Calculer le centre de masse de groupe ;
Finsi
Calculer la direction vers le centre de groupe ;
Finpour
Finpour
Fin
```

c.3 La séparation

Le comportement de séparation donne à l'acteur la capacité de maintenir une certaine distance de séparation des autres caractères qui lui sont proches. Cela peut être employé pour empêcher les acteurs de s'entasser ensemble. . [19]

- **Pseudo code**

```
Début
Pour (chaque groupe) faire
Pour (chaque membre de groupe) faire
Si (dist (pos actuel, pos voisin de mm groupe <espace de personnel)alors
Calculer la force d'éloignement ;
Finsi
Finpour
Finpour
Fin
```

c.4 Evitement de collision

L'idée principale de l'évitement de collision est de générer une force de direction permettant d'éviter un obstacle ou les agents qui sont avec lui sur le même chemin, et même si l'environnement est peuplé d'un grand nombre d'obstacle, ce comportement utilisera un à la fois pour calculer la force d'évitement

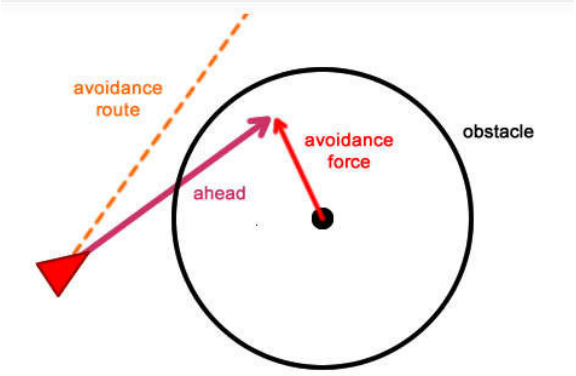


Figure3.9 : la force d'evitement [8]

- **Pseudo code**

```
Début
Ahead = position +velocité.normalise () * max _see_ahead ;
Ahead_2=pos+vélocité. Normalise ()*max_see_ahead*0.5 ;
Avoid_force= Ahead – le centre de l'obstacle
Avoidance_force= Avoidance_force.normalise ()*max avoid force.
Return (force_avoidance)
Fin
```

c.5 Le comportement émergent

Le comportement émergent est un comportement qui est le résultat d'un ensemble d'autres comportements simples, dans ce cas un nouveau comportement va apparaitre comme nouveau comportement appelé Flocking, ce comportement est la combinaison de comportement simples de groupes (l'alignement + la séparation + la cohésion) se comportement complexe est le résultat des les trois comportements précédent , ce comportement est utilisé beaucoup plus dans les système avec une foule de grande densité comme les simulation de cas de paniques .

- **Pseudo code**

```
Début
Pour (chaque groupe) faire
Pour (chaque membre de groupe) faire
Cohésion(i) = calculer cohésion (agent) ;
Alignement(i ) = calculer alignement(agent) ;
Separation (i) = calculer separation (agent) ;
Flocking  =cohesion(i) + alignment(i) + separation(i) ;
Return force (Flocking)
```

5) Conclusion

Dans ce chapitre nous avons focalisé sur la conception de système qui consiste à faire une simulation comportementale dans un environnement bien planifié ce dernier est simple et peuplé par des agents et un nombre d'obstacles. nous avons commencer par par motionner la représentation de la conception globale de système puis nous avons détaillé dans les différents modules qui sont nécessaires pour la réalisation de notre système.

Dans le chapitre suivant on va entamer la phase de la programmation ou bien la réalisation pour transformer notre système de la version papier à une version exécutable et ainsi nous allons citer les différents outils qu'on avait utilisé pour cette réalisation.

Chapitre 4

« Résultats et implémentation »

1) Introduction

Dans ce chapitre, on va voir les contenus de projet (les classes, fonction utilisées)
Et les résultats obtenus par la simulation de différents comportements.
Nous présentons d'abords les outils de développement qu'on a utilisé puis on va détailler un peux pour montrer les différents modules (fonctions) et en fin en donnera un aperçu de la scene via une interface graphique.

2) Outils de développements

Qt Creator

a) Qt_creator est un Framework écrit en c++ développé par **QT_Project**, Ce dernier, les applications et les bibliothèques qui l'utilisent peuvent être compilés par n'importe quel compilateur c++ conforme aux normes telque Clang, GCC, ICC.

- **Pourquoi Qt_creator**
  - ❖ Il intègre directement dans l'interface un déboguer.
  - ❖ LA création des interfaces graphiques.

b) Ogre 3D (Object-Oriented Graphics Rendering Engine)

Ogre est un moteur de rendu 3D en temps réel, orienté sur les scènes, le logiciel est open source .LA première version de logiciel à été publié en février 2005, le logiciel est écrit en c++ et à été porté sur Windows, Linux et Pocket PC et autre Systems d'exploitations.

- **Pourquoi Ogre3D**

- ❖ Interface orienté objet simple et facile à utiliser conçue pour minimiser l'effort requis pour le rendu de la scene 3D.
- ❖ Rapide et simple de lancer l'application.
- ❖ Le design est bien organisé et documentation complète de toutes les classes de moteurs.

3) Langage de programmation

- ❖ **C++**est un langage de programmation polyvalent crée par Bjarne Strostrup en tant qu'extension de langage c, il comporte des fonctionnalités fonctionnelles génériques et orienté objet.

- **Caractéristiques de c++**

- ❖ Simple et facile à apprendre et à écrire.
- ❖ Utilisé pour la conception des compilateurs, les systèmes d'exploitation, les jeux et les bases de données.
- ❖ Utilisation des pointeurs.

4) Architecture de l'application

4.1.Attributs

- ❖ Pour chaque entité on a ces attributs :

  - **Ogre :: Real** Walkspeed : qui désigne la vitesse de l'entité .
  - **Ogre :: vector3** Direction **:** Direction de l'entité**.**
  - **Ogre :AnimationState***AnimationState :l'animation de l'entité.
  - **Ogre :: vector3** Destination **:** c'est le nœud ou l'entité doit y'aller .
  - **std:: deque<Ogre::Vector3> WalkList :** est liste qui contient les nœuds pour que l'entité se déplace dans la scene .

- ❖ Pour les groupes on a les mêmes attributs sauf ils sont sous forme de tableaux

- **Ogre :: Real W**alk_speed[n].
- **Ogre :: vector3 m**Direction[n].
- **Ogre::AnimationState***mAnimationState[n].
- **Ogre :: vector3 m**Destination **[n].**
- **std:: deque<Ogre::Vector3>mWalkList[n]**

4.2.Les fonctions et les procédures

Dans l'application il existe deux principales fonctions :

1. Void QTOgreWindow :: create_scene () ;
    - Cette fonction s'occupe de la création de la scene.
2. bool QTOgreWindow:frameRenderingQueued (const Ogre ::FrameEvent evt)
    - c'est dans cette fonction ou se passe la boucle de rendu en parallèle avec le calcul.
    - Et on fait appelles à toutes les fonctions pour s'exécuter, les fonctions de différents comportements (individuels ou de groupes).
3. Il y'a d'autre fonction qui sont occupée de paramétrage de la scene :
    - Paramétrage de la camera.
    - Paramétrage de l'éclairage de la scene.
4. Maintenant je vais citer les différentes fonctions de comportements :
    - **Comportement individuels**

      Seek () ;

      Flee and arrival () ;

      Path following () ;

      Pursuite() ;

      Evade() ;
    - **Comportement de groupe**

      Evitement de collision () ;

      Séparation () ;

      Suivie d'un leader () ;

      Alignement () ;

Cohésion () ;

4.3. Les classes

Dans ce travail, il ya trois classes et un formulaire pour la modification de l'interface graphique.

- **Classe1 : QTOgreWindow.cpp** : Cette classe contient les fonctions des comportements et la scene crée avec la boucle de rendu.
- **Classe2 : MainWindow.cpp** : Cette classe relie le projet avec l'interface graphique, elle contient le fonctionnement de l'interface.
- **Classe3 : Delaunay.cpp** : la structure de donné de Delaunay
- **Classe4 : A_star.cpp :** Contient l'implémentation de A*
- **Classe5 :Nœud.cpp :** représente les nœuds utilisé dans Delaunay et A*
- **Classe 3 main.cpp:**C'est la classe principale de projet.

5) Les structures de données

5.1 Structure de la triangulation de Delaunay

Notre scène est représentée par un maillage de navigation triangulaire.

```
Struct SVertex          //  structure d'un sommet
{
        X, Y, Z : double ;      //coordonnées du sommet

        Obs. : entier;   //  identifiant de l'obstacle
};

Struct Sedge            // structure d'un segment
{
        V1, V2 : entier ;        // indices des extrémités du segment

};

Struct STriangle                //  structure d'un triangle
{
        V1, V2, V3 : entier ;    //  les indices des  3 sommets du triangle

        E1, E2, E3 : entier ;    //les indices des segments formant le triangle
```

```
        ETAT : entier ;          //   Navigable ou Non
};

StructSEdges
{
        V1, V2, T1, T2 : entier ;     //segments communs entre 2 triangles adjacents

};
```

Les sommets, les segments et les triangles générés sont sauvegardés dans des tableaux de structure approprie à chaque type.

5.2 Fonction A*

a) Représentation d'un nœud

On a utilisé la classe Nœud pour représenter l'élément de base de notre graphe de recherche (liste ouverte et liste fermée).

```
Class Nœud {
    X, Y, cout G, cout H, cout F, état : entier ;
    Parent : pointeur vers Nœud ;}
```

b) Représentation des listes des nœuds

La fonction A* utilise trois listes essentielles :

- une liste ouverte qui contient les nœuds à explorer,
- une liste fermée qui contient les nœuds choisis,
- une dernière liste qui contient les nœuds du chemin final.

On a implémenté ces trois listes comme étant des tableaux dynamiques en utilisant la class **vector** de la bibliothèque standard de C++. Cette classe possède plusieurs fonctions qui facilitent la manipulation sur les tableaux dynamiques.

push (): Ajouter un élément.

top () : Consulter le dernier élément ajouté .

pop () : Supprimer le dernier élément ajouté.

sort() : Pour trier le contenu de la liste.

c) Fonction de calcul de distance

Cette fonction consiste à calculer l'heuristique entre un nœud quelconque et le nœud destination.

Soit X_{dest}, Y_{dest} : les coordonnées du point destination,

X, Y : coordonnées d'un nœud.

- Distance de Manhattan

Fonction calculer_H_M(X, X_{dest}, Y, Y_{dest} : entier) : entier ;
debut
Return (|X_{dest} - X |+| Y_{dest} - Y|);
Fin;

- Distance euclidienne

Fonction calculer_H_E(X, X_{dest}, Y, Y_{dest} : entier) : entier ;
debut
Return (Racine ((X_{dest} - X)* (X_{dest} - X) + (Y_{dest} – Y)* (Y_{dest} – Y))) ;
Fin;

6) Résultats Obtenus

Nous avons eu quelques résultats qu'on va montrer dans cette partie.

Commençant par l'environnement :

Les figure **4.1** et **4.2** son l'état de notre environnement avant et âpres la triangulation de Delaunay.

Figure4.1 : environnement avant la triangulation

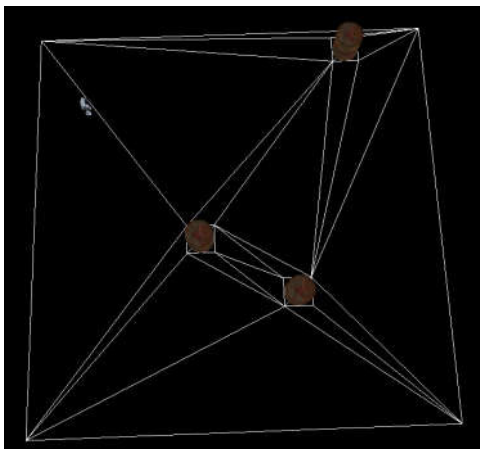


Figure4.2 : représentation de Delaunay

L'environnement à une configuration dynamique on relançant l'exécution on va avoir une nouvelle représentation.

On passe maintenant pour montrer le chemin vide d'obstacle généré par A*.

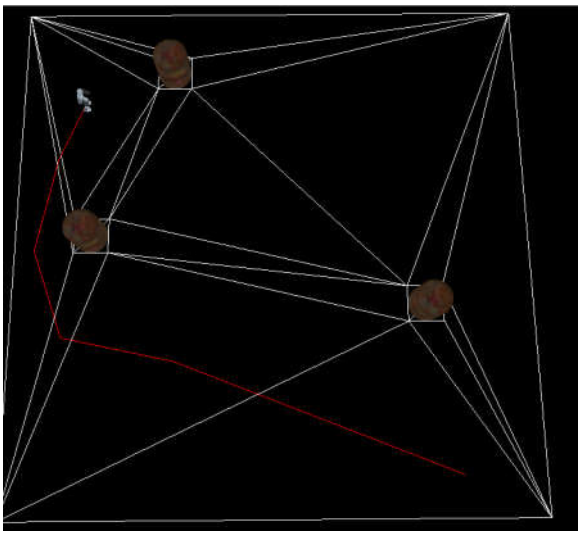


Figure4.3 : chemin généré par A*

Le suivie de chemin

Le résultat de suivie de chemin avec les différents variables rayon qui désigne la distance qui reste pour que l'agent atteindre son sous but et **curving** est la force de direction qui guide

La force utilisée ici est la force de recherche (Seek), le chemin en blanc est l'ensemble de point passés par l'agent et celui en rouge est le chemin obtenu de (l'algorithme A*)

Le comportement de poursuite

Pour ce comportement on a capturé les résultats comme suit :

La figure 4.7 montre l'agent atteins son but avant qu'il change de position.

Figure 4.7 : atteindre le but avant de changer Sa position

La figure 4.8 : illustre la poursuite apres le changement de position de but .

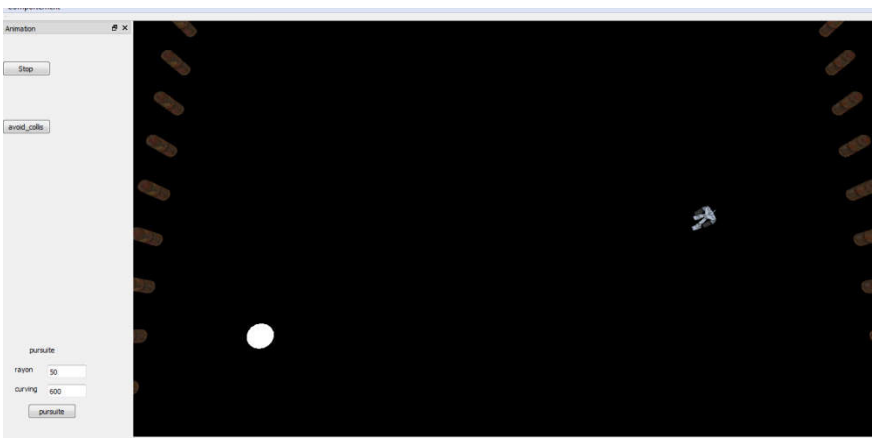


Figure 4.8 : le suivi de chemin vers la nouvelle position de but.

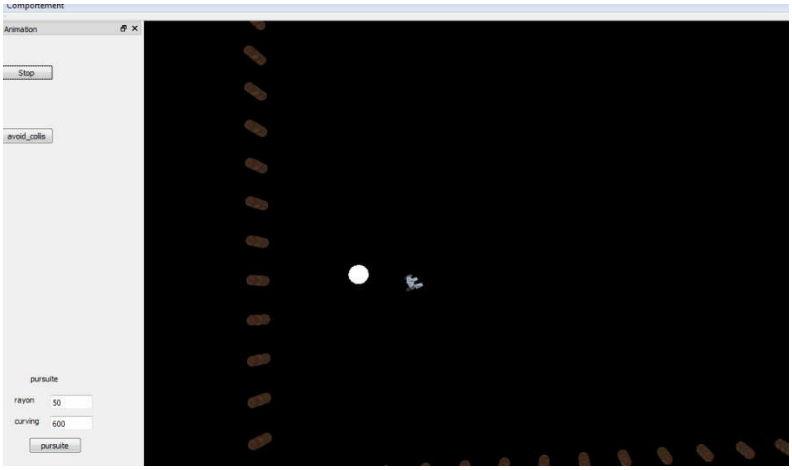


Figure 4.9 : l'arrivé à la nouvelle position de but

L'évitement de collision

La **figure 4.10** montre l'évitement de collision entre un ensemble d'agent virtuel chaque agent fait sa propre détection de collision à l'aide de la sphère query qu'il lui associé.

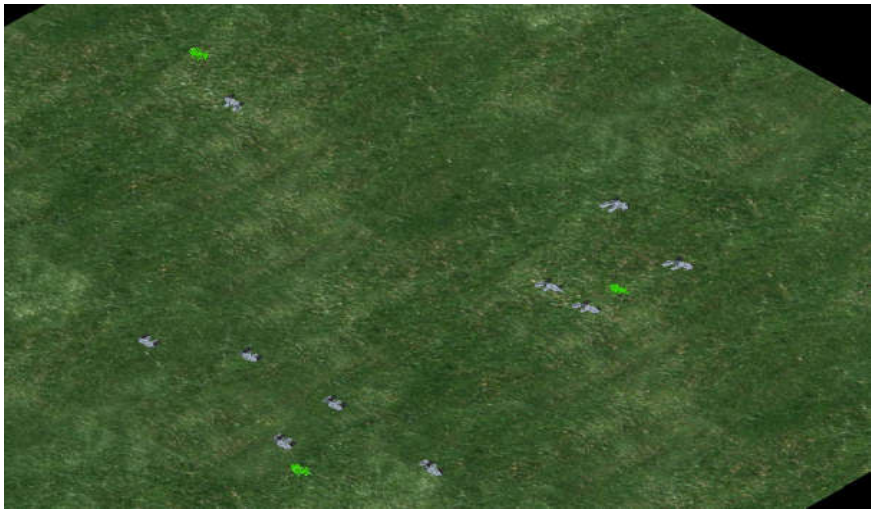


Figure 4.10 : la première étape avant de lancer l'évitement.

Chapitre 04 : Résultats et implémentations

Figure 4.11 : le départ de l'évitement.

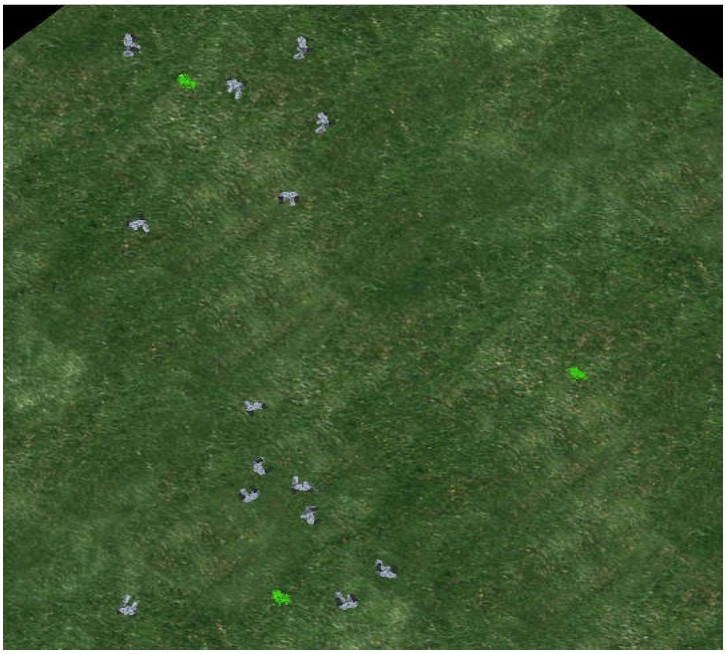


Figure 4.12 : la suite de l'évitement

Le comportement de suivie de chef :

Dans ce comportement le groupe est composé de 4 membres on va illustrer les résultats de ce comportement ci-dessus.

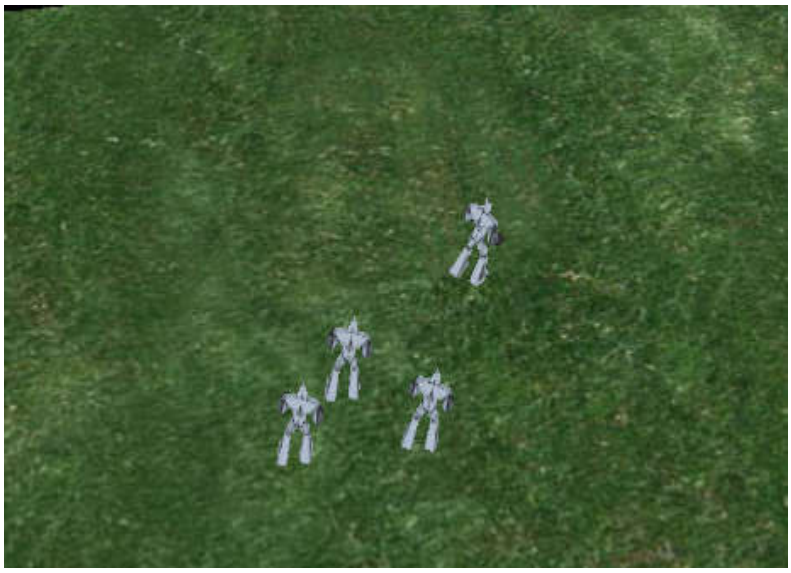


Figure 4.13 : le depart de suivie de chef.

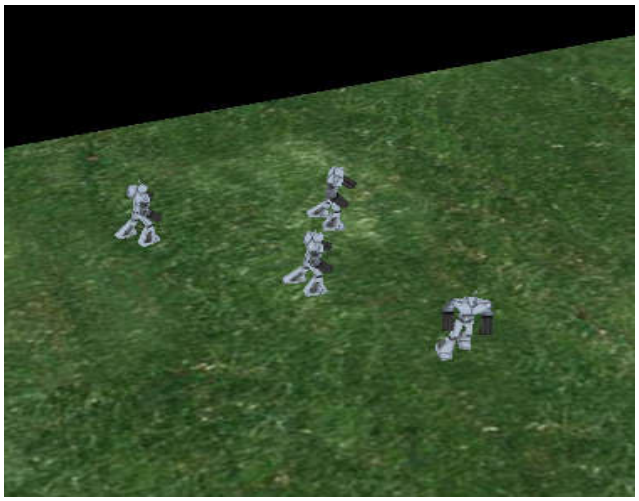


Figure 4.14 :la suite de suivie de chef

Conclusion generale

La simulation comportementale est un sujet d'actualité dans le domaine de l'infographie. L'importance de lancer des simulations de comportement d'un ensemble d'agents dans différentes situations réside dans l'impossibilité d'accomplir d'une manière rapide le comportement réel avec des humains. Pour cette raison, la simulation de comportent concerne beaucoup de domaines d'applications tels que Le divertissement, les effets visuels par exemple La production de films et La production du jeu. Dans le domaine de la gestion des interventions d'urgence par exempleL'incendie et l'intoxication au gaz et les foules surexcitées et attentat à la bombe. En génie civil par exemple, on s'intéresse aux caractéristiques de flux de foules des piétons afin d'assurer une évacuation sécurisée dans des situations d'urgences. Dans la psychologie pour valider les modèles de comportement de l'esprit humain. Dans l'urbanisme et la conception des bâtiments, la simulation des foules des piétons est utilisée pour tester la fiabilité des équipements publics et des conceptions architecturales.

Notre travail s'inscrit dans le cadre de simulation de comportements d'agent

virtuels. L'objectif principal était d'implémenter une structure de donnée cohérente qui permet de planifier le chemin correctement, la simulation d'un ensemble de

comportement d'une façon plus réaliste, Notre implémentation est basée principalement sur le modèle de comportement de C.Reynolds développé initialement pour reproduire les comportements individuels ainsi que les groupements des agents autonomes sur quelque espèce d'animaux comme les oiseaux et les poissons.

Les résultats Obtenus sont satisfaisants mais il reste quelque perspectives afin d'améliorer le travail.

- Optimiser la structure de donnée de la triangulation de Delaunay.
- Optimiser l'interface graphique de terme qu'elle soit manipulé a travers la souris .

Références bibliographique

Références bibliographique

[01] Mubasir,Kapadia,Norman,s.Badler ,Navigation and streeting for Autonomous Virtual Humans ,Phd thesis,University of Penusylamia.

[02] Bechiraz , P and Thalman, D.Abehavioral animation system for Autonomous Actors Personified by emotions,Proceeding of the first work _shop on Embodied conversational character(WECC'98') Lake tahoe , California pp8-17 ,1998.

[03] Autonomous Agent ,https//en wikipedia.org , consulté le 27/03/2019.

[05] Pamella Carreno Madrano , simulation comportementale dans les environnements virtuels ,2012.

[06] Ferguson,Touring Machines Architecture for dynamique , Rational , Mobile Agent , university of Cambridge , UK , 1992.

[07] J.Muller .An architecture for dynamically interaction agents . Phd thesis ,DFKI, University of Saarland , Saarbruken , 1996.

[08] understanding steering behaviors ,modifié le 23/05/2016,https://game developpement .tutsplus.com .consulté le 23/06/2019, 89Ko .

[09] Reynolds , C.W , 1999 –steering behaviours for autonomous characters , Proceeding of Game developers Conference 1999 , Miller free man game group , San fransisco ,California ,pp-763-782 .

[10] A modular framework for adaptive agent based steering .Singh, shawn and Kapadia ,Mubassir and Hemeletee, Billy and Reinman,Glenn and Flontsos ,Petros , 2011,symposing on interactive 3D Graphics and games.

[11] Global path plannning using artificial potential field ,Warren, C.W ,1989,Robotic and autonation , vol .1,pp 316-321.

[12] Multiple robot path coording using artificial potential field .Warrren , C.W 1990,Robotic Rbotic and automation ,Vol -1,pp 500-505.

Références bibliographique

[13] Barnouti ,NH, AI-Dabbagh, SSM and Nacer ,MAS (2016), path finding in strategy Games and Maze solving using A* Search Algorithm ,Journal of computer and communications,4,15-25.

[15] Planification , partie 1 ; la recherche de chemin ,publié le 9 mai 2015 .

[16] Rozeum Bouville , Interopabilité ds environnements virtuels 3D ,Modele de reconciliation des contenus et des composants logiciels ,Synthese d'image et realité virtuelle ,Insa de Rennes , 2012.

[17] B.Delaunay ,sur la sphere vide ,Congres international des matematiciens ,1924,p 695-700.

[18] Triangulation de delaunay ,telechargé le 22/06/,2019, https//fr.wikipedia .org , consulté le 24/06/2019.

[19] understanding steering behaviors ,modifié le 23/05/2016,https://game developpement .tutsplus.com .consulté le 23/06/2019, 89Ko.

[20] graphe de visibilité , https//en.wikipedia.org consulté le 25,04,2019 .

[21] Recherche de chemin par l'algorithm A* , Khayyam90, publié le 23 Aout 2008, derniere modification 26 fevrier 2008.

Table de Figures

Chapitre 1 : Animation comportementale

Chapitre2 : Environnement virtuels

Chapitre 03 : La conception

Chapitre 04 : Résultats et implémentation

Table des matières