الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Master

Filière Électronique

Spécialité Électronique des systèmes embarqués

présenté par

Mamoudou Moumouni Abdoul Karim

&

Tangara Bouba

Poursuite laser de cibles multiples par micromiroirs

Proposé par : Bougherira Hamida & Meliani Zoheir

Année Universitaire 2018-2019

Remerciements

Nos remerciements les plus sincères envers Dieu le tout puissant de nous avoir donné des parents exceptionnels, ainsi que de nous avoir octroyé les capacités nécessaires afin de nous permettre de réussir et survivre jusqu'à ce jour.

Nous remercions nos parents de vive voix, ayant toujours été présents autant physiquement que moralement et financièrement, nous montrant la droiture et l'éthique de la société, ils seront à jamais des exemples à suivre.

Nous remercions grandement Mme Bougherira Hamida d'avoir encadré notre travail, tout en nous encourageant et une source de connaissance et sans qui le projet n'aurait pas pu voir le jour.

Nous remercions Mr Melliani d'avoir grandement contribué à la conception du dispositif optique et d'avoir été d'une grande patience envers nous.

Nous remercions l'Université de Saad Dahleb de nous avoir prodigué un bon enseignement dont on a grandement tiré profit pour notre projet.

Nous remercions nos frères et sœurs de nous avoir été d'une grande aide morale, qui nous a permis de prendre courage tout au long. Enfin nous remercions tous nos proches d'avoir contribué à notre évolution personnelle.

Résumé :

Pour notre projet nous avons conçu un système opto-électromécanique, permettant de pallier aux problèmes de précision et de temps de traitement par laser de la rétinopathie. Cela est assuré à travers la création d'un réseau de neurone à convolution (par Deep Learning), ayant pour rôle la détection et la poursuite de maladies rétiniennes pour résoudre le problème de précision. Nous avons implémenté une deuxième méthode basée sur l'utilisation de la fonction de poursuite de Matlab afin d'illustrer le principe de poursuite par DMD. Afin de palier au problème du temps de traitement, avons utilisé le composant DMD, avec son contrôleur Vialux V-7000, pour façonner un faisceau laser compact, en un ensemble de rayons lasers projetés simultanément sur plusieurs zones atteintes de la rétine, afin de les traiter. Nous présentons l'ensemble des résultats de nos implémentations logicielles et matérielles dans notre mémoire.

Mots clés : DMD, modulation de faisceau laser, système optique, rétinopathie, deep learning, convolution, intelligence artificielle, poursuite.

Abstract :

For our project we have designed an opto-electromechanical system, to overcome the problems of precision and laser treatment time of retinopathy. This is ensured through the creation of a convolutional neuron network (by Deep Learning), whose role is the detection and tracking of targets to solve the problem of precision. We have implemented a second method based on the use of the Matlab tracking function to illustrate the principle of tracking using the DMD. In order to overcome the problem of processing time, we used the DMD component, with its Vialux V-7000 controller, to shape a compact laser beam, into a set of laser rays projected simultaneously on the several affected areas of the retina, in order to treat them. We present all the results of our software and hardware implementations in our thesis.

Keywords: DMD, laser beam modulation, optical system, retinopathy, deep learning, convolution, artificial intelligence, pursuit.

ملخص

بالنسبة لمشروعنا ، قمنا بتصميم نظام كهروميكانيكي بصري ، للتغلب على مشاكل الدقة والوقت في علاج اعتلال الشبكية بالليزر. يتم ضمان ذلك من خلال إنشاء شبكة عصبية تلافيفية (بالتعلم العميق) ، يتمثل دورها في اكتشاف وتتبع الأهداف من .DMDلتوضيح مبدأ التتبع بال Matlab لحل مشكلة الدقة. قمنا بتطبيق طريقة ثانية تعتمد على استخدام وظيفة تتبع الخاص به ، Vialux V-7000 ، مع جهاز التحكم DMD أجل التغلب على مشكلة وقت المعالجة ، استخدمنا مكون لتشكيل حزمة ليزر مدمجة، في مجموعة من أشعة الليزر لاسقاطها في وقت واحد على كل المناطق المتأثرة في الشبكية ، من أجل علاجها. نقدم جميع نتائج برامجنا وتطبيقات الأجهزة في مذكرتنا.

كلمات

DMD, تعديل شعاع الليزر, النظام البصري, اعتلال الشبكية, التفاف, تعلم عميق, ذكاء إصطناعي, مطاردة.

Listes des acronymes et abréviations

DMD : Digital Mirror Divice

FPGA : Field Programmable Gate Arrays

L1 : lentille une

L2 : lentille deux

L3 : lentille trois

C1 : cible une

C2 : cible deux

DL : Deep learning

ML : Machine learning

IA : Intelligence artificielle

CNN : Convolution Neural Network

MOEMS : Micro-Opto-Electro-Mechanical-Systems

DCNN : Deep Convolution Neural Network

VB : Visual Basic

Fig : Figure

Table des Matières

Introduction Générale 1

Chapitre I Généralités sur la rétine, la rétinopathie diabétique, les dispositifs optiques et l'apprentissage profond 4

I.1 Introduction 5

I.2 Généralités sur la rétine et la rétinopathie 5

I.2.1 Description de l'œil humain 5

I.2.2 Anatomie de l'œil 6

I.2.3 La rétine 6

I.2.4 La rétinopathie 6

I.2.5 La rétinopathie diabétique 7

I.2.6 La photocoagulation au laser 8

I.3 Dispositifs optiques et DMD 9

I.3.1 Le DMD 9

I.3.2 Contrôleur FPGA 10

I.3.3 La lentille convergente 11

I.3.4 Le laser 12

I.4 Apprentissage profond et Réseaux de neurones à convolution 13

I.4.1 Apprentissage automatique 13

I.4.2 Apprentissage profond 14

I.4.3 Différence entre l'apprentissage profond et l'apprentissage automatique 14

I.4.4 Réseaux de neurones à convolution CNN 15

I.5 Conclusion 17

Chapitre II Conception du système opto-électromécanique et de sa commande pour la poursuite de cibles par DMD 18

II.1 Introduction 19

II.2 Commande et contrôle du DMD 19

II.2.1 Commande du DMD par Vialux 19
II.2.2 Commande du DMD par Visual Studio 2017 20
II.3 Conception du dispositif 29
II.3.1 Disposition des éléments 29
II.3.2 Mesure et calcul 30
II.4 Principe de fonctionnement du dispositif optique 31
II.4.1 Phénomène de diffraction dû au DMD 32
II.4.2 Affichage simultané sur les cibles 32
II.5 Conclusion 32
Chapitre III Implémentations logicielles 33
III.1 Introduction 34
III.2 Programme de contrôle du DMD implémenté sous Visual Studio 2017 34
III.2.1 La structure du sous-programme de l'interface de contrôle 35
III.2.2 Le sous-programme de l'interface de la webcam 35
III.2.3 Sous-programme de l'interface de contrôle du DMD 37
III.3 Programmes implémentés sous Matlab pour la création des bases de données pour le DL 38
III.3.1 Implémentation de la première base de données 38
III.3.2 Implémentation de la deuxième base de données 45
III.4 Algorithme de DL pour la pour poursuite 46
III.4.1 Environnement : Python Tensorflow, Anaconda 46
III.4.2 Les outils nécessaires pour l'exécution du modèle 48
III.4.3 Programme d'appels des bibliothèques et des fonctions 50
III.5 Algorithme de poursuite par comparaison de pixel et par la fonction Vision Template Matcher de Matlab 52
III.5.1 Algorithme de poursuite par comparaison de pixel 52
III.5.2 Algorithme de poursuite par la fonction Vision Template Matcher 55
III.6 Conclusion 56

Chapitre IV Implémentations et résultats 57
IV.1 Introduction 58
IV.2 Implémentation et commande du dispositif optique 58
IV.2.1 Affichage visuel sur le DMD 59
IV.2.2 Les différents états du faisceau laser 59
IV.2.3 Génération des motifs à afficher sur le DMD sous VB 62
IV.2.4 Poursuite de la cible par Deep learning 64
IV.2.5 Poursuite de la cible avec l'algorithme développé sous Matlab 67
IV.3 Conclusion 72
Conclusion Générale 73
Bibliographie 76
Annexes 79

Liste des figures

Figure I.1. Les différentes couches et composants de l'œil 5
Figure I.2. Schéma coupe de la rétine centrale 6
Figure I.3. Différence entre l'oeil normal et d'un oeil atteint de la rétinophathie diabétique 7
Figure I.4. Photocoagulation au laser (Argon) d'une RD 8
Figure I.5. DMD et son contrôleur 9
Figure I.6. Représentation du DMD avec les micromiroirs 10
Figure I.7. Schéma de transfert de donner du V-7000 10
Figure I.8. Lentille convergente 11
Figure I.9. Lentille convergent le faisceau lumineux 11
Figure I.10 Exemple de Laser 12
Figure I.11 Relation entre AI, ML et DL 13
Figure I.12. Procéder de l'apprentissage automatique comparé à l'apprentissage profond 15
Figure I.13. Couches de réseaux de neurones convolutifs 16
Figure II.1. Fenêtre principale de Vialux Gmbh - ALP Demo 19
Figure II.2. Gestionnaire de configurations- par défaut (Any CPU) 20
Figure II.3. Modification dans le gestionnaire de configurations 20
Figure II.4. Création de nouvelle plateforme 21
Figure II.5. Nouvelle plateforme « x86 » 21
Figure II.6. Procédure d'ajout du fichier .dll 22
Figure II.7. Importation du fichier alpV42.dll 22
Figure II.8. Interface de contrôle 23
Figure II.9. Création de projet Visual Basic Forms 23
Figure II.10. Organigramme de communication du DMD et de le Webcam 25
Figure II.11. Interface de contrôle DMD 26
Figure II.12. Interface de commande de la Webcam 26
Figure II.13. Etape 1 importation de VideoCap 27
Figure II.14. Etape 2 importation de VideoCap 28
Figure II.15. Visualisation de la webcam 28
Figure II.16. Dispositif opto-électromécanique 29
Figure II.17. Distance focale 30

Figure II.18. Schéma de fonctionnement du dispositif optique..... 31
Figure III.1. Organigramme fonctionnel de l'interface graphique de Contrôle 34
Figure III.2. Sous-programme de l'interface de contrôle..... 35
Figure III.3. Sous-programme du bouton Turn On 35
Figure III.4. Sous-programme du bouton Capture pour l'acquisition de l'image de la cible 36
Figure III.5. Sous-programme des boutons Turn Off et Close (fermeture de la webcam et de l'interface)..... 36
Figure III.6. Sous-programme du bouton Alloc 37
Figure III.7. Sous-programme du bouton Pattern 1..... 37
Figure III.8. Sous-programme du bouton Pattern 1..... 38
Figure III.9. Etapes de création de la base d'apprentissage 39
Figure III.10. Nombres aléatoires [10x10] obtenues avec la fonction Random..... 40
Figure III.11. Nombres arrondis avec la fonction Round..... 40
Figure III.12. Lecture de l'image de puis le répertoire de sauvegarde 40
Figure III.13. Binarisation de l'image lu 41
Figure III.14. Image modèle (10x10) de rétine atteinte d'une anomalie..... 41
Figure III.15. Principe utilisé pour insérer l'image (Img) 10x10 dans l'image (Imgf) 32x32 42
Figure III.16. Image 10x10 insérée dans l'image 32x32 42
Figure III.17. Exemples de déplacement de l'image (Img) 10x10 dans l'image (Imgf) 32x32 43
Figure III.18. Photographie du fond d'œil atteinte d'une rétinopathie diabétique et image modèle simulant une rétine atteinte 44
Figure III.19. Le principe suivi pour créer des simulations d'images 32x32 de la base de données 44
Figure III.20. Rétine atteinte de la RD 45
Figure III.21. Image de la rétine insérée dans un fond noir et déplacée..... 46
Figure III.22. Logo d'Anaconda et de la bibliothèque Tensorflow..... 47
Figure III.23. Commande Prompt d'Anaconda..... 47
Figure III.24.interface de LabelImg 49
Figure III.25. Importations des bibliothèques 50
Figure III.26. Importation du module de détection 51
Figure III.27. Préparation du modèle 51

Figure III.28. Fonction de la détection 52
Figure III.29. Organigramme du programme 53
Figure III.30. Organigramme fonctionnel des étapes 55
Figure III.31. Programme pour la poursuite et son organigramme fonctionnel 55
Figure IV.1. Schéma de fonctionnement du dispositif et de l'interface DMD 58
Figure IV.2. Dispositif opto-électromécanique 58
Figure IV.3. Affichage d'image de damier et d'un motif 59
Figure IV.4. Etat du laser avant et après traversée de la lentille L1 59
Figure IV.5. Faisceau laser avant L1 60
Figure IV.6. Faisceau laser après L1 60
Figure IV.7. Etat du laser avant et après traversée des lentilles L2, L3 61
Figure IV.8. Faisceau laser avant L2 ou L3 61
Figure IV.9. Faisceau laser après les lentilles L2 et L3 61
Figure IV.10 Représentation l'image en damier 62
Figure IV.11. Image damier projetée par le DMD sur la cible 63
Figure IV.12. Représentation du motif généré 64
Figure IV.13. Image projetée par le DMD sur la cible 64
Figure IV.14. Exemples d'images de la 1ère base de données 65
Figure IV.15. Résultat obtenu de la première base de données 65
Figure IV.16. Exemples d'images de la deuxième base de données 66
Figure IV.17. Début d'exécution de l'entrainement 66
Figure IV.18. Fin d'exécution de l'entrainement 66
Figure IV.19. Graphe obtenu à la fin de l'entrainement 67
Figure IV.20. Images utilisées pour la détection 68
Figure IV.21. Résultats obtenue par comparaison de pixels 68
Figure IV.22. Position du modèle (10x10) dans l'image (32x32) 69
Figure IV.23. Poursuite et détection du modèle 69
Figure IV.24. Résultats obtenu avec vision template matcher 70
Figure IV.25. Les coordonnées obtenues avec vision template matcher 70
Figure IV.26. Images capturées de la cible 71
Figure IV.27. Résultats obtenus en condition réelle 71
Figure IV.28. Résultat obtenu de l'image droite 71
Figure IV.29. Résultat obtenu de l'image de gauche 72

Liste des tableaux

Tableau I-1. Différence entre apprentissage profond et apprentissage automatique……15

Introduction Générale

Introduction Générale

La rétinopathie désigne tout dommage de la rétine des yeux pouvant entrainer une déficience visuelle. Elle fait souvent référence à une maladie vasculaire rétinienne ou à des lésions de la rétine causée par un flux sanguin anormal.

Les maladies retiennes constituent une cause majeure de malvoyance et leurs origines sont diverses. On a la dégénérescence maculaire dû au vieillissement, la rétinite pigmentaire qui est d'origine héréditaire et bien d'autres qui peuvent être causées par des complications du diabète [1].

La rétinopathie diabétique (maladie rétinienne causer par le diabète) est la principale cause de cécité chez les personnes en âge de travailler. Selon une étude de 2008, la rétinopathie (maladie rétinienne) représente 5% de la cécité dans la population mondiale et a été classée parmi les maladies oculaires prioritaires par l'organisation mondiale de la santé [2]. Il existe des moyens de traitement mais les problèmes majeurs de ces techniques sont la précision et le temps de traitement, qui sont la cause d'inconfort et de dommages irréversibles parfois pour le patient et l'ophtalmologiste, et qui constitueront la problématique de notre projet.

Le but de notre projet est d'aider à pallier aux problèmes de précision et de temps de traitement par laser de certains types de maladies rétiniennes.

Cela est réalisé à travers la conception et réalisation d'un dispositif optique, qui a pour composant principal le module V-7000 de Vialux, et l'implémentation d'un algorithme d'apprentissage profond (Deep Learning) pour le positionnement de rayons laser.

Le V-module possède un DMD (Digital Micromirrors Device), fabriqué par Texas Instrument, constitué d'une matrice de micro miroirs (1024x768). Contrairement aux méthodes actuelles de traitement standard de photocoagulation par laser, qui traitent les zones malades, une à la fois, notre système opto-électromécanique nous permettra d'effectuer un traitement simultané des zones à problèmes, réduisant ainsi considérablement le temps de traitement. Le réseau de neurones à convolution (CNN) créé à la suite de l'apprentissage profond, aura pour rôle de détecter tout mouvement de la rétine (sur des images obtenues en temps réel par une WEBCAM) lors du traitement, et de repositionner le faisceau laser façonné sur les zones atteintes; ce qui permettra d'obtenir une grande précision. Le faisceau laser façonné, en un ensemble de rayons laser, est obtenu par réflexion, d'un faisceau laser plein, sur les micromiroirs du DMD sur lequel est affichée une image Noir et Blanc indiquant les positions exactes des zones atteintes de la rétine.

Afin de décrire notre travail nous avons organisé notre mémoire comme suit :

Le premier chapitre expose des généralités sur la rétine, les dispositifs et éléments optiques de base, ainsi que le DMD.

Le chapitre deux quant à lui rentre dans le vif du sujet, à savoir, la conception du dispositif optique. Ce dernier est composé de deux parties : une unité de contrôle et une unité optique. Pour l'unité de contrôle, il y est expliqué comment nous contrôlons le V-7000 par PC. Pour la partie optique, le calcul des paramètres optiques nécessaires à la conception du dispositif optique (autour du DMD), et le principe de fonctionnement du système y sont détaillés.

Le chapitre trois est dédié à l'ensemble des logiciels que nous avons développés. Il contient notamment le programme de la création de notre base de données implémenté sous Matlab, les programmes de contrôle du V-7000 implémentés sous Visual Basic sous l'environnement de développement Visual Studio, et, pour la détection et la poursuite de cible, le programme de deep learning développé sous python, et le programme de poursuite de Matlab.

Le chapitre quatre étant le dernier, met en évidence tous les résultats obtenus lors des différentes implémentations. Notamment l'affichage des images numériques par le DMD, la création des rayons laser à partir d'un faisceau laser par le DMD, ainsi que la détection et la poursuite de cible à travers la webcam.

Chapitre I

Généralités sur la rétine, la rétinopathie diabétique, les dispositifs optiques et l'apprentissage profond

I.1 Introduction

Dans ce chapitre nous allons présenter les notions de base. Dans la première partie, on va définir la rétine, la rétinopathie, montrer le principe du traitement par photocoagulation par laser. Après on va définir les éléments constitutifs de notre dispositif optique et enfin on va donner des notions de bases sur l'apprentissage profond et les réseaux de neurones à convolution.

I.2 Généralités sur la rétine et la rétinopathie

I.2.1 Description de l'œil humain

Pour mieux cerner ce qu'est la rétine et où elle se situe dans l'œil, une étude de l'œil humain est nécessaire.

L'œil, appelé aussi globe oculaire est l'organe de la vue de l'humain. Son but est de transformer l'information lumineuse captée en influx nerveux transmis au cerveau. Il correspond à une petite boule de 2,5 cm de diamètre et pèse 7 à 8 grammes. Il comprend différentes régions qui lui permettent d'assurer sa fonction, de la cornée jusqu'à la rétine [3].

La figure I.1 illustre les composants de l'œil.

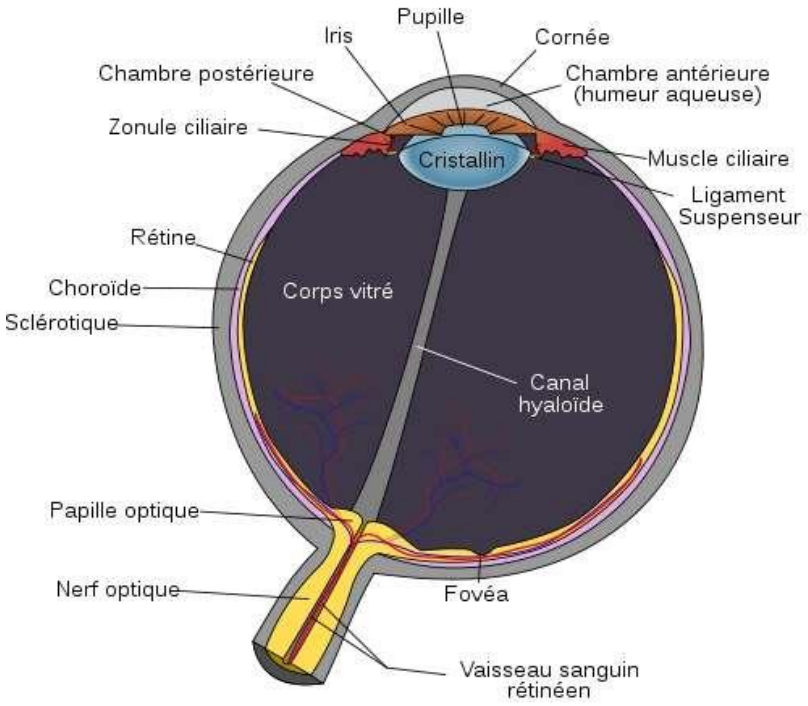


Figure I.1. Les différentes couches et composants de l'œil

I.2.2 Anatomie de l'œil

L'œil comprend plusieurs parties :

- Les parties externes de l'œil : la cornée, la sclère, la conjonctive, les paupières, les larmes.
- Les parties internes visibles : l'iris, la pupille.
- Les parties internes non visibles : la rétine, la fovéa ou macula, le nerf optique, le corps vitré [4].

I.2.3 La rétine

La rétine se situe dans le fond du globe oculaire, permettant la transformation des rayons lumineux en un signal nerveux. Elle est composée de cellules en cônes et en bâtonnets, sensibles aux détails de l'environnement et à la luminosité. C'est à ce niveau que le signal lumineux va être traduit en message nerveux, et être envoyé au cerveau par le nerf optique [5].

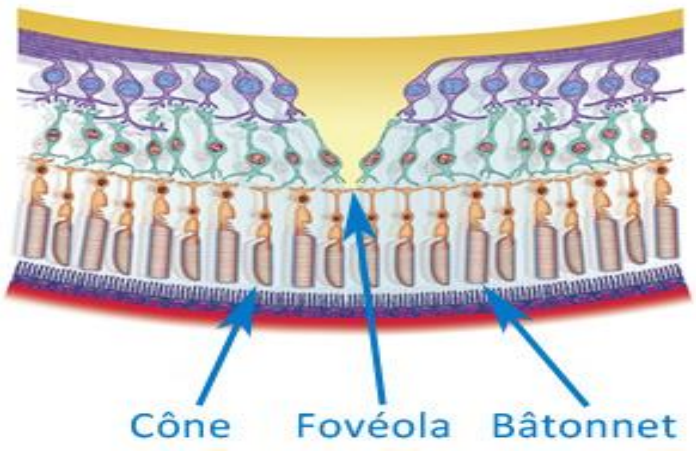


Figure I.2. Schéma coupe de la rétine centrale

I.2.4 La rétinopathie

La rétinopathie est une maladie de la rétine, la membrane transparente au fond de l'œil qui assure notre vision. Elle peut mener à une cécité (devenir aveugle) totale ou partielle si elle n'est pas repérée à temps. Elle est principalement causée par le diabète ou l'hypertension. Ses conséquences vont d'une gêne de la vision, à une cécité totale (devenir aveugle).

Elle est traitée selon sa cause, contrôle glycémique (le taux de sucre dans le sang) ou de pression artérielle, voire chirurgie laser [6].

Il existe différents types de rétinopathie :

- La rétinopathie diabétique
- La rétinopathie hypertensive
- La rétinopathie du prématuré
- La rétinopathie photique
- La rétinopathie des radiations

Bien que les rétinopathies les plus fréquentes restent La rétinopathie diabétique et La rétinopathie hypertensive.

I.2.5 La rétinopathie diabétique

La rétinopathie diabétique est une des complications graves du diabète de type 2. Cette maladie est causée par un niveau de sucre trop élevé dans le sang, fragilisant ainsi les petits vaisseaux de la rétine. Ces derniers, perdent leur étanchéité puis éclatent. Il en découle une baisse de l'acuité visuelle causée par le manque d'oxygène de certaines régions de la rétine qui ne sont plus irriguées. De nouveaux vaisseaux sanguins sont produits par la rétine pour c compenser la perte des précédents, créant un épaississement du centre de la rétine (la macula) et un œdème maculaire apparaît [7].

Au début des stades précoces de la rétinopathie diabétique, il est possible qu'aucun symptôme ne soit perceptible. Au fur que l'affection évolue, les symptômes comportent : une vision floue, des fluctuations de la vision, des taches aveugles, modification de la perception etc…

La figure I.3 illustre une comparaison d'un œil normal et d'un œil atteint de la RD [8].

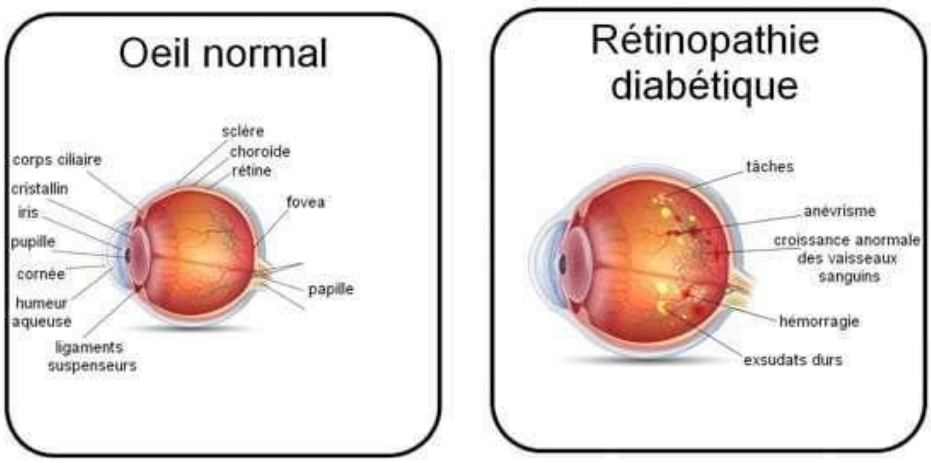


Figure I.3. Différence entre l'oeil normal et d'un oeil atteint de la rétinophathie diabétique

I.2.6 La photocoagulation au laser

La photocoagulation au laser est une méthode thérapeutique pratiquée en ophtalmologie, elle consiste à appliquer un faisceau laser sur la structure la plus profonde de l'œil, la rétine, avec pour objectif de produire une brûlure thérapeutique dans une zone sélectionnée.

Le Laser Argon est une lumière colorée, à haute énergie, il peut être centré de manière très précise sur un point de la rétine. Il est possible de traiter une petite zone de la rétine, ainsi qu'une zone plus grande. La photocoagulation a entre autre permis d'éviter la dégradation de la capacité visuelle qui accompagne certaines maladies, comme **la rétinopathie diabétique** et le décollement de rétine [9].

La Figure schématise un exemple de Photocoagulation par laser lors d'un traitement d'une rétinopathie diabétique pour coaguler des vaisseaux qui sont anormaux.

1. Laser
2. Pointe de laser
3. Vaisseaux sanguins anormaux

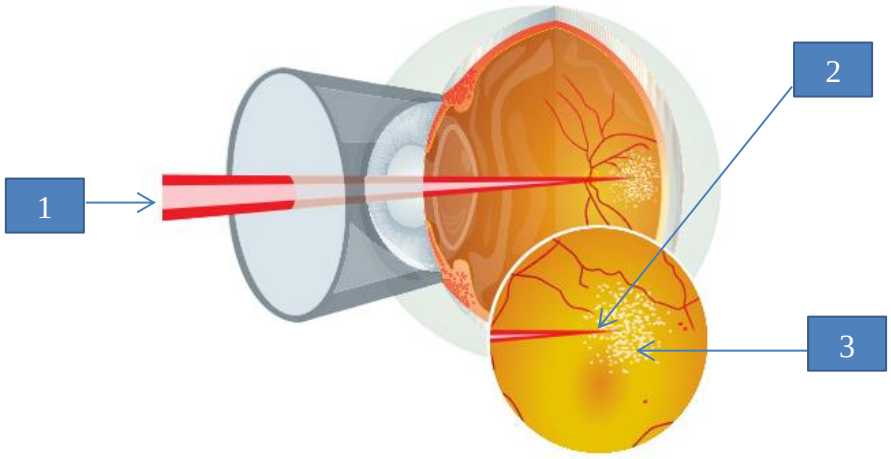


Figure I.4. Photocoagulation au laser (Argon) d'une RD

La technique se déroule à l'aide d'un verre de contact et d'un microscope doté d'un laser. Habituellement, on utilise un laser de type Argon. En règle générale, le patient peut retourner chez lui après le traitement. Dans les premières heures suivant l'intervention, une vision floue, un mal de tête et une douleur peuvent survenir. Notre projet a pour but de diminuer cet état

d'inconfort en proposant une solution qui permet de projeter plusieurs pointes de laser simultanément sur tous les points atteints de la rétine.

Le principe utilise des micromiroirs (digital micromirrors device DMD) pour façonner un faisceau laser uniforme en un ensemble de rayons lasers projetés sur les points d'hémorragie dans la rétine. Nous décrivons le DMD dans la section suivante.

I.3 Dispositifs optiques et DMD

Le dispositif optique est constitué de divers composants adapté à l'utilisation prévus. Il est composé principalement d'un DMD (Digital Mirror Device), d'une carte de contrôle FPGA (File Programmable Gate Arrays) plus précisément le VIALUX, d'un Laser, de 3 lentilles convergentes, de dioptries variables et d'un ordinateur portable avec une webcam pour le contrôle.

I.3.1 Le DMD

Le Digital Mirror Device communément appelé DMD, est un microsystème optique électromécanique fabriqué par Texas Instrument, constitué d'une matrice de micro miroir. Il permet par interaction avec un faisceau lumineux de reconstituer une image numérique par la réflexion des pixels sur ces micromiroirs. Ces derniers ne peuvent basculer que sur deux positions, On et Off selon le pixel de l'image numérique. Lorsqu'il procède à l'affichage on obtient deux images numériques, l'une positive, l'autre négative et les images numériques obtenues auront la même longueur d'onde que le faisceau utilisé. Le DMD utilisé dans notre dispositif optique a une matrice de micro miroir de 1024 x 768, chacun faisant 13,6 µm de long.

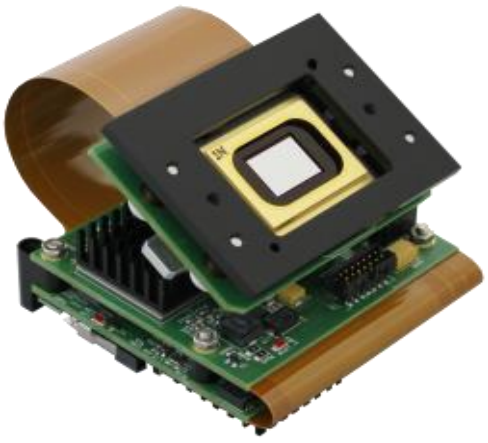


Figure I.5. DMD et son contrôleur

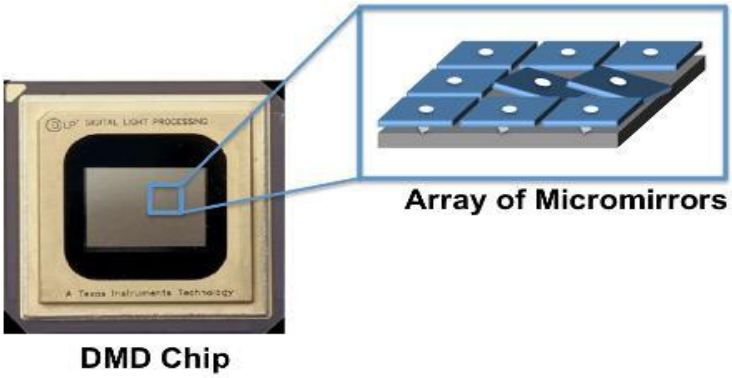


Figure I.6. Représentation du DMD avec les micromiroirs

I.3.2 Contrôleur FPGA

Le Field Programmable Gate Arrays ou communément appelé FPGA est un circuit intégré pouvant être reprogrammé, grâce aux lignes de connections internes du circuit pouvant être repositionné à tout moment par l'utilisateur. Ayant une plus vaste potentialité que les circuits intégrés standards, le FPGA est utilisé de diverses manières pour effectuer divers tâches. Dans notre cas on utilisera le VIALUX V-7000 adapté au DMD.

Le V-7000 est un V-module de grande vitesse, il offre par conséquent un ensemble de haute performance avec un facteur de forme réduite. Il est doté d'une puce DLP7000 d'une vitesse maximale de 22727 commutateurs de réseau mondiaux par seconde. L'interface de l'ordinateur USB 2.0 est accélérée par une compression sans perte, ce qui permet une communication presque instantanée [10].

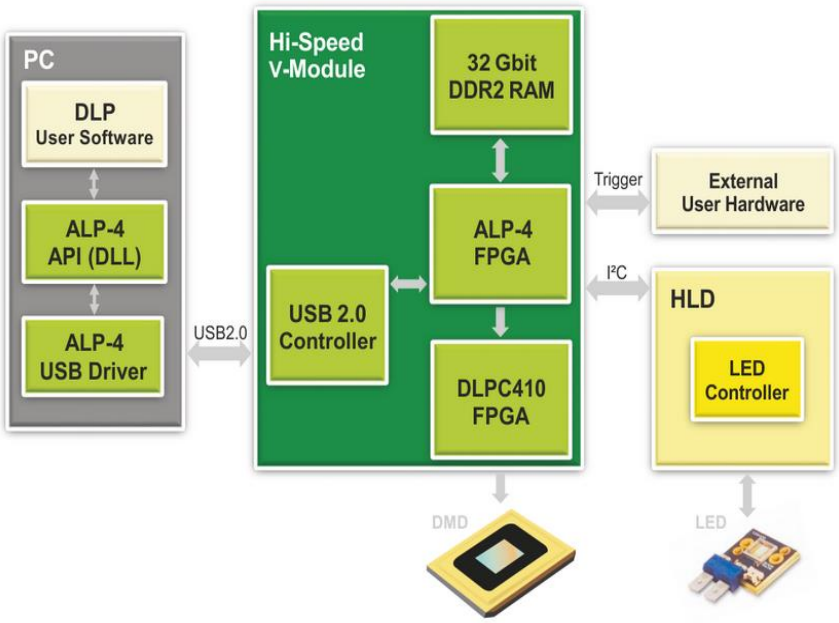


Figure I.7. Schéma de transfert de donner du V-7000

I.3.3 La lentille convergente

Une lentille est un morceau de verre transparent permettant d'agrandir ou de diminuer la taille d'un objet. Lorsque l'on dit qu'elle est convergente cela implique quelle focalise les faisceaux parallèles arrivant en amont (d'un côté vers l'autre). Ainsi tout faisceau parallèle passant d'un côté d'une lentille convergente par l'autre passe forcément par l'axe optique et ce croise au niveau du foyer.

Figure I.8. Lentille convergente

I.3.3.1 Le phénomène de focalisation

Ce phénomène est expliqué par le fait que les faisceaux parallèles venant de la droite vers la gauche (fig. I.9), passent par deux états. Le premier état étant le passage des faisceaux lumineux d'un état plus réfrangeant à un état moins réfrangeant, la deuxième étant d'un état moins réfrangeant à un état plus réfrangeant. Cela provoque une déviation angulaire selon l'angle d'entré, mais comme nos faisceaux lumineux sont parallèles ils passent tous par le foyer de l'autre côté de la lentille convergente, c'est la règle numéro une des lentilles convergentes.

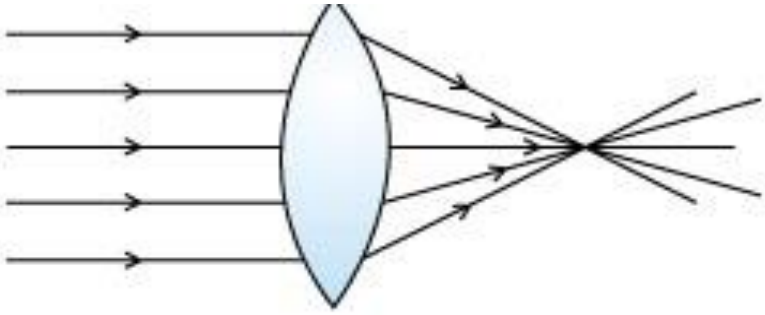


Figure I.9. Lentille convergent le faisceau lumineux

I.3.3.2 Le phénomène de défocalisation

Supposant que notre source lumineuse se trouve toujours à droite comme sur la figure I.9, les faisceaux sont parallèles en théorie. Mais en pratique, cela correspond au fait que notre source lumineuse est située à une distance infinie. Partant de ce principe, lorsque l'on ramène cette source au niveau du foyer on obtient des faisceaux parallèles de l'autre côté ; c'est ce qu'on appelle le phénomène de défocalisation.

I.3.4 Le laser

Un laser est un dispositif qui nous permet d'obtenir un faisceau lumineux centré et directionnel sur une distance infinie en théorie. Cela est dû à l'amplification des faisceaux lumineux d'une diode dans une cavité résonnante, qui fournit un rayonnement d'onde cohérente et monochromatique par émission simulée. Son utilisation dépend du contexte, mais quel qu'en soit son utilité, le paramètre le plus important à prendre en compte est son intensité lumineuse.

Son utilité dans le cas du traitement de maladies rétiniennes, est de nous permettre d'effectuer des points de suture délicate. Vu que les maladies rétiniennes provoquent des lésions ou une formation de Cailloutes dans les veines rétiniennes, qui sont situées sur une partie extrêmement sensible, on ne peut prendre de risque en utilisant un outil d'extraction métallique d'où la nécessiter du laser.

Figure I.10 Exemple de Laser

I.4 Apprentissage profond et Réseaux de neurones à convolution

Dans notre projet nous voulons apprendre à notre système opto-électromécanique, à suivre la rétine pour ajuster le positionnement des rayons laser sur les points à saturer en cas de déplacement de la rétine.

L'intelligence artificielle (IA), ou « Artificial Intelligence (AI) » en anglais, consiste à mettre en œuvre un certain nombre de méthodes visant à permettre aux machines d'imiter une forme d'intelligence réelle.

Cette partie décrit le fonctionnement et la structure des réseaux de neurones à convolution et également de l'apprentissage profond qui est une forme d'intelligence artificielle, dérivée de l'apprentissage automatique (Machine Learning). Pour comprendre ce qu'est l'apprentissage profond, il convient donc de comprendre ce qu'est l'apprentissage automatique.

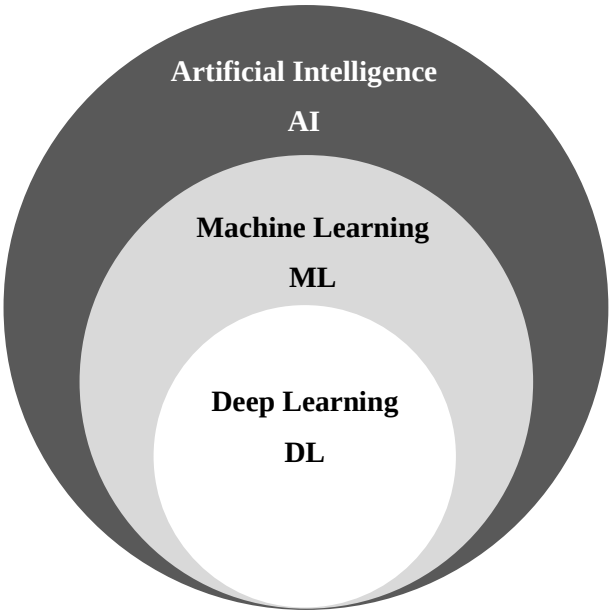


Figure I.11 Relation entre AI, ML et DL

I.4.1 Apprentissage automatique

Le machine learning ou « apprentissage automatique » en français est un concept qui se rapporte au domaine de l'intelligence artificielle. Il s'agit d'une sorte de programme permettant à un ordinateur ou à une machine un apprentissage automatisé.

L'objectif visé est de rendre la machine ou l'ordinateur capable d'apporter des solutions à des problèmes compliqués, par le traitement d'une grande quantité d'informations [11].

I.4.1.1 Les différents procédés d'apprentissage

Le machine learning implique deux principaux systèmes d'apprentissage qui définissent ses différents modes de fonctionnement. Il s'agit de :

- **L'apprentissage supervisé ou analyse discriminatoire :**
  La machine s'appuie sur des classes prédéterminées et sur un certain nombre de paradigmes connus pour mettre en place un système de classement à partir de modèles déjà étiquetés.
- **L'apprentissage non-supervisé ou clustering :**
  Ce mode fonctionnement, le machine learning ne s'appuie pas sur des éléments prédéfinis. La tache revient à la machine de procéder toute seule à la classification des données.

I.4.1.2 Différence entre l'apprentissage supervisé et non supervisé

La différence entre ces deux principes de fonctionnement réside dans le fait que l'apprentissage supervisé peut être influencé par des à priori (bruit) au moment de l'étiquetage des données. Ce n'est pas le cas de l'apprentissage non-supervisé, qui se révèle ainsi beaucoup plus fiable dans la mesure où les réponses obtenues vont plus loin que la compréhension humaine des faits. Par ailleurs, il faut noter que le machine learning peut également faire intervenir un mode de fonctionnement mixte qui utilise les deux types d'apprentissage pour arriver à des résultats plus précis.

I.4.2 Apprentissage profond

L'apprentissage profond ou Deep Learning est une branche de l'apprentissage automatique (Machine Learning) entièrement basée sur des réseaux de neurones artificiels s'inspirant du cerveau humain. Ce réseau est composé de dizaines voire de centaines de « couches » de neurones, chacune recevant et interprétant les informations de la couche précédente. Le système apprendra par exemple à reconnaître les lettres avant de s'attaquer aux mots dans un texte, ou détermine s'il y a un visage sur une photo avant de découvrir de quelle personne il s'agit [12].

I.4.3 Différence entre l'apprentissage profond et l'apprentissage automatique

Le tableau ci-dessous illustre une comparaison entre l'apprentissage automatique et l'apprentissage en profondeur.

Apprentissage automatique	Apprentissage Pronfond
Fonctionne sur une petite quantité de jeu de données pour plus de précision	Fonctionne sur une grande quantité de données
Dépend de la machine de basse gamme	Fortement dépendant de la machine haut de gamme
Divise les taches en sous-tâches, les résout individuellement et enfin combine les résultats	Résout le problème de bout en bout
Prend moins de temps à former	Prend plus de temps à former
Le temps de test peut augmenter	Moins de temps pour tester les données

Tableau I-1. Différence entre apprentissage profond et apprentissage automatique

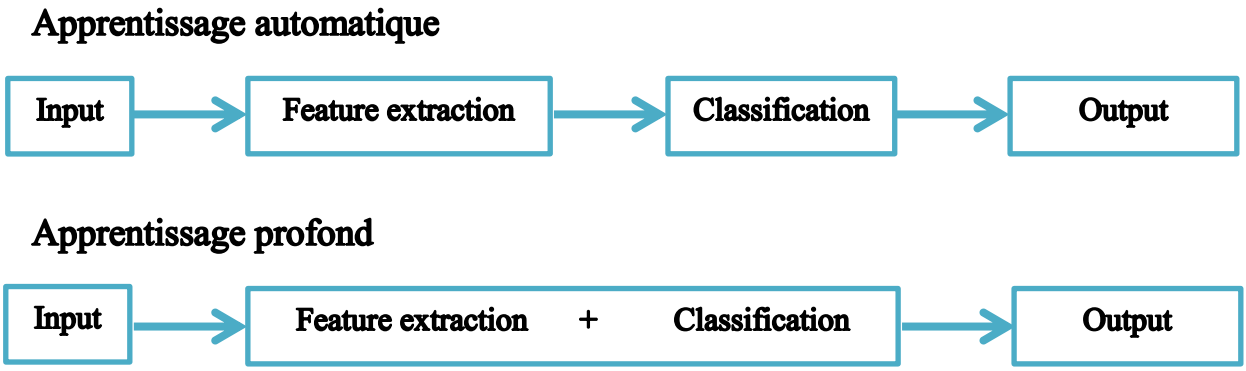


Figure I.12. Procéder de l'apprentissage automatique comparé à l'apprentissage profond

I.4.4 Réseaux de neurones à convolution CNN

Les réseaux de neurones convolutifs sont directement inspirés du cortex visuel des vertébrés. Un réseau de neurones à convolution, appelé aussi convnet (pour «Convolutional Network»), ou encore CNN (pour «Convolutional Neural Network»).

On distingue deux parties, une première partie que l'on appelle la partie convolutive du modèle et la seconde partie, que l'on va appeler la partie classification du modèle qui correspond à un modèle MLP (Multi Layers Perceptron).

Il existe quatre types de couches pour un réseau de neurones convolutifs: la couche de convolution, la couche de pooling, la couche de correction ReLU et la couche fully-connected [13].

La figure (I.13) illustre les différentes couches de réseaux de neurones convolutifs [14].

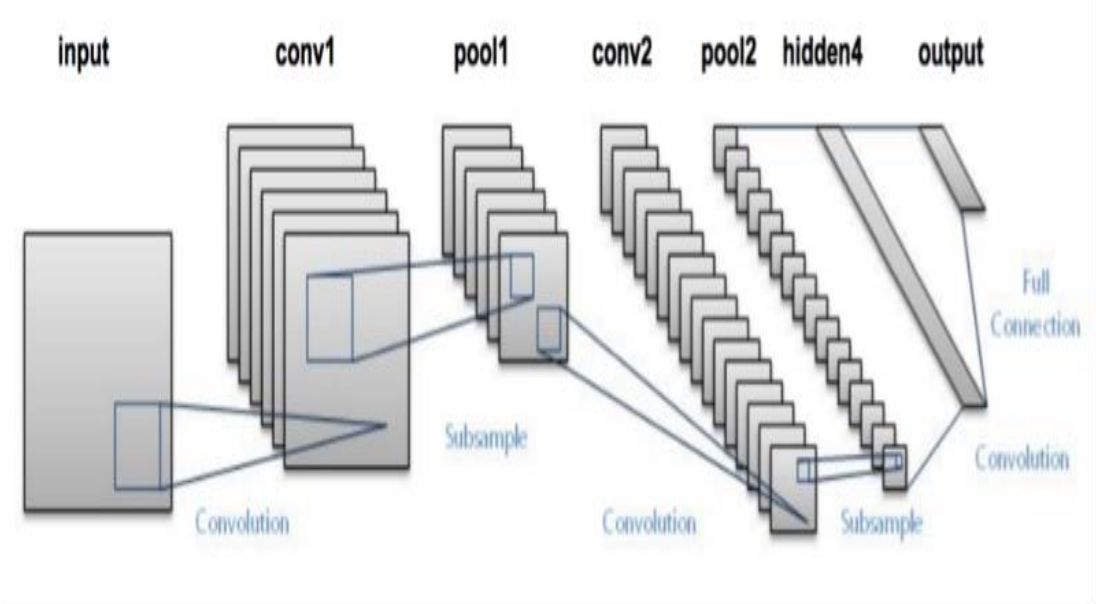


Figure I.13. Couches de réseaux de neurones convolutifs

- **La couche de convolution :**

  La couche de convolution est la composante clé des réseaux de neurones convolutifs, et constitue toujours au moins leur première couche. Son but est de repérer la présence d'un ensemble de features dans les images reçues en entrée.

$$W_0 = \frac{Wi-K+2p}{S} + 1$$

  W_0 : la taille spatiale du volume de sortie

  W_i : la taille volume d'entrée

  K : la surface de traitement

  S : le pas

  P : la taille de la marge

- **La couche de Pooling :**

  Cette couche est souvent placée entre deux couches de convolution : elle reçoit en entrée plusieurs features maps, et applique à chacune d'entre elles l'opération de pooling.

- **La couche de correction ReLU :**

  Pour améliorer l'efficacité du traitement en intercalant entre les couches de traitement une couche qui va opérer une fonction mathématique (fonction d'activation) sur les signaux de sortie. Dans ce cadre on trouve ReLU (Rectified Linear Units) désigne la fonction réelle non-linéaire définie par ReLU(x) = max (0, x).

x : la valeur de sortie des couches de convolutions.

La couche de correction ReLU remplace donc toutes les valeurs négatives reçues en entrées par des zéros. Elle joue le rôle de fonction d'activation.

- **La couche fully-connected :**

  Elle constitue toujours la dernière couche d'un réseau de neurone convolutif ou non, elle n'est donc pas caractéristique d'un CNN. Ce type de couche reçoit un vecteur en entrée et produit un nouveau vecteur en sortie. Pour cela, elle applique une combinaison linéaire puis éventuellement une fonction d'activation aux valeurs reçues en entrée.

I.5 Conclusion

Dans ce chapitre nous avons donné un aperçu général sur l'aspect médical de la rétine et la rétinopathie, afin de mieux cerner ce qu'est la rétine et où elle se situe dans l'œil. Nous avons également donné un résumé sur le traitement par photocoagulation, une méthode qui a fait ses preuves dans le monde de l'ophtalmologie. Nous avons défini les éléments qui composent notre dispositif optique et des notions sur l'apprentissage profond (Deep Learning).

Le principe utilisé pour le traitement par photocoagulation se fait à l'aide d'une assistance d'un ophtalmologiste. Quant à notre projet, nous concevons un dispositif optique autonome. Le chapitre qui suit explique le principe de conception du système opto-électromécanique.

Chapitre II

Conception du système opto-électromécanique et de sa commande pour la poursuite de cibles par DMD

II.1 Introduction

Ce chapitre consiste à mettre en avant une description générale de la conception du dispositif opto-électromagnétique qui se compose de deux parties : une partie dédiée au commande et contrôle du DMD et une partie pour la conception du dispositif optique. Le composant majeur de cette conception est basé sur le dispositif DMD relié de manière industrielle à son contrôleur VIALUX. Nous décrivons la conception du dispositif optique destiné à façonner un faisceau laser par le DMD pour le positionner sur une cible qui se déplace. Il existe différentes manières de contrôler le DMD, dont un logiciel préconçu nommer ALP Tools, mais dans le cadre de notre travail le contrôle sera effectué par un ensemble de programmes que nous avons développés sous le logiciel Visual Studio.

II.2 Commande et contrôle du DMD

Dans cette partie nous présentons les logiciels de commande que nous avons utilisés pour le contrôle du DMD. Dans un premier temps nous utiliserons le logiciel Vialux et dans un deuxième temps le logiciel Visual Studio pour implémenter nos programmes avec le langage de programmation Visual Basic. Pour rappel notre objectif est de générer une image pour l'afficher sur le DMD, afin de façonner le faisceau laser de façon à positionner le motif du faisceau laser (réfléchi par le DMD) sur la cible qui c'est déplacer.

II.2.1 Commande du DMD par Vialux

Nous avons commencé par installer les drivers sur notre PC, Vialux est installé en même temps. Vialux est le logiciel de commande par défaut du DMD via le contrôleur FPGA.

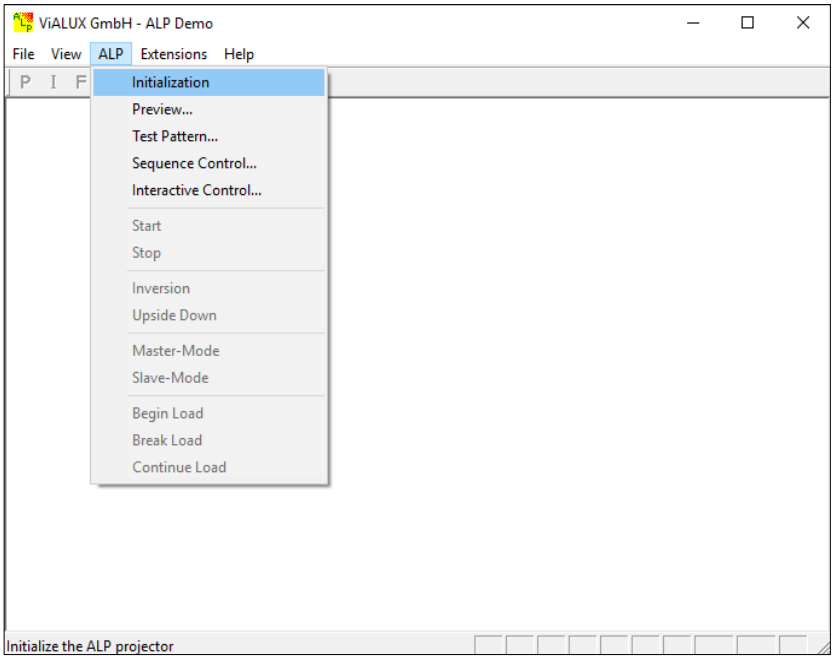


Figure II.1. Fenêtre principale de Vialux Gmbh - ALP Demo

Après connexion du DMD au PC via le port USB, nous avons démarré le logiciel et procédé à la liaison du DMD au logiciel comme suit :

- La liaison du DMD est activée par le bouton « Initialization », la fenêtre qui apparait propose des versions de Alp (dans notre cas Alp-4.2 device) ;
- Choix de séquence « sequence control », pour sélectionner la séquence d'images à envoyer au DMD ;
- Pour charger les images, dans « files » appuyer sur « open » et le choisir le dossier de répertoire.

Le logiciel Vialux nous permet de charger que 5 images sur une séquence, avec un délai de projection variable (nous donnons une fréquence d'affichage des images).

II.2.2 Commande du DMD par Visual Studio 2017

Le logiciel Vialux présente certains inconvénients (délai de projection variable) qui ne nous permettent pas d'atteindre notre but. Nous avons implémenté notre application qui permet de charger nos images, ou générer des images pour envoyer au DMD.

II.2.2.1 Configuration du système pour la communication avec le DMD par Visual Studio

a. Configuration du PC

La configuration dépend du PC sur lequel est exécuté le programme, par défaut c'est « Any CPU » qui est sélectionné. Nous avons configuré selon le CPU de notre PC « x86 » (fig. II.2, II.3, II.4, II.5).

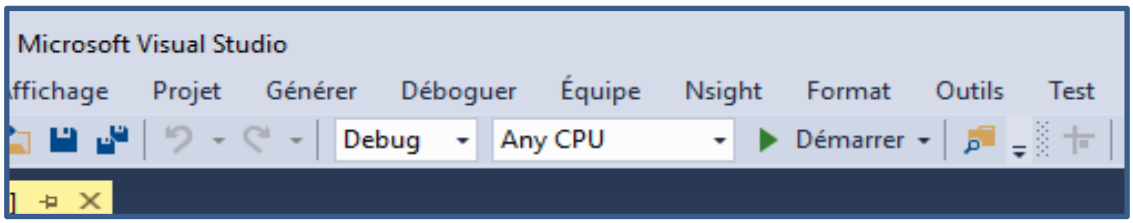


Figure II.2. Gestionnaire de configurations- par défaut (Any CPU)

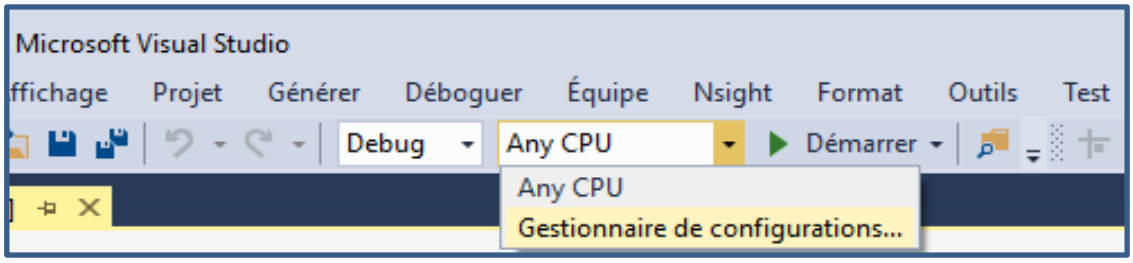


Figure II.3. Modification dans le gestionnaire de configurations

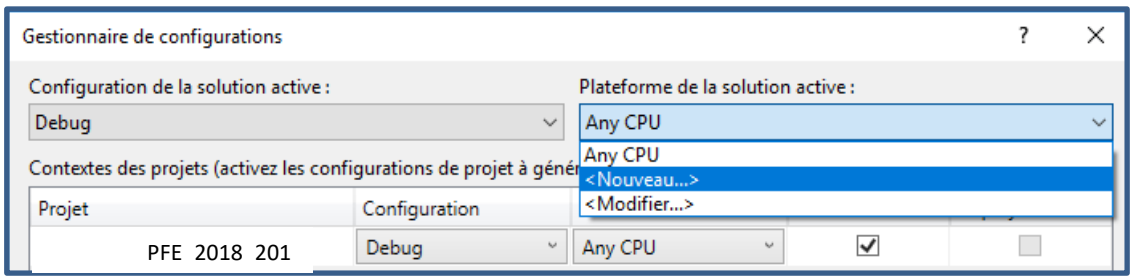


Figure II.4. Création de nouvelle plateforme

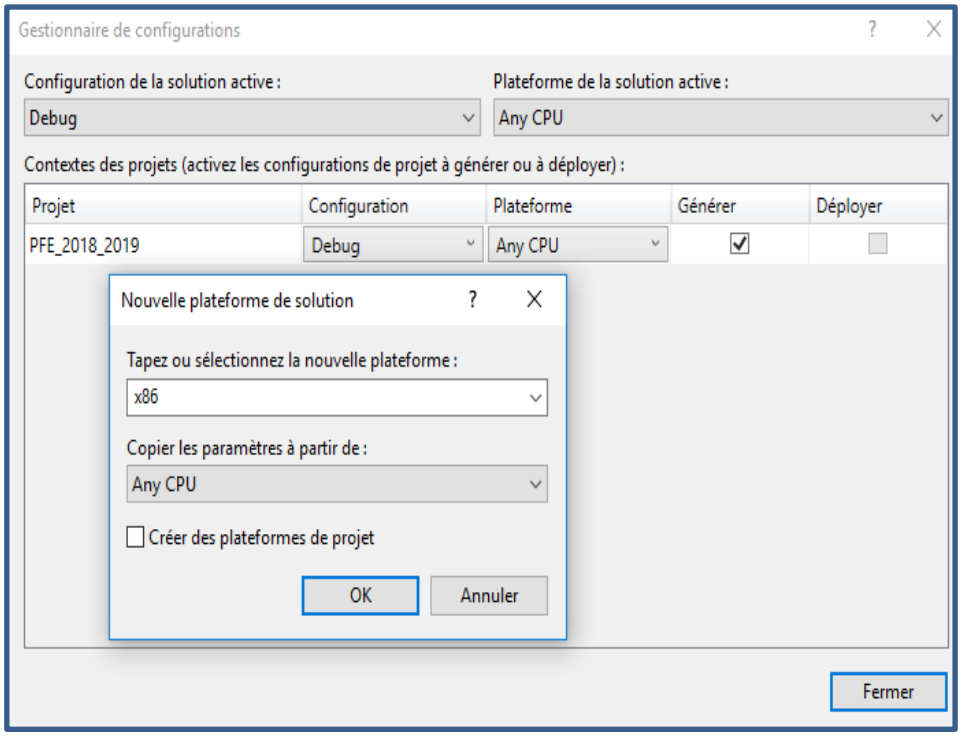


Figure II.5. Nouvelle plateforme « x86 »

b. Emplacement du fichier ALPV42.dll

Le fichier « alpV42.dll » doit être ajouté et placé dans la bibliothèque de notre projet, et aussi dans le dossier « C:\Program Files\ALP-4.2\ALP-4.2 high-speed API\Samples\ALP high-speed VB2005\bin\x86\Debug » (fig. II.6, II.7).

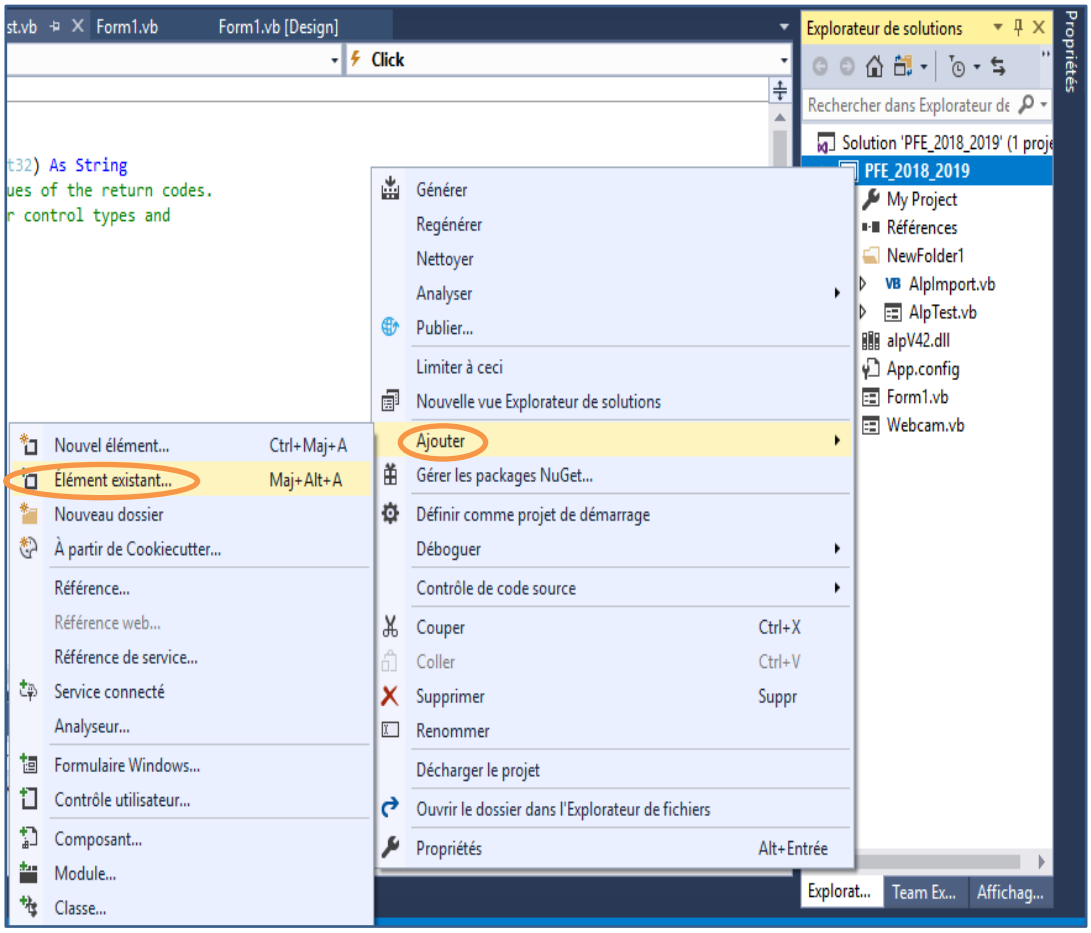


Figure II.6. Procédure d'ajout du fichier .dll

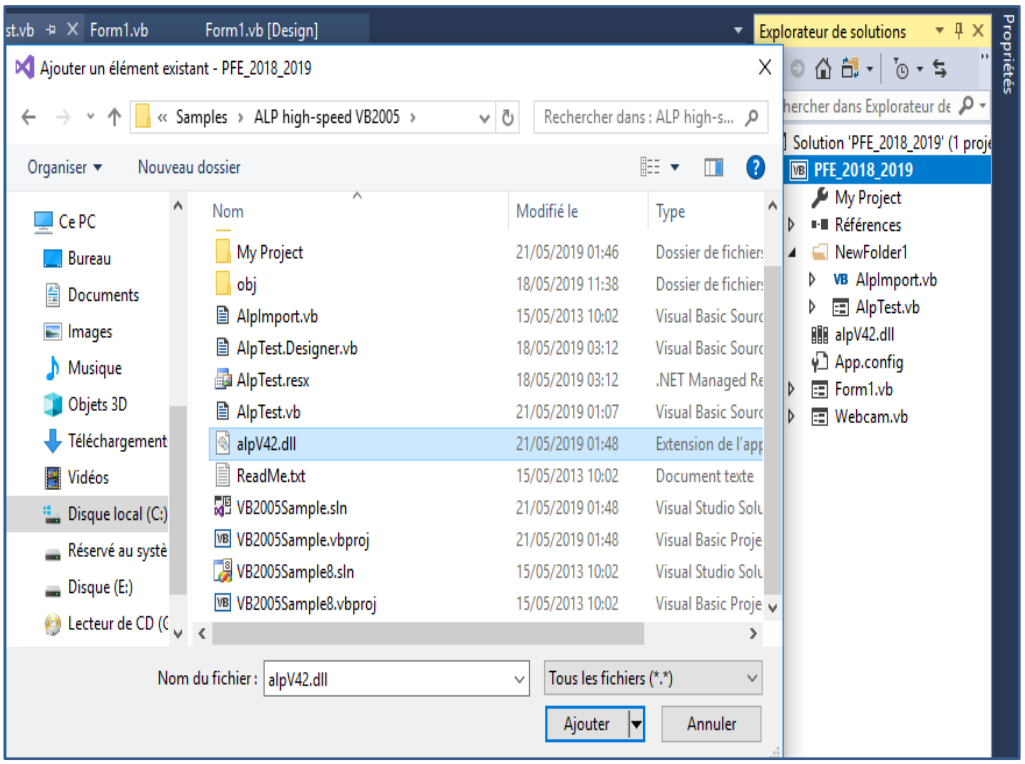


Figure II.7. Importation du fichier alpV42.dll

II.2.2.2 Conception de l'interface graphique pour la commande du DMD

Nous avons créé une première interface (Control) qui contient 3 boutons : DMD, Webcam et Close (voir figure ci-dessous). Les boutons DMD, Webcam permettent d'accéder respectivement à leurs interfaces de commande et le bouton Close pour fermer l'application.

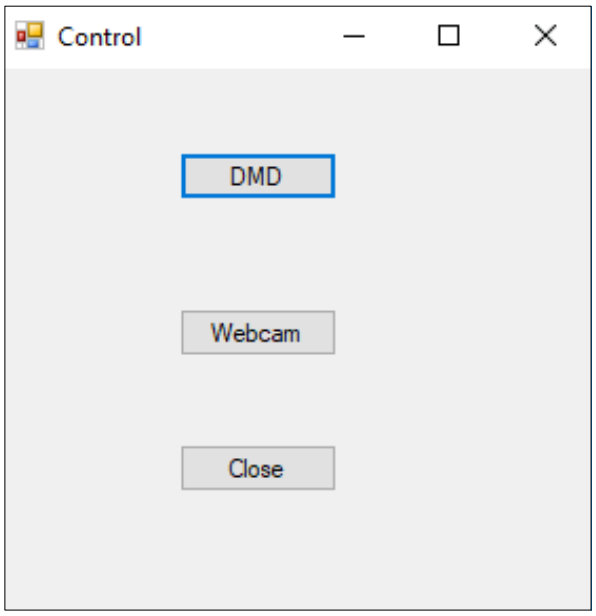


Figure II.8. Interface de contrôle

Nous utilisons le langage de programmation Visual Basic « VB Net » qui est un des outils de Visual Studio. Nous avons créé un nouveau projet et avons utilisé « Application Windows Forms » qui permet de générer des interfaces (fig. II.9).

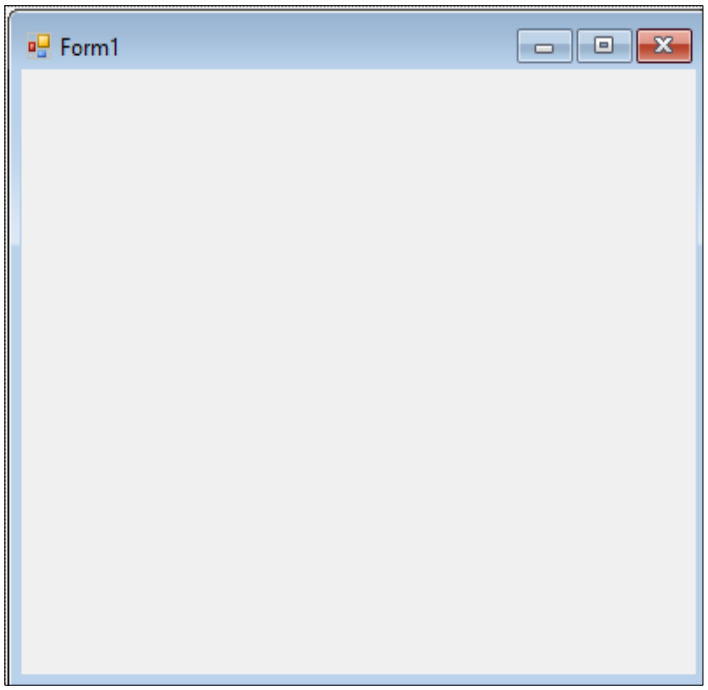


Figure II.9. Création de projet Visual Basic Forms

Le contrôle pour la poursuite de cible, consiste à lire l'image envoyée par la webcam, détecter si un déplacement de la cible a eu lieu, et dans le cas positif générer une nouvelle image pour façonner le rayon laser réfléchi par le DMD de manière à le projeter sur la cible.

L'organigramme (fig. II.10) est une représentation schématique des liens et des relations fonctionnelles de la structure des programmes que nous avons implémentés et que nous décrivons dans la section (a, b).

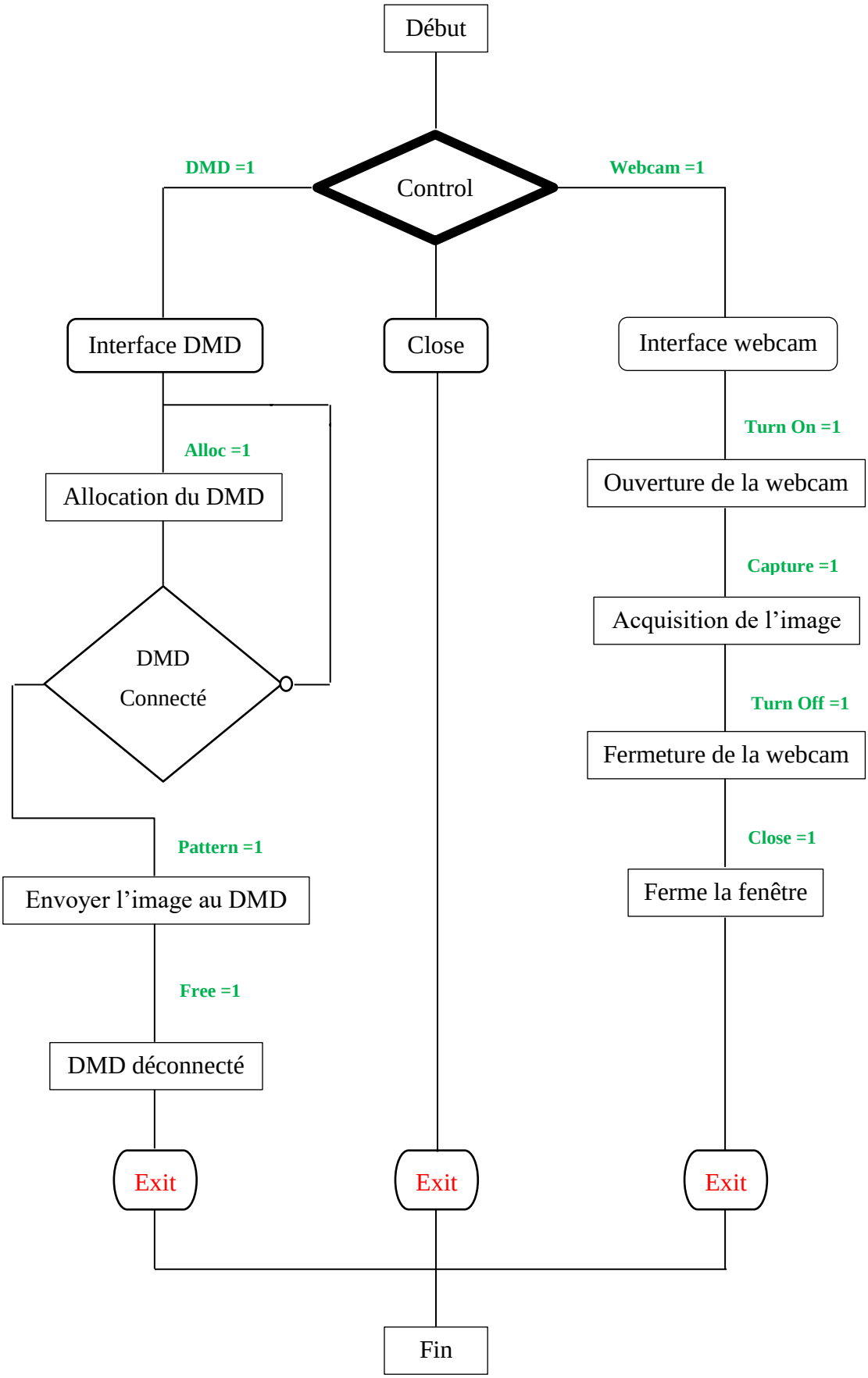


Figure II.10. Organigramme de communication du DMD et de le Webcam

a. Fenêtre de contrôle du DMD

La sélection du bouton DMD de l'interface de contrôle (fig. 1.3) fait appel à l'interface de commande du DMD appelée « ALP Test ». Cette interface permet de faire l'allocation (connexion), d'envoyer des images et de déconnecter le DMD.

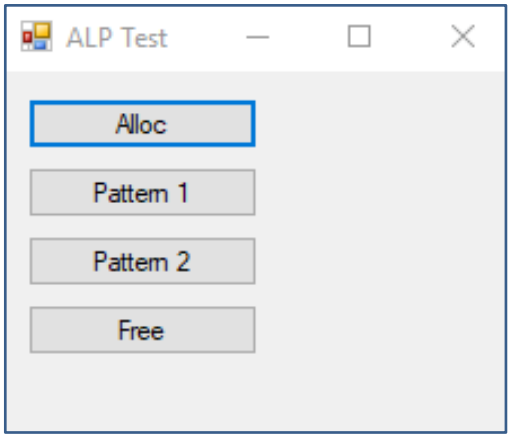


Figure II.11. Interface de contrôle DMD

- Le bouton « **Alloc** » fait appel à la fonction **AlpDevAlloc** responsable de l'allocation du DMD.
- Les boutons « **Pattern 1 & 2** » active la fonction **AlpProjStartCont** qui envoie un pattern au DMD.
- Le bouton « **Free** » fait appel à la fonction **AlpDevHalt** responsable de déconnection du DMD.

b. Fenêtre de contrôle de la webcam

L'interface de la webcam est composée de 4 boutons et une fenêtre « SaveFileDialog ».

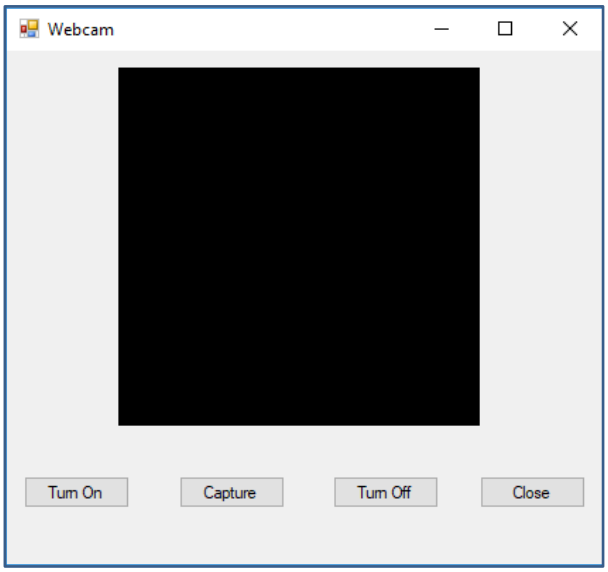


Figure II.12. Interface de commande de la

- Le bouton « Turn On » permet d'activée la webcam du PC.
- Le bouton « Capture » permet de faire l'acquisition d'image via la webcam du PC.
- Le bouton « Turn Off » permet de désactivée la webcam du PC.
- Le bouton « Close » permet de ferme la fenêtre de l'interface de la webcam.
- La fenêtre « SaveFileDialog », c'est l'écran noir situé dans l'interface de commande de la webcam, il permet l'affichage en temps réel lorsque la webcam est activé (bouton Turn ON).

Pour ajouter cette fenêtre nous avons fait quelques configurations : le téléchargement du logiciel VideoCap « https://www.commentcamarche.net/download/telecharger-34060911-videocap » et son importation dans notre projet (fig. II.13, II.14). VideoCap a pour tâche de faciliter les enregistrements d'images et de vidéos capturées par les webcams.

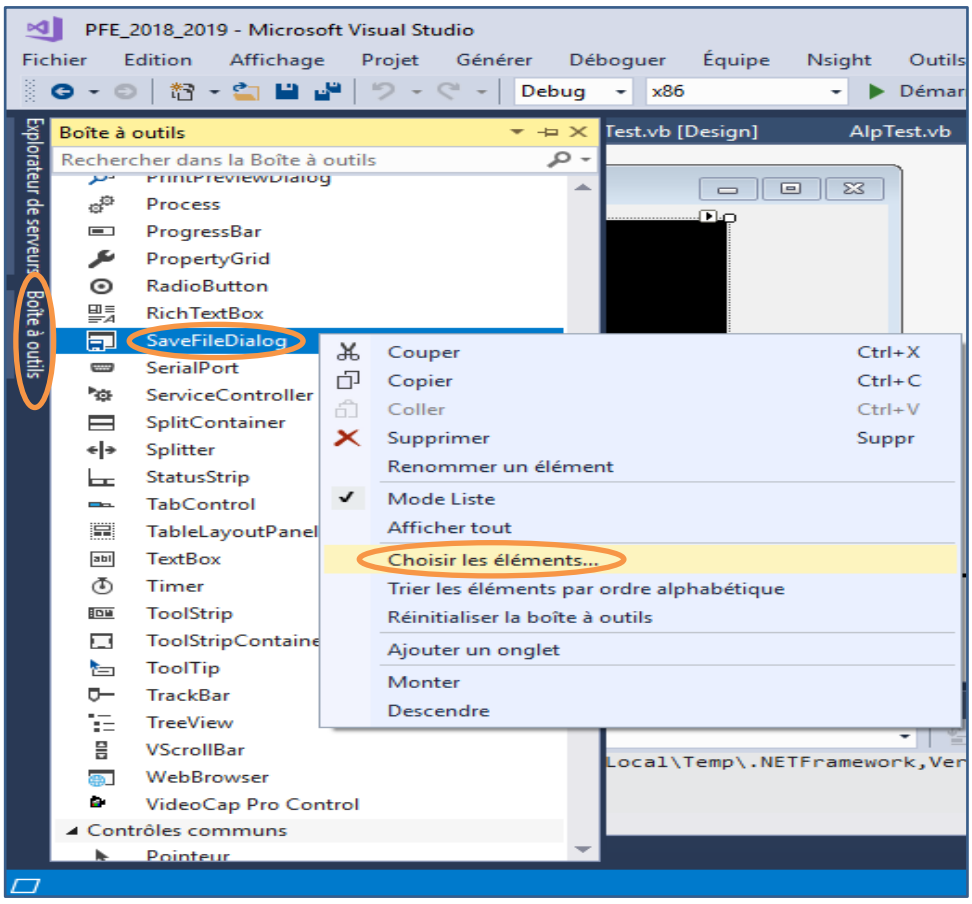


Figure II.13. Etape 1 importation de VideoCap

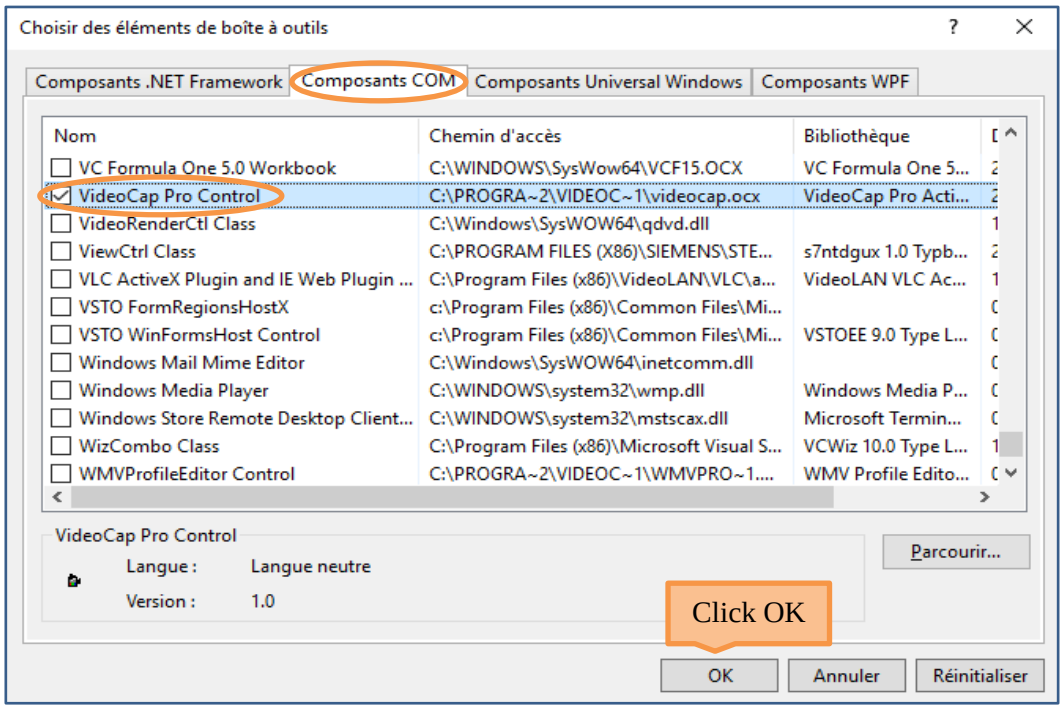


Figure II.14. Etape 2 importation de VideoCap

Après l'importation nous avons tracé notre fenêtre SaveFileDialog dans l'interface de la webcam.

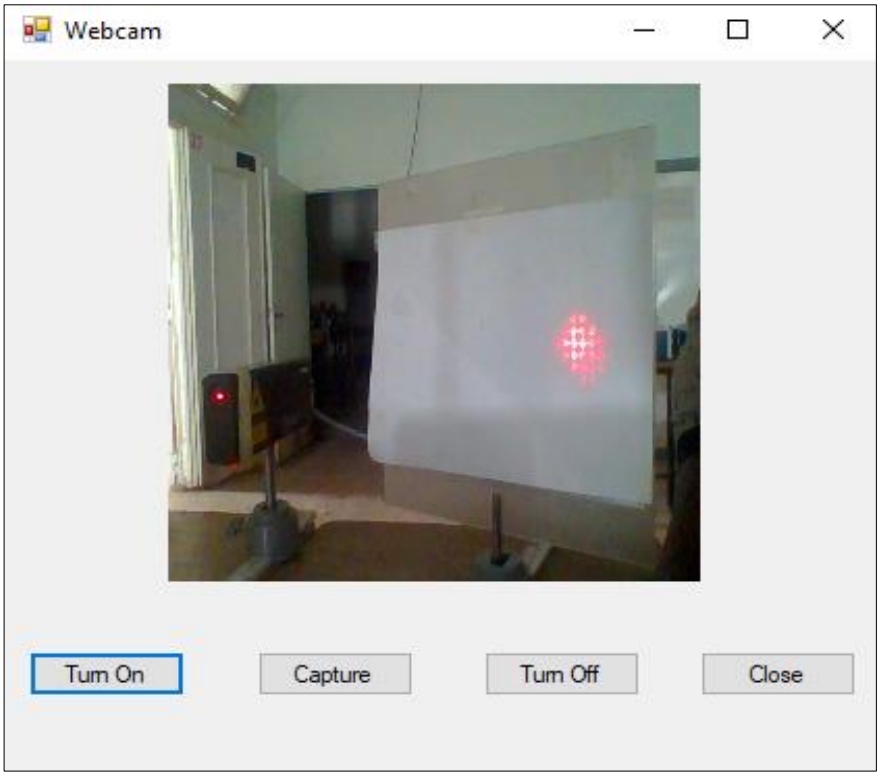


Figure II.15. Visualisation de la webcam

II.3 Conception du dispositif

L'assemblage fit effectuer en premier lieu dans un laboratoire de physique, sur une grande surface plane afin de nous permettre de faire différent calcule. Ces calcules nous ont permis d'obtenir le résultat voulu. Ce résultat étant un affichage des images positif et négative refléter par le module DMD (digital Mirror Divice), sur une courte distance pouvant être place sur une table d'étude.

II.3.1 Disposition des éléments

On a commencé par disposer deux règles graduées sous forme de triangle, dont le sommet fait vingt-quatre degré (24°) car le **DMD** une fois situé au milieu ; Effectue l'affichage à douze degré (12°) de chaque côté. Le triangle étant formé, on le Sépare au milieu par une troisième règle graduée pour connaitre la ligne de positionnement du laser. La disposition ainsi effectuée, on place les éléments. Le laser, la première lentille (L1) et le **DMD** sont situés sur la ligne de la règle graduée du milieu. Les deux autres lentilles (L2, L3) ainsi que les cibles (C1, C2), seront eux placées sur les deux autres règles graduées formant le sommet du triangle.

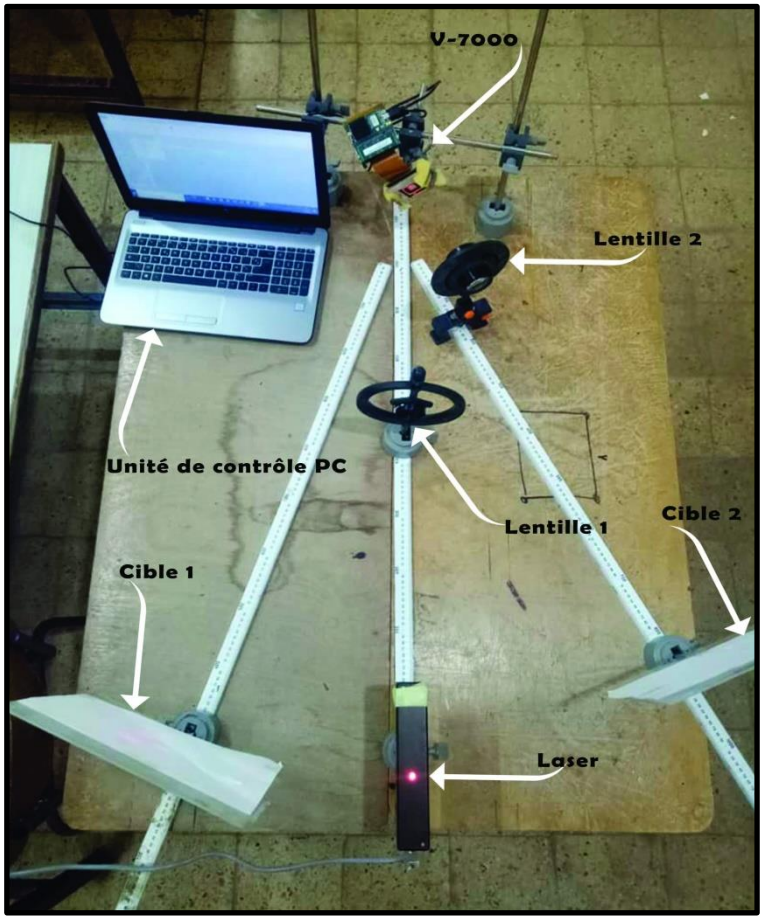


Figure II.16. Dispositif opto-électromécanique

II.3.2 Mesure et calcul

Le laser est situé à 50 cm de la L1 (2 dioptrie), afin d'obtenir des faisceaux parallèles pour éclairer le **DMD** qui est placé à 45 cm de L1. La L2 et L3 ont les mêmes valeurs en dioptrie, elles sont situées sur la même distance qui est de 5 cm par rapport au sommet du triangle. ET les cibles (C1, C2) sont situées à 56 cm par rapport à leur lentille respective.

Les différentes valeurs ici ont été obtenues par calcule des points focaux des différentes lentilles (L1, L2, L3). Prenons le cas du laser par exemple, pour pouvoir obtenir des faisceaux lumineux parallèle, il faut que le laser soit sur le point focale de la lentille (Foyer). Pour déterminer ce point focale (foyer objet) dans notre cas, on utilisera la formule qui lie la focale (f en mètre) a la vergence (V en dioptrie) des lentilles.

$$\boldsymbol{V = \frac{1}{f}}$$

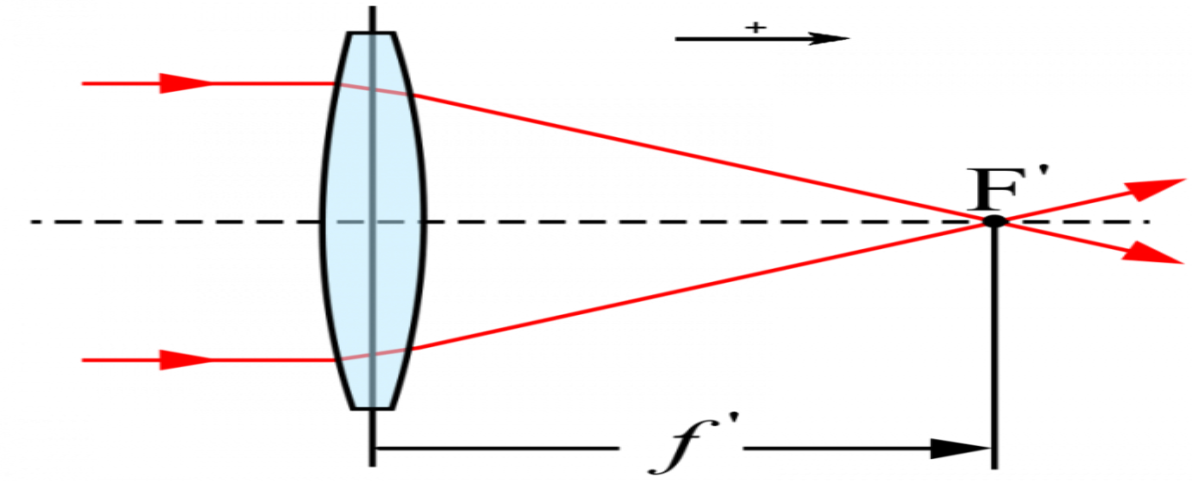


Figure II.17. Distance focale

Dans le cas de l'affichage au travers les lentilles (L2, L3), leur cible respective est située au niveau de leur distance focale (Foyer Image) car c'est là où on obtient une image nette. Pour le déterminer, on utilise la même formule que dans le cas précédent. Dans le cas où la vergence de la lentille est inconnue, on peut utiliser la formule suivante :

$$\boldsymbol{f = \frac{P \times Q}{P + Q}}$$

f : la distance focale

P : la distance qui sépare la lentille de l'image

Q : la distance qui sépare la lentille de l'objet

La distance focale ainsi déterminer, on peut calculer les dimensions de notre image numérique refléter au travers des lentilles (L2, L3). Afin d'obtenir ces dimensions de l'image numérique, on détermine la hauteur (H) et la largeur (L) de l'image sur la cible en fonction de la focale (f) et de la distance (d).

$$O_H = arctan(\frac{l}{2 \times f})$$

$$O_V = arctan(\frac{h}{2 \times f})$$

O_H = angle horizontal

O_V = angle vertical

II.4 Principe de fonctionnement du dispositif optique

Les faisceaux lumineux émis par la diode laser, traverse la lentille une (L1) qui les rend parallèle par phénomène de défocalisation afin d'éclairer toute les pixels du DMD. Lequel à son tour reflètera l'image numérique affichée sur DMD dans la direction douze degré (+12°). L'image numérique réfléchit passe au travers de la lentille deux (L2), qui recentre les faisceaux en les focalisant vers le foyer où sera situé notre cible une (C1) qu'on utilisera en cas pratique comme la rétine de l'œil. La figure II.18 illustre le schéma de fonctionnement du dispositif optique [15].

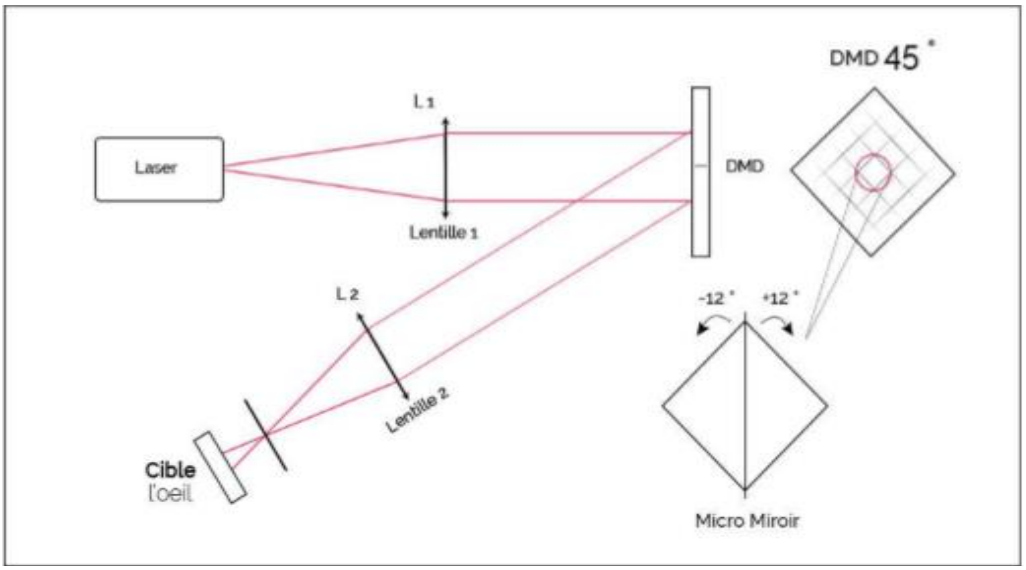


Figure II.18. Schéma de fonctionnement du dispositif optique

II.4.1 Phénomène de diffraction dû au DMD

Lors du premier essaie de projection, on a observé qu'il y a deux problèmes à résoudre. Le premier étant le phénomène de diffraction dû au DMD. Cela s'identifiant par un affichage de part et d'autre de plusieurs copies à intensité réduite de l'image numérique affichée par le DMD. Ainsi pour pallier à cela, la présence des lentilles (L2, L3) est nécessaire afin d'effectuer une deuxième convergence (sélection de l'image originale).

II.4.2 Affichage simultané sur les cibles

Ce dernier étant notre deuxième problème est d'ordre angulaire, car en effet le DMD doit être incliné à un angle de 45° afin d'effectuer un affichage équitable à l'horizontal sur les deux cibles.

II.5 Conclusion

Nous avons fait la conception de l'interface de contrôle permettant d'établir la communication du DMD et l'interface de la webcam pour l'acquisition des images sur les cibles. Nous avons également conçu un dispositif optique à travers les différentes étapes de réalisation. Afin de réaliser l'expérience pratique pour la poursuite de cible par le façonnage de faisceau laser comme nous le montrons dans le chapitre implémentations et résultats.

Chapitre III

Implémentations logicielles

III.1 Introduction

Dans le chapitre précèdent nous avons conçu les interfaces graphiques permettant le contrôle du DMD pour la projection de l'image sur la cible, et de la Webcam pour la visualisation et l'acquisition de l'image de la cible.

Ce chapitre comprend l'ensemble des programmes que nous avons implémentés pour le contrôle du DMD et de la Webcam avec le logiciel Visual Basic, la création de notre base de données pour le Deep Learning avec le logiciel Matlab. Notre algorithme développé aussi sous Matlab, l'algorithme de poursuite par la fonction Vision Template Matcher de Matlab et l'algorithme de poursuite avec Python Tensorflow.

III.2 Programme de contrôle du DMD implémenté sous Visual Studio 2017

L'activation du programme de contrôle du DMD, ouvre une interface graphique permettant d'accéder aux fonctionnalités du DMD (en cliquant sur les boutons correspondant selon les besoins de nos différentes expériences).

Il permet également d'interagir avec la webcam qui fournit les images pour la détection de mouvement de la cible pour la poursuite de la cible par le DMD. Les différents programmes sont décrits dans cette section.

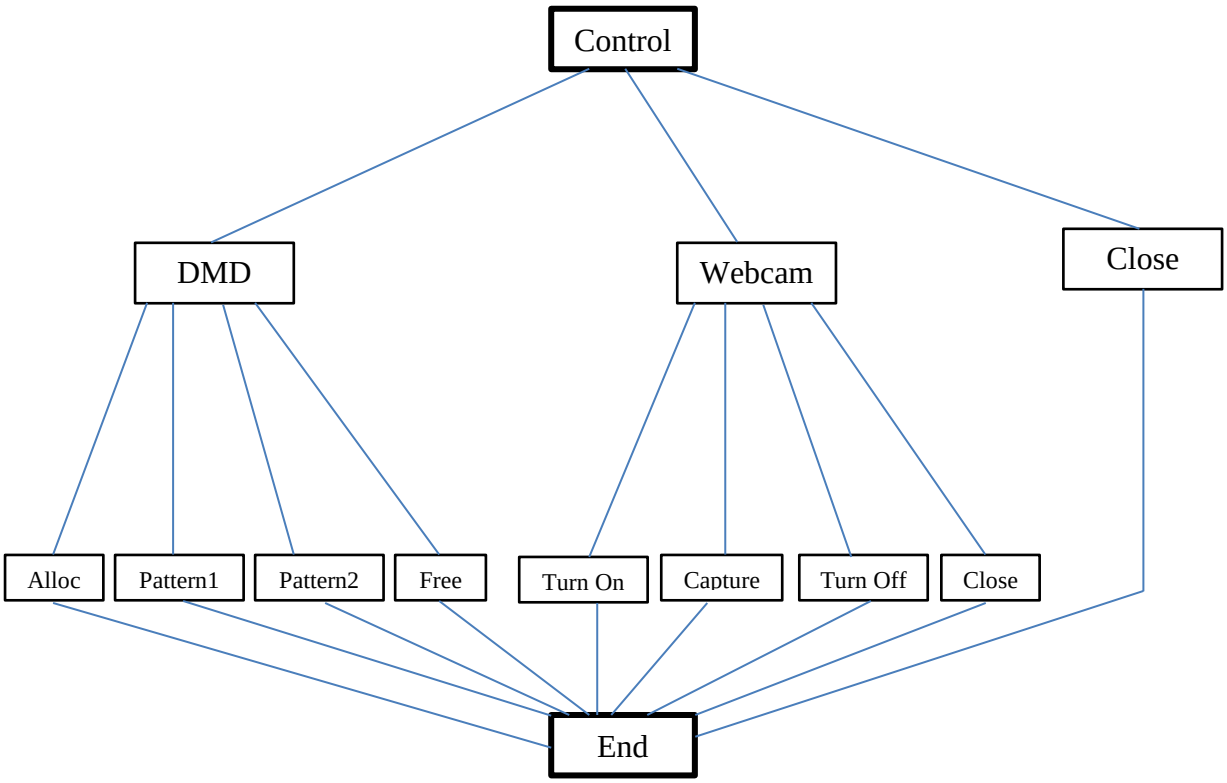


Figure III.1. Organigramme fonctionnel de l'interface graphique de Contrôle

III.2.1 La structure du sous-programme de l'interface de contrôle

Le sous-programme est représenté comme suit :

```
Public Class Form1

    Private Sub DMDbtn_Click(sender As Object, e As EventArgs) Handles
Bouton1.Click

        AlpTest.Show()    // Appel de l'interface de contrôle du DMD

    End Sub

    Private Sub Webcambtn_Click(sender As Object, e As EventArgs) Handles
Bouton2.Click

        Webcam1.Show()    // Appel de l'interface de la Webcam

    End Sub

    Private Sub Closebtn_Click(sender As Object, e As EventArgs) Handles
Bouton3.Click

        Close()           // Ferme la fenêtre de l'interface de contrôle

    End Sub

End Class
```

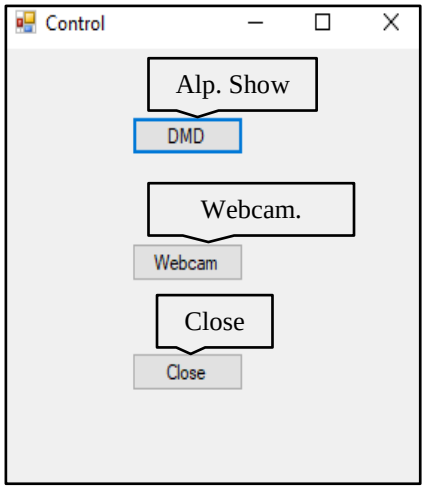


Figure III.2. Sous-programme de l'interface de contrôle

III.2.2 Le sous-programme de l'interface de la webcam

Le sous-programme de l'interface de la webcam pour la visualisation et l'acquisition de l'image de la cible est représenté comme suit :

```
Public Class Webcam1

    Private webcam As Object

    Private Sub TurnOn_btn_Click(sender As Object, e As EventArgs) Handles
Button1.Click

// Ouverture de la Webcam pour la visualisation

        AxVideoCap1.Device = 0

        AxVideoCap1.VideoFormat = 0

        AxVideoCap1.SyncMode = 1

        AxVideoCap1.VideoStandard = 0

        AxVideoCap1.AudioInputPin = 0

        AxVideoCap1.Start()

    End Sub
```

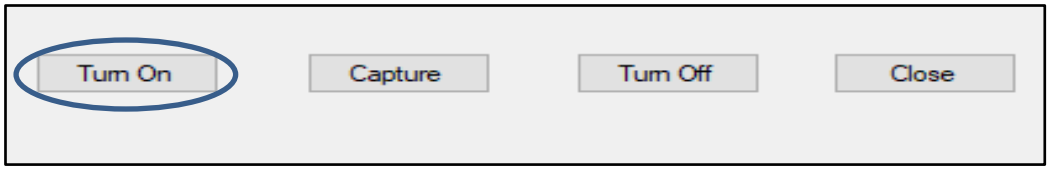


Figure III.3. Sous-programme du bouton Turn On

```
            // Acquisition ou capture de l'image sur la cible

    Private Sub Capture_btn_Click(sender As Object, e As EventArgs)
Handles Button2.Click

        Me.SaveFileDialog1.Filter = "JPEG File (*.jpg)|*.jpg"

        Me.SaveFileDialog1.ShowDialog()

        AxVideoCap1.SnapShotJPEG(SaveFileDialog1.FileName, 90)

    End Sub
```

Figure III.4. Sous-programme du bouton Capture pour l'acquisition de l'image de la cible

```
            // Fermeture de la Webcam

      Private Sub TurnOff_btn_Click(sender As Object, e As EventArgs)
Handles Button3.Click

        AxVideoCap1.StopIPCamera()

    End Sub

            // Ferme la fenêtre de l'interface de la webcam

    Private Sub Close_btn_Click(sender As Object, e As EventArgs) Handles
Button4.Click

        Close()

    End Sub

End Class
```

Figure III.5. Sous-programme des boutons Turn Off et Close (fermeture de la webcam et de l'interface)

III.2.3 Sous-programme de l'interface de contrôle du DMD

Le sous-programme de l'interface de contrôle du DMD pour l'allocation et l'envoi des images est représenté comme suit :

```
            // Allocation du DMD

      Private Sub Alloc_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Alloc.Click

        Dim nRet As Int32

        nRet = AlpDevAlloc(0, 0, m_AlpDeviceId)

        If nRet <> 0 Then

            MessageBox.Show("AlpDevAlloc Error" & _

                vbCrLf & AlpErrorString(nRet))

            Exit Sub

        End If

      MessageBox.Show("Success." & vbCrLf & _

          "Allocated ALP device and 2 sequences." & vbCrLf & _

          "Sent image data of 2 sequences.")
```

Figure III.6. Sous-programme du bouton Alloc

```
            // Bouton Pattern

      Private Sub Pattern1_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles Pattern1.Click

        Dim nRet As Int32

        nRet = AlpProjStartCont(m_AlpDeviceId, m_AlpSeqId1)

        If nRet <> 0 Then

            MessageBox.Show("AlpProjStartCont(1) Error" & _

                vbCrLf & AlpErrorString(nRet))

            Exit Sub

        End If

    End Sub
```

Figure III.7. Sous-programme du bouton Pattern 1

```
        // Bouton Free
    nRet = AlpDevFree(m_AlpDeviceId)
    If nRet <> 0 Then
        MessageBox.Show("AlpDevFree Error" & _
            vbCrLf & AlpErrorString(nRet))
        Exit Sub
    End If
    MessageBox.Show("Success." & vbCrLf & _
        "Halted and released ALP device.")
```

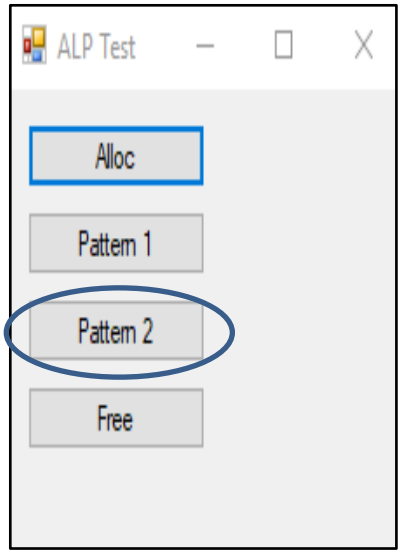


Figure III.8. Sous-programme du bouton Pattern 1

III.3 Programmes implémentés sous Matlab pour la création des bases de données pour le DL

Avec le logiciel Matlab nous avons créé notre base de données qui servira d'apprentissage au Deep learning pour la poursuite de la rétine.

III.3.1 Implémentation de la première base de données

Le but de cette partie est de créer une base de données d'images d'angiographie de la rétine. Nous générons une image blanche qui représente l'œil et une autre image binaire (noire et blanche) qui représente la rétine avec des anomalies représentées par les parties noires (fig.III.19).

Nous avons utilisé le logiciel Matlab pour générer notre base de données d'images binaire. Nous décrivons dans cette partie les étapes de création de la base d'apprentissage.

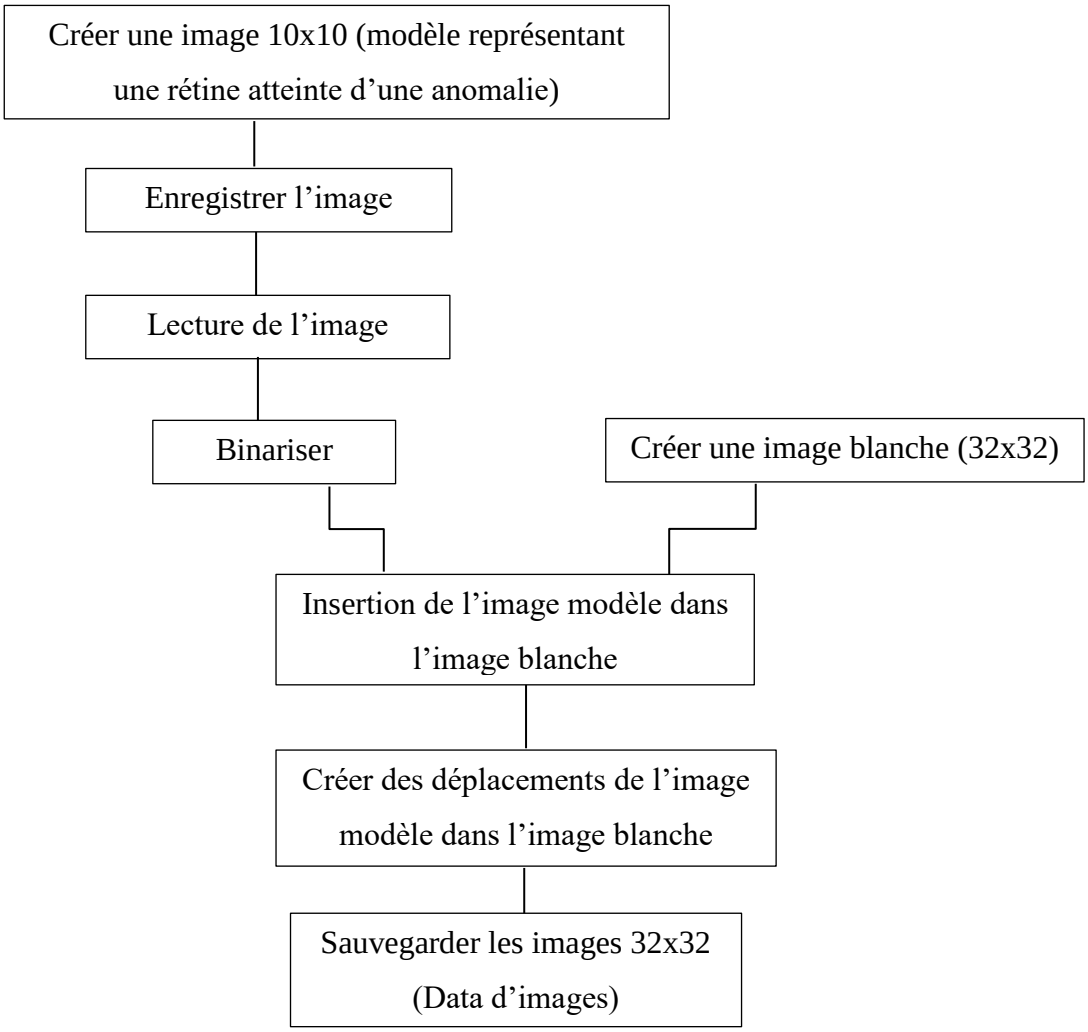


Figure III.9. Etapes de création de la base d'apprentissage

- Première étape : elle consiste à créer notre image modèle qui représente une rétine atteinte d'une anomalie.

```
Img=round (rand(10));

Img=imread ('0.jpg');
Img=imbinarize(Img);
figure; imshow(Img);
```

Pour cela nous avons utilisé la fonction Random « rand » qui permet de générer des tableaux (lignes et colonnes) de nombres aléatoires. Ces nombres sont des flottants dans l'intervalle de [0 à 1], nous obtenons ainsi une image à niveau de gris, ce qui n'est pas notre objectif, donc nous avons fait appel à la fonction « round » qui nous permet d'arrondir les nombres flottants pour avoir une image binaire noir et blanc.

```
img =

    0.0740    0.8711    0.4226    0.6537    0.6816    0.7027    0.3258    0.1635    0.6712    0.5056
    0.6841    0.3508    0.3596    0.9569    0.4633    0.1536    0.5464    0.9211    0.7152    0.6357
    0.4024    0.6855    0.5583    0.9357    0.2122    0.9535    0.3989    0.7947    0.6421    0.9509
    0.9828    0.2941    0.7425    0.4579    0.0985    0.5409    0.4151    0.5774    0.4190    0.4440
    0.4022    0.5306    0.4243    0.2405    0.8236    0.6797    0.1807    0.4400    0.3908    0.0600
    0.6207    0.8324    0.4294    0.7639    0.1750    0.0366    0.2554    0.2576    0.8161    0.8667
    0.1544    0.5975    0.1249    0.7593    0.1636    0.8092    0.0205    0.7519    0.3174    0.6312
    0.3813    0.3353    0.0244    0.7406    0.6660    0.7486    0.9237    0.2287    0.8145    0.3551
    0.1611    0.2992    0.2902    0.7437    0.8944    0.1202    0.6537    0.0642    0.7891    0.9970
    0.7581    0.4526    0.3175    0.1059    0.5166    0.5250    0.9326    0.7673    0.8523    0.2242
```

Figure III.10. Nombres aléatoires [10x10] obtenues avec la fonction Random

```
Command Window

  img =

       1     1     0     0     0     0     0     0     1     0
       1     0     1     0     0     1     0     1     1     0
       1     0     1     1     0     1     1     1     1     0
       1     1     1     0     0     0     1     0     1     1
       0     1     1     1     0     1     0     1     0     0
       1     1     1     0     0     0     0     1     0     1
       1     1     1     1     0     0     1     1     0     1
       1     1     1     1     0     0     0     0     0     1
       1     0     1     0     0     1     0     1     1     0
       0     1     1     1     1     1     0     1     0     0

fx >>
```

Figure III.11. Nombres arrondis avec la fonction Round

A chaque exécution du programme, les nombres changent. Pour conserver le même modèle, nous avons enregistré notre image, ensuite nous faisons une lecture de cette image grâce à la fonction « imread » depuis le répertoire de sauvegarde. La fonction « imbinarize » permet de binariser l'image, car une fois lue, l'image n'est pas binaire.

```
img =

  10×10 uint8 matrix

  250    10   249     0     4   245    10     0   246   255
   13   241   255   240    18   255     0   251   255     0
  245     5   255   255   232   250    12   252   255     0
  255     0   255   254     0    16   231   255   249     0
    0   246     5     3     0   236    13     0   255   255
    9     8   253     0   255   255     0   255     0   241
  255   233   255     0   255     2     0     1   255   237
  250   255   255     3   235     2    16     0   248    16
  255   235   252   255     0     8   255     0   240   255
    0   255    17   253   248   255   227   255    12   233
```

Figure III.12. Lecture de l'image de puis le répertoire de sauvegarde

```
img =

  10×10 logical array

   1   0   1   0   0   1   0   0   1   1
   0   1   1   1   0   1   0   1   1   0
   1   0   1   1   1   1   0   1   1   0
   1   0   1   1   0   0   1   1   1   0
   0   1   0   0   0   1   0   0   1   1
   0   0   1   0   1   1   0   1   0   1
   1   1   1   0   1   0   0   0   1   1
   1   1   1   0   1   0   0   0   1   0
   1   1   1   1   0   0   1   0   1   1
   0   1   0   1   1   1   1   1   0   1
```

Figure III.13. Binarisation de l'image lu

La fonction « imshow », nous permet d'afficher l'image de notre modèle.

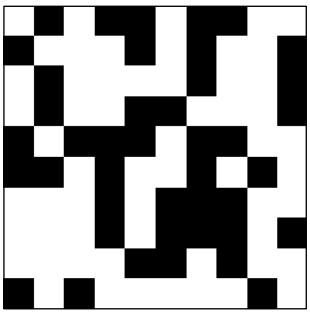


Figure III.14. Image modèle (10x10) de rétine atteinte d'une anomalie

- ❖ Deuxième étape : elle consiste à créer une image blanche (32x32) dans laquelle nous allons insérer notre image modèle.

La fonction « ones », permet de créer une matrice (n, p) composée du nombre 1, ce qui nous donne une image blanche (Imgf) de taille 32x32. Ensuite nous avons inséré l'image (Img) du modèle dans l'image blanche (Imgf) en faisant **Imgf (12 :21, 12 :21) = Img**, la figure III.15 montre le principe que nous avons utilisé.

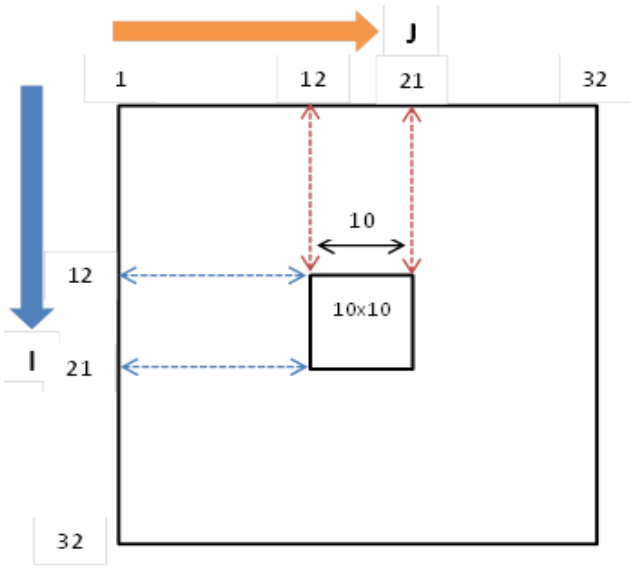


Figure III.15. Principe utilisé pour insérer l'image (Img) 10x10 dans l'image (Imgf) 32x32

1. Image blanche 32x32
2. Image modèle 10x10

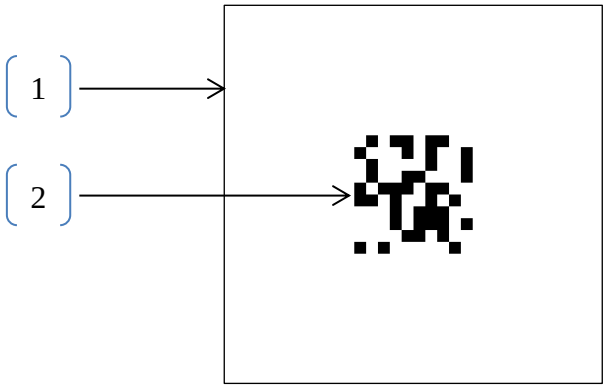


Figure III.16. Image 10x10 insérée dans l'image 32x32

❖ Troisième étape : elle consiste à générer des déplacements de l'image modèle dans l'image blanche et enregistrer chaque cas de déplacement dans un dossier qui représente notre base de données.

```
for i=1:100
      Imgf = ones(32) ;
      Imgf (12:21, 12:21)=Img;
      Imgf (12:21, 12:21)=1;

      x(i)=randi ([-11,11]);
      y(i)=randi ([-11,11]);

      Imgf (12+x:21+x, 12+y:21+y)=Img;

      figure
      imshow (Imgf);

      imwrite (Imgf, sprintf ('%d.bmp', i))
end
```

Dans le programme ci-dessus nous avons créé une boucle **for**, qui permet de faire à chaque itération un déplacement et enregistrer l'image. Nous avons vu à quoi servent les deux premières instructions du programme dans l'étape précédente.

La fonction « **randi** » renvoie un nombre entier aléatoire, nous avons donné comme argument à cette fonction un intervalle [-11,11], cet intervalle est le minimum et maximum à laquelle l'image (Img) 10x10 peut être déplacé dans l'image (Imgf) 32x32.

Les variables x et y sont des vecteurs qui contiennent les valeurs des déplacements des images à chaque itération. Cette instruction du programme « **Imgf (12+x:21+x, 12+y:21+y)=Img** » permet de gérer les déplacements, la figure III.17 montre des exemples de déplacements de l'image modèle.

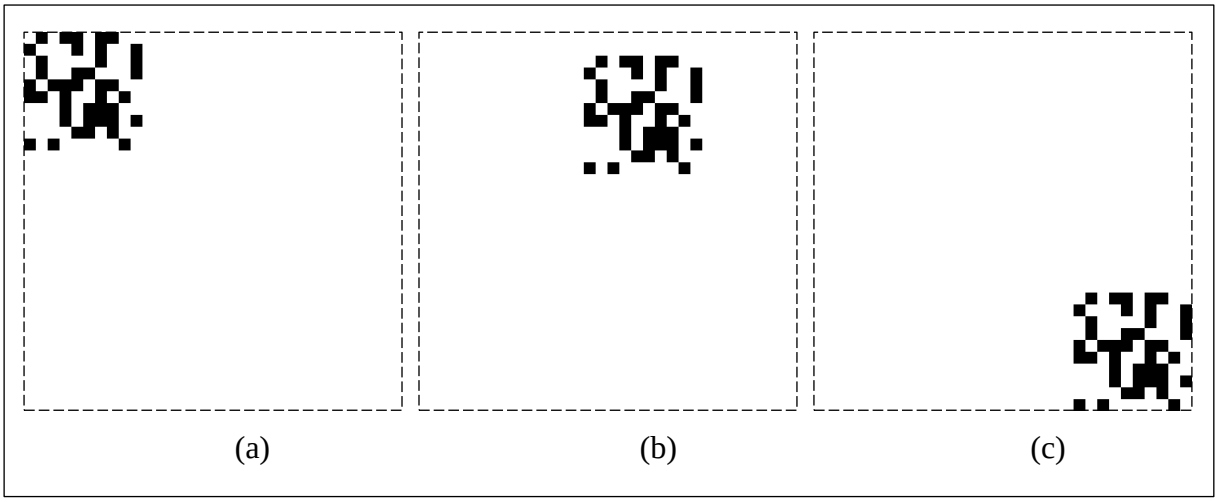


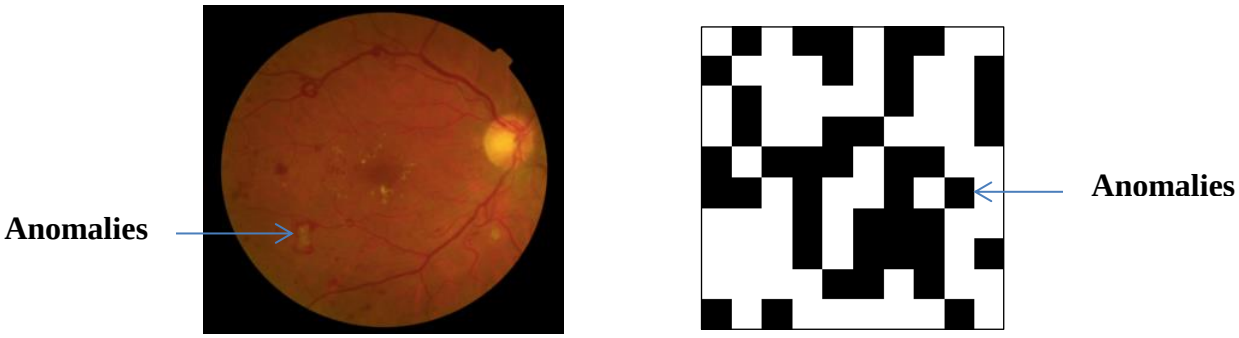
Figure III.17. Exemples de déplacement de l'image (Img) 10x10 dans l'image (Imgf) 32x32

Les vecteurs x et y contiennent respectivement [-11 -9 11], [-11 3 11], en remplaçant ses valeurs dans l'instruction **Imgf (12+x:21+x, 12+y:21+y)=Img**, nous avons :

- La figure III.17-a à pour coordonner de déplacement x= -11 et y= -11,
  Imgf (12-11:21-11, 12-11:21-11)=Img → **Imgf (1:10, 1:10)=Img**.
- La figure III.17-b de coordonnée x= -9 et y = 3,
  Imgf (12-9:21-9, 12+3:21+3)=Img → **Imgf (3:12, 15:24)=Img**.
- La figure III.17-c de coordonnée x = 11 et y= 11,
  Imgf (12+11:21+11, 12+11:21+11)=Img → **Imgf (23:32, 23:32)=Img**.

L'instruction **imwrite (Imgf, sprintf ('%d.bmp', i))** permet d'enregistrer une image à chaque itération dans un dossier cible.

La figure III.18-a représente une photographie du fond d'œil (rétine) atteinte d'une rétinopathie diabétique, nous avons donc généré notre programme pour simuler une rétine atteinte (figure III.18-b), les tâches noires de cette image sont considérées comme les anomalies.



(a) (b)

Figure III.18. Photographie du fond d'œil atteinte d'une rétinopathie diabétique et image modèle simulant une rétine atteinte

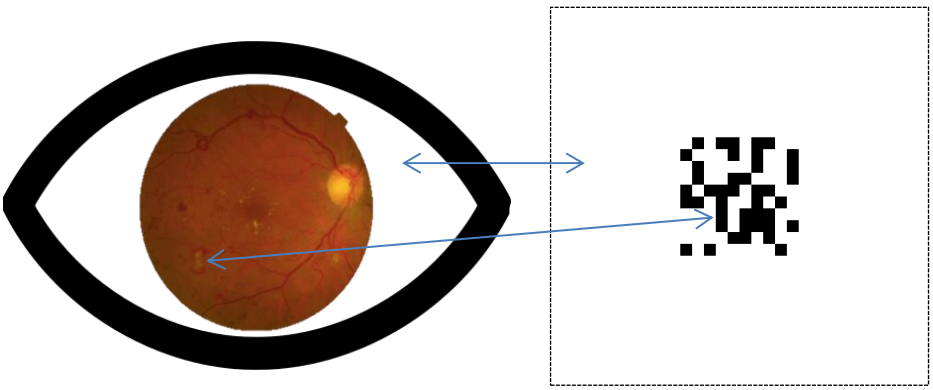


Figure III.19. Le principe suivi pour créer des simulations d'images 32x32 de la base de données

III.3.2 Implémentation de la deuxième base de données

Le principe utilisé pour créer la base de donnée de la partie (III.3.1) est la même pour la création de cette base de données, les facteurs qui changent sont le fond de l'image qui est noir et l'image insérée est une image du fond de l'œil (rétine) atteinte d'une rétinopathie diabétique (RD).

Figure III.20. Rétine atteinte de la RD

```
Img = imread ('modele.jpg');


for i=1:100

      Imgf= zeros (512, 512, 3,'uint8');

      Imgf (158:355, 108:404,1)=Img (:,:,1);
      Imgf (158:355, 108:404,2)=Img (:,:,2);
      Imgf (158:355, 108:404,3)=Img (:,:,3);

      Imgf (158:355, 108:404,1)=0;
      Imgf (158:355, 108:404,2)=0;
      Imgf (158:355, 108:404,3)=0;

      x = randi ([-157,157]);
      y = randi ([-107,107]);

      Imgf (158+x:355+x, 108+y:404+y, 1)=Img (:,:,1);
      Imgf (158+x:355+x, 108+y:404+y, 2)=Img (:,:,2);
      Imgf (158+x:355+x, 108+y:404+y, 3)=Img (:,:,3);

      figure
      imshow (Imgf);
      imwrite (Imgf, sprintf ('%d.jpg',i))

end
```

Le programme ci-dessus permet d'insérer l'image de la rétine tout en lui générant des déplacements aléatoires.

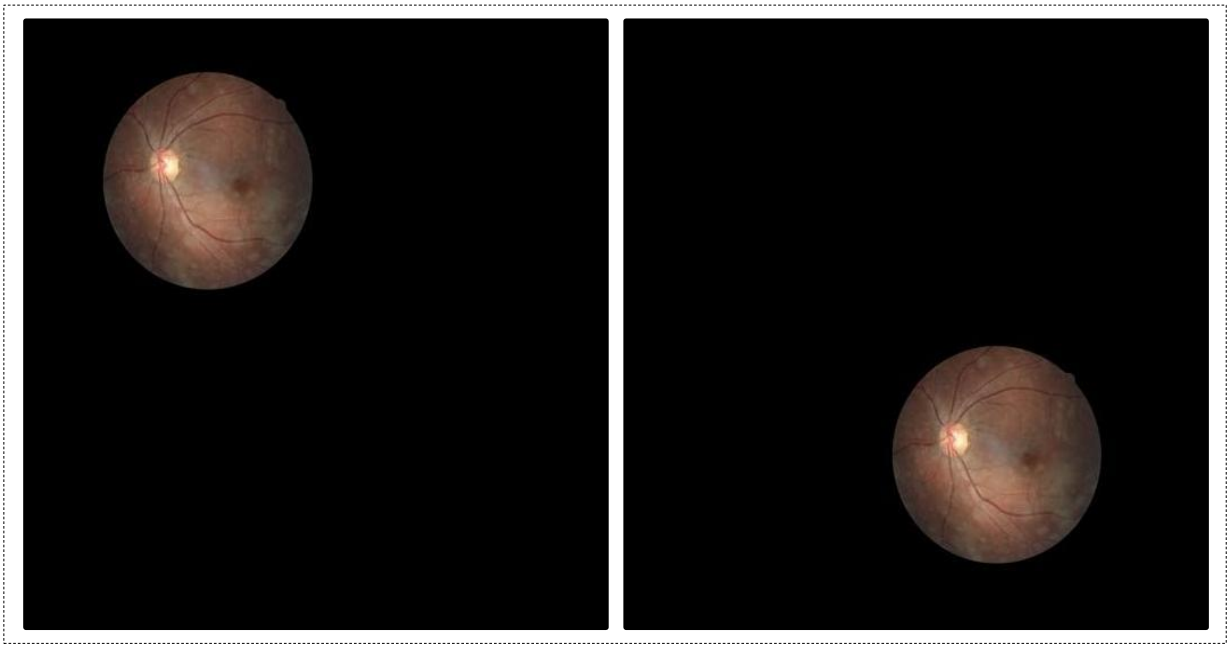


Figure III.21. Image de la rétine insérée dans un fond noir et déplacée

III.4 Algorithme de DL pour la pour poursuite

Nous utilisons l'API de détection d'objet de Tensorflow préformé, pour former notre algorithme de détection d'objet de réseau de neurones convolutifs.

III.4.1 Environnement : Python Tensorflow, Anaconda

Tensorflow est une bibliothèque Python pour les calculs numériques hautes performances qui permet aux utilisateurs de créer des applications sophistiquées d'apprentissage approfondi et d'apprentissage automatique. L'API de détection d'objet Tensorflow nécessite l'utilisation de la structure de répertoire et également de plusieurs packages Python, des ajouts spécifiques aux variables d'environnements PATH et PYTHONPATH.

Un certain nombre de méthodes peuvent être utilisées pour installer Tensorflow, telles que l'utilisation de pip et packages conda. Dans notre projet on a installé Tensorflow avec conda, qui offre de nombreux avantages, notamment un système de gestion complet des packages. Pour cela on a installé Anaconda, qui une distribution libre et open source de langage de programmation python [16].

Figure III.22. Logo d'Anaconda et de la bibliothèque Tensorflow

On a téléchargement et installé Anaconda Python 3.7 version pour Windows sur le site « https://www.anaconda.com/download/ ». Anaconda offre une interface de commande (Command Prompt) qui nous permet d'exécuté des instructions.

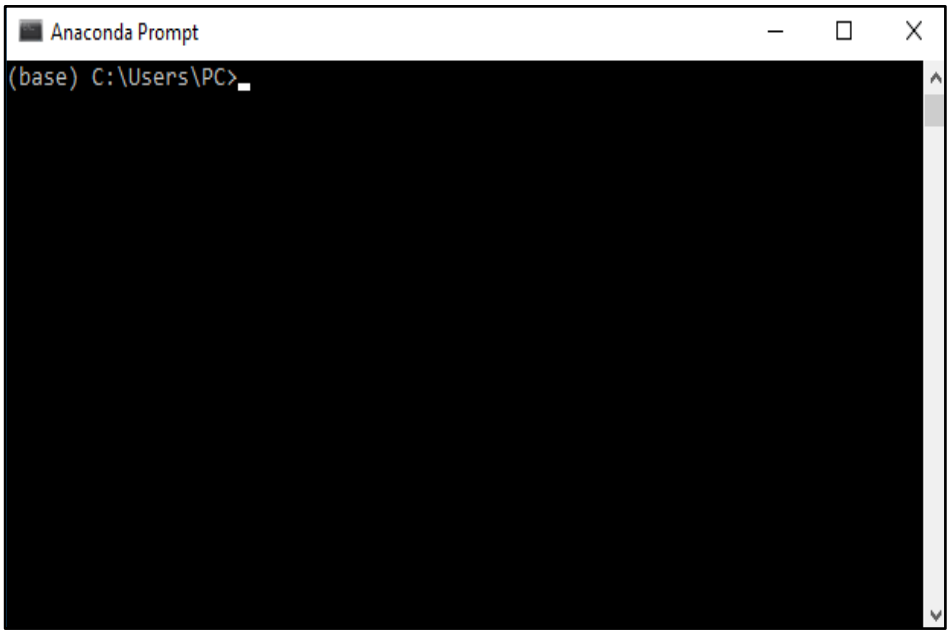


Figure III.23. Commande Prompt d'Anaconda

Il existe deux variantes génériques de Tensorflow, qui utilisent un matériel différent sur les ordinateurs pour exécuter leurs algorithmes de calcul lourd. A savoir : tensorflow_cpu et tensorflow_gpu. La différence entre ses deux variantes, est la performance. Le tensorflow_cpu est lent en termes de performance, le tensorflow_gpu nécessite la possession de carte graphique et l'installation de CUDA pour accélérer les calculs effectués par Tensorflow. Pour notre cas nous utilisons tensorflow_cpu parce que notre ordinateur ne possède pas de carte graphique.

Pour installer Tensorflow on a d'abord créé un nouvel environnement virtuel et ensuite on a procédé à l'installation en exécutant les commandes ci-dessous dans Command Prompt de Anaconda.

- Création de l'environnement virtuel :
  > conda create -n tensorflow_cpu pip python=3.6
  > conda activate tensorflow_cpu
- Installation de tensorflow_cpu :
  > pip install --ignore-installed –upgrade tensorflow==1.9

III.4.2 Les outils nécessaires pour l'exécution du modèle

Afin de mieux cerner cette section nous détaillons étapes par étapes [17].

- **Etape 1 : installations des packages**

Avant d'installer le modèle de détection d'objet pré-entrainé, nous avons installés certains packages nécessaires. Ces packages se composent de : pillow, lxml, jupyter, matplotlib, opencv. On les a installés en exécutant la commande suivante :

> **conda install pillow, lxml, jupyter, matplotlib, opencv**

- **Etape 2 : téléchargement du modèle Tensorflow**

On a téléchargé le modèle sur le site de GitHub, dont le lien est « https://github.com/tensorflow/models ». On a renommé le fichier téléchargé **models.**

Nous avons créé un dossier nommé Tensorflow pour une bonne organisation, ce dossier doit contenir le fichier téléchargé et d'autres fichiers qu'on va ajouter après.

- **Etape 3 : téléchargement du modèle Faster-RCNN-Inception-V2-COCO**

Tensorflow fournit plusieurs modèles de détection d'objets, c'est-à-dire des classificateurs préformés avec des architectures de réseau neuronal spécifiques. On a téléchargé Faster-RCNN-Inception-V2-COCO à partir du modèle zoo de Tensorflow, le lien : « https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md ». Le fichier obtenu est décompressé directement dans le répertoire C:\Tensorflow\models\research\object_detection.

- **Etape 4 : configuration de la variable d'environnement PYTHONPATH**

On a créé une variable PYTHONPATH pointant vers les répertoires \models ; \models\research ; \models\research\slim. La commande suivante permet de faire cela :

> **set PYTHONPATH=C:\Tensorflow\models; C:\Tensorflow\models\research\; C:\Tensorflow\models\research\slim;**

❖ **Etape 5 : compilation des Protobufs**

La compilation des fichiers Protobufs utilisés par Tensorflow permet la configuration des paramètres de modèles et formation. Ces fichiers (.proto) se trouvent dans le répertoire \models\research\object_detection\protos. On a remplacé le répertoire de la commande prompt d'Anaconda et exécuté ses commandes :

> **cd C:\Tensorflow\models\research**

> protoc object_detection\protos*.proto --python__out=.

> **python setup.py build**

> **python setup.py install**

❖ **Etape 6 : étiquetage des images de la base de données**

Nous devons donnés une centaine d'images d'un objet à Tensorflow pour former un bon classificateur de détection. Nous voulons détectés un objet dans les images de notre base de données (les points noirs qui représentent des anomalies rétinienne).

Avant l'étiquetage, nous avons déplacés les 20% des images vers le répertoire \object_detection\images\test et 80% des images vers le répertoire \object_detection\images\train

Pour étiqueter l'objet dans les images, nous avons utilisés **LabelImg** qui est un outil d'étiquetage d'image.

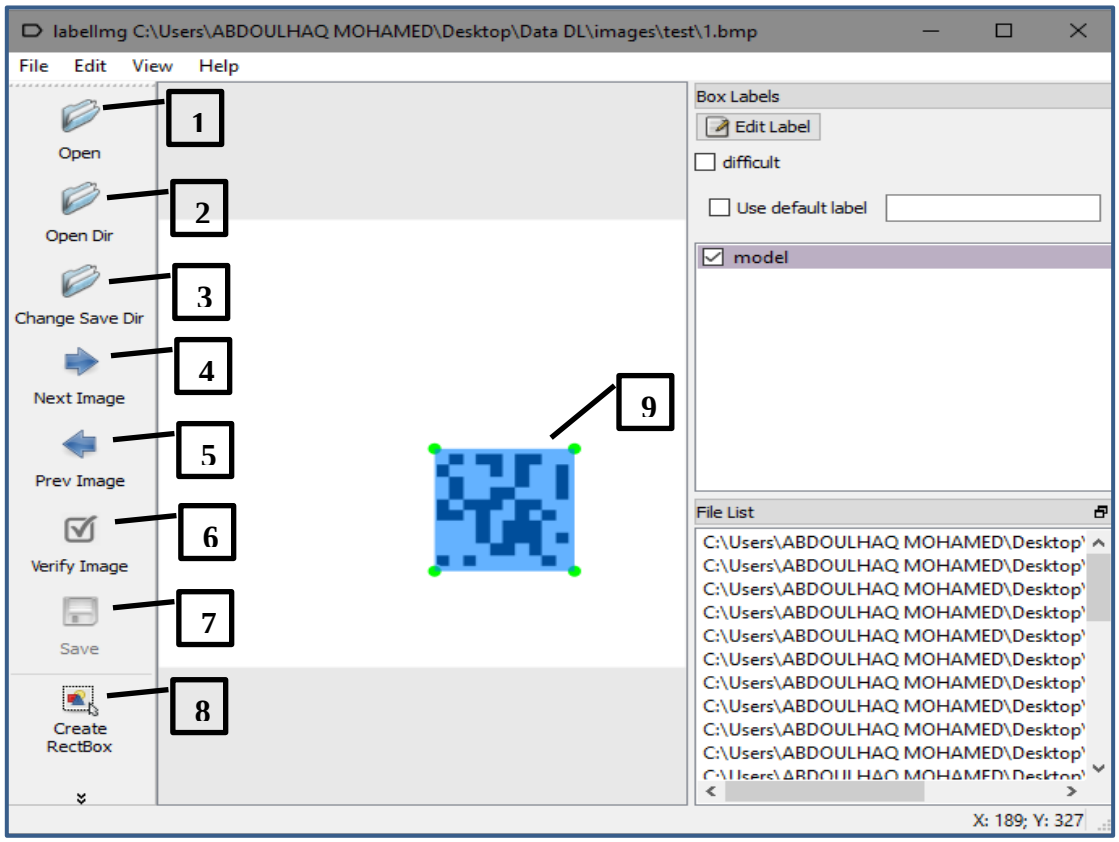


Figure III.24.interface de LabelImg

1. Charger une image
2. Charger tous les images d'un répertoire
3. Changer de dossier de sauvegarde des fichiers .xml
4. Afficher l'image suivant
5. Afficher l'image précédente
6. Vérifie si l'image est étiquetée
7. Enregistrer l'image étiquetée
8. Crée un rectangle pour étiquetée l'image
9. Image étiqueté et enregistrer

Après avoir étiqueté les images, il y aura un fichier .xml pour chaque image dans les répertoires test et train. Ensuite on a généré des TFRecords qui servent de données d'entrée au modèle de formation Tensorflow. On a aussi créé des fichiers .csv contenant toutes les données du train et des images de test à partir des fichiers .xml, en exécutant la commande :

> **python xml_to_csv.py**

Cela crée un fichier train_labels.csv et test_labels.csv dans le dossier \object_detection\images.

III.4.3 Programme d'appels des bibliothèques et des fonctions

III.4.3.1 Importations des bibliothèques

Imports

```
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile

from distutils.version import StrictVersion
from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image

sys.path.append("..")
from object_detection.utils import ops as utils_ops

if StrictVersion(tf.__version__) < StrictVersion('1.12.0'):
  raise ImportError('Please upgrade your TensorFlow installation to v1.12.*.')
```

Figure III.25. Importations des bibliothèques

Object detection imports

les importations du module de détection d'objet.

```
from utils import label_map_util

from utils import visualization_utils as vis_util
```

Figure III.26. Importation du module de détection

III.4.3.2 Préparation du modèle

Model preparation

Variables

```
# What model to download.
MODEL_NAME = 'ssd_mobilenet_v1_coco_2017_11_17'
MODEL_FILE = MODEL_NAME + '.tar.gz'
DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object_detection/'

# Path to frozen detection graph. This is the actual model that is used for
PATH_TO_FROZEN_GRAPH = MODEL_NAME + '/frozen_inference_graph.pb'

# List of the strings that is used to add correct label for each box.
PATH_TO_LABELS = os.path.join('data', 'mscoco_label_map.pbtxt')
```

Download Model

```
opener = urllib.request.URLopener()
opener.retrieve(DOWNLOAD_BASE + MODEL_FILE, MODEL_FILE)
tar_file = tarfile.open(MODEL_FILE)
for file in tar_file.getmembers():
  file_name = os.path.basename(file.name)
  if 'frozen_inference_graph.pb' in file_name:
    tar_file.extract(file, os.getcwd())
```

Load a (frozen) Tensorflow model into memory.

```
detection_graph = tf.Graph()
with detection_graph.as_default():
  od_graph_def = tf.GraphDef()
  with tf.gfile.GFile(PATH_TO_FROZEN_GRAPH, 'rb') as fid:
    serialized_graph = fid.read()
    od_graph_def.ParseFromString(serialized_graph)
    tf.import_graph_def(od_graph_def, name='')
```

Loading label map

Label maps map indices to category names, so that when our convolution network predicts `5`, we know that this corresponds to `a`
internal utility functions, but anything that returns a dictionary mapping integers to appropriate string labels would be fine

```
category_index = label_map_util.create_category_index_from_labelmap(PATH_TO_LABELS, use_display_name=True)
```

Helper code

```
def load_image_into_numpy_array(image):
  (im_width, im_height) = image.size
  return np.array(image.getdata()).reshape(
      (im_height, im_width, 3)).astype(np.uint8)
```

Figure III.27. Préparation du modèle

Detection

```
# For the sake of simplicity we will use only 2 images:
# image1.jpg
# image2.jpg
# If you want to test the code with your images, just add path to the images to the TEST_IMAGE_PATHS.
PATH_TO_TEST_IMAGES_DIR = 'test_images'
TEST_IMAGE_PATHS = [ os.path.join(PATH_TO_TEST_IMAGES_DIR, 'image{}.jpg'.format(i)) for i in range(1, 3) ]

# Size, in inches, of the output images.
IMAGE_SIZE = (12, 8)
```

```
def run_inference_for_single_image(image, graph):
  with graph.as_default():
    with tf.Session() as sess:
      # Get handles to input and output tensors
      ops = tf.get_default_graph().get_operations()
      all_tensor_names = {output.name for op in ops for output in op.outputs}
      tensor_dict = {}
      for key in [
          'num_detections', 'detection_boxes', 'detection_scores',
          'detection_classes', 'detection_masks'
      ]:
        tensor_name = key + ':0'
        if tensor_name in all_tensor_names:
          tensor_dict[key] = tf.get_default_graph().get_tensor_by_name(
              tensor_name)
```

Figure III.28. Fonction de la détection

III.5 Algorithme de poursuite par comparaison de pixel et par la fonction Vision Template Matcher de Matlab

III.5.1 Algorithme de poursuite par comparaison de pixel

Cette méthode consiste à faire une comparaison logique de deux images pixels par pixels afin de déterminer le déplacement (coordonnées x, y) de la cible. Le principe utilisé est illustré par l'organigramme (fig.III.29).

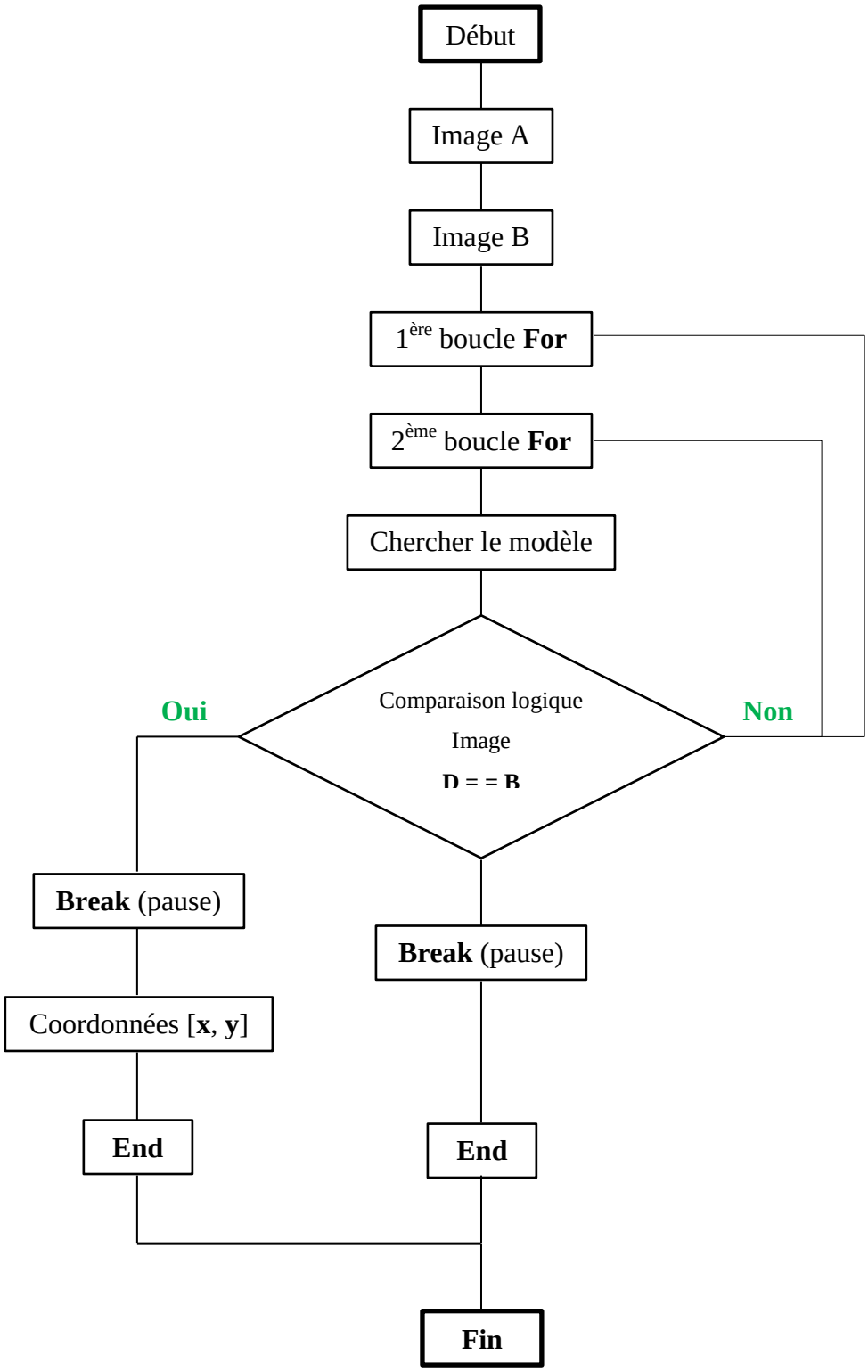


Figure III.29. Organigramme du programme

```
A = imread ('image A');
B = imread ('image B');

for i=-11:11
    for j=-11:11
        D = imtranslate (A, [i, j],'FillValues',255);
        Imshow (D)
        if D==B
            x = j;
            y = i;
            break
        end
    end
        if D==B
            break;
        end
end
```

- **Etape 1 :** elle consiste à faire la lecture des images A et B avec la fonction « imread ».
- **Etape 2 :** elle consiste à créer deux boucles **for** associé aux vecteurs (i, j) qui permettent de balayer l'image du début à la fin.
- **Etape 3 :** nous avons une variable D au quelle nous affectons la fonction « **imtranslate** », cette fonction permet de faire une translation (décalage) d'image, elle prend comme argument l'image A, les valeurs des vecteurs [i, j] à chaque itérations des boucles, FillValues qui est un tableau contenant une ou plusieurs valeurs de remplissage (parce que l'image est un tableau de plusieurs valeurs), et « 255 » parce que les images une fois importées (fonction imread), sont des images à niveau de gris.
  L'instruction **D = imtranslate (A, [i, j],'FillValues',255)** nous permet d'avoir un balayage pixel par pixel par pixel [18].
- **Etape 4 :** nous procédons ici à la comparaison logique des deux images D et B contenant l'image modèle disposée aléatoirement dans D et B. L'image D va parcourir l'image B pixel par pixel, si (**if D= =B)** elle trouve la position du modèle dans l'image B, elle nous renvoie les coordonnées (x, y). La 2ème condition (**if D==B**) permet d'arrêter (**break**) le balayage une fois que le modèle est retrouvé.

III.5.2 Algorithme de poursuite par la fonction Vision Template Matcher

Dans cette section nous déterminons la position (coordonnée x, y) de déplacement de l'image modèle de la cible. Nous utilisons l'interface de contrôle de la webcam pour la visualisation et l'acquisition des images de la cible.

Pour localiser et déterminer la position du modèle dans l'image, nous utilisons **Vision.TemplateMatcher** qui est un objet Matlab. Les objets système sont conçus spécifiquement pour l'implémentation et la simulation de systèmes dynamiques avec des entrées évoluant dans le temps [19]. Ensuite nous appelons l'objet avec des arguments, comme s'il s'agissait d'une fonction. Le principe est décrit à travers l'organigramme (fig.III.30).

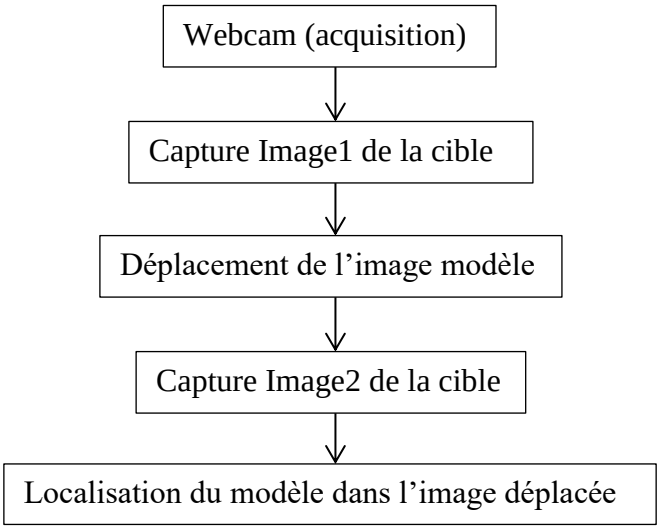


Figure III.30. Organigramme fonctionnel des étapes

```
htm = vision.TemplateMatcher;

image1 = imread ('image1.jpg');
image2 = imread ('image2.jpg');

image1 = rgb2gray (image1);
image2 = rgb2gray (image2);

image1 = imresize (image1, 0.5);
image2 = imresize (image2, 0.5);

image1 = image1 (x, y);

Loc = step (htm, image2, image1);
```

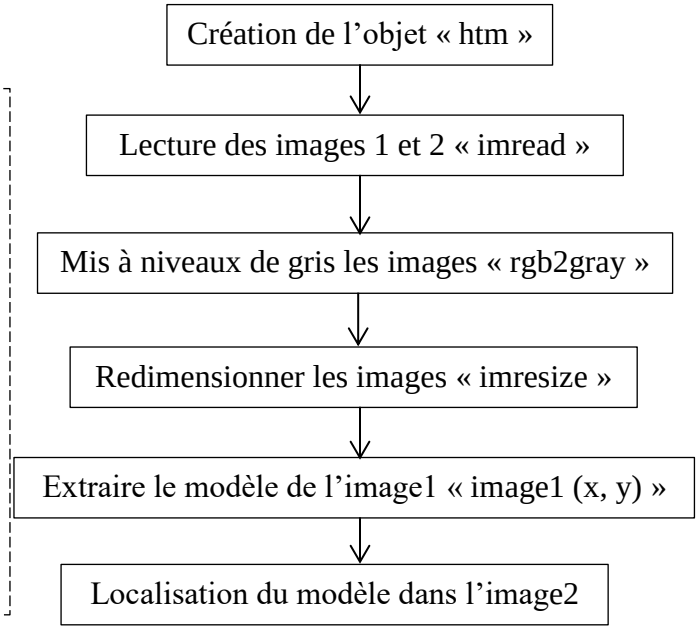


Figure III.31. Programme pour la poursuite et son organigramme fonctionnel

Après avoir exécuté les étapes des organigrammes (fig. III.30, III.31), à la fin de l'exécution l'instruction **Loc = step (htm, image2, image1),** nous renvoie les coordonnées (x, y). Ses coordonnées nous les utilisons dans notre programme de Visual Basic afin de repositionné en cas de déplacement de la cible les faisceaux lumineux aux coordonnées obtenues.

III.6 Conclusion

Dans ce chapitre nous avons procédé à l'implémentation des sous-programmes liés aux interfaces graphiques permettant le contrôle du DMD (allocation, chargement des images) et le contrôle de la webcam pour la visualisation et capture d'image grâce au logiciel Visual Basic. Nous avons utilisé un modèle préformé (détection d'objet Tensorflow) de Deep Learning pour la poursuite de la cible. Nous avons aussi créé notre base de données pour l'apprentissage du Deep learning et des algorithmes pour la poursuite avec le logiciel Matlab.
Dans le chapitre suivant, nous réaliserons des expériences avec notre dispositif opto-électromécanique pour la poursuite de la cible commandé par les programmes que nous avons implémentés.

Chapitre IV

Implémentations et résultats

IV.1 Introduction

Dans ce chapitre nous effectuons des expériences pour la mise en évidence des paramètres du dispositif opto-électromécanique. Nous montrons les différents états du faisceau laser ainsi que des motifs d'images reflétés par le DMD sur la cible. Nous décrivons les solutions pour la poursuite de la cible avec le deep learning et les deux algorithmes que nous avons implémentés sur Matlab. Nous présentons dans ce chapitre les expériences et les résultats obtenus et leurs discussions.

IV.2 Implémentation et commande du dispositif optique

Nous construisons, dans cette partie, le dispositif optique conçu dans le chapitre II autour du DMD. Nous exécutons les commandes de notre interface graphique sur PC pour mettre en évidence le bon fonctionnement de notre dispositif.

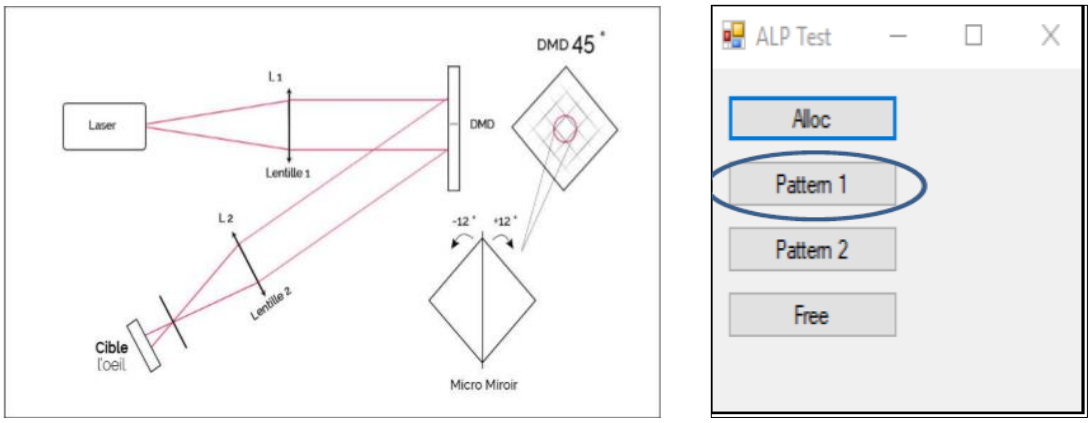


Figure IV.1. Schéma de fonctionnement du dispositif et de l'interface DMD

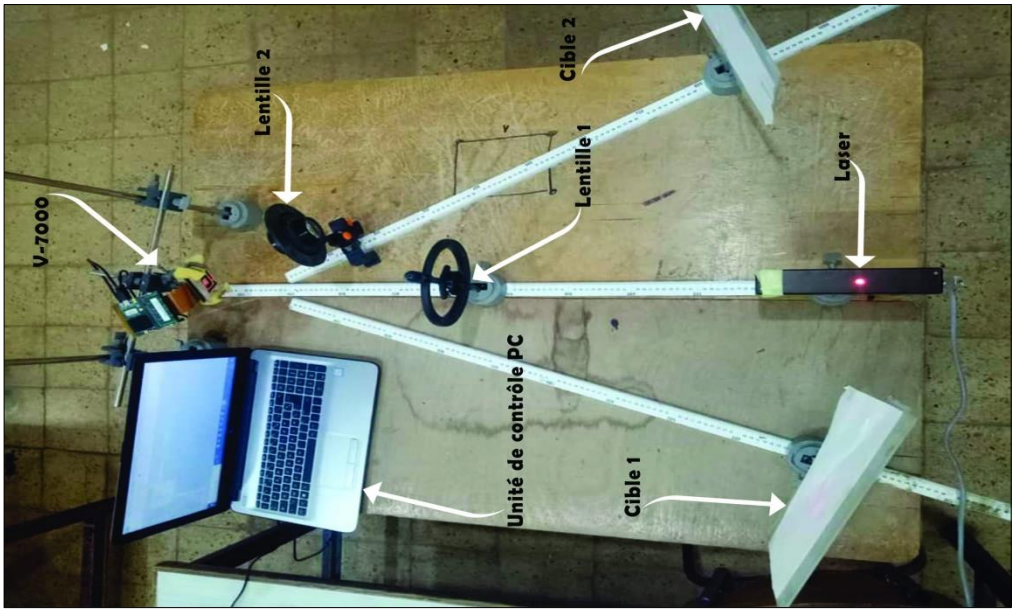


Figure IV.2. Dispositif opto-électromécanique

IV.2.1 Affichage visuel sur le DMD

Le DMD effectue son affichage en basculant ses micromiroirs entre l'état on et off, ce qui est présenté comme suit (fig. IV.3) :

Figure IV.3. Affichage d'image de damier et d'un motif

IV.2.2 Les différents états du faisceau laser

Certains résultats obtenus à travers les différentes états prisent par les faisceaux lumineux du laser sont difficilement imaginable, d'où la nécessite d'une représentation visuelle.

IV.2.2.1 L'état du laser avant et après traversée de la lentille (L1)

Les faisceaux lumineux du laser avant de traverser la lentille L1, sont cohérents et monochromatiques. Après avoir traversé L1 qui est une lentille convergente, les faisceaux lumineux du laser se retrouvent de l'autre côté étant diffracté.

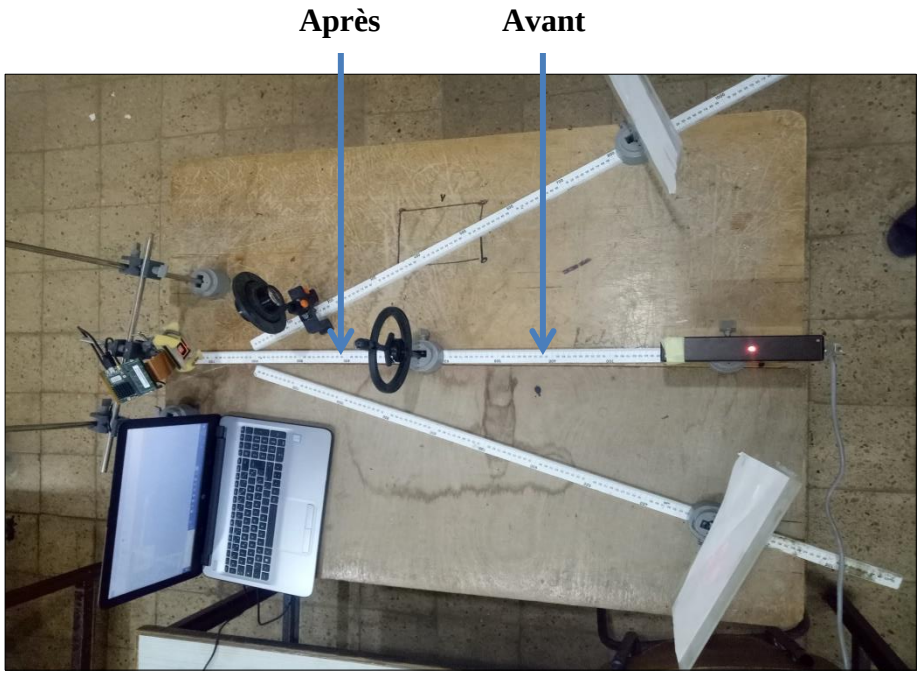


Figure IV.4. Etat du laser avant et après traversée de la lentille L1

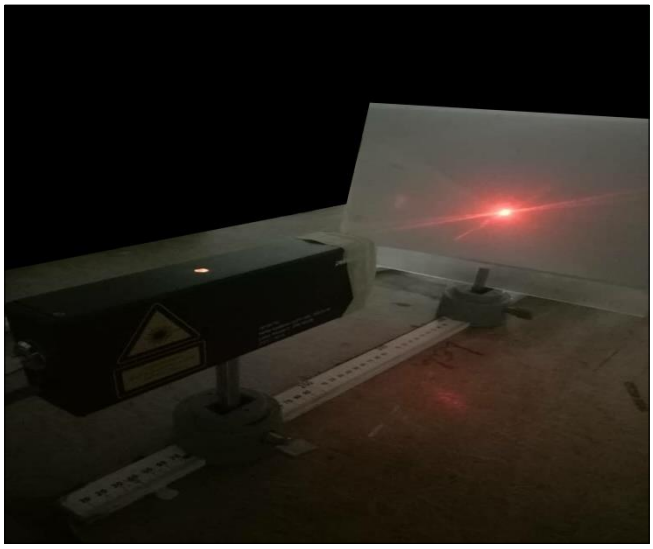


Figure IV.5. Faisceau laser avant L1

Figure IV.6. Faisceau laser après L1

Il est possible de les différentiés a vu d'œil en comparant la taille du point sur la cible (fig.IV.5, IV.6). La plus grande étant le faisceau diffracté et l'autre le monochromatique.

IV.2.2.2 L'état du laser avant et après traversée des lentilles (L2, L3)

Les faisceaux lumineux une fois réfléchies par le DMD, avant de passer au travers de la L2 et L3 est parallèle donc présente des diffractions de l'image.

Après avoir traversé la L2 et L3 qui sont des lentilles convergentes, les faisceaux lumineux du laser reflétés se retrouvent de l'autre côté focalisés vers le foyer d'où on obtient une image nette.

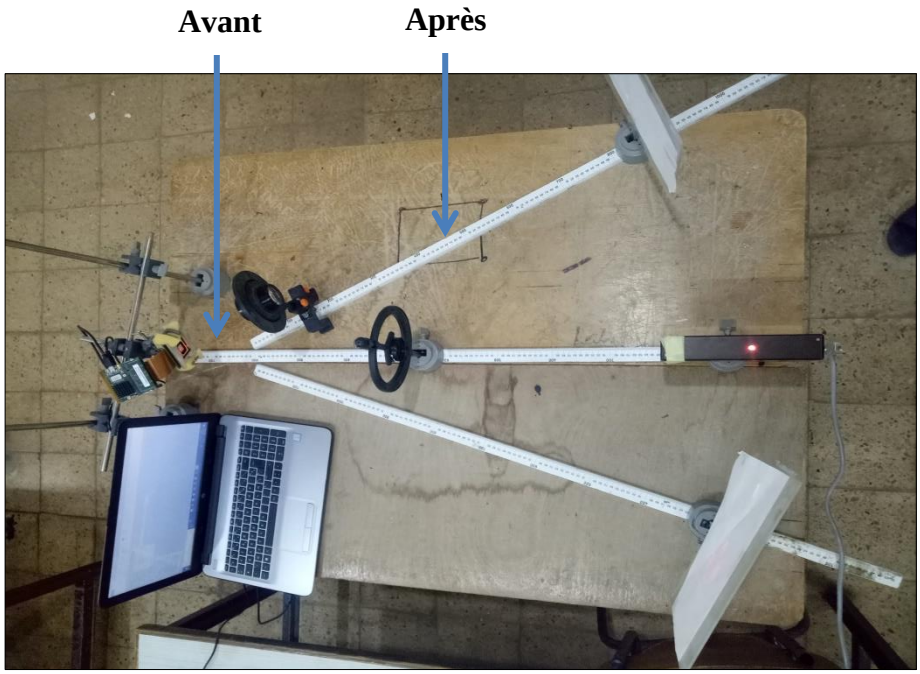


Figure IV.7. Etat du laser avant et après traversée des lentilles L2, L3

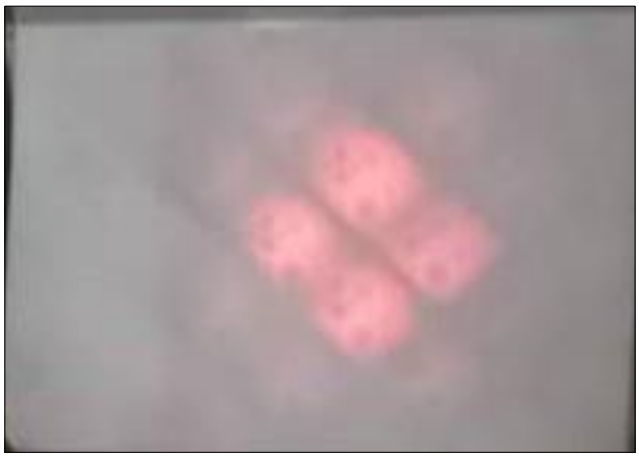


Figure IV.8. Faisceau laser avant L2 ou L3

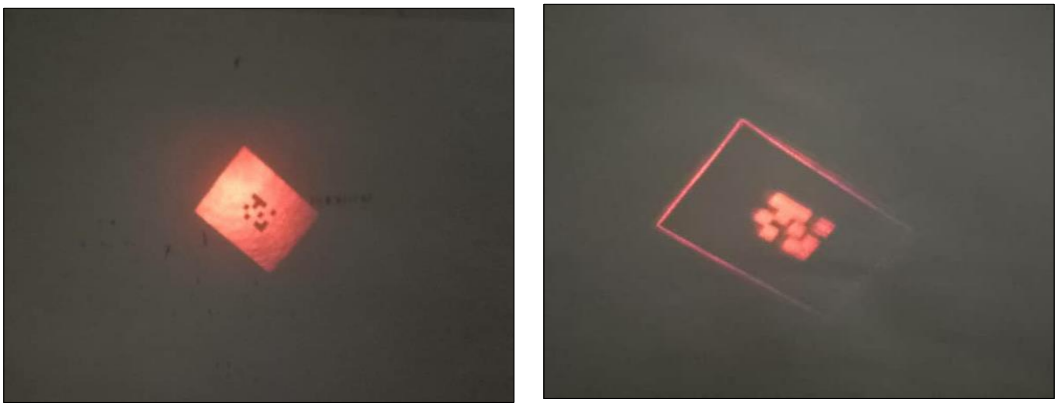


Figure IV.9. Faisceau laser après les lentilles L2 et L3

IV.2.3 Génération des motifs à afficher sur le DMD sous VB

Ce programme représente le motif d'un damier (fig. IV.10), le motif couvre toute la surface du DMD et chaque carré est de 64 pixels.

```
Dim Pattern(m_nSizeX * m_nSizeY) As Byte, x As Int32, y As Int32
        For y = 0 To m_nSizeY - 1 Step 1
            For x = 0 To m_nSizeX - 1 Step 1
                If ((x Xor y) And 64) = 0 Then
                    Pattern(y * m_nSizeX + x) = 128
                Else
                    Pattern(y * m_nSizeX + x) = 0
                End If
            Next
        Next
```

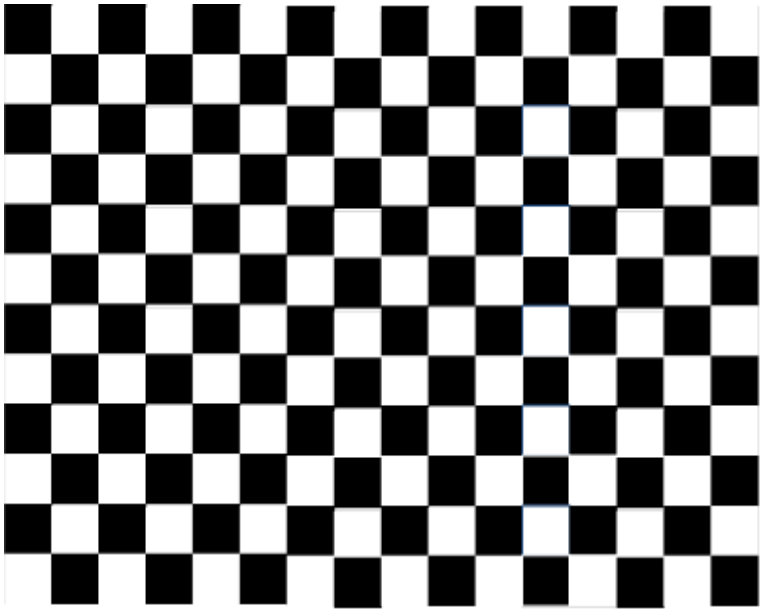


Figure IV.10 Représentation l'image en damier

L'image est envoyée au DMD via le bouton pattern de l'interface de contrôle du DMD, cette image est ensuite projetée sur la cible par l'émission des faisceaux laser sur le DMD.

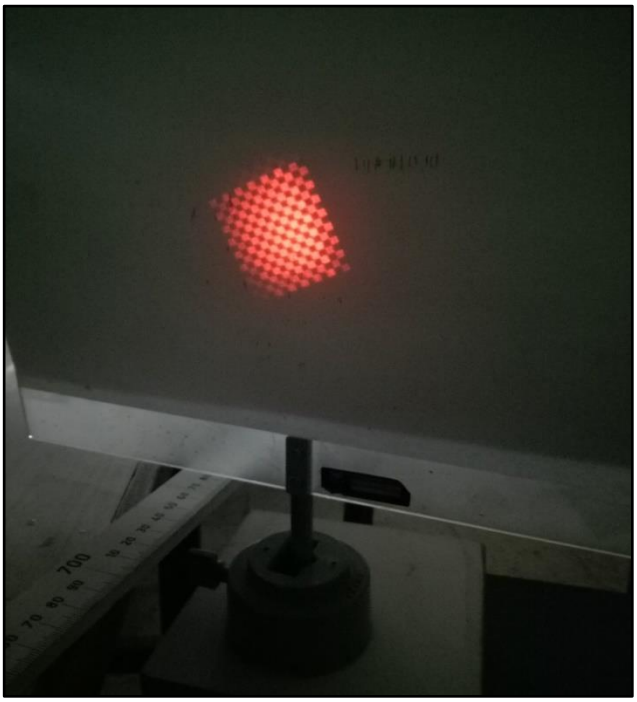


Figure IV.11. Image damier projetée par le DMD sur la cible

Nous avons généré un nouvel programme pour visualiser un autre motif représenté sur la figure IV.12.

```
Dim W As Int32 = 154
Dim H As Int32 = 100
Dim Pattern(m_nSizeX * m_nSizeY) As Byte, x As Int32, y As Int32
        For y = 0 To m_nSizeY - 1 Step 1
            For x = 0 To m_nSizeX - 1 Step 1
                Pattern(y * m_nSizeX + x) = 128
            Next
        Next
        For y = 256 + H To 320 - 1 + H Step 1
            For x = 416 + W To 480 - 1 + W Step 1
                Pattern(y * m_nSizeX + x) = 0
            Next
        Next
        For y = 320 + H To 384 - 1 + H Step 1
            For x = 352 + W To 416 - 1 + W Step 1
                Pattern(y * m_nSizeX + x) = 0
            Next
        Next
        For y = 256 + H To 320 - 1 + H Step 1
            For x = 544 + W To 672 - 1 + W Step 1
                Pattern(y * m_nSizeX + x) = 0
            Next
        Next
        For y = 320 + H To 384 - 1 + H Step 1
            For x = 608 + W To 672 - 1 + W Step 1
                Pattern(y * m_nSizeX + x) = 0
            Next
        Next
```

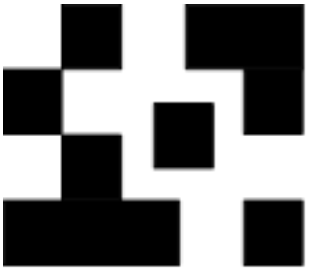


Figure IV.12. Représentation du motif généré

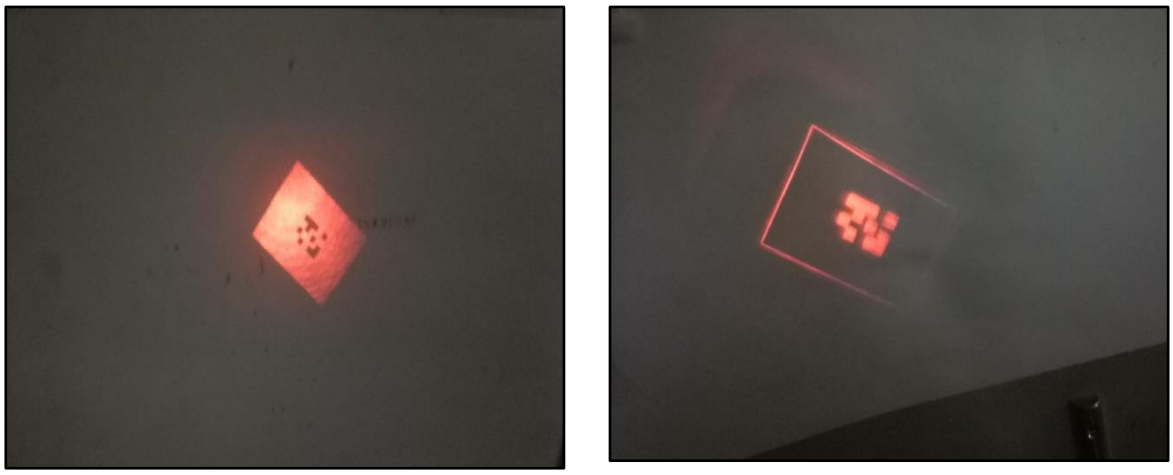


Figure IV.13. Image projetée par le DMD sur la cible

L'image à gauche représente l'image réelle projetée par le DMD et celle à droite représente l'image négative (image inverse de l'image réelle).

IV.2.4 Poursuite de la cible par Deep learning

Le principe d'utilisation du deep learning, est de fournir une base d'apprentissage à un réseau de type Deep CNN, et de présenter au réseau obtenu les images afin qu'il détecte le nouvel emplacement des objets déplacés.

Nous avons entrainés notre modèle de détecteur d'objet de Tensorflow avec les bases de données que nous avons générées sur Matlab.

Notre première base de données comprend des images avec un fond blanc et du noir qui simule les anomalies liées à la rétine. Le but est de détecter le déplacement des points noirs dans l'image et de soustraire les coordonnées (x, y) du déplacement afin que le DMD réoriente les faisceaux du laser vers la cible déplacée.

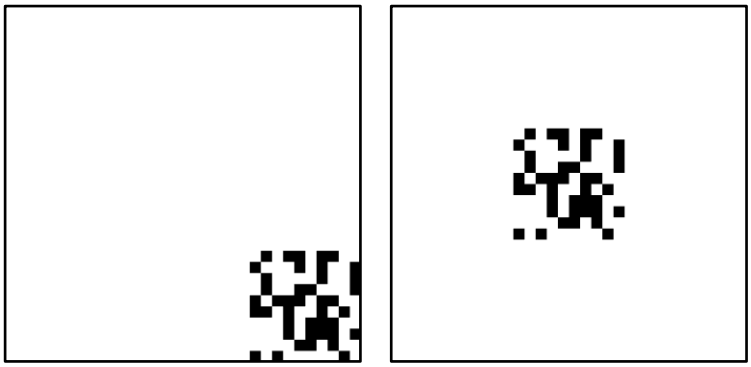


Figure IV.14. Exemples d'images de la 1ère base de données

On a étiqueté ces images avec le logiciel LabelImg, ensuite scindé les images (20% pour le test et 80% pour le train). Nous avons lancé l'entrainement avec la commande Prompt de Anaconda.

Figure IV.15. Résultat obtenu de la première base de données

Le résultat obtenu de l'entrainement du deep learning avec les images de la première base de données est mauvais. L'entrainement a duré des heures et on remarque que le **loss** qui s'affiche est anormale.

C'est d'ailleurs ce qui nous a conduit a créé une nouvelle base de données, dont les images contient un fond noir et une image de rétine atteinte.

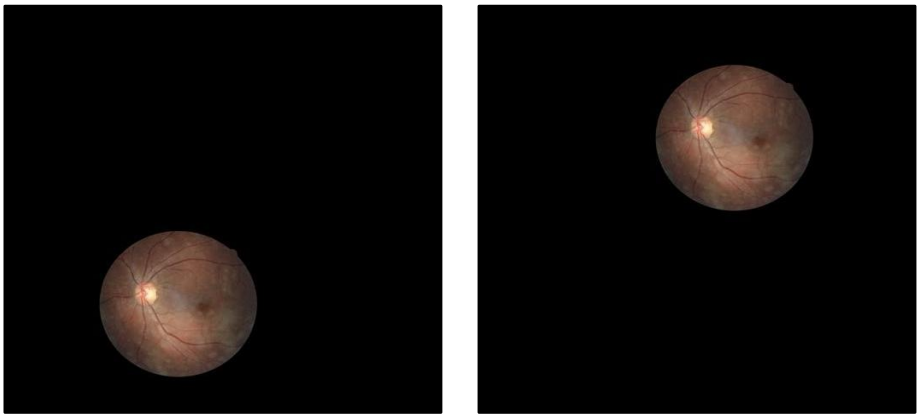


Figure IV.16. Exemples d'images de la deuxième base de données

Nous avons lancé l'entrainement avec les images de la deuxième base de données.

Figure IV.17. Début d'exécution de l'entrainement

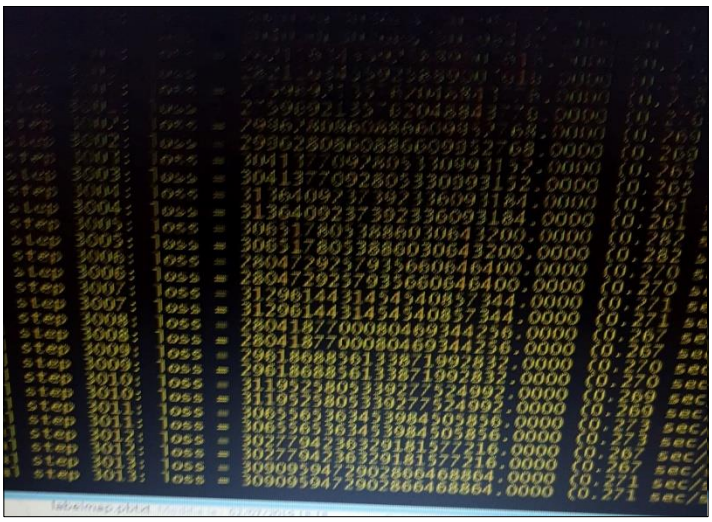


Figure IV.18. Fin d'exécution de l'entrainement

Au début de l'entrainement (fig.IV.17) nous avons un **loss** variant de 0.6000 à 1.000, ce résultat obtenu n'est pas satisfaisant. Ensuite on obtient un **loss** anormale (fig.IV.14) comme le résultat obtenu pour l'entrainement de la première base de données.

A la fin de l'exécution on a obtenu un **loss** égale à zéro (loss = 0), ce résultat est illustré sur la figure IV.19.

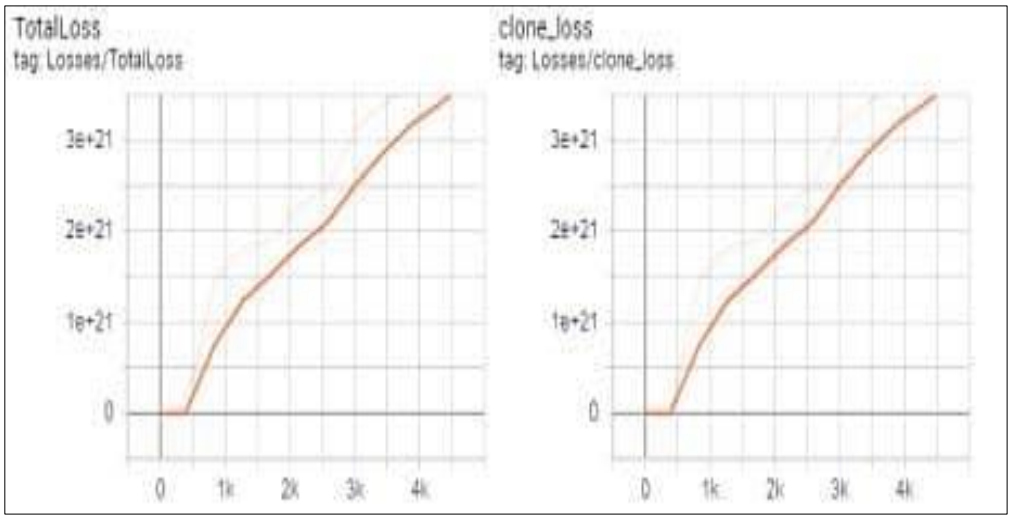


Figure IV.19. Graphe obtenu à la fin de l'entrainement

Nous avons déduit de ses résultats obtenus que nos bases de données ne soient pas adaptées pour ce modèle deep learning. Le modèle requière des images d'apprentissage comportant des objets aléatoires dans l'image, ainsi que de divers arrière-plans et conditions d'éclairage…

IV.2.5 Poursuite de la cible avec l'algorithme développé sous Matlab

Le modèle de deep learning sur lequel nous nous sommes basé ne nous a pas fourni des résultats satisfaisant, nous avons pensé à d'autres alternatives. Nous avons développé deux algorithmes de poursuites et détection de la cible avec le logiciel Matlab.

Le premier algorithme consiste à comparer les images pixel par pixel et le deuxième algorithme utilise Vision Template Matcher qui est un objet de Matlab.

IV.2.5.1 Premier algorithme (comparaison logique)

Nous donnons à algorithme deux images selon le programme qu'on a implémenté qui est décrit dans le chapitre III.

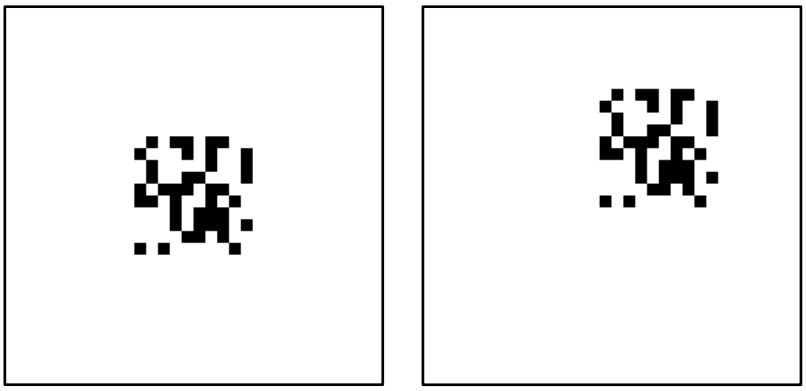


Image A Image B

Figure IV.20. Images utilisées pour la détection

L'image A est la translation de l'image D qui est à l'origine du balayage pixel par pixel de l'image B en faisant une comparaison logique des pixels des images. Une fois que l'image D trouve la position à laquelle l'image B est déplacée, l'exécution s'arrête et nous donne les coordonnées (x, y).

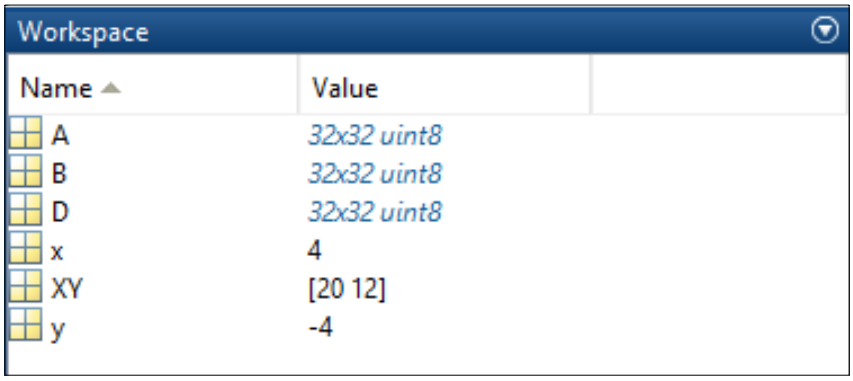


Figure IV.21. Résultats obtenue par comparaison de pixels

Nous avons obtenues les coordonnées x = 4, y = - 4 qui correspondent à l'emplacement de l'image modèle (10x10) dans le l'image de fond blanc (32x32). Voir chapitre III

En remplaçant ses coordonnées dans l'instruction Imgf (12+ x : 21+ x, 12+ y : 21+ y) = Img ;

On a >> Imgf (16 : 25, 8 : 17).

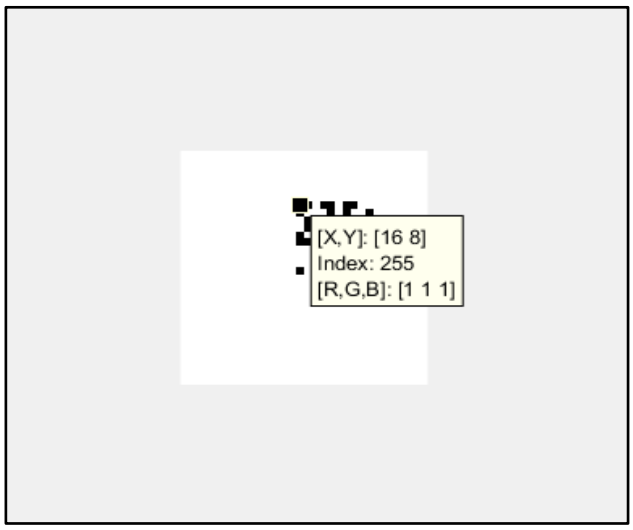


Figure IV.22. Position du modèle (10x10) dans l'image (32x32)

On a utilisé une autre instruction pour cibler le milieu de l'image modèle, ainsi au lieu d'avoir Imgf (16 : 25, 8 : 17) qui nous donne les extrémités du modèle dans l'image, on utilise :
XY = [16+x, 16+y]. En remplaçant les coordonnées x = 4 et y = - 4 dans l'instruction, on a XY = [20, 12].

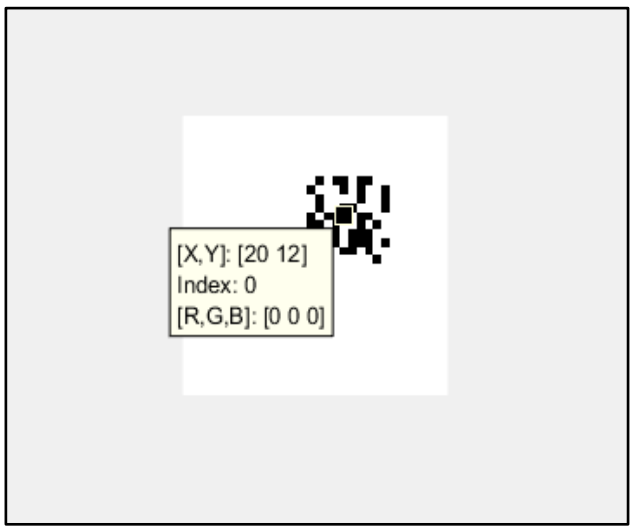


Figure IV.23. Poursuite et détection du modèle

Nous avons pu faire la poursuite et obtenir les coordonnées, mais dans les conditions réelles d'utilisation, cet algorithme ne va pas fonctionner. Parce que nous utilisons là des images de la base de données que nous avons générées, alors que dans les conditions réelle il y'a beaucoup de facteurs qui entre en jeu dans l'acquisition d'image, à savoir : la luminance, le teint, la saturation de l'image. Il est difficile de faire des captures d'images et ayant les mêmes

valeurs de ces facteurs. Car notre algorithme fait une comparaison logique pixel par pixel, il suffit d'une valeur de pixel qui ne soit pas les mêmes dans les images pour que tout soit faux.

IV.2.5.2 Deuxième algorithme (Vision Template Matcher)

Nous utilisons un objet de Matlab Vision Template Matcher qui permet de localiser un modèle dans une image et renvoie ses coordonnées x, y dans l'image. On a détaillé l'algorithme dans le chapitre III. Nous utilisons les mêmes images de la base données de la figure IV.20 ci-dessus.

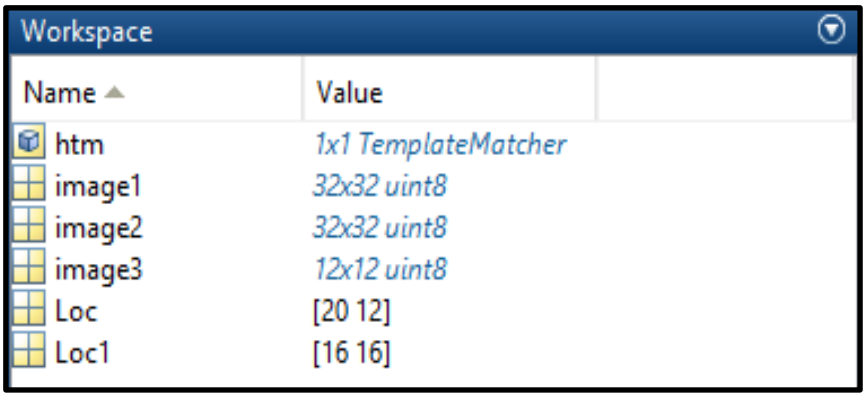


Figure IV.24. Résultats obtenu avec vision template matcher

Le résultat obtenu après exécution du programme. L'image3 contient le modèle et nous avons déterminé les coordonnées du modèle dans les images 1 et 2.

L'instruction **Loc = step (htm, image2, image3)**, prend en paramètre l'objet créer Vision Template Matcher, l'image dans laquelle on cherche le modèle et le modèle.

Les coordonnées obtenues Loc et Loc1 sont respectivement de l'image2 (image B) et l'image1 (image A).

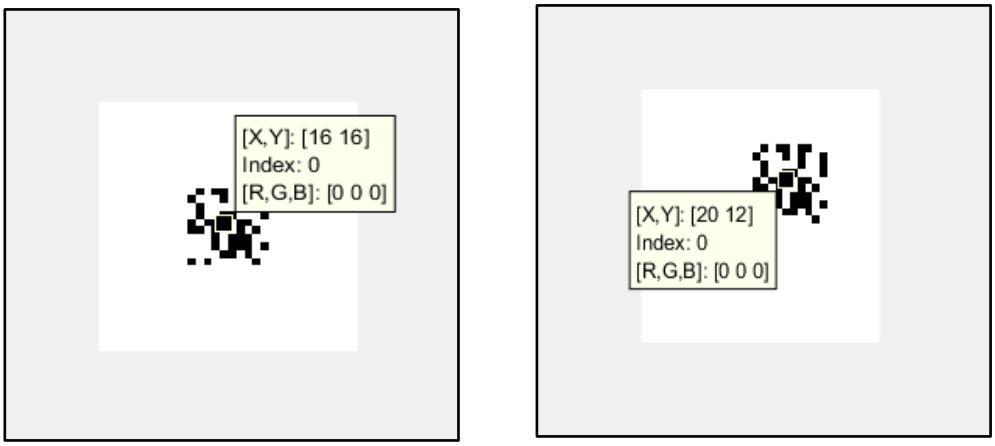


Figure IV.25. Les coordonnées obtenues avec vision template matcher

Nous avons utilisés cet algorithme dans les conditions réelles pour obtenir les coordonnées d'un motif sur la cible.

Figure IV.26. Images capturées de la cible

L'image de gauche est considérée comme étant initiale, celle de droite est l'image où le motif est déplacé. Nous avons déterminé la position du motif dans l'image de gauche ensuite de celle de droite pour montrer que le motif est vraiment déplacé.

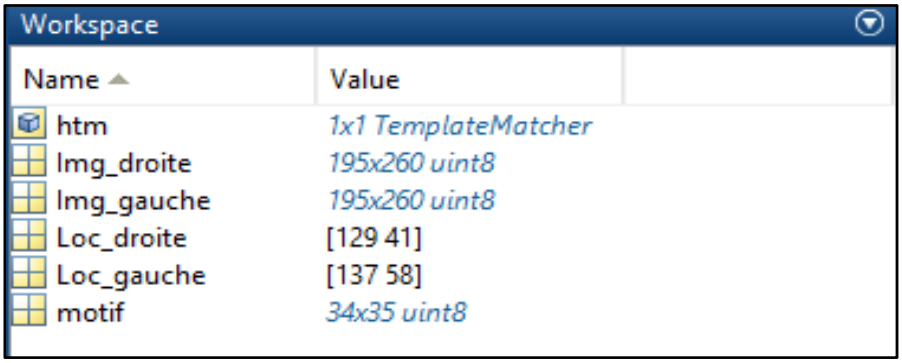


Figure IV.27. Résultats obtenus en condition réelle

Figure IV.28. Résultat obtenu de l'image droite

Figure IV.29. Résultat obtenu de l'image de gauche

D'après les résultats obtenus nous pouvons affirmer que l'image est déplacée et qu'on a pu obtenir de façon précise ses coordonnées dans les conditions réelles.

Afin de pouvoir rediriger les micromiroirs du DMD, on envoie les deltas x et celui de y multiplier par le facteur d'agrandissement G comme les nouvelles coordonnées de l'image numérique dans notre programme implémenter sur Visual basic.

$$G = i / o$$

i : la hauteur de l'image sur la cible

o : la hauteur de l'objet (la hauteur de l'espace micro-miroir du DMD)

IV.3 Conclusion

Dans ce chapitre nous avons montré les résultats obtenus pour les réglages que nous avons faits sur le dispositif optique afin d'obtenir une meilleure projection d'image sur la cible. Nous avons obtenu des résultats satisfaisants, pour la poursuite d'une cible en mouvement avec les programmes implémentés sous Matlab. Faute de temps, nous n'avons cependant pas pu étudier l'algorithme de Deep Learning, afin de mettre en évidence les problèmes, et les résoudre.

Conclusion Générale

Conclusion Générale

Ce projet représente un grand avenir dans le traitement des maladies rétiniennes. Il permet de résoudre le problème de temps de traitement et de précision, qui sont deux points cruciaux lors du traitement par laser d'une partie de ces maladies.

Les micromiroirs du DMD du module V7000 permettent de moduler un faisceau laser plein en un ensemble de rayons lasers pouvant être projetés (grâce au dispositif optique que nous avons conçu et réalisé autour du DMD et sa commande par PC, par les programmes API, pilotes … que nous avons développés sous Visual Basic, sous Visual Studio) simultanément, et avec précision, sur les points de la rétine à traiter, ou sur toute autre cible ; cela permet de réduire plusieurs fois (des dizaines à des centaines de fois) le temps d'exposition et de traitement du patient, réduisant ainsi l'inconfort du patient, et celui de l'ophtalmologiste.

Le fait que la rétine soit un organe assez instable, vu qu'un simple battement de cœur peut la déplacer, nous avons eu pour tâche de contrôler les points d'impact des rayons laser sur la rétine par le DMD, en affichant sur le DMD l'image déplacée de la cible (déplacement évalué grâce aux images de la cible, saisies par WEBCAM), pour positionner les rayons laser sur la cible déplacée. Nous avons utilisé trois méthodes : un algorithme de poursuite que nous avons élaboré sous Matlab, pour illustrer le principe, un algorithme basé sur la fonction de poursuite de Matlab, et un algorithme de deep learning pour la détection développé sous Matlab. Les deux premières nous ont donné de bons résultats, cependant, faute de temps, nous n'avons pas pu aller au bout de nos investigations pour faire aboutir la troisième.

Le deep Learning étant au cœur des avancées scientifiques, ainsi que ses hautes performances avérées, et l'utilisation judicieuse des micromiroirs du DMD nous permet d'envisager que ce projet puisse être innovateur. Sachant que de plus en plus de personne sont atteinte de cette maladie, notre projet présente de nombreux avantages :

- Gain de temps de traitement
- Gain de précision dans le traitement
- Réduction des coûts de traitement
- Réduction de budget matériel etc…

En perspective, le V-module dispositif optique utilisé peut-être adapté à de multiples tâches, parmi lesquelles on peut citer la fonction scanné qui est une perspective à développer. Il est possible de remarquer qu'il y a deux problèmes majeurs dans notre projet. Notamment la taille de l'unité optique, ainsi que l'échec de l'apprentissage profond (deep Learning). Par rapport au premier problème, il est possible d'y palier en jouant sur la focale des lentilles L1, L2 et L3 ; on pourra avoir un dispositif opto-électromécanique que l'on pourra embarquer,

avec son contrôleur, sur un petit système, voire un microsystème (MOEMS). Et le deuxième peut être résolu en utilisant des images réelles fournies par des ophtalmologues, et en testant des DCNN de structures différentes.

Bibliographie

Bibliographie

1. FRM : Maladies de la rétine. Dernier accès, juillet 2019, Du site internet : https://www.frm.org/recherches-autres-maladies/maladies-de-la-retine.
2. Wikipédia : Retinopathy. Dernier accès, juillet 2019, Du site internet : https://en.wikipedia.org/wiki/Retinopathy
3. FUTURA SANTÉ : Corps humain-Œil. Dernier accès, juillet 2019, Du site internet : https://www.futura-sciences.com/sante/definitions/corps-humain-%C5%93il-14131/
4. Fleury Opticiens : Anatomie de l'œil. Dernier accès, juillet 2019, Du site internet : https://www.fleury.ch/anatomie-de-l-oeil/anatomie-de-l-oeil.
5. FUTURA SANTÉ : Corps humain-Œil. Dernier accès, juillet 2019, Du site internet : https://www.futura-sciences.com/sante/definitions/corps-humain-%C5%93il-14131/
6. Passeport Santé : La rétinopathie, qu'est-ce que c'est ? Dernier accès, juillet 2019, Du site internet : https://www.passeportsante.net/fr/Maux/Problemes/Fiche.aspx?doc=retinopathie.
7. Medisite : Qu'est-ce que la rétinopathie diabétique ? Dernier accès, juillet 2019, Du site internet : https://www.medisite.fr/autres-maladies-des-yeux-retinopathie-diabetique-quels-sont-les-traitements.1217118.146448.html.
8. EVision : La rétinopathie diabétique. Dernier accès, juillet 2019, Du site internet : https://evision.fr/traitements/la-retinopathie-diabetique/
9. INSTITUT DE LA MACULA : Photocoagulation au laser. Dernier accès, juillet 2019, Du site internet : http://www.institutmacula.com/fr/tratamiento/fotocoagulacion-laser/
10. VIALUX: Hi-speed-v-modules. Dernier accès, juillet 2019, Du site internet : https://www.vialux.de/en/hi-speed-v-modules.html.
11. DIGITALINSIDERS : Qu'est-ce que le Machine Learning ? Dernier accès, juillet 2019, Du site internet : https://digitalinsiders.feelandclic.com/construire/definition-quest-machine-learning.
12. FUTURA TECH : Deep Learning. Dernier accès, juillet 2019, Du site internet : https://www.futura-sciences.com/tech/definitions/intelligence-artificielle-deep-learning-17262/
13. DataScienceToday : Les réseaux de neurones convolutifs. Dernier accès, juillet 2019, Du site internet : https://www.datasciencetoday.net/index.php/fr/deep-learning/173-les-reseaux-de-neurones-convolutifs.
14. One pointe. Wave : mécanisme d'attention simple astuce mécanisme universel. Dernier accès, juillet 2019, Du site internet : https://weave.eu/mecanisme-dattention-simple-astuce-mecanisme-universel/

15. Bouali Mohamed, Boueubdllah Rabah. Contrôle par PC des micro-miroirs D'un <<DMD>> pour la modulation et le positionnement X, Y sur la rétine, d'un faisceau laser issu d'une source lumineuse fixe. Mémoire en électronique. Université Saad Dahlab de Blida. 2017.
16. Anaconda : Tensorflow a anaconda. Dernier accès, juillet 2019, Du site internet : https://www.anaconda.com/tensorflow-in-anaconda/
17. Github : TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10. Dernier accès, juillet 2019, Du site internet : https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10.
18. Mathworks : imtranslate. Dernier accès, juillet 2019, Du site internet : https://www.mathworks.com/help/images/ref/imtranslate.html
19. Mathworks : vision.TemplateMatcher. Dernier accès, juillet 2019, Du site internet : https://www.mathworks.com/help/vision/ref/vision.templatematcher-system-object.html

Annexes

Le programme complet pour le contrôle du DMD sous Visual Studio :

```
Imports PFE_2018_2019.AlpImport

Public Class AlpTest
    Private Function AlpErrorString(ByVal nRet As Int32) As String
        ' See also the C header file "alp.h" for values of the return codes.
        ' This file also contains the values of other control types and
        ' special values.
        Select Case nRet
            Case AlpReturnCodes.ALP_OK
                AlpErrorString = "ALP_OK"
            Case AlpReturnCodes.ALP_NOT_ONLINE
                AlpErrorString = "ALP_NOT_ONLINE"
            Case AlpReturnCodes.ALP_NOT_IDLE
                AlpErrorString = "ALP_NOT_IDLE"
            Case AlpReturnCodes.ALP_NOT_AVAILABLE
                AlpErrorString = "ALP_NOT_AVAILABLE"
            Case AlpReturnCodes.ALP_NOT_READY
                AlpErrorString = "ALP_NOT_READY"
            Case AlpReturnCodes.ALP_PARM_INVALID
                AlpErrorString = "ALP_PARM_INVALID"
            Case AlpReturnCodes.ALP_ADDR_INVALID
                AlpErrorString = "ALP_ADDR_INVALID"
            Case AlpReturnCodes.ALP_MEMORY_FULL
                AlpErrorString = "ALP_MEMORY_FULL"
            Case AlpReturnCodes.ALP_SEQ_IN_USE
                AlpErrorString = "ALP_SEQ_IN_USE"
            Case AlpReturnCodes.ALP_HALTED
                AlpErrorString = "ALP_HALTED"
            Case AlpReturnCodes.ALP_ERROR_INIT
                AlpErrorString = "ALP_ERROR_INIT"
            Case AlpReturnCodes.ALP_ERROR_COMM
                AlpErrorString = "ALP_ERROR_COMM"
            Case AlpReturnCodes.ALP_DEVICE_REMOVED
                AlpErrorString = "ALP_DEVICE_REMOVED"
            Case AlpReturnCodes.ALP_NOT_CONFIGURED
                AlpErrorString = "ALP_NOT_CONFIGURED"
            Case AlpReturnCodes.ALP_LOADER_VERSION
                AlpErrorString = "ALP_LOADER_VERSION"
            Case AlpReturnCodes.ALP_ERROR_POWER_DOWN
                AlpErrorString = "ALP_ERROR_POWER_DOWN"
            Case Else
                                AlpErrorString = "(unknown error" & Format(nRet) & ")"
        End Select
    End Function


    Private m_AlpDeviceId As Int32, m_AlpSeqId1 As Int32, m_AlpSeqId2 As Int32
```

```
    Private m_nDmdType As AlpDevValues, m_nSizeX As Int32, m_nSizeY As Int32

    Private Sub Alloc_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Alloc.Click
        Dim nRet As Int32

        ' Allocate ALP device (no precautions if already allocated!)

        ' Known Errors:
        ' System.DllNotFoundException -> ALP DLL must be available, e.g. in the same
directory as this exe file
        ' System.BadImageFormatException -> ALP DLL platform does not match. Try the
Win32 (x86) or the x64 version.
        nRet = AlpDevAlloc(0, 0, m_AlpDeviceId)
        If nRet <> 0 Then
            MessageBox.Show("AlpDevAlloc Error" & _
                vbCrLf & AlpErrorString(nRet))
            Exit Sub
        End If

        ' Inquire the DMD type in order to retrieve data size information
        nRet = AlpDevInquire(m_AlpDeviceId, AlpDevTypes.ALP_DEV_DMDTYPE,
m_nDmdType)
        If nRet <> 0 Then
            MessageBox.Show("AlpDevInquire (DMD TYPE) Error" & _
                vbCrLf & AlpErrorString(nRet))
            Exit Sub
        End If
        Select Case m_nDmdType
            Case AlpDevValues.ALP_DMDTYPE_XGA_055A, _
                AlpDevValues.ALP_DMDTYPE_XGA_055X, _
                AlpDevValues.ALP_DMDTYPE_XGA_07A
                m_nSizeX = 1024
                m_nSizeY = 768
            Case AlpDevValues.ALP_DMDTYPE_1080P_095A, _
                AlpDevValues.ALP_DMDTYPE_DISCONNECT
                m_nSizeX = 1920
                m_nSizeY = 1080
            Case AlpDevValues.ALP_DMDTYPE_WUXGA_096A
                m_nSizeX = 1920
                m_nSizeY = 1200
            Case Else
                MessageBox.Show("Unknown DMD type: " & Format(m_nDmdType))
                Exit Sub
        End Select

        ' Allocate 2 ALP sequences: 1 bit, 1 picture each
        nRet = AlpSeqAlloc(m_AlpDeviceId, 1, 1, m_AlpSeqId1)
        If nRet <> 0 Then
            MessageBox.Show("AlpSeqAlloc(1) Error" & _
```

```
        vbCrLf & AlpErrorString(nRet))
    Exit Sub
End If
nRet = AlpSeqAlloc(m_AlpDeviceId, 1, 1, m_AlpSeqId2)
If nRet <> 0 Then
    MessageBox.Show("AlpSeqAlloc(2) Error" & _
        vbCrLf & AlpErrorString(nRet))
    Exit Sub
End If

' Send image data of sequence 1 (checkered pattern, 64x64 squares)
Dim Pattern(m_nSizeX * m_nSizeY) As Byte, x As Int32, y As Int32
For y = 0 To m_nSizeY - 1 Step 1
    For x = 0 To m_nSizeX - 1 Step 1
        If ((x Xor y) And 64) = 0 Then
            Pattern(y * m_nSizeX + x) = 128
        Else
            Pattern(y * m_nSizeX + x) = 0
        End If
    Next
Next
nRet = AlpSeqPut(m_AlpDeviceId, m_AlpSeqId1, 0, 1, Pattern)
If nRet <> 0 Then
    MessageBox.Show("AlpSeqPut(1) Error" & _
        vbCrLf & AlpErrorString(nRet))
    Exit Sub
End If

' Send image data of sequence 2 (checkered pattern, 128x128 squares)
For y = 0 To m_nSizeY - 1 Step 1
    For x = 0 To m_nSizeX - 1 Step 1
        If ((x Xor y) And 128) = 0 Then
            Pattern(y * m_nSizeX + x) = 128
        Else
            Pattern(y * m_nSizeX + x) = 0
        End If
    Next
Next
nRet = AlpSeqPut(m_AlpDeviceId, m_AlpSeqId2, 0, 1, Pattern)
If nRet <> 0 Then
    MessageBox.Show("AlpSeqPut(2) Error" & _
        vbCrLf & AlpErrorString(nRet))
    Exit Sub
End If

' Example: pulse Synch Output 1 each first of three frames.
            ' (Not available prior to ALP-4)
nRet = AlpDevControlEx_SynchGate(m_AlpDeviceId, 1, True, New Byte() {1, 0, 0})
If nRet <> 0 Then
    MessageBox.Show("AlpDevControlEx_SynchGate Error" & _
```

```
            vbCrLf & AlpErrorString(nRet))
         Exit Sub
      End If

      MessageBox.Show("Success." & vbCrLf & _
         "Allocated ALP device and 2 sequences." & vbCrLf & _
         "Sent image data of 2 sequences.")
   End Sub

   Private Sub Free_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Free.Click
      Dim nRet As Int32

      ' Disable SynchGate1 output: Omit "Gate" parameter
                  ' (Not available prior to ALP-4)
      nRet = AlpDevControlEx_SynchGate(m_AlpDeviceId, 1, True, New Byte() {})

      ' Halt ALP device (no precautions if not allocated!)
      nRet = AlpDevHalt(m_AlpDeviceId)
      If nRet <> 0 Then
         MessageBox.Show("AlpDevHalt Error" & _
            vbCrLf & AlpErrorString(nRet))
         Exit Sub
      End If
      ' Free ALP device
      nRet = AlpDevFree(m_AlpDeviceId)
      If nRet <> 0 Then
         MessageBox.Show("AlpDevFree Error" & _
            vbCrLf & AlpErrorString(nRet))
         Exit Sub
      End If
      MessageBox.Show("Success." & vbCrLf & _
         "Halted and released ALP device.")
   End Sub

   Private Sub Pattern1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Pattern1.Click
      Dim nRet As Int32

      ' Display sequence 1 (no precautions if not allocated!)
      nRet = AlpProjStartCont(m_AlpDeviceId, m_AlpSeqId1)
      If nRet <> 0 Then
         MessageBox.Show("AlpProjStartCont(1) Error" & _
            vbCrLf & AlpErrorString(nRet))
         Exit Sub
      End If
   End Sub

   Private Sub Pattern2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Pattern2.Click
```

```
        Dim nRet As Int32

        ' Display sequence 2 (no precautions if not allocated!)
        nRet = AlpProjStartCont(m_AlpDeviceId, m_AlpSeqId2)
        If nRet <> 0 Then
            MessageBox.Show("AlpProjStartCont(2) Error" & _
                vbCrLf & AlpErrorString(nRet))
            Exit Sub
        End If
    End Sub

    Private Sub AlpTest_Load(sender As System.Object, e As System.EventArgs) Handles
MyBase.Load

    End Sub
End Class
```