REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université Saad DAHLEB - BLIDA

Institut d'Aéronautique et des études spatiales

Projet de fin d'étude
En vue de l'obtention du diplôme Master

Spécialité : CNS/ATM

Communication, Navigation et surveillance / Air trafic Management

THÈME

Transmission d'une vidéo en temps réel

Présenté par :
KHELIFA SARAH

Promoteur : Mme AZINE HOURIA

Promotion : 2016-2017

SOMMAIRE

REMERCIEMENT

À ceux qui ont tout sacrifié pour moi, Mes chers parents et mon époux Bilel.
À celui qui a tout changé à mes yeux, mon petit ange Anes.

C'est avec un grand plaisir que je réserve ces lignes en témoignage de ma reconnaissance à tous ceux qui m'ont accompagné, soutenu et aidé pour accomplir les travaux présentés dans ce manuscrit.
J'exprime mes profonds remerciements à Mme H.Azine, ma directrice de thèse, qui a bien accepté de superviser mes travaux.

Je suis très reconnaissante envers Monsieur K.Radouane qui a été d'une grande aide, A mon époux L.Bilel qui m'a encouragé, soutenu et enfin aidé à la relecture de ce document.

Je ne manquerais pas de remercier mes chers parents qui m'ont soutenu depuis le tout début de mes études, et qui n'ont jamais cessé de m'encourager de si loin.

Enfin, je tiens aussi à remercier mes frères et sœurs et toute ma famille pour leur soutien inconditionnel.

RESUME

Cette thèse présente une implémentation de la transmission de la vidéo en temps réel suivant un algorithme de communication en langage C#.

La compression de la vidéo a été effectuée en adoptant le standard JPEG de compression des images.

Le Protocol de communication utilisé est l'UDP en appliquant la méthode de programmation en sockets en mode non connecté.

L'implémentation a été validée par une série des expériences où les caractéristiques essentielles ont été abordées, suivie par une étude exhaustive de la robustesse de cette implémentation en agissants sur les paramètres les plus susceptibles dans la communication.

Mots clés : Compression vidéo, JPEG, Sockets, UDP.

ABSTRACT

This thesis presents an implementation of the real time video transmission according to a communication algorithm using C# language.

The compression of the video was carried out by adopting the standard JPEG of Images compression.

The communication protocol used is the UDP by applying the method of programming in sockets in unconnected mode.

The implementation was validated by a series of experiments where the essential characteristics were addressed, followed by a comprehensive study of the robustness of this implementation by acting on the communication most essential parameters.

Keywords: Video Compression, JPEG, Sockets, UDP

ملخص

هذه الأطروحة تعرض تنفيذ نقل فيديو في الوقت الحقيقي باستعمال خوارزمية الاتصالات المبرمجة حسب لغة البرمجة C#.

تقليص حجم الفيديو تم بإتباع المعيار العالمي JPEG لضغط الصور.

البروتوكول المتبع لتنفيذ هذا النقل هو UDP من خلال تطبيق البرمجة ب SOCKETS على الطريقة غير المتصلة.

التنفيذ تم بتطبيق سلسلة من التجارب أين تم تناول أهم المواصفات, تليها دراسة شاملة لإبراز فعالية هذا التنفيذ من خلال العمل على المعايير الأكثر تأثيرا في الاتصالات.

الكلمات الرئيسية: تقليص الفيديو JPEG, SOCKETS, UDP.

NOMENCLATURE

ARP: Address Resolution Protocol.

ARQ: messages d'acquittement.

BMP: Bitmap.

BSD: Barkeley software distribution.

CODEC: codeur/decodeur.

DCT: Discrete Cosine Transform.

DV: Digital vidéo.

EOB: end of bits.

FFT : fast fourier transform

GIF: Graphics interchange format.

ICMP: Internet control message protocol

IPC: inter processus communication.

JPEG: Joint Photographic Expert Group.

LLC: Logical Link Control.

MAC: Media access control

MPEG : Moving Picture Experts Group.

OSI: Open systems interconnection.

OS: operating system.

PCX : Un format d'image numérique dont l'encodage est basé sur une forme de RLE.

PNG : portable network graphics.

RTP: Real-time Transport Protocol.

RVB/RGB: rouge vert bleu/ red green blue

RLE: Run Length Encoding.

SDI: Serial digital interface

TCP / IP: transmission control protocol/internet protocol

TGA: Truevision Targa

UDP: User Datagram Protocol

VLC: Video Lan client.

VHS: Video home system

WMF: Windows media format.

Y'Cb et Cr : Y'(rouge+vert+bleu), Cb(Y'-bleu) et Cr(Y'-rouge).

LISTE DES FIGURES

CHAPITRE 1

CHAPITRE 2

CHAPITRE 3

CHAPITRE 4

LISTE DES TABLEAUX

CHAPITRE 2

CHAPITRE 4

INTRODUCTION GENERALE

Depuis l'avènement de l'informatique moderne, les ordinateurs ont été utilisés pour afficher le contenu multimédia. Au début ces contenus étaient relativement simples, puis, au fil du temps, les ordinateurs sont devenus capables d'afficher des contenus de plus en plus complexes. Puis l'idée de transmettre ces vidéos en temps-réel d'une machine à une autre est née. Il restait cependant de nombreux défis à relever. En effet, transmettre des vidéos en temps-réel nécessite des processeurs assez puissants pour supporter le décodage de la vidéo ainsi qu'une bande passante assez élevée pour la transmettre. Cependant, le débit disponible était très limité et la seule solution viable était de télécharger le média pour le visionner ultérieurement.

Les données vidéo ont besoin d'un protocole de transport pour être acheminées de la source vers la destination. La conception d'un tel protocole doit prendre en compte la contrainte temps réel qu'impose le visionnage en direct du contenu. UDP répond à cette exigence mais il ne garantit pas l'acheminement des données et c'est à l'application réceptrice de détecter les pertes de données et de les restituer.

Une image d'une vidéo non compressée occupe une taille d'environ 1 Mo. Afin d'obtenir une vidéo paraissante fluide, il est nécessaire d'avoir une fréquence d'au moins 25 ou 30 images par seconde, ce qui produit un flux de données d'environ 30 Mo/s, soit plus de 1.5 Go par minute. Il est évident que ce type de débit est peu compatible avec les espaces de stockage des ordinateurs personnels ni même avec les connexions réseau de particuliers ou de petites et moyennes entreprises.

Ainsi, afin de pallier à cette difficulté, il est possible de recourir à des algorithmes permettant de réduire significativement les flux de données en compressant/décompressant les données vidéos appelés communément CoDec où on trouve généralement l'utilisation du standard JPEG pour des images constituant une séquence vidéo.

Le principal défi de ce projet était d'implémenter un algorithme de communication réseau capable de répondre aux exigences suivantes:

- Faire un choix judicieux du matériel à utiliser.
- Proposer un algorithme qui assure la transmission de la vidéo après sa compression et la réception de cette dernière.
- Implémenter cet algorithme pour arriver à réaliser une transmission vidéo en temps réel d'une manière fiable et pour des distances appréciables.

- Tester et valider cette implémentation en utilisant des scènes réelles pour des durées considérables.

La contribution de cette thèse est dans le fait que cet algorithme, testé et validé par des expériences, peut être utilisé dans des applications futures plus complexes.

Le plan de la thèse est décliné en quatre chapitres :

- Chapitre1 présente le matériel nécessaire pour la réalisation de notre travail.
- Chapitre2 explique la transmission réseau ainsi que quelques protocoles utilisés dans la pratique, il se termine par une présentation de la méthode de programmation dite en sockets.
- Chapitre3 évoque les concepts élémentaires de l'image et de la vidéo dans une chaîne de communication classique suivie par les principes de la compression d'images.
- Chapitre4 s'occupe de la partie réalisation ou implémentation de notre programmation où les résultats des différentes expériences et essais sont présentés, il les discute et analyse en détail avant d'émettre des conclusions.
- Ce travail est terminé par une conclusion générale et des perspectives.

CHAPITRE 1 : HARDWARE

1.1 Introduction

Le hardware désigne l'ensemble de l'équipement matériel, mécanique, magnétique, électrique et électronique, qui entre dans la conception d'un ordinateur, ou des machines de traitement de l'information en général.

1.2 Cartes d'acquisition vidéo

Plusieurs types de cartes d'acquisition vidéo sont utilisés. Le type de carte requis dépendra des choix de configuration que vous aurez effectués. Les cartes vidéo possèdent six caractéristiques principales :

- ✓ Types d'entrée/sortie vidéo analogique pris en charge
- ✓ Types d'entrée/sortie vidéo numérique pris en charge
- ✓ Types de compression pris en charge
- ✓ Types de traitements spéciaux pris en charge
- ✓ Types de logiciels inclus
- ✓ Types d'audio pris en charge

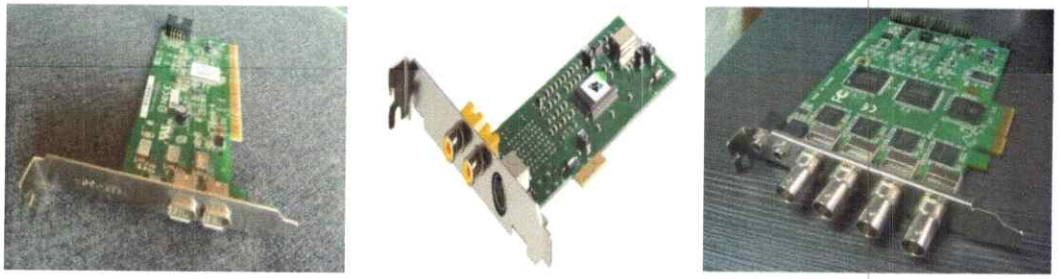

Figure 1.1 Différentes cartes d'acquisition vidéo

1.2.1 Types de cartes

(a) Carte d'interface IEEE 1394

La carte d'interface IEEE 1394 est utilisée pour doter d'une interface IEEE 1394 les ordinateurs qui n'en disposent pas en standard. Cette carte ne prend en charge aucune fonction d'E/S vidéo analogique, de compression, ni aucun traitement spécifique. En fait, elle n'effectue pas vraiment d'« acquisition » vidéo, elle permet simplement de «transférer» de la vidéo numérique sur votre ordinateur [1].

L'interface IEEE 1394 peut aussi être utilisée pour assurer un grand nombre d'autres connexions, à des disques durs, à des scanners et à des réseaux, par exemple [18] .

Bien que la vidéo numérique transmise par la carte IEEE 1394 soit compressée, la carte elle-même n'effectue aucune opération de compression ou de décompression.

Ces tâches sont gérées par la caméra et l'ordinateur. Si vous achetez une carte IEEE 1394, vérifiez qu'elle comporte les pilotes aptes à prendre en charge le logiciel NLE que vous utilisez, tel qu'Adobe Première.

(b) **Cartes d'acquisition analogique**

Les cartes d'acquisition analogiques assurent les tâches de conversion de vidéo du format analogique vers le numérique et inversement. Il existe plusieurs sources de vidéos analogiques, telles que les cassettes VHS et Beta-SP, les caméras Hi-8, etc.

La vidéo de type Composite, issue d'une platine VHS par exemple, est la moins coûteuse d'entre elles. La plus onéreuse est la vidéo de type Composante, issue d'une platine Beta-SP.

En plus du type d'entrée analogique pris en charge, vous devez également prêter attention au type de compression utilisé. Pendant de nombreuses années, le format de compression MJPEG a fait office de standard pour la vidéo, sur le marché grand public comme sur le marché professionnel.

Dernièrement, des formats plus récents tels que DV et MPEG-2 ont acquis un certain niveau de popularité [1].

Parmi les cartes les plus récentes, certaines vous permettent d'effectuer facilement des conversions d'un format à un autre.

Il est ainsi possible de monter une vidéo au format DV puis de la transcoder en MPEG-2 en vue de la distribution.

(c) **Cartes d'acquisition temps réel**

Si vous vouliez intégrer un effet spécial comme une transition, il vous fallait attendre que l'application bureautique calcule l'effet, alors que dans l'application haut de gamme, celui-ci était créé instantanément au moyen d'un matériel spécifique intégré.

Parfois le calcul de l'effet (appelé rendu) dans l'application bureautique demandait plusieurs minutes, voire des heures, ce qui ralentissait considérablement la production.

Ce fossé de productivité entre les applications haut de gamme et les applications bureautiques s'est désormais comblé avec l'introduction de nouvelles cartes vidéo, lesquelles fonctionnent avec Adobe Première et intègrent des processeurs spécialisés aptes à traiter le nombre énorme de calculs requis par les effets vidéo.

Chaque image de vidéo contient environ 1 Mo de données, et ces images volumineuses parviennent à un rythme voisin de 30 images par seconde.

Un effet, tel qu'une transition, résulte de la fusion mathématique de deux flux vidéo en vue de créer un élément vidéo.

Ceci signifie que même l'effet le plus simple nécessite environ 60 millions de calculs par seconde pour créer le nouvel élément de vidéo. Quels types d'effets peut-on réaliser en temps réel ? Ceci dépend de la carte utilisée.

Dans l'ensemble, la plupart des cartes fonctionnant en temps réel savent gérer les types d'effets les plus répandus, tels que les transitions et les titres.

Certaines cartes peuvent gérer une palette d'effets plus étendue, avec notamment la possibilité de « faire voler » la vidéo en 3D en temps réel.

Les cartes vidéo temps réel diffèrent également par le type de compression (MJPEG, DV, MPEG-2, non compressé, etc.) ainsi que par leurs options d'entrée/sortie (composite, composante, 1394, SDI, etc.) [1]

1.3 Les supports de transmission

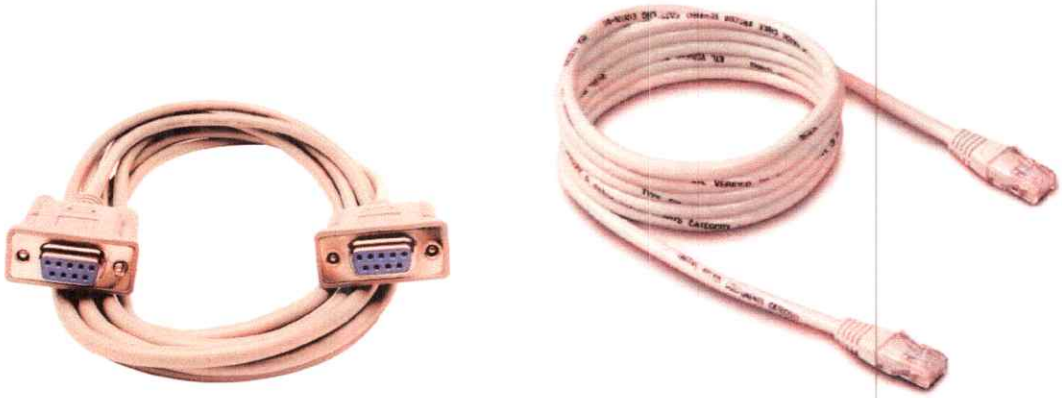

Figure 1.2 Câble réseau RS232 vs Câble réseau RJ45 RS232

Deux types de support de transmission série ou parallèle, leur vitesse de transmission varie considérablement, de quelque kilo de bite par second pour le câble DB9 (utilisant le Protocole RS232 jusqu'a quelques méga bite par second pour le câble réseau RJ5.

Pour les applications de vidéo où une grande quantité de donnée doit être acheminée en temps réel on opte pour le support de transmission réseau Rj45.

1.4 Les cameras USB

La camera PC, également appelée webcam, est un périphérique vidéo connecte à la carte graphique qui permet de communiquer par vidéo sur internet, notamment en vidéo-conférence.

Une caméra PC peut également permettre de prendre des photos numériques, de réaliser de petits films vidéo et peut même être utilisée comme vidéo de surveillance en configurant la détection de mouvement.

La plupart des cameras PC que l'on trouve sur le marché se branchent en USB.

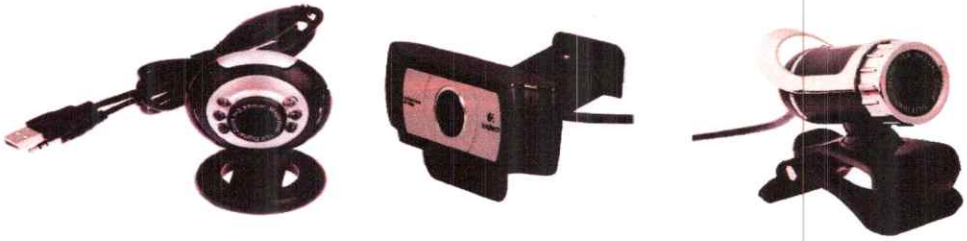

Figure 1.3 Modèles de camera USB

Il existe plusieurs types de camera de standard USB sur le marché toute de bonnes performances.

CHAPITRE 2 : TRANSMISSION RESEAU

2.1 Introduction

Toute entreprise possède aujourd'hui un ou plusieurs systèmes de télécommunication qui véhiculent les différentes informations nécessaires à sa vie et à son développement. Ces systèmes sont organisés en réseaux, qu'on peut définir comme des ensembles d'équipements et de supports de transmission dont une des fonctions est de permettre le transfert d'informations. Nous sommes entrés dans l'ère de la communication où le volume et la diversité de ces informations se font de plus en plus grands. Dans les années 80, cette diversité conduisait à l'adoption de solutions de communication distinctes et différentes suivant la nature des informations à transmettre : réseau téléphonique pour la transmission de la voix, réseau spécialisé dans la transmission de données sur longue distance comme Transpac en France ou sur courte distance comme les réseaux locaux d'entreprise, réseau hertzien ou câblé pour la télévision. Aujourd'hui les progrès de l'informatique rendent possible le traitement d'informations de natures différentes sur le même ordinateur : séquences vidéos et sonores, présentation de documents. C'est le domaine du multimédia.

De plus, les progrès des techniques de transmission permettent de transférer sur un même support (une fibre optique par exemple) ces informations variées. Les frontières entre les différents réseaux tendent à s'estomper. Par exemple, le réseau mondial Internet, initialement destiné exclusivement à la transmission de données, transmet des communications téléphoniques sûres. Les solutions à un besoin de communication sont multiples et les progrès techniques rendent foisonnant le domaine des réseaux.

2.2 Le modèle en couches OSI

Le modèle OSI définit l'interface réseau commune à tous les terminaux à l'aide d'une architecture en pile composée de sept couches. Le synoptique du modèle OSI est représenté sur la Figure 2.1 Le modèle OSI ne décrit pas les protocoles et les systèmes à utiliser au niveau de chaque couche, mais il définit plutôt les fonctions et le rôle de chacune de ces couches.

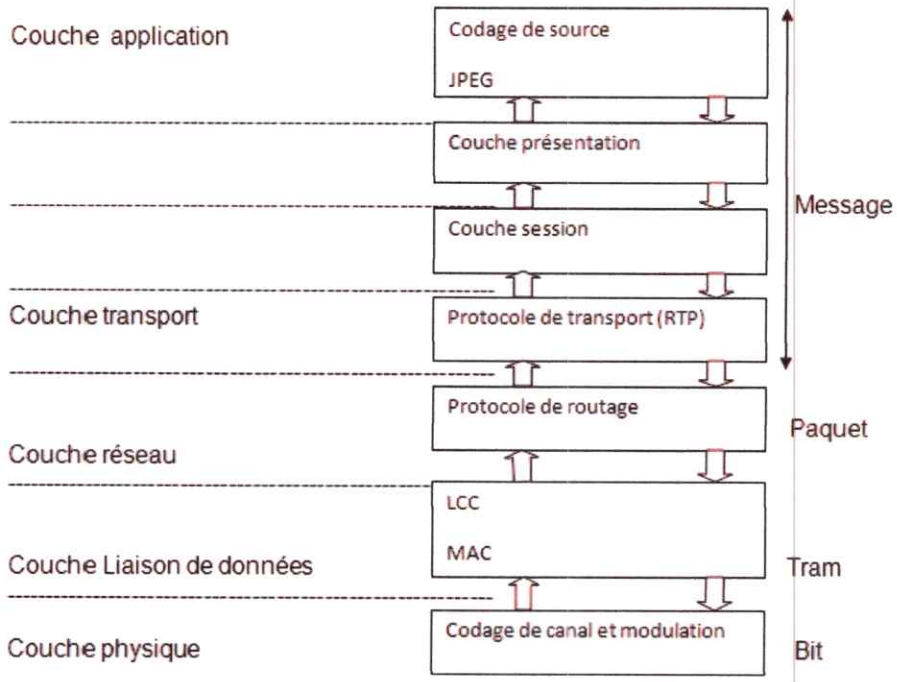

Figure 2.1 Structure en couches du modèle OSI

- ✓ La couche application offre à l'utilisateur l'accès aux différentes applications du réseau.

  C'est au niveau de cette couche que l'information est traitée avant son acheminement par les autres couches vers la destination. La couche présentation traduit l'information à transmettre à travers le réseau en un langage commun. L'établissement des communications de bout en bout, c'est-à-dire entre une source et une destination, est géré par la couche session.
- ✓ La couche transport assure la fiabilité et la régulation des échanges de paquets entre la source et la destination. Les protocoles utilisés pour les réseaux ad hoc sont similaires aux protocoles adoptés dans les réseaux filaires, à savoir les protocoles UDP, TCP/IP et le protocole RTP pour les applications temps réel.
- ✓ La couche réseau oriente et achemine les paquets de la source jusqu'à la destination. Les protocoles de routage couramment utilisés dans les réseaux filaires et sans fil avec une infrastructure fixe (cellulaires) ne peuvent pas être directement appliqués aux réseaux ad hoc. Les protocoles adoptés dans les réseaux ad hoc gèrent le routage, à travers des messages de contrôle, équitablement à partir des différents terminaux. Quoi que, il existe des

protocoles de routage hiérarchiques, où le routage est géré par des terminaux chefs (en anglais head) élus dans le réseau suivant des critères spécifiques, tels que la zone de couverture et l'énergie des terminaux.

- ✓ La couche liaison, composée de deux sous couches de données : LLC et MAC, assure l'échange et la fiabilité de réception des trames entre deux terminaux voisins (émetteur-récepteur) communiquant à travers un lien direct. Cette couche envoie des messages d'acquittements (ARQ) à la réception de chaque trame. Dans le cas où l'émetteur ne reçoit pas l'acquittement d'une trame émise, il retransmet cette trame jusqu'à ce qu'elle soit reçue correctement, ou que le nombre de retransmissions autorisé soit atteint.

Enfin, la couche physique à travers des techniques de traitement du signal permet d'adapter l'information à transmettre au canal radio.

Il est important de préciser que le modèle en couches OSI traite l'information à transmettre par les différentes couches d'une manière indépendante. L'objectif du modèle OSI classique est d'assurer un transfert sans perte de paquets sans pour autant garantir une qualité de service à une application donnée. Ce mode de fonctionnement autorise alors la mise en place des stratégies de transmission entre deux couches non voisines dans le modèle OSI [3].

2.3 Le protocole UDP

Le protocole UDP est basé en couche 4 (Transport), c'est un protocole de transport sans connexion qui fonctionne au-dessus du protocole de réseau IP (couche 3 du modèle OSI).

Le rôle de ce protocole est de permettre la transmission de données de manière très simple entre deux entités, chacune étant définie par une adresse IP et un numéro de port. Contrairement au protocole TCP, il fonctionne en mode non-connecté : il n'existe pas de procédure de connexion préalable à l'envoi des données, et il n'y a pas de garantie de bonne livraison d'un datagramme à sa destination, l'ordre d'arrivée des datagrammes peut différer de l'ordre d'envoi. Il est également possible que des datagrammes soient dupliqués. Les fonctions assurant la retransmission et le ré-ordonnancement doivent être assurées par les protocoles de la couche supérieure si elles sont souhaitées. L'intégrité des données est assurée par une somme de contrôle, l'utilisation de celle-ci est cependant facultative en IPv4mais obligatoire avec IPv6. Si un hôte n'a pas calculé la somme de contrôle d'un

paquet émis, la valeur de celle-ci est fixée à zéro. La somme de contrôle inclut un pseudo en-tête qui inclut les adresses IP source et destination.

2.3.1 Caractéristique d'UDP

- ✓ Il Possède un mécanisme permettant d'identifier les processus d'application à l'aide de numéros de port UDP.
- ✓ UDP est orienté datagrammes (sans connexion), ce qui évite les problèmes liés à l'ouverture, au maintien et à la fermeture des connexions.
- ✓ il est efficace pour les applications en diffusion/multidiffusion. Les applications satisfaisant à un modèle du type (interrogation-réponse) peuvent également utiliser UDP. La réponse peut être utilisée comme étant un accusé de réception positif à l'interrogation .si une réponse n'est pas reçue dans un certain intervalle de temps, l'application envoie simplement une autre interrogation.
- ✓ UDP ne séquence pas les données. La remise conforme des données n'est pas garantie.
- ✓ UDP peut éventuellement vérifier l'intégrité des données avec un total de contrôle.
- ✓ UDP est plus rapide, plus simple et plus efficace que TCP mais il est moins robuste.
- ✓ il permet une transmission sans connexion, mais aussi sans sécurité [19].

2.3.2 Structure du datagramme UDP

Le paquet UDP est encapsulé dans un paquet IP. Il comporte un en-tête suivi des données proprement dites à transporter.

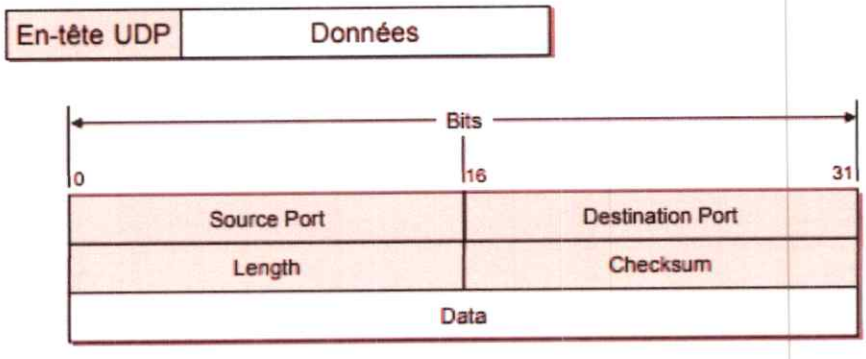

Figure 2.2 Structure du datagramme UDP

Format général :

a) Un en-tête de taille fixe de 8 octets

b) Un champ de données de taille variable.

c) Le champ source port occupe 16 bits. Il indique :

- ✓ le numéro de port du processus émetteur
- ✓ le numéro de port où on peut adresser les réponses lorsque l'on ne dispose d'aucun autre renseignement
- ✓ si sa valeur est 0, cela signifie qu'aucun numéro de port n'est attribué

d) Le champ destination port identifie le processus correspondant à l'adresse IP de destination auquel on envoie les données UDP. UDP effectue le démultiplexage des données à l'aide de numéros de port. Lorsqu'UDP reçoit un datagramme sans numéro de port, il génère un message d'erreur ICMP indiquant qu'il est impossible de contacter le port et il rejette le datagramme.

e) Le champ lenght contient la longueur du paquet UDP en octets (en tête+données) la valeur minimale est 8 et correspond à un paquet où le champ de données est vide.

UDP n'utilise pas d'accusé de réception, donc le transfert de données est rapide. C'est pourquoi UDP est utilisé quand on à des flux (streaming) audio ou vidéo à transporter, exemple vidéo de surveillance.

2.4 Le protocole TCP

Dans le Protocol TCP dont les données encapsulées dans un en-tête TCP sont des " paquets TCP ".

Les applications utilisent TCP pour garantir une transmission des données fiable. TCP est un protocole fiable, orienté connexion, à flots d'octets.

2.4.1 Caractéristique du TCP

TCP est bien plus compliqué qu'UDP, il apporte en contrepartie des services beaucoup plus élaborés. Cinq points principaux caractérisent ce protocole :

a) TCP contient un mécanisme pour assurer le bon acheminement des données. Cette possibilité est absolument indispensable dès lors que les applications doivent transmettre de gros volumes de données et de façon fiable.

Il faut préciser que les paquets de données sont acquittés de bout en bout et non de point en point.

b) Le protocole TCP permet l'établissement d'un circuit virtuel entre les deux points qui échangent de l'information. On dit aussi que TCP fonctionne en mode connecté (par opposition à UDP qui est en mode non connecté ou encore mode datagramme).

- ✓ Avant le transfert les deux applications se mettent en relation avec leurs OS respectifs, les informent de leurs désirs d'établir ou de recevoir une communication.
- ✓ Pratiquement, l'une des deux applications doit effectuer un appel que l'autre doit accepter.
- ✓ Les protocoles des deux OS communiquent alors en s'envoyant des messages au travers du réseau pour vérifier que le transfert est possible (autorisé) et que les deux applications sont prêtes pour leurs rôles.
- ✓ Une fois ces préliminaires établis, les modules de protocole informent les applications respectives que la connexion est établie et que le transfert peut débuter.
- ✓ Durant le transfert, le dialogue entre les protocoles continue, pour vérifier le bon acheminement des données.

c) TCP a la capacité de mémoriser des données :

- ✓ Aux deux extrémités du circuit virtuel, les applications s'envoient des volumes de données absolument quelconques, allant de 0 octet à des centaines (ou plus) de Mo.
- ✓ A la réception, le protocole délivre les octets exactement comme ils ont été envoyés.
- ✓ Le protocole est libre de fragmenter le flux de données en paquets de tailles adaptées aux réseaux traversés. Il lui incombe cependant d'effectuer le réassemblage et donc de stocker temporairement les fragments avant de les présenter dans le bon ordre à l'application.

d) TCP est indépendant vis à vis des données transportées, c'est un flux d'octets non structuré sur lequel il n'agit pas.

e) TCP simule une connexion en " full duplex ". Pour chacune des deux applications en connexion par un circuit virtuel, l'opération qui consiste à lire des données peut s'effectuer indépendamment de celle qui consiste à en

écrire. Le protocole autorise la clôture du flot dans une direction tandis que l'autre continue à être active. Le circuit virtuel est rompu quand les deux parties ont clos le flux [19].

2.4.2 Structure du TCP

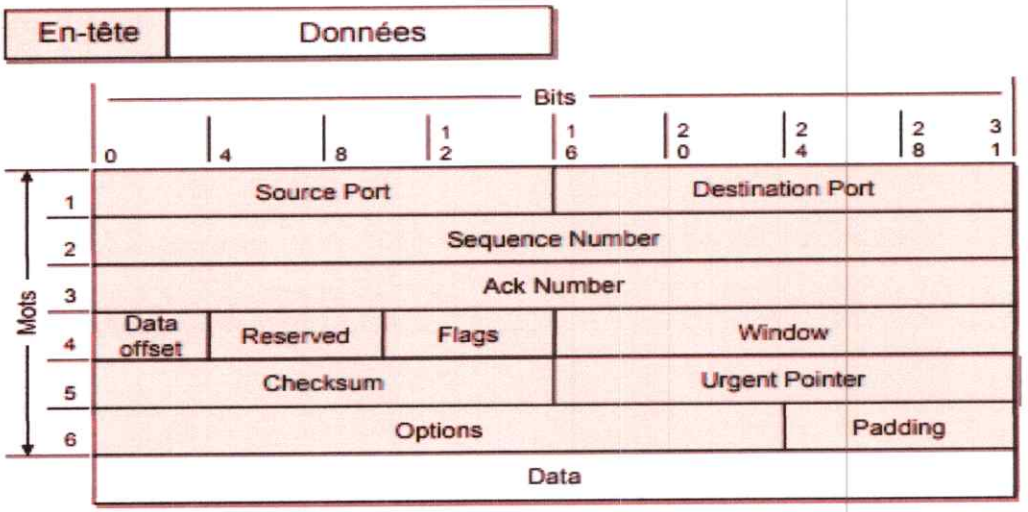

Figure 2.3 Structure du datagramme TCP

Les différentes informations du champ en-tête sont les suivantes :

- ✓ Port source(16) : port source de l'application sur la machine source
- ✓ Port destination(16) : port destination de l'application sur la machine destination
- ✓ Numéro de séquence(32) : numéro du premier octet du paquet dans l'ensemble du flux de données transmis, 0 pour le premier paquet
- ✓ Numéro d'acquittement(32) numéro de séquence du prochain octet attendu
- ✓ Taille de l'en-tête : longueur de l'en-tête en mots de 32 bits (les options font partie de l'en-tête)
- ✓ Réservé : réservé pour un usage futur
- ✓ Fenêtre : taille de fenêtre demandée, c'est-à-dire le nombre d'octets que le récepteur souhaite recevoir sans accusé de réception
- ✓ Somme de contrôle : somme de contrôle calculée sur l'ensemble de l'en-tête TCP et des données, mais aussi sur un pseudo en-tête (extrait de l'en-tête IP)
- ✓ Pointeur de données urgentes : position relative des dernières données urgentes
- ✓ Options : facultative

- ✓ Remplissage : zéros ajoutés pour aligner les champs suivants du paquet sur 32 bits, si nécessaire
- ✓ Données : séquences d'octets transmis par l'application [4]

2.5 La différence entre les protocoles TCP et UDP

TCP Segment Header Format

Bit #	0	7	8	15	16	23	24	31
0	Source Port				Destination Port			
32	Sequence Number							
64	Acknowledgment Number							
96	Data Offset	Res	Flags		Window Size			
128	Header and Data Checksum				Urgent Pointer			
160...	Options							

UDP Datagram Header Format

Bit #	0	7	8	15	16	23	24	31
0	Source Port				Destination Port			
32	Length				Header and Data Checksum			

Figure 2.4 En-tête TCP et UDP

Caracteristiques	UDP	TCP
Démarrage	Aucun établissement de connexion	Etablissement de la connexion en 3temps
Fiabilité	Aucun acquittement des messages, aucune assurance de livraison	Acquittement de messages pour assurer la transmission, l'acquittement sélectif est optionnel en TCP
Ordre de la livraison	Aucun mécanisme mis en œuvre pour assurer l'ordre de messages	Numérotation des messages pour assurer l'ordre
Contrôle de congestion	Aucun mécanisme mis en œuvre	Mis en œuvre des mécanismes de contrôle de congestion

Caracteristiques	UDP	TCP
Procédures de fermeture	Aucune (protocole non orienté connexion)	Fermeture de la connexion spécifiée
SYN attack	Insensible (aucune connexion effectuée)	C'est un de grands soucis de TCP
Blocage en-tête de ligne	Insensible	Partiellement résolue en ouvrant plusieurs connexions
Multi accès dynamique	Pas supporté	Pas supporté

Tableau 2.1 La différence entre le protocole TCP et UDP

2.6 Types d'adressages

2.6.1 Unicast

L'adressage unicast est le plus utilisé et le plus simple. Les ordinateurs possédant chacun une adresse IP, on peut envoyer les trames en spécifiant l'adresse IP de l'ordinateur à qui on veut envoyer les informations. Les éléments actifs et passifs du réseau (répéteurs, commutateurs, routeurs...) dirigent l'information dans la bonne direction pour que les trames arrivent au bon endroit. Seule la machine ayant l'adresse contenue dans la trame regarde et traite l'information [5].

2.6.2 Broadcast

Le broadcast (diffusion) consiste à envoyer l'information à tous les ordinateurs du réseau ou sous-réseau où on se situe (domaine de diffusion). Au lieu d'envoyer les informations en unicast vers l'adresse IP de chaque machine, on envoie la trame à tous les ordinateurs du sous-réseau en utilisant l'adresse de broadcast, Cette adresse est réservée à cet usage. Chacun des ordinateurs du sous-réseau regarde et traite la trame comme si elle leur était personnellement adressée. Les trames de broadcast ont une caractéristique particulière : c'est de ne pas pouvoir passer les routeurs puisqu'il s'adresse uniquement à tous les ordinateurs d'un même sous-réseau [5].

2.6.3 Multicast

Plutôt que d'envoyer les fichiers du serveur vers chacune des machines clientes (unicast) on peut envoyer l'information qu'une seule fois et chaque ordinateur client

la récupère. En effet, dans un réseau Ethernet par exemple, toutes les trames qui circulent passent par tous les ordinateurs. C'est le principe du multicast : on envoie l'information à une adresse et tous les clients écoutent cette adresse. En Ipv4, en plus des classes d'adresses A, B et C, existe une quatrième classe (D) réservée aux adresses multicast. Les adresses IPv4 entre 224.0.0.0 et 239.255.255.255 appartiennent à cette classe. Les 4 bits les plus significatifs de l'adresse IP autorisent des valeurs comprises entre 224 et 239. Les autres 28 bits, moins significatifs, sont réservés à l'identificateur du groupe multicast, comme montré dans la figure ci-dessous :

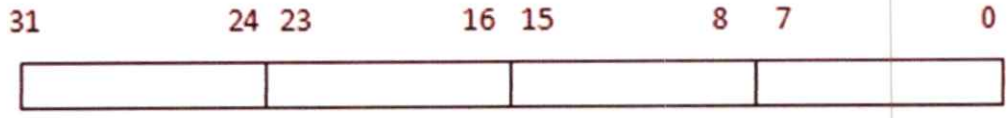

Figure 2.5 Identificateur du groupe multicast

Chaque client multicast s'enregistre donc avec une adresse IP multicast de classe D (entre 224.0.0.0 et 239.255.255.255 sauf 224.0.0.0 non utilisée et 224.0.0.1 qui correspond au "broadcast du multicast"). C'est sur cette adresse que les informations vont être envoyées. Les clients écoutent ce qui arrive sur cette adresse et suivent la procédure décrite par le protocole multicast implémenté [5].

2.7 Programmation en sockets

2.7.1 Définition de socket

La notion de sockets a été introduite dans les distributions de Berkeley (un fameux système de type UNIX, dont beaucoup de distributions actuelles utilisent des morceaux de code), c'est la raison pour laquelle on parle parfois de sockets BSD.

Il s'agit d'un modèle permettant la communication inter processus (IPC) afin de permettre à divers processus de communiquer aussi bien sur une même machine qu'à travers un réseau TCP/IP. La communication par socket est souvent comparée aux communications humaines.

Un Socket désigne l'extrémité d'un canal de communication bidirectionnel. Un socket est donc une interface entre les applications des utilisateurs et la couche transport.

Modèle des sockets	Modèle OSI
Application utilisant les sockets	Application Présentation Session
UDP/TCP	Transport
IP/ARP	Réseau
Ethernet, x25…	Liaison Physique

Tableau 2.2 Position des sockets dans le modèle OSI

Les sockets peuvent être utilisés en deux modes de communication :

– **Le mode connecté** (comparable à une communication téléphonique), utilisant le protocole TCP. Dans ce mode de communication, une connexion durable est établie entre les deux processus, de telle façon que l'adresse de destination n'est pas nécessaire à chaque envoi de données.

– **Le mode non connecté** (analogue à une communication par courrier), utilisant le protocole UDP. Ce mode nécessite l'adresse de destination à chaque envoi, et aucun accusé de réception n'est donné. Pour faciliter leur utilisation les sockets sont conçus en conservant la sémantique des primitives d'E/S système exactement comme les fichiers (création, ouverture, lecture, écriture, fermeture). Pour communiquer des données entre deux sockets, dans un mode connecté, on doit faire une distinction entre celle utilisée par le programme demandeur de connexion (Client) et celle utilisée par le programme qui accepte les connexions (Serveur). – Un serveur est un programme qui attend les connexions à travers une Socket et qui gère ensuite toutes les connexions qui arrivent. – Un client est un programme qui se connecte à un serveur à travers un socket. Le schéma suivant montre les différentes étapes exécutées par un client et un serveur pour communiquer [6] :

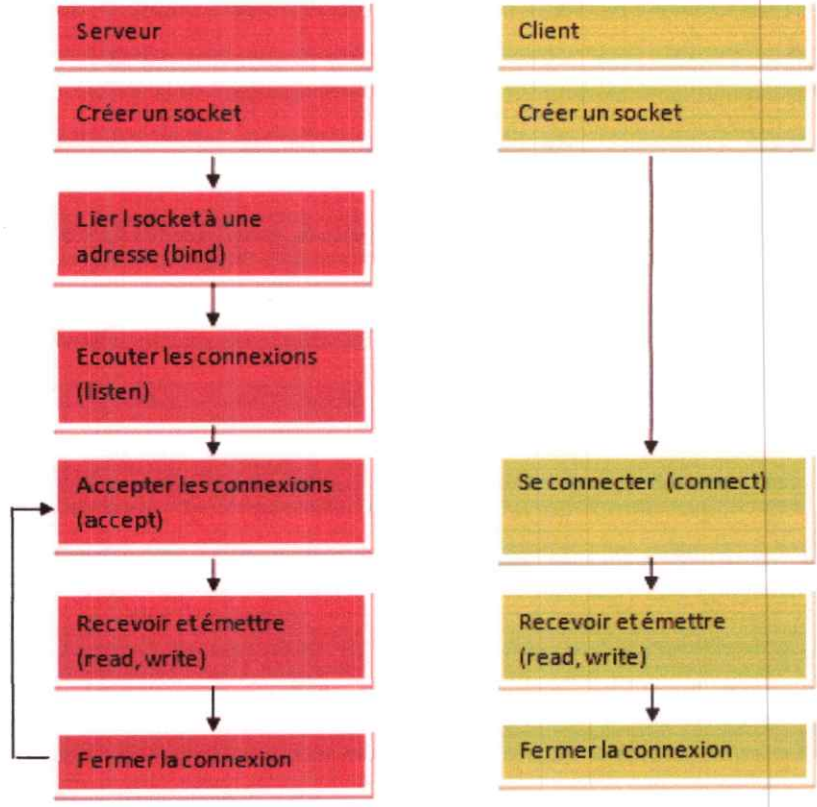

Figure 2.6 Modèle serveur/client mode connecté

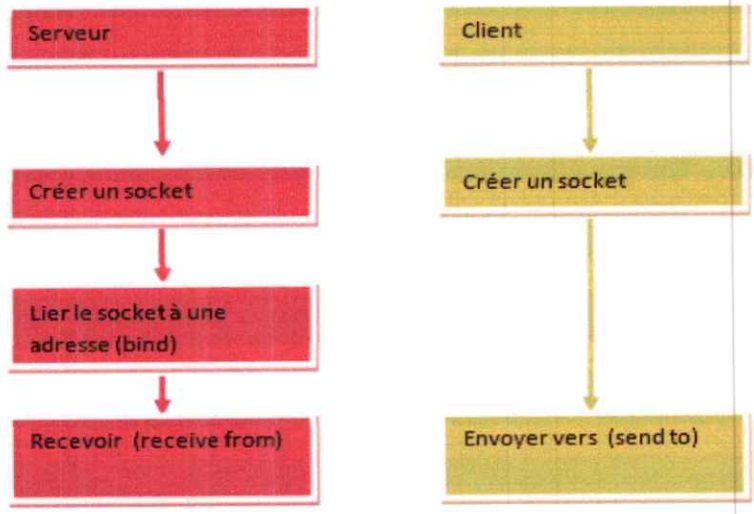

Figure 2.7 Modèle serveur/client mode non connecté

2.7.2 Utilisation des sockets

Les sockets sont généralement implémentés en langage C, et utilisent des fonctions et des structures disponibles dans la librairie <sys/socket.h>.

Pour recevoir des appels téléphoniques, il faut d'abord installer votre téléphone. Pour ce faire, il faut créer un socket pour écouter les connections, un procédé qui se passe en plusieurs étapes :

- ✓ **Création d'un socket ! Socket()**

Ce qui ressemble à se faire installer une ligne de téléphone par la compagnie des télécoms.

La fonction socket() nécessite de Spécifier le type de la socket. Les deux types les Plus connus sont SOCK_STREAM et SOCK_DGRAM.

SOCK_STREAM indique que les données seront transportées par le protocole TCP, alors que SOCK_DGRAM indique que les données seront transportées par le protocole UDP.

Nous ne nous intéresserons ici qu'aux sockets SOCK_STREAM, qui sont les plus courantes.

- ✓ **Lier la Socket à une adresse ! Bind()**

Après avoir créé un socket, il faut lui donner une adresse à écouter, de la même façon qu'on prend un numéro de téléphone pour pouvoir recevoir des appels, mais avant ça, on doit spécifier la famille d'adresse à utiliser. Dans le cas de TCP/IP, la famille d'adresse est "AF_INET" où chaque socket a une adresse unique composée de deux éléments : une adresse IP et un numéro de port.

– La première partie est l'adresse IP (comme 192.9.200.10), avec lequel on veut communiquer.

Malheureusement, les nombres sont difficiles à retenir, surtout quand on doit travailler avec beaucoup de nombres différents. L'interface la plus utilisée pour retrouver une adresse est la fonction gethostbyname(), qui prend le nom d'un ordinateur et vous renvoie son adresse IP. De même, il est possible de retrouver le nom d'un ordinateur quand on a son adresse IP en utilisant la fonction gethostbyaddr().

– La seconde partie est le numéro de port, qui autorise plusieurs conversations simultanées sur chaque ordinateur. Une application peut soit prendre un numéro de port réservé pour son type d'application, soit en demander un au hasard du système lorsqu'il lie une adresse à sa socket ; une application peut en utiliser plusieurs (un

serveur par exemple exploite un socket par client connecté). Les numéros de port sont uniques sur un système donné.

Une connexion est identifiée, donc, de façon unique par la donnée de deux couples, une adresse IP et un numéro de port, un pour le client et un autre pour le serveur. Il est important de noter qu'un dialogue client/serveur n'a pas forcément lieu via un réseau. Il est en effet possible de faire communiquer un client et un serveur sur une même machine, via ce qu'on appelle l'interface de loopback, représentée par convention par l'adresse IP 127.0.0.1.

Pour lier un socket à une adresse on utilise la fonction bind() (to bind veut dire "lier").

Une adresse de socket Internet est spécifiée en utilisant la structure sockaddr, qui contient les champs qui spécifient la famille d'adresse, l'adresse et sa longueur.

- ✓ **Ecouter les appels ! Listen()**

Les sockets de type SOCK_STREAM ont la capacité de mettre les requêtes de connexion en files d'attentes, ce qui ressemble au téléphone qui sonne en attendant que l'on réponde. Si c'est occupé, la connexion va attendre qu'on libère la ligne. La fonction listen() est utilisée pour donner le nombre maximum de requêtes en attente avant de refuser les connexions.

- ✓ **Accepter les appels ! Accept()**

Après avoir créé un socket pour recevoir des appels, il faut accepter les appels vers ce socket.

La fonction accept() est utilisée pour se faire. Appeler la fonction accept() est équivalent à prendre le combiné lorsque le téléphone sonne. Accept() renvoie un nouveau socket qui est connecté à celui qui appelle.

- ✓ **Appeler un socket ! Connect()**

Après avoir créé un socket, et après lui avoir donné une adresse, il faut utiliser la fonction connect() pour essayer de se connecter à une socket qui attend les appels ce qui est équivalent à composer numéro d'un téléphone.

- ✓ **Emettre et recevoir des données ! Send() &Recv()**

Maintenant qu'on a une connexion entre deux sockets, on peut envoyer des données entre elles. On utilise les fonctions send() and recv().

- ✓ **Fermer une connexion ! close socket()**

De la même façon qu'on raccroche après avoir eu quelqu'un au téléphone, il faut fermer la connexion entre les deux sockets. La fonction close socket() est utilisée pour

fermer chaque extrémité de la connexion. Si une des extrémité est fermée et qu'on essaye d'utiliser la fonction send() à l'autre, la fonction send() renverra une erreur. Un recv() qui attend quand la connexion à l'autre extrémité est fermé ne retournera aucun octet[7].

CHAPITRE 3 : COMPRESSION D'IMAGE

3.1 Introduction

Pour utiliser les images numériques, il faut comprimer les fichiers dans lesquels elles sont enregistrées. Les chercheurs ont imaginé de nombreuses méthodes de compression, que l'on classe en deux catégories : celles qui se contentent de comprimer les données sans les altérer, et celles qui les compactent en les modifiant. Les premières, dites non destructives, permettent de reconstituer, au bit près, le fichier dans l'état où il était avant la compression.

Pour réduire le volume global des images un moyen consiste en la compression des images avec le minimum de dégradation et le maximum d'efficacité possible. Les principaux critères d'évaluation de toute méthode de compression sont :

- La qualité de reconstitution de l'image
- Le taux de compression
- La rapidité du codeur et décodeur (codec).

3.2 Concepts élémentaires de l'image

3.2.1 Image numérique

C'est une matrice de X×Y pixels (picture element) correspondant à l'échantillonnage et la quantification d'un signal acquis avec une caméra. Chaque pixel est associé à un niveau de gris n ou des niveaux de composantes couleur codé sur N bits et qui représentent respectivement le niveau de luminosité ou de couleur de la zone correspondante dans la scène observée[8], comme le montre l'exemple suivant :

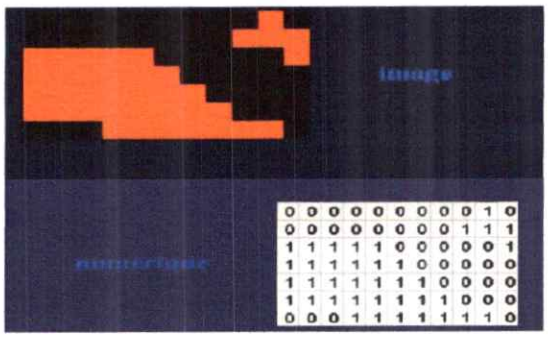

Figure 3.1 Détail d'une image binaire.

3.2.2 Les pixels d'une image

Une image numérique en niveaux de gris est un tableau de valeurs chaque case de ce tableau qui stocke une valeur se nomme un pixel En notant n le nombre de lignes et p le nombre de colonnes de l'image, on manipule ainsi un tableau de n×p pixels.

La figure ci-dessous montre une visualisation d'un tableau carré avec n=p=240, ce qui représente 240×240=57600 pixels. Les appareils photos numériques peuvent enregistrer des images beaucoup plus grandes, avec plusieurs millions de pixels [12].

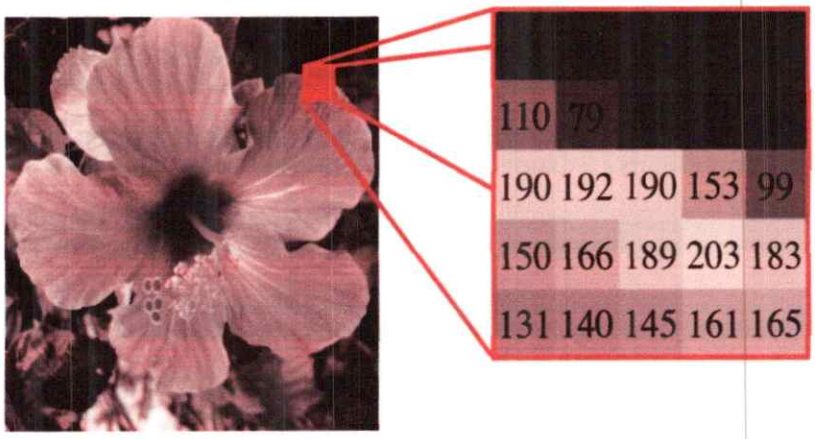

Figure 3.2 Une image en niveaux de gris et une sous image de taille 5×5.

3.2.3 Les types d'images utilisés

a) **Image binaire :**

Dans une image binaire les pixels sont représentés par deux états logiques '0' noir et '1' blanc, c'est un codage de l'image sur 1Bits.

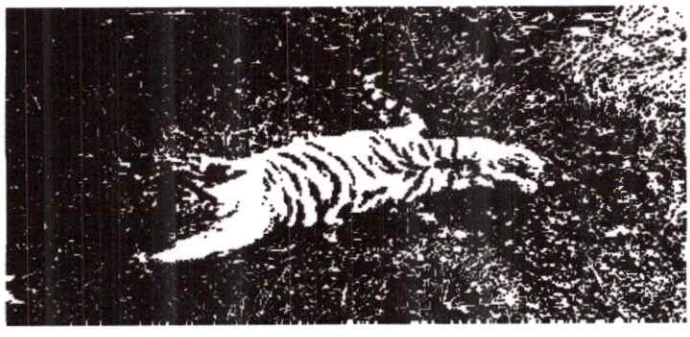

Figure 3.3 Image binaire

b) **Image à niveaux de gris :** (intensité ou luminance)

Chaque pixel est codé sur N bits, ce qui lui confère des valeurs entières comprises entre 0 (noir) et 2 N-1 (blanc).

Figure 3.4 Image a niveaux de gris

c) **Image couleur :**

Une image couleur correspond à la synthèse additive de 3 images, rouge, vert et bleu. Chaque pixel est donc codé sur 3× N bits.

La couleur d'un pixel est représentée par 3 composantes couleurs et donne naissance à un point dans un espace tridimensionnel [20].

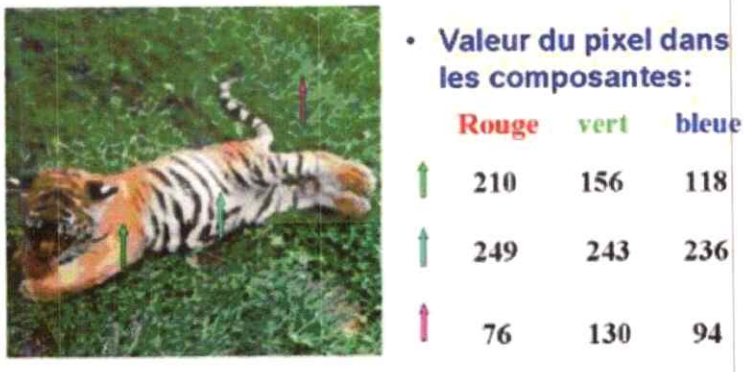

Figure 3.5 Image couleur

d) **Image indexée :**

La couleur ou l'intensité des niveaux de gris est déterminé par un index auquel correspond la couleur en question [9].

3.2.4 Stocker une image

Stocker de grandes images sur le disque dur d'un ordinateur prend beaucoup de place. On peut écrire toute valeur de pixel entre 0 et 255 sur 8 bits. Il y a en effet 256 valeurs possibles, et $256=2^8$. Pour stocker l'image complète, on a donc besoin den×p×8 bits. On utilise le plus souvent l'octet (8 bits) comme unité, par exemple une image de taille nécessite *57,6*ko (kilo octets) [8].

Afin de réduire la place de stockage d'une image, on peut réduire sa résolution, c'est-à-dire diminuer le nombre de pixels.

La façon la plus simple d'effectuer cette réduction consiste à supprimer des lignes et des colonnes dans l'image de départ.

La figure suivante montre ce que l'on obtient si l'on retient une ligne sur 4 et une colonne sur 4.les résultats obtenus en gardant de moins en moins de lignes et de colonnes. Bien entendu, la qualité de l'image se dégrade vite.

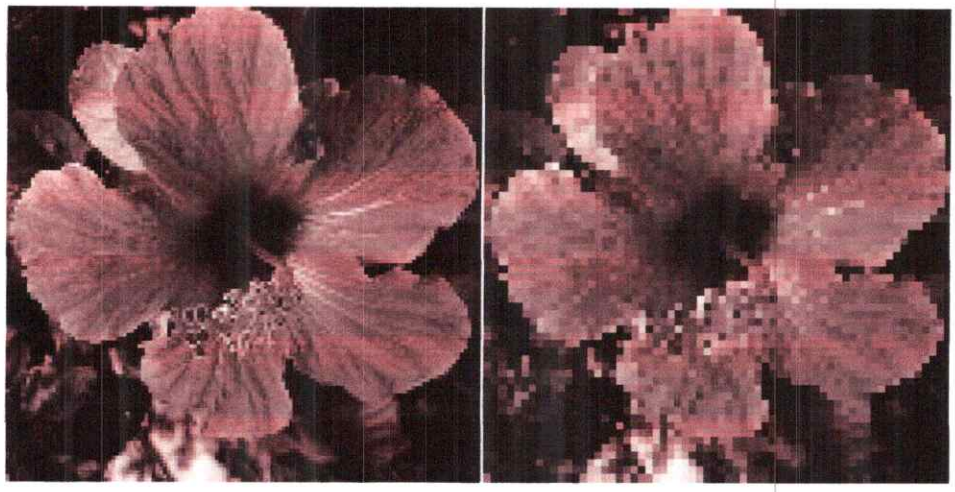

Figure 3.6 Une ligne/colonne sur 2Une ligne/colonne sur 4

3.2.5 Formats des fichiers images

Il n'existe pas de format idéal. Selon le cas, certains formats peuvent être préférables. Le terme bitmap employé ci-dessous fait référence aux formats de fichier qui stockent les pixels sous forme de tableau de points, avec une gestion des couleurs soit en couleur vraie soit par le biais d'une palette indexée.

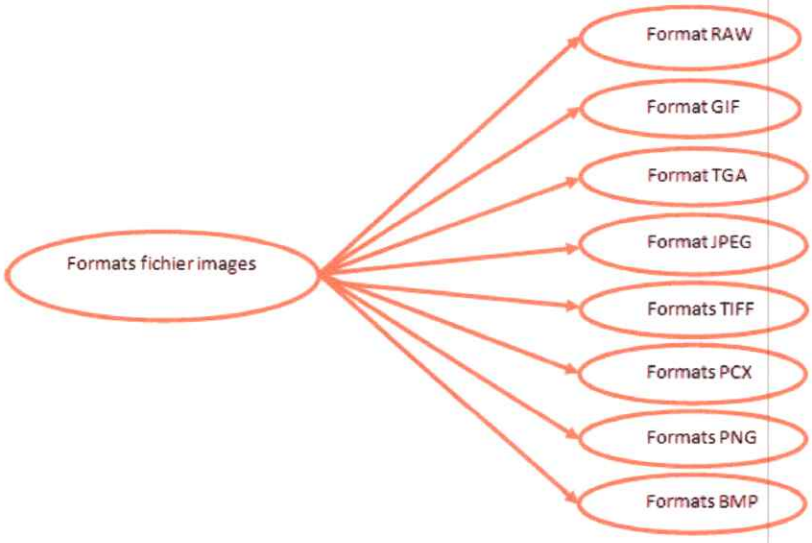

Figure 3.7 Formats des fichiers images

- ✓ Format RAW : Format de stockage des données brutes.
- ✓ Format GIF : Utilisé pour des images codées sur 8 bits au plus.
- ✓ Format TGA : utilisé pour stocker les images codées sur 24 ou 32 bits.
- ✓ Format JPEG : Compresses en utilisant le standard.
- ✓ Format TIFF : Stocker des images en noir et blanc, en couleurs réelles ainsi que des images indexées.
- ✓ Format PCX: pour des images codées sur 1,4, 8 ou 24 bit
- ✓ Format PNG : Stocker images en noir et blanc, en couleurs réelles et des images indexées, faisant usage d'une palette de 256couleurs.
- ✓ Format BMP : Format le plus simple, développé par Microsoft et IBM, il est très répandu sur les plates-formes Windows [14].

3.3 Compression d'image fixe

Un système de codage d'image fixe est généralement constitué de trois étapes principales : la transformée, la quantification et le codage entropique.

La figure illustre les différents blocs d'un système de compression d'image.

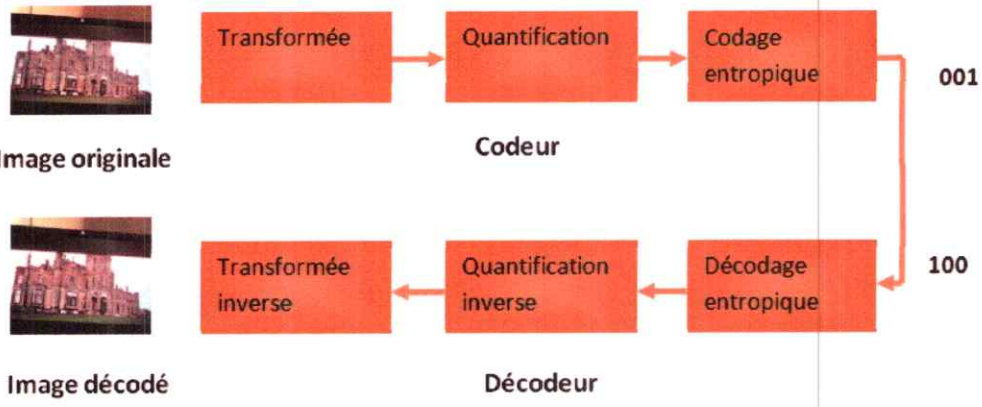

Figure 3.8 Schéma fonctionnel d'un codeur/décodeur de source pour l'image

a) Transformée

La transformée permet de dé-corréler les pixels d'une image et de compacter l'énergie dans un nombre restreint de coefficients. Le choix de la transformée est primordial lors de la conception d'un nouveau codeur d'image, car les spécificités de la transformée adoptée caractériseront les performances et les fonctionnalités du codeur [26].

Plusieurs transformées linéaires et réversibles sont utilisées dans le domaine de compression d'image, telles que la transformée en cosinus discrète (DCT).

b) Quantification

L'étape de quantification consiste à attribuer à l'ensemble des coefficients issus de la transformée des valeurs prises d'un ensemble dénombrable fini.

Contrairement à l'étape précédente, la quantification est une étape irréversible et introduit de la distorsion, ainsi cette étape est uniquement appliquée à un codage avec perte [16].

c) Codage entropique

Le codage entropique exploite la statistique d'apparition des coefficients de la transformée quantifiée ou pas pour réduire la longueur du code binaire.

Plus précisément, un codeur entropique consiste à accorder un nombre de bits différents (VLC) aux coefficients à coder de façon à ce que les coefficients d'occurrences fréquentes soient représentés par des mots binaires courts, et que des mots plus longs soient attribués aux coefficients moins fréquents[17].

3.4 Concepts élémentaires de la vidéo

3.4.1 Définition de la vidéo

Une vidéo est une succession d'images à une certaine cadence. L'œil humain a comme caractéristique d'être capable de distinguer environ 20 images par seconde. Ainsi, en affichant plus de 20 images par seconde, il est possible de tromper l'œil et de lui faire croire à une image animée [15].

3.4.2 Importance de la compression de la vidéo

La compression vidéo est une méthode de compression de données, qui consiste à réduire la quantité de données, en minimisant l'impact sur la qualité visuelle de la vidéo. Cela permet de réduire l'espace de stockage nécessaire et facilite aussi sa diffusion [10].

A elle seule, une image vidéo non compressée requiert près d'1 méga-octet (Mo) d'espace de stockage [13]. Vous pouvez faire ce calcul en multipliant la résolution horizontale (720 pixels) par la résolution verticale (486 pixels), puis en multipliant le tout par 3 octets, correspondant aux valeurs des couleurs RVB. Au taux standard de 29,97 images par seconde actuellement en vigueur pour les vidéos, il faudrait donc disposer de 30 Mo pour stocker chaque seconde de vidéo non compressée ! La minute, elle, nécessiterait 1,5 giga-octet (Go) ! Pour visionner et manipuler de la vidéo non compressée, il vous faudrait disposer d'un disque extrêmement rapide et coûteux, capable de fournir des volumes de données au processeur de votre ordinateur avec une vitesse suffisante.

L'objectif de la compression est de réduire le volume de données tout en conservant une bonne qualité d'image.

Le taux de compression appliqué dépend de l'usage que l'on fait de la vidéo. Le format DV compresse les données dans un rapport de 5 à 1 (autrement dit, il compresse la vidéo à un cinquième de sa taille d'origine).

Pour les vidéos accessibles sur Internet, le taux de compression utilisé peut aller jusqu'à un rapport de 50 à 1, voire davantage [11].

Remarque : Il faudrait plus d'1,5 Go (giga-octets) pour enregistrer une minute de vidéo non compressée.

3.4.3 Compression selon la norme JPEG

JPEG est Le nom du comité qui a créé la norme JPEG, La norme JPEG définit les étapes standards Pour compresser une image dans un flux d'octets et la décompresser à nouveau pour régénérer l'image originale [29].

JPEG peut être ajusté pour le taux de compression et la qualité de l'image selon l'exigence de l'utilisateur.

L'algorithme de compression JPEG fonctionne mieux avec des photographies et des tableaux de scènes réalistes avec une variation uniforme de la couleur et du ton.

Pour l'application Web, JPEG est très populaire en raison de sa capacité à réduire la bande passante pour le transfert de l'image sur le réseau [27].

JPEG est le format le plus commun utilisé pour l'ordinateur. JPEG n'est pas aussi bon pour le fichier qui subit plusieurs modifications car la qualité d'image est perdue chaque fois que l'image est compressée et décompressée avec la méthode de compression JPEG [21].

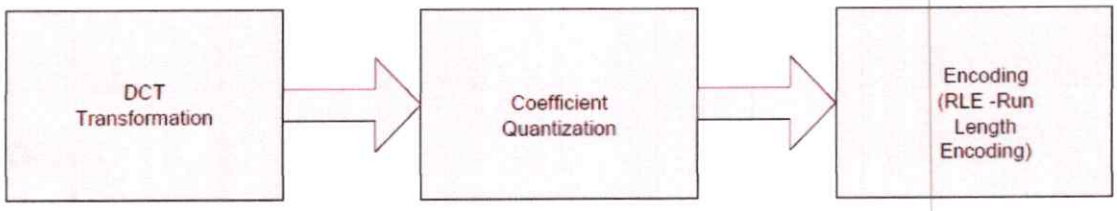

Figure 3.9 Etapes de compression JPEG

Trois étapes ci-dessus sont des étapes de compression puissantes. La qualité finale de l'image de sortie dépend du contenu de l'image.

Mais généralement, JPEG utilise ces étapes de compression et atteint un taux de compression élevé sans une dégradation notable de la qualité.

L'étape de codage finale a de nombreuses variantes avec l'utilisation d'un algorithme de codage différent comme RLE, Huffman Encoding, etc.

a) **Compression :**

La compression JPEG peut être divisée dans les petites étapes suivantes :

- ✓ Transformez l'image en espace de couleur approprié.
- ✓ L'échantillon descendant du composant Cb et Cr comme les yeux humains ne peut pas voir considérablement dans la luminosité de l'image.
- ✓ Divisez l'image en petits blocs de 8x8 pixels pour un traitement ultérieur.
- ✓ Appliquer une transformée de cosinus discrète (DCT) sur chaque bloc individuel de 8x8 pixels.
- ✓ Quantifier chaque bloc 8x8 avec la fonction de pondération optimisée pour les yeux humains.
- ✓ Réorganisez le coefficient de chaque bloc 8x8 dans l'ordre "Zig-Zag" pour unencodage supplémentaire.
- ✓ Encoder le coefficient résultant en utilisant l'une des méthodes de codage selon les exigences de qualité (RLE et Huffman)

Les étapes inverses seront appliquées pour le processus de la décompression.

La représentation graphique montre les étapes de compression JPEG [23].

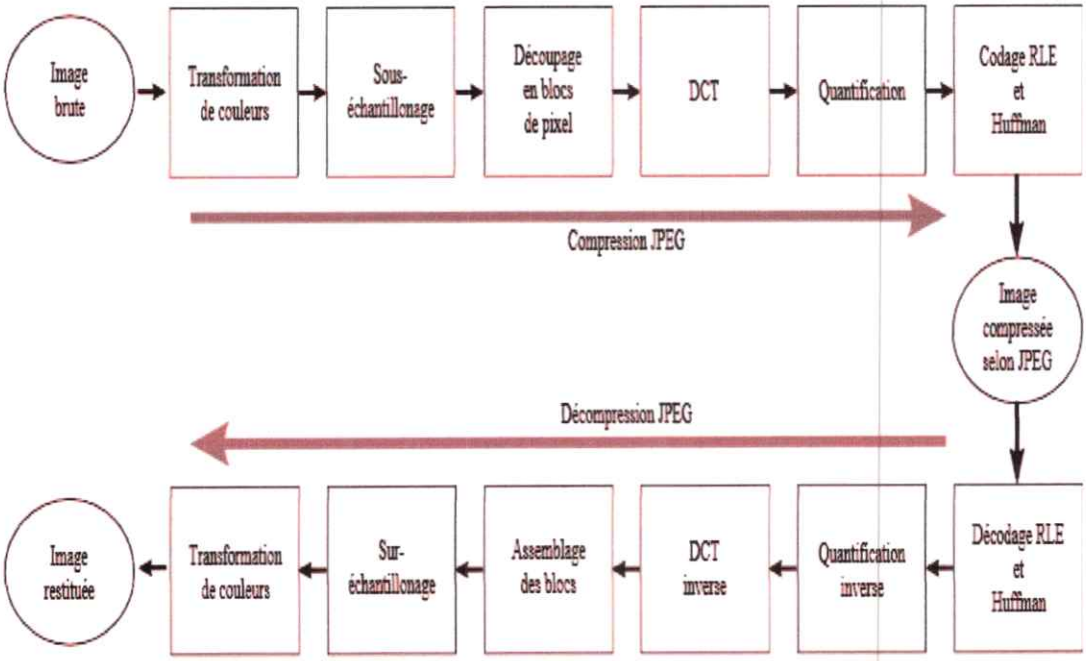

Figure 3.10 Etapes de compression JPEG.

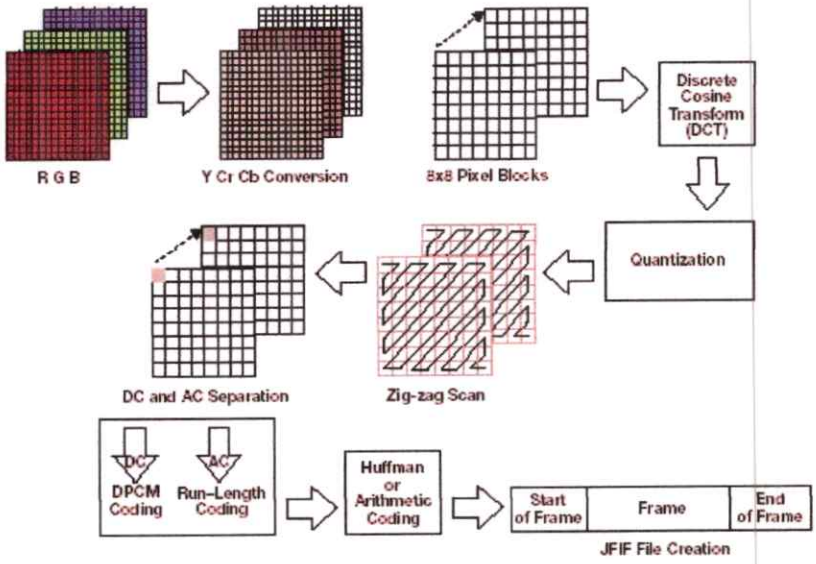

Figure 3.11 Schéma détaillé de la compression JPEG.

Cette image à échelle de gris est divisée en petits blocs de 8x8 pixels pour un traitement ultérieur.

Après cette étape, toutes les autres étapes sont effectuées sur chaque bloc de 8x8 pixels individuellement de manière séquentielle.

1) DCT pour la compression

La clé du processus de compression est la DCT.

La DCT est une transformée fort semblable à la FFT : la transformée de Fourier rapide, travaillant sur un signal discret unidimensionnel.

Elle prend un ensemble de points d'un domaine spatial et les transforme en une représentation équivalente dans le domaine fréquentiel [28].

La DCT est donc effectuée sur chaque matrice 8x8 de valeurs de pixels, et elle donne une matrice 8x8 de coefficients de fréquence.

Dans le signal Img(x,y), les axes X et Y représentent les dimensions horizontales et verticales de l'image. Par contre dans la transformée en cosinus discrète du signal DCT (i,j), les axes représentent les fréquences du signal en deux dimensions [21] .

2) La quantification

On quantifie ensuite chaque bloc de coefficients après transformation DCT.

Concrètement, cela revient à diviser ces matrices par une matrice de quantification afin de réduire le nombre de coefficients.

C'est cette étape qui fait perdre le plus de qualité à l'image mais aussi qui fait gagner le plus de place.

Avec Q la matrice de quantification, F la matrice transformée après DCT et F* la matrice quantifiée.

La matrice des coefficients est ensuite lue en zigzag et la chaine se termine par un caractère de fin (EOB pour signaler qu'il ne reste que des zéros dans la matrice) [24].

3) La sequence en zigzag

Cette étape consiste à codé la matrice DCT quantifiée. Ce codage est réalisé sans perte d'informations, Cela produira en général un très petit nombre.

Le codage du reste de la matrice DCT quantifiée va se faire en parcourant les éléments dans l'ordre imposé par une séquence particulière appelée séquence zigzag [21]:

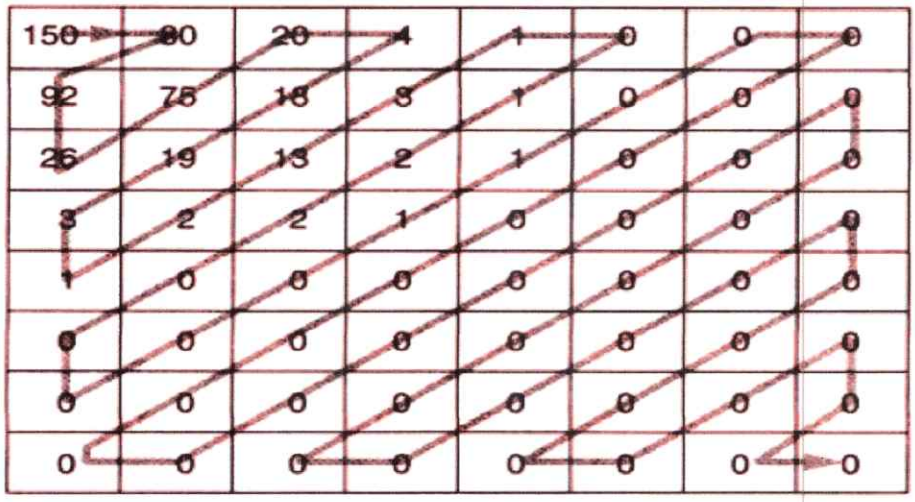

Figure 3.12 Séquence en zigzag.

Ce qui donne la suite suivante : 150, 80, 92, 26, 75, 20, 4, 18, 19, 3, 1, 2, 13, 3, 1, 0, 1, 2, 2, 0, 0, 0, 0, 0, 1, 1, 0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, etc.

Cette séquence à la propriété de parcourir les éléments en commençant par les basses fréquences et de traiter les fréquences de plus en plus hautes. Puisque la matrice DCT quantifiée contient beaucoup de composantes de hautes fréquences nulles, l'ordre de la séquence zigzag va engendrer de longues suites de 0 consécutifs.

Deux mécanismes sont mis en œuvre pour comprimer la matrice DCT quantifiée.

D'une part, les suites de valeurs nulles sont simplement codées en donnant le nombre de 0 successifs.

D'autre part, les valeurs non nulles seront codées en utilisant une méthode statistique de type Huffman ou arithmétique.

4) Codage RLE

Le principe du codage RLE ou le codage par plage est de Créer une nouvelle séquence dans laquelle le deuxième élément correspond au niveau de gris et le premier élément correspond au nombre de pixels consécutif possédant ce niveau de gris, On code séparément le niveau de gris et l'occurrence de chaque pixel Doc un Fort taux de compression pour des images possédant de nombreuses zones de régions homogènes [22].

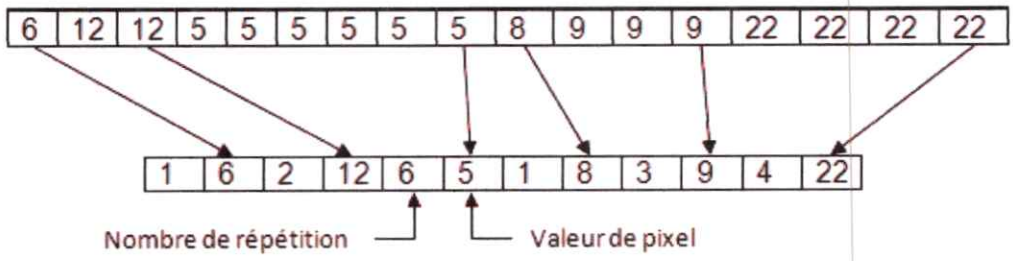

Figure 3.13 RLE (Run Length Encoding).

5) Codage Huffman

Dans l'algorithme de Huffman, on procède différemment pour un résultat comparable. On commence par choisir les deux symboles qui ont le moins d'occurrences, on leur donne les codes 0 et 1 et on les regroupe dans un arbre binaire auquel on attribue un nombre d'occurrences: la somme des deux symboles qu'il regroupe et on continue ainsi [25].

Exemple : Soit le signal S = "CAGATAMAGAGAA". On commence par l'ensemble :{C/1, T/1, M/1, G/3, E/1, A/7}. L'algorithme groupe les deux symboles ou arbres les moins représentés: c/1 et t/1 et crée un arbre pour eux: {C, T}/2. Dans cette notation, on met la branche 0 à gauche, la branche 1 à droite etc…

L'arbre suivant explique les étapes du codage :

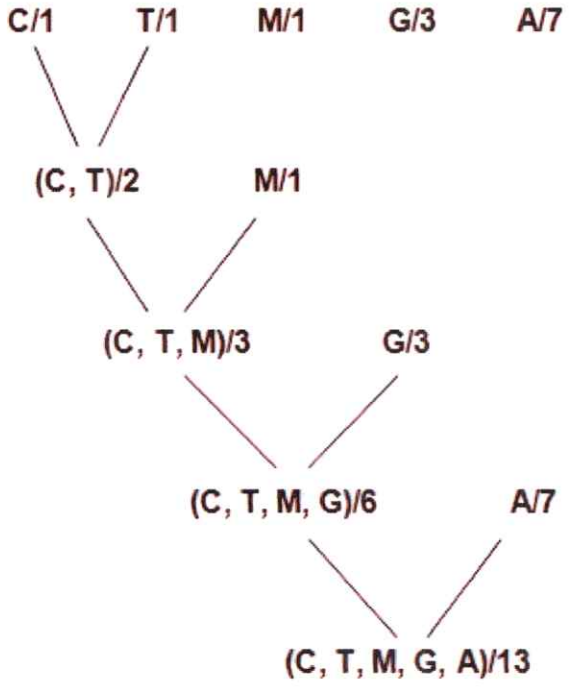

Figure 3.14 Arbre de Huffman.

b) **Decompression:**

Il s'agit de faire le chemin inverse de la compression. Pour cette raison, nous allons nous restreindre à ses étapes principales.

Un logiciel qui va décompresser une image au format JPEG va suivre les étapes principales suivantes:

- Ouverture du fichier concerné

- Remise en forme de la matrice quantifiée, en suivant le chemin inverse de la méthode utilisée pour la compression

- Produit terme à terme des coefficients de la matrice DCT quantifiée, par les coefficients de la matrice de quantification.

- Régénération de la matrice de pixels en appliquant la DCT inverse.

La DCT inverse se fait rapidement en utilisant les notations matricielles [30].

CHAPITRE 4 : IMPLEMENTATION, RESULTATS ET DISCUSION

4.1 Introduction

Ce chapitre résume les principales étapes que nous avons suivies pour réaliser notre implémentation. Il est débuté par un choix technologique des équipements à utiliser tels que la caméra, le support de transmission et les deux ordinateurs émetteur et récepteur, suivi par le choix logiciel tels que le langage de programmation ainsi que la méthode de transmission réseau, le Protocol et la méthode de compression des images.

Les principaux résultats des essais et expériences sont présentés et commentés puis des conclusions sont tirées. Une étude de validation et de robustesse de notre algorithme est menée.

4.2 Choix technologique

4.2.1 Choix de langage de programmation

Notre choix de langage de programmation a été fixé sur le C# version 2012 « Visual studio », du fait de l'habitude de son utilisation et donc sa maitrise relative.

4.2.2 Matériel utilisé

Le choix du matériel à utiliser se résume dans les équipements suivants:

4.2.2.1 Une Caméra USB

Nous avons choisit une caméra PC type *Logitech HD C310* comme montré dans la Figure 4.1.

Figure 4.1 Camera USB Logitech C310

Les principales caractéristiques de la camera sont présentées dans le tableau ci-dessous [2]:

Caractéristiques techniques	Caméra USB logitech C310
Image (s) disponible(s)	Image avant
Type de connexion	USB filaire
Microphone	Microphone disponible avec suppression du bruit

Caractéristiques techniques	Caméra USB logitech C310
Type de lentille et de capteur	Plastique CMOS
Type de mise au point	Fixe
Champ de vision (CDV)	Diagonale de CDV 60°
Distance focale	4.4 mm
Résolution optique réelle	1280x960 VGA
Capture d'image (4 :3 SD)	320x240, 640x480, 1.2 MP, 5MP
Capture d'image (16 :9 W)	360p, 480p, 720p
Capture vidéo (4 : 3 SD)	320x240, 640x480, 800x600
Capture vidéo (16 : 9 W)	360p, 480p, 720p
Débit d'image (maximal)	30 images par seconde en 640x480

Tableau 4.1 Caractéristiques de la camera.

4.2.2.2 Support de Transmission

Notre support de transmission est un câble réseau RJ45 de longueur de 1 m et un autre câble de longueur 20 m.

Figure 4.2 Câble réseau RJ45

Caractéristique	Valeur
Vitesse	1-10Mo/s
Longueur	1 et 20m
Type de transmission	Série

Tableau 4.2 Caractéristiques du Câble Réseau RJ45

4.2.2.3 L'Emetteur: Ordinateur1

L'ordinateur utilisé pour l'acquisition de la vidéo et la transmission est de type PACKARD-BELL EasyNote TK et dont les caractéristiques sont présentées dans le tableau 4.3.

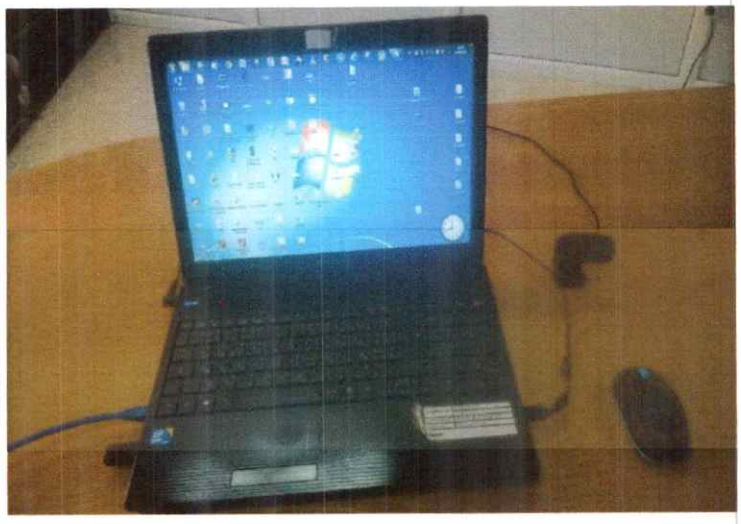

Figure 4.3 Emetteur (ordinateur 1)

4.2.2.4 Le Récepteur: Ordinateur 2

L'ordinateur utilisé pour la réception de la vidéo est de type SAMSUNG dont les caractéristiques sont présentées dans le tableau 4.3.

Figure 4.4 Récepteur (ordinateur 2)

4.2.3 Le banc d'essai complet

Figure 4.5 Banc d'essai.

Le banc d'essai que nous avons utilisé consiste en deux ordinateurs juxtaposés fonctionnant comme émetteur et récepteur et reliés par un câble réseau de type Rj45.

L'émetteur est connecté à notre caméra Logitech via une connexion USB, il abrite le programme1 pour assurer la compression et l'émission sous réseau.

Le récepteur abrite le programme2 pour assurer la réception et l'affichage de la vidéo reçue.

Les caractéristiques des deux ordinateurs sont présentes dans le tableau suivant :

	Ord1 (Emetteur)	**Ord2 (Recepteur)**
Processeur	Intel(R) Core(TM) i3 CPU M 370 @2.40GHz	Intel(R)Core(TM) i5-3230M CPU @2.60GHz
Memoire installée(RAM)	4.00 Go	8.00 Go
Type de systeme	Système d'exploitation 64 bits	Système d'exploitation 64 bits
Systeme d'éxploitation	Windows 7 Edition Integrale	Windows7 Edition Integrale
Disc-dur	500 Go	1 To
Carte Graphique	Intel HD Graphics	Intel HD Graphics

Tableau 4.3 Caractéristiques de l'émetteur et le récepteur

4.3 Procédures de l'émission réception

Le schéma suivant représente la succession des étapes de transmission de la vidéo entre les deux machines (ordinateur1 et ordinateur2)

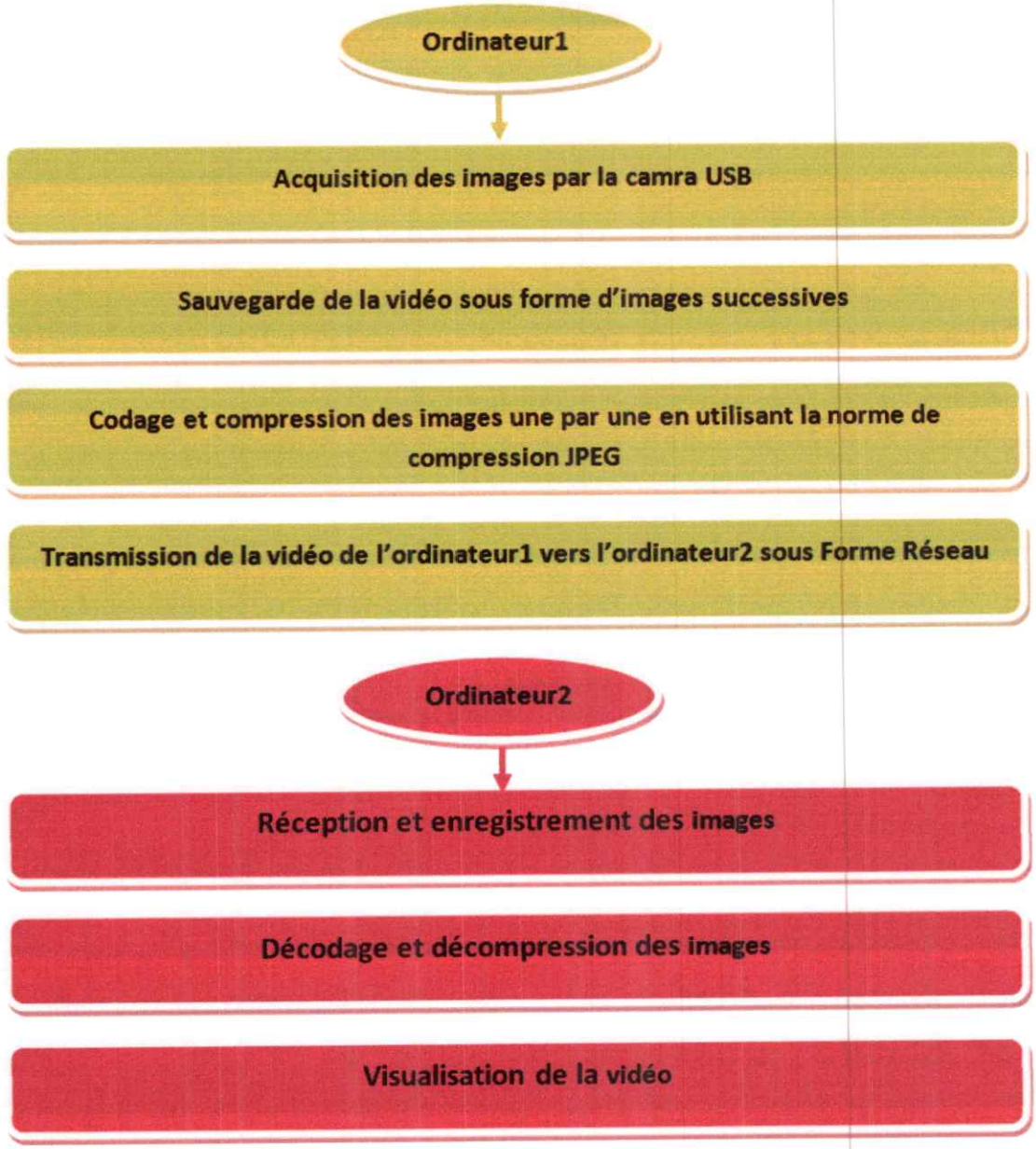

Figure 4.6 Schéma de la transmission et réception.

Pour la transmission de la vidéo nous allons utiliser la programmation en socket en mode non connecté.

Le mode non connecté (analogue à une communication par courrier), utilisant le protocole UDP.

Ce mode nécessite l'adresse de destination à chaque envoi, et aucun accusé de réception n'est donné.

- Créer un nouveau socket.
- Connecter le socket send (computer1) avec le socket receive (computer2).

1) Principe :

L'image prise par la caméra est sauvegardée dans l'ordinateur1.

Elle sera compressée suivant la norme JPEG en 1/24s.

Puis la transformer d'une image matricielle en une image linéaire pour la transmettre, (la taille de l'image avant la compression est de 46k Octets et 18k Octets après la compression).

L'image sera segmentée en paquets de 1024 Octets, On y rajoute 5 Octets à chaque paquet (1024+5 : 1 Octet pour le flag et 4 Octets pour la taille de l'image).

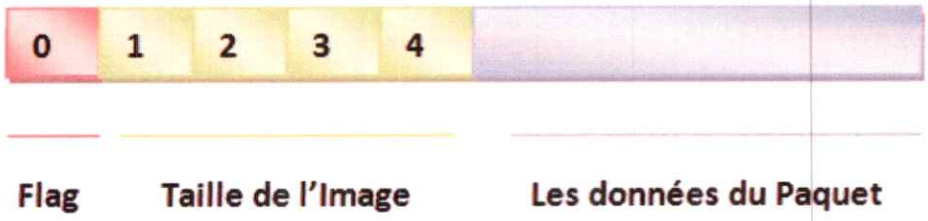

Figure 4.7 Format du Paquet.

On appelle les 5 premiers octets l'entête (flag+ la taille de l'image à envoyer).

Chaque paquet sera préparé pour l'émission selon le protocole suivant :

Les valeurs du flag seront :

0, pour le début de l'image, 1 pour le milieu de l'image et 2 indique la fin de l'image.

L'opération de l'émission débute par copier les premiers 1024 octets de l'image dans :

- Le paquet numéro1 et y ajouter son entête (flag=0 et les 4 Octets de la taille de l'image)
- Le paquet numéro2 et son entête (flag=1 et les 4octets vides)……jusqu'au l'avant dernier paquet (flag=1 +4octets vides)
- Le dernier paquet : la fin de l'image est connue dès que la taille des données restantes est inférieure à 1024octets (flag=2 ; pour informer le récepteur que c'est le dernier paquet portant la queue de l'image).

A la réception l'image sera identifiée selon le protocole inverse.

La réception débute lorsque :

- le récepteur détecte le premier paquet de l'image (c'est le début de l'image flag=0 +la taille de l'image à recevoir et à stocker).
    - Le récepteur reçoit le reste des paquets (flag=1+la taille de l'image).
    - Et enfin le dernier paquet qui porte la fin de l'image reconnue par flag=2 + le reste de l'image.
    - L'image est ensuite enregistrée sur l'odinateur2.

La même procédure est appliquée pour toutes les images stockées sur l'ordinateur1.

C'est le protocole UDP qui assure la synchronisation entre émetteur/récepteur depuis le début de la connexion au niveau des deux instructions:

- *Serversocket.connect(ipep)*
- *Clientsocket.receive(datastream).*

Organigramme de l'emission:

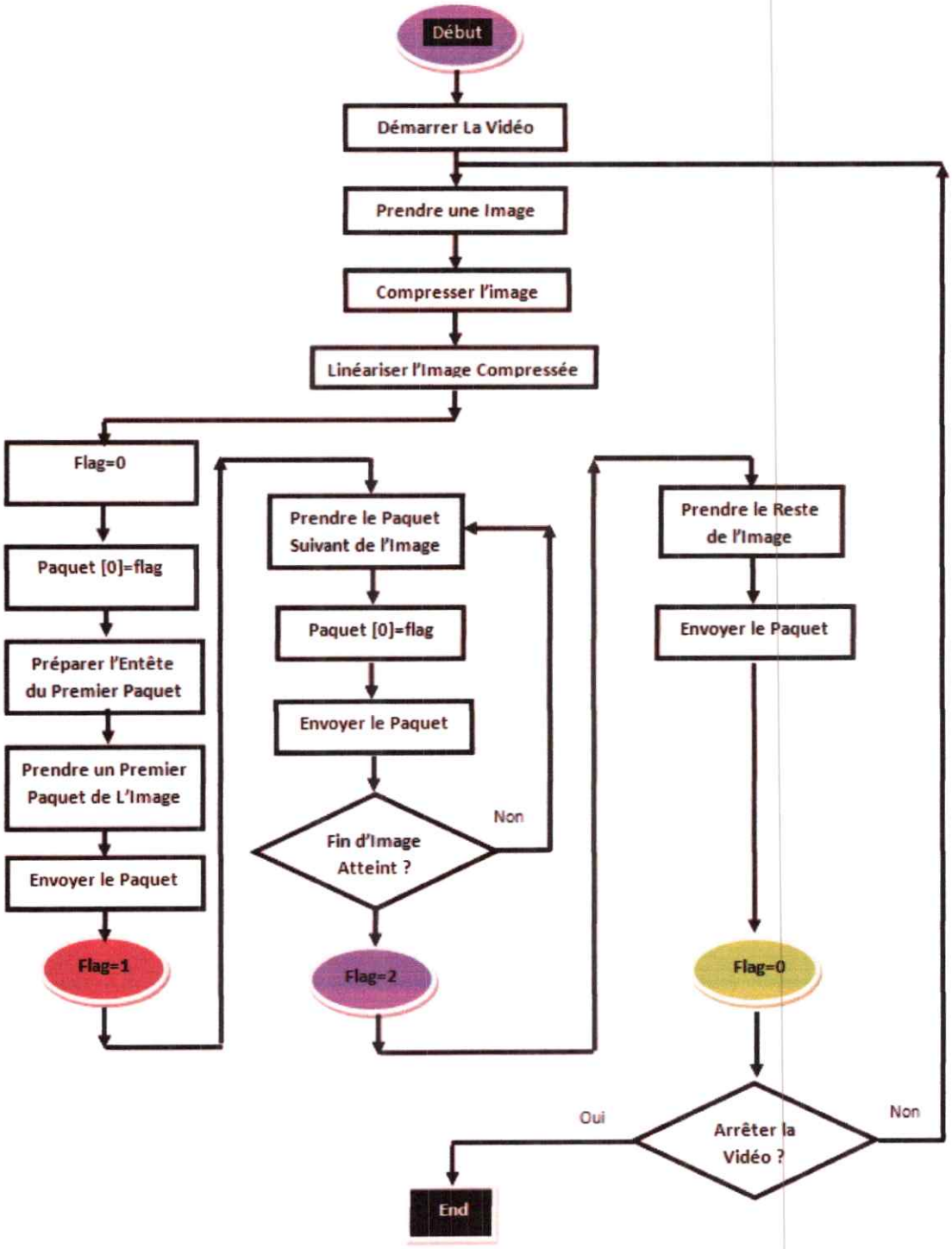

Figure 4.8 Schéma de l'organigramme de l'émission.

L'algorithme illustré par cet organigramme, développé en interne, a été implémenté en C# et il travail sur l'ordinateur émetteur.

Organigramme de la réception :

L'algorithme illustré par cet organigramme de réception suit les mêmes étapes que celles de l'émission, a été implémenté en C# et il travail sur l'ordinateur récepteur mais en un ordre inversé et en ajoutant l'affichage de la vidéo reçue.

Il est à noter que ces organigrammes ont été développés par nous même en choisissant 1 octet pour le flag et 04 octet pour transmettre la taille de l'image.

2) Mode Opératoire

Les images capturées, compressées et reçues sont enregistrées sur les disques des deux ordinateurs,

Le temps de transmission de chaque image est enregistré dans un fichier texte,

Les fonctions conçues sous MATLAB ont été écrites pour : extraire le temps de transmission des fichiers texte et l'ordonner, extraire la taille des images, dessiner les différents graphs.

Les fonctions sont :

- *ProgramGlobal.m* : Programme principale, il appelle les autres fonctions.
- *CompareImages.m* : Soustrait les images compressées des images reçues.
- *ElapsedTimeExtraction.m* : extrait le temps de transmission depuis les fichiers texte.
- *CompressionCaracteristique* : Extrait les tailles des images.
- *PlotGraphs*: Dessiner les Graphs nécessaires.

4.4 Résultats et discussion

4.4.1 Validation de l'implémentation

La mise en marche de notre programme a donné les résultats qui permettent la validation de notre implémentation, ou on peut remarquer à l'eoil que la vidéo reçue est une copie fidèle de la vidéo injectée à partir de la caméra, comme c'est montré dans les figures suivantes :

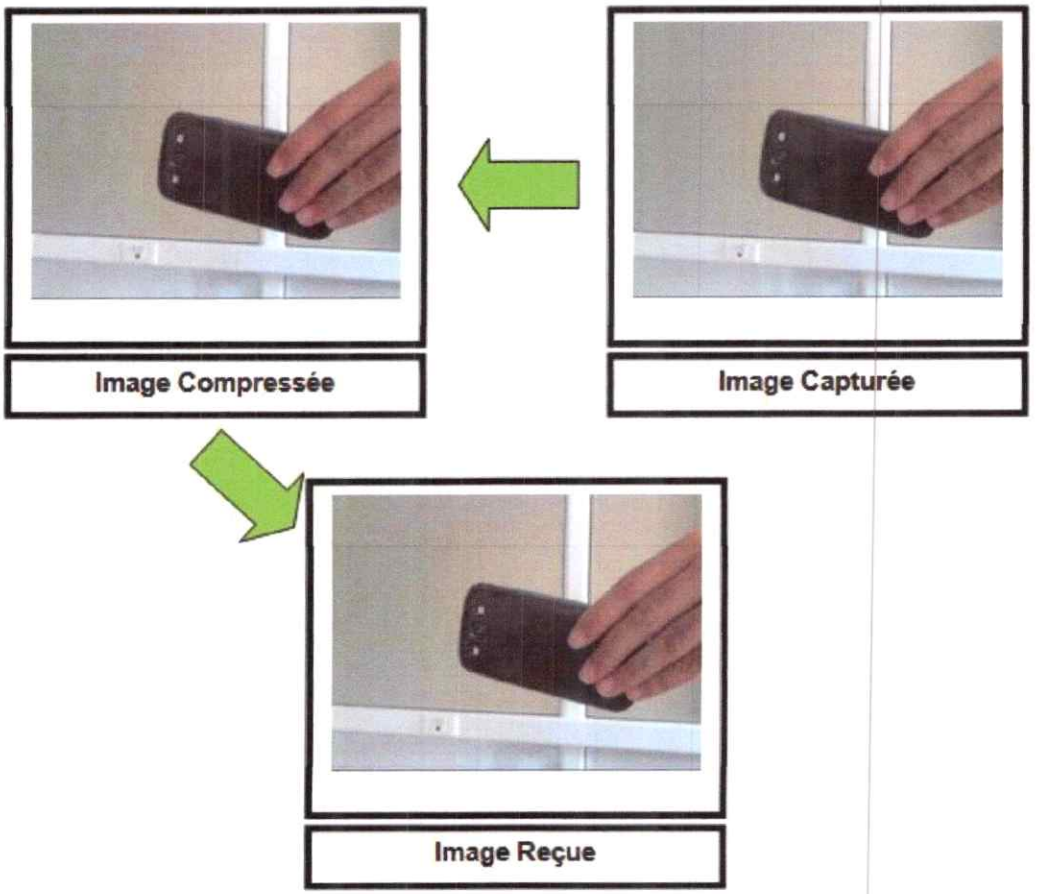

Figure 4.9 Comparaison entre image capturée, compressée et reçue sauvegardée dans le disque.

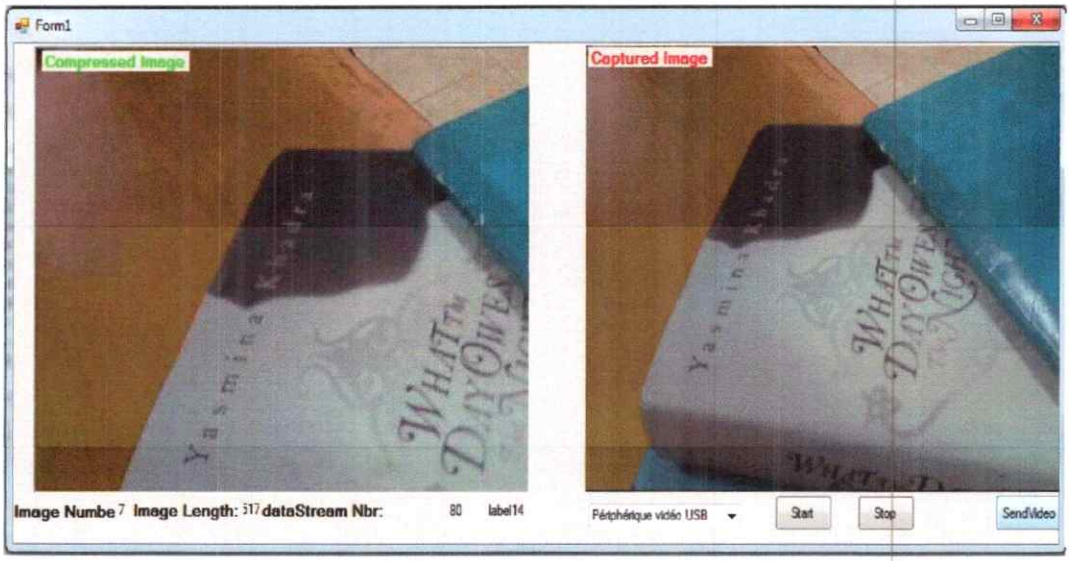

Figure 4.10 Comparaison des résultats depuis le Programme en marche.

La Figures 4.9 et 4.10 montre les résultats de l'implémentation de l'algorithme de transmission de vidéo en temps réel en marche, elle montre que les résultats obtenus sont satisfaisants, si on compare l'image reçue avec celle envoyée.

4.4.2 La compression

4.4.2.1 Taux de compression

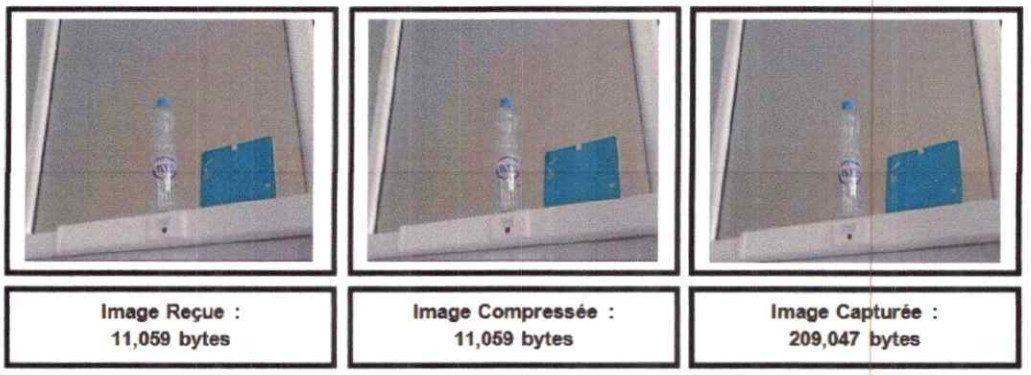

Figure 4.11 Comparaison de la taille entre image capturée, compressée et reçue.

Le taux de compression est un paramètre pour caractériser le compresseur utilise, il est calculé en utilisant la formule Taux de Compression (TC)= taille image capturée / taille image reçue.

Pour cet exemple on le trouve Taux de Compression TC= 209,047 Octet /11,059 Octet =18.9.

4.4.2.2 Caractérisation du Compresseur JPEG Utilisé

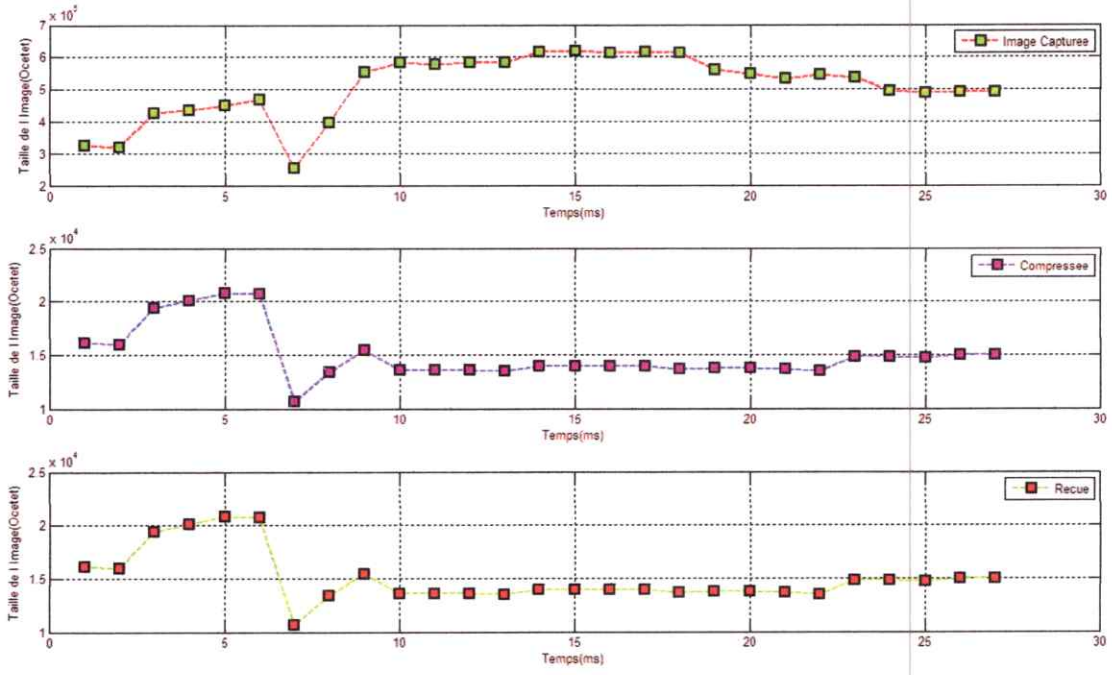

Figure 4.12 Caractérisation du Compresseur pour 30 Images.

La Figure 4.12 montre le comportement du compresseur JPEG utilisé, on remarque que la compression a été appliquée sur les images capturées avec un taux de compression légèrement variable.

Les images compressées et qui ont été envoyées dans le canal de transmission ont été reçues d'une manière presque identique.

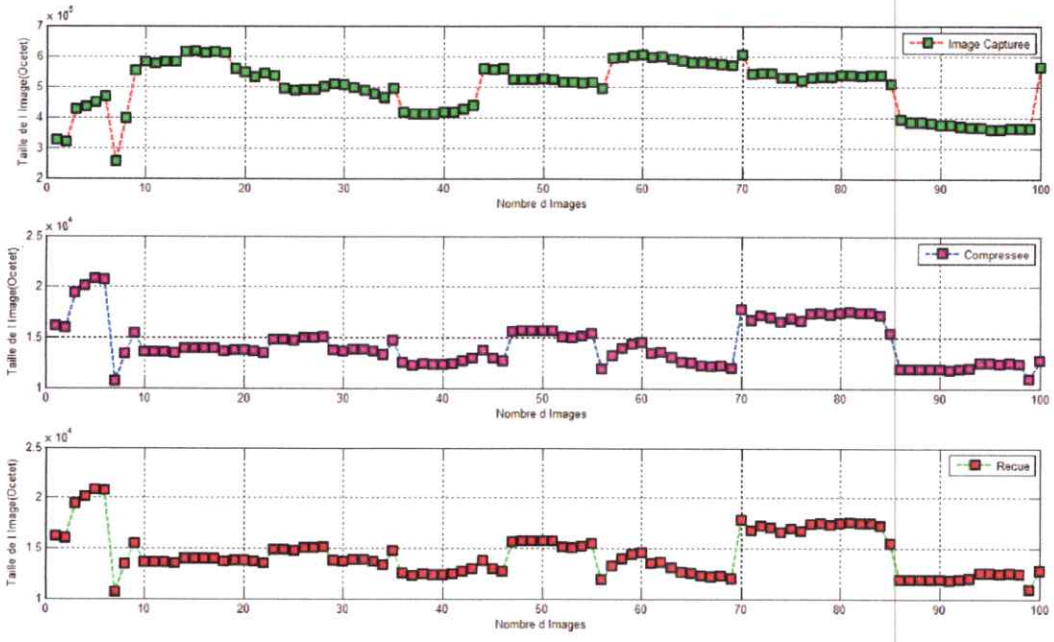

Figure 4.13 Caractérisation du Compresseur pour 100 Images.

La Figure 4.13 montre les mêmes résultats que l'expérience précédente avec un nombre plus élevé des images(100), on remarque le même comportement de notre compresseur et aussi une ressemblance entre les images compressées et celles reçues.

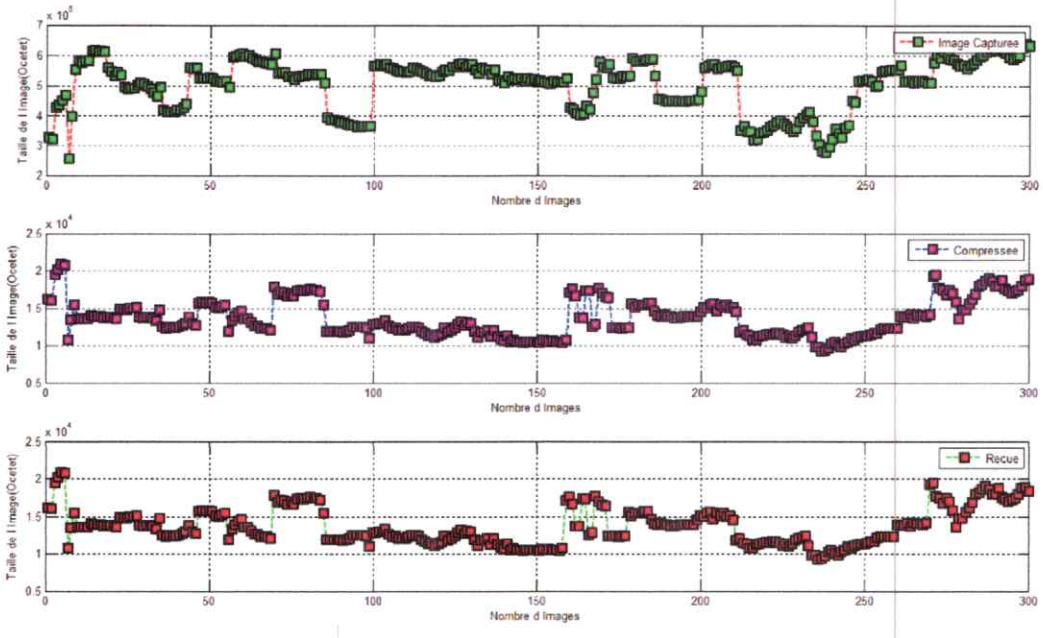

Figure 4.14 Caractérisation du Compresseur pour 300 Images

La Figure 4.14 montre les mêmes résultats que les deux expériences précédentes avec un nombre encore plus élevé des images(300), on remarque le même comportement de notre compresseur et aussi une ressemblance entre les images compressées et celles reçues.

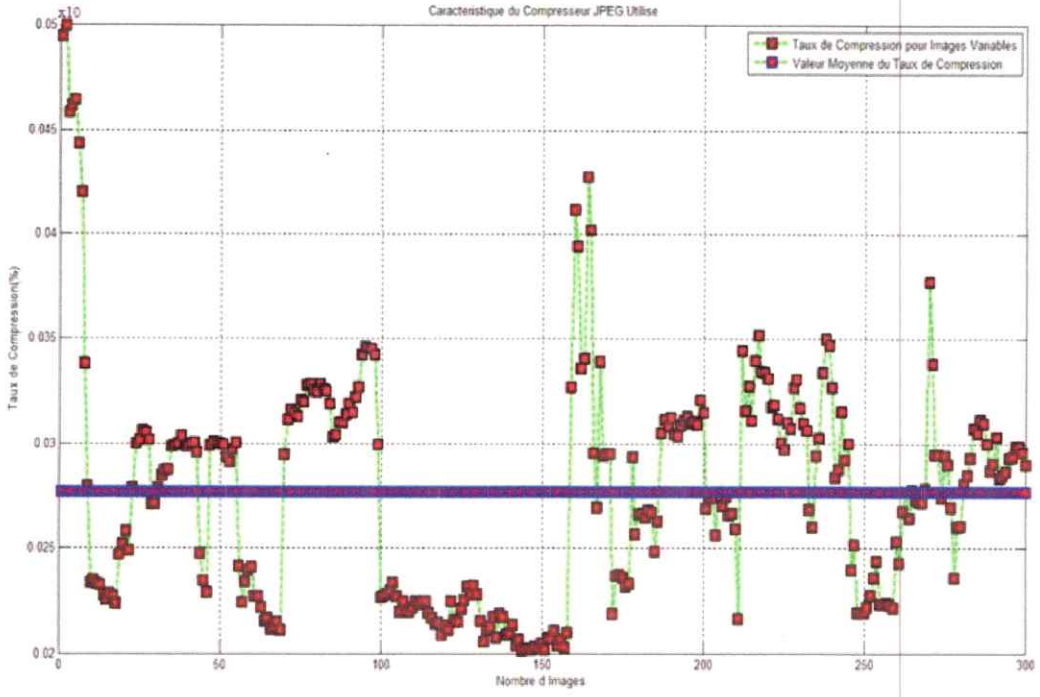

Figure 4.15 Etude de la variation du Taux de compression

La figure 4.15 montre la variation du taux de compression en fonction de la taille de chaque image capturée qui varie, dans notre cas, de 20% et 43%.

4.4.3 Caractérisation du support de transmission

a) Expérience

1. Envoyer une vidéo d'une scène sans mouvement (même image sera envoyée) et observée le temps de transmission pour chaque image.
2. Faire cette expérience pour une distance Ord1-Ord2=1m et pour 20m.

b) Résultats et Discussion

1. **Pour D=1m**

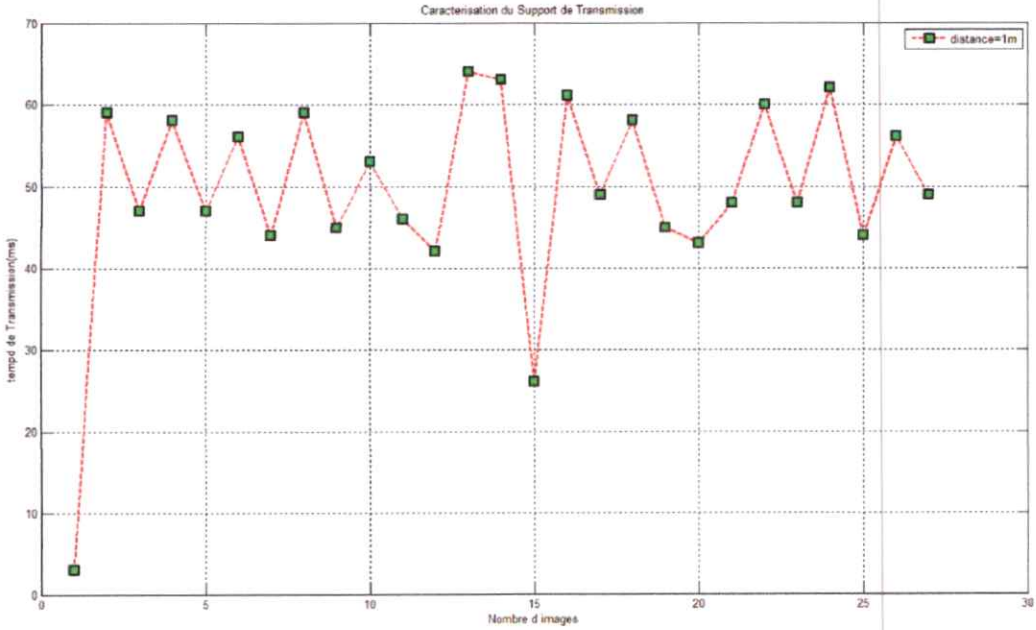

Figure 4.16 Caractérisation du Support de Transmission pour D=1m.

La figure 4.16 montre que le temps de transmission de notre support pour une distance de 1m reste relativement constant (autour d'une moyenne 50 ms) et ce pour la même taille des images, ce qui montre la non affectibilité par le bruit de transmission qui est pour ce cas +/-15ms.

2. **Pour D=20m**

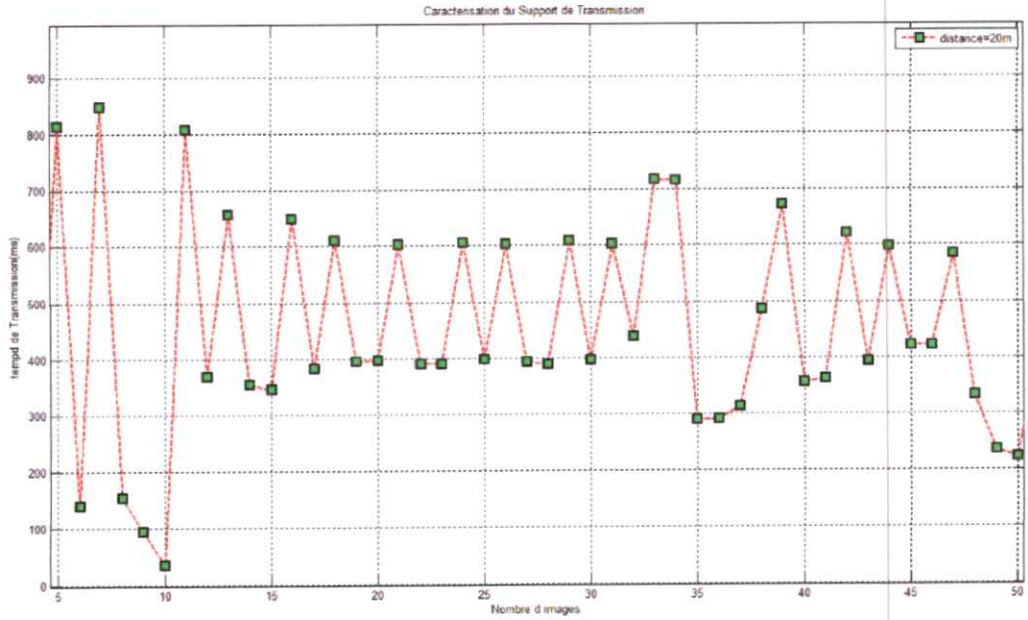

Figure 4.17 Caractérisation du Support de Transmission pour D=20m.

La figure 4.17 montre que le temps de transmission de notre support pour une distance de 20m tourne autour d'une moyenne de 500 ms, ce qui montre l'effet de la distance et d'un bruit de transmission de +/-350ms qui est plus considérable que celui de la distance de 1m.

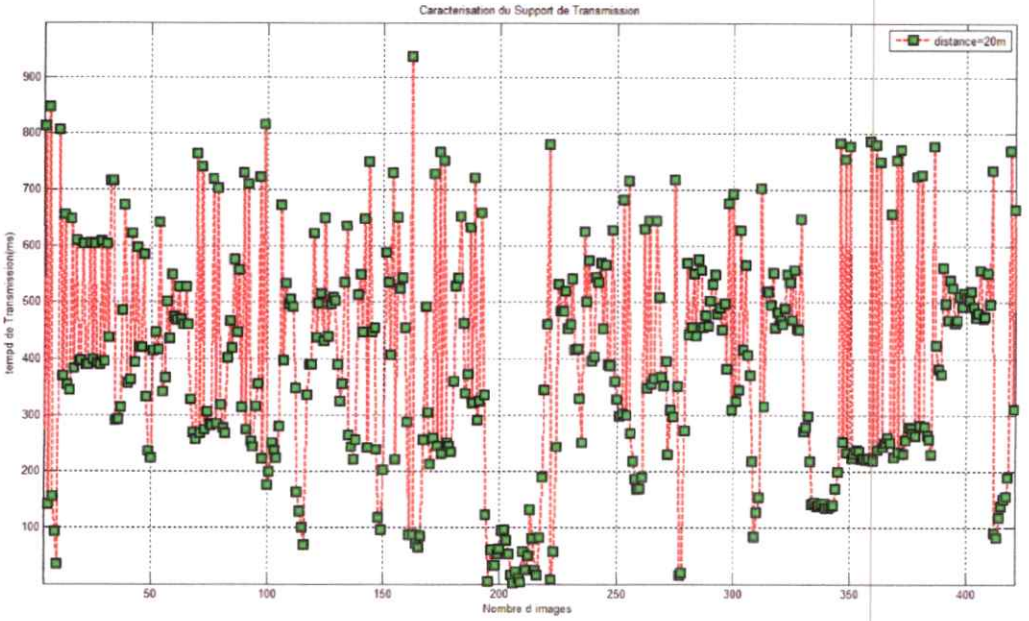

Figure 4.18 Caractérisation du Support de Transmission pour Un nombre enlevé d'images, et distance D=20m.

La figure 4.18 montre un effet plus remarquable du bruit de transmission vu que l'expérience a été faite pour une durée plus longue (>400 images) et pour une distance de 20m.

c) Conclusion

Le support de transmission suit une loi linéaire en fonction de la taille des paquets jusqu'à la valeur de saturation qui est pour notre cas autour de 2700 Octets, au-delà duquel un comportement pseudo-aléatoire du support est observé.

L'effet linéaire du support (une valeur moyenne constante pour toutes les expériences) et un comportement similaire à chaque fois qu'on envoie la même image.

Les expériences ont montré que le support de transmission se comporte de la même manière pour une distance donnée.

Il a été aussi relevé que l'augmentation de nombre des images envoyées à un effet minime sur le temps de la transmission.

4.4.4 Effet de la variation de la taille des tampons sur le temps de transmission

a) Expérience

1. Envoyer une vidéo d'une scène avec mouvement (tailles d'images variables) et observée le temps de transmission pour chaque image.
2. Faire cette expérience en variant la taille des paquets pour 512 Octets, 1024, 1536, 2700 et 2800.
3. Conclure sur l'effet de la taille des paquets utilisée sur la transmission.

b) Résultats et Discussions

1. **Tampon=512 Octets**

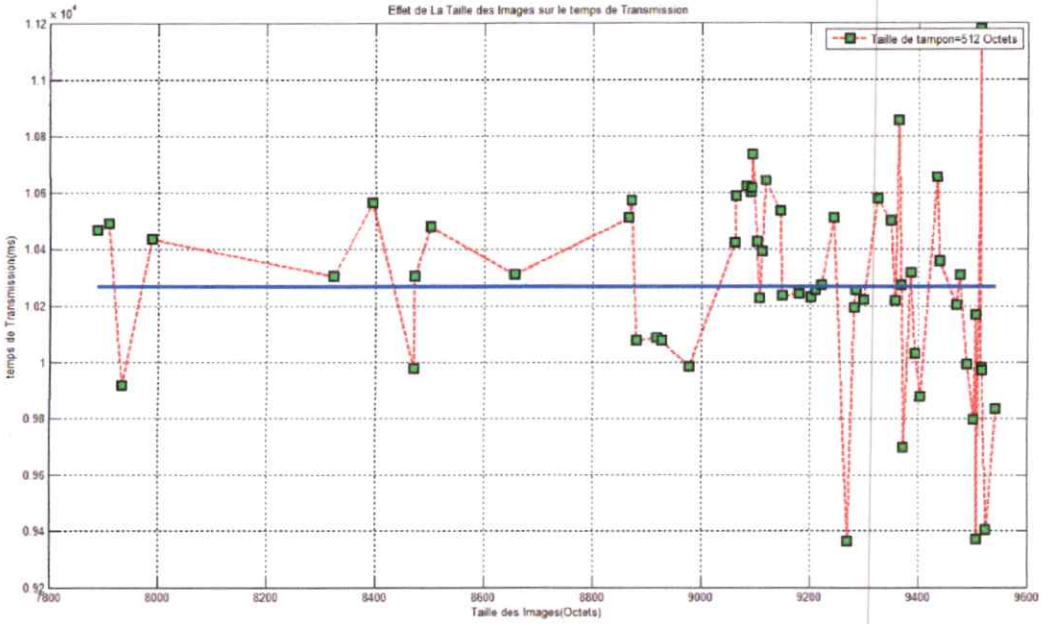

Figure 4.19 Effet de la taille des Images sur le temps de transmission pour tampon de 512 Octets.

La Figure 4.19 montre que le temps de transmission varie autour d'une valeur moyenne de 1.03ms.

2. Tampon=1024 Octets

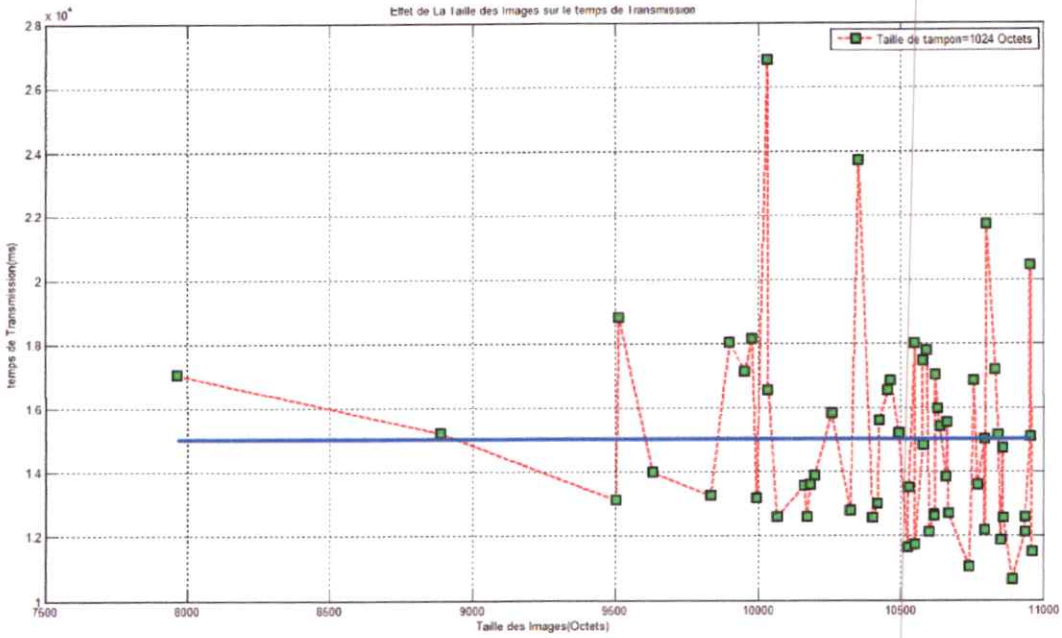

Figure 4.20 Effet de la taille des Images sur le temps de transmission pur tampon de 1024 Octets.

La Figure 4.20 montre que le temps de transmission augmente en fonction de la taille des images envoyées moins rapidement si on la compare avec l'utilisation d'une taille de tampon de 1024 Octet et tourne autour d'une valeur de 1.5ms.

3. Tampon=3*512 Octets

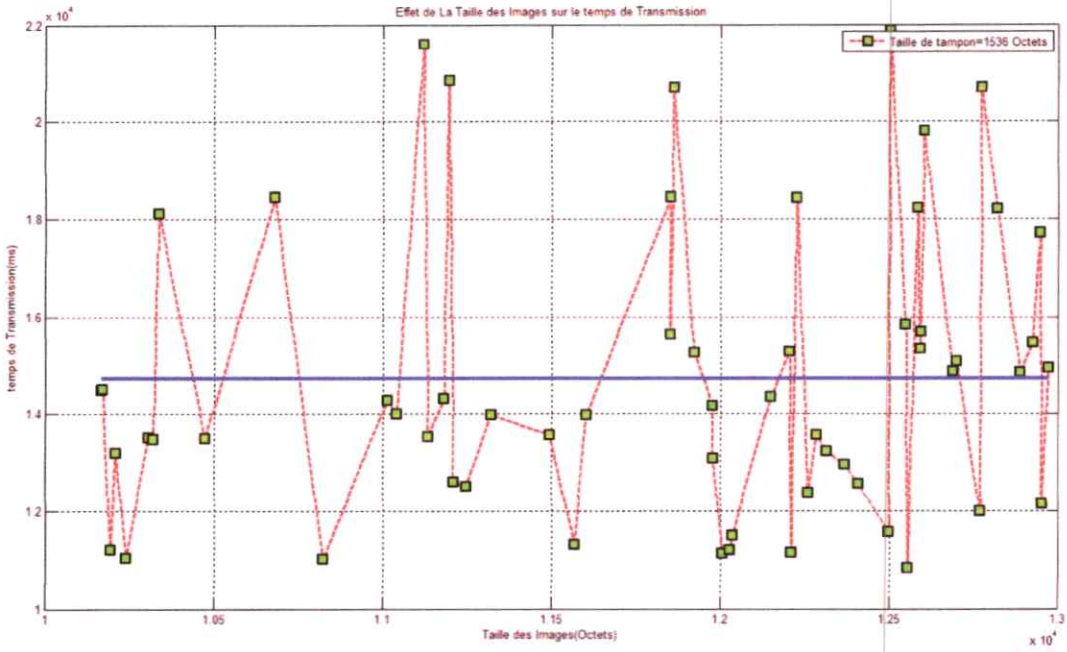

Figure 4.21 Effet de la taille des Images sur le temps de transmission pur tampon de 1536 Octets.

La Figure 4.21 montre que le temps de transmission augmente en fonction de la taille des images envoyées moins rapidement si on la compare avec l'utilisation d'une taille de tampon de 1536 Octet et tourne autour d'une valeur de 1.5ms.

4. **Tampon=2700 Octets**

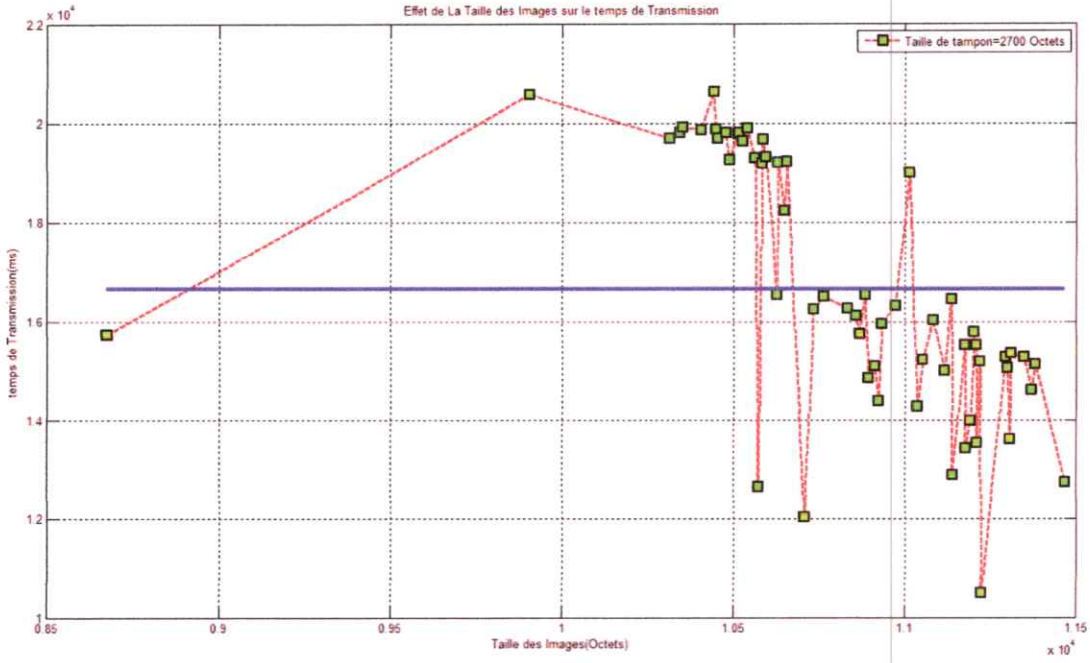

Figure 4.22 Effet de la taille des Images sur le temps de transmission pur tampon de 2700 Octets.

La Figure 4.22 montre que le temps de transmission augmente en fonction de la taille des images envoyées moins rapidement si on la compare avec l'utilisation d'une taille de tampon de 2700 Octet et tourne autour d'une valeur de 1.7ms.

5. **Tampon=2800 Octets**

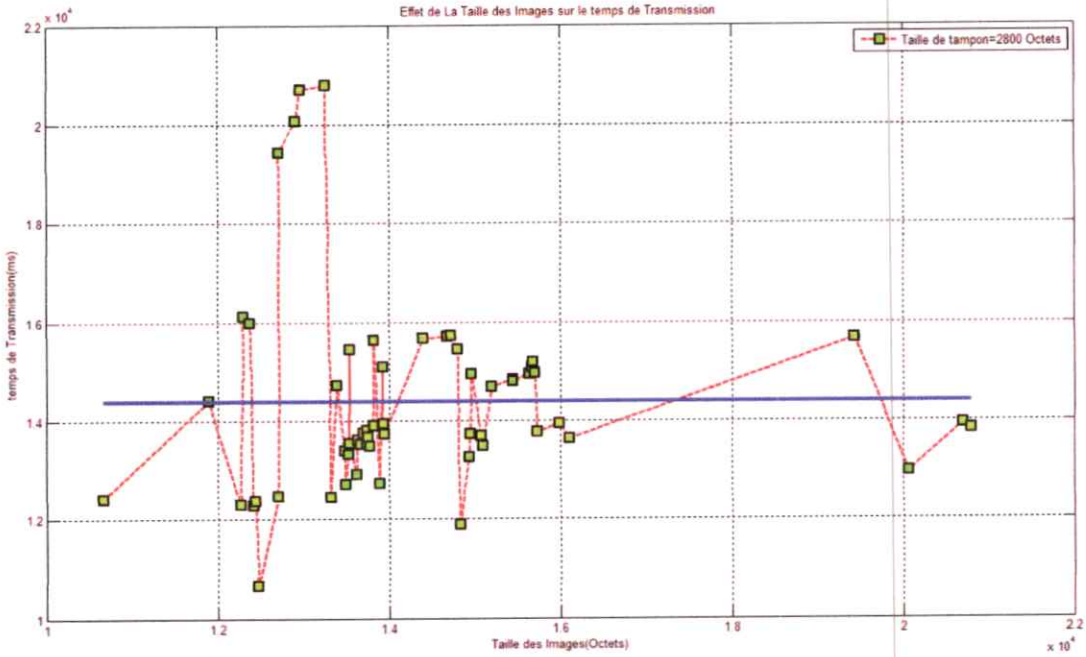

Figure 4.23 Effet de la taille des Images sur le temps de transmission pur tampon de 2800 Octets.

La Figure 4.23 montre que le temps de transmission augmente en fonction de la taille des images envoyées moins rapidement si on la compare avec l'utilisation d'une taille de tampon de 2800 Octet et tourne autour d'une valeur de 1.45ms.

Dans la Figure 4.23 on a remarqué qu'en utilisant la taille 2800octets des paquets, des coupures répétées de la connexion ce qui montre qu'il y a saturation de la transmission à partir de cette taille, et donc aucune conclusion ne peut être tirée.

c) Conclusion

Vus les résultats sur l'effet de la variation de la taille des tampons sur le temps de transmission : 1.03, 1.5, 1.5, 1.7, 1.45ms pour les taille 512,1024,1536,2700 et 2800 Octets respectivement, on peut dire que le temps de transmission augment si en augment la taille de tampon a l'exception celui de 2800 Octet qui peut s'expliquer par le faite que il y a eu des coupure répétées de la connexion vu à notre sens à la saturation de la mémoire et/ou du support de transmission.

4.4.5 Effet de la variation de la distance sur le temps de transmission

a) Experience

1. envoyer une vidéo d'une scène avec mouvement (tailles d'images variables) et observée le temps de transmission pour chaque image en utilisant la même taille des paquets de 1024 Octets.
2. Faire l'expérience pour distance Ord1-Ord2 de 1m et 20m.
3. Conclure sur l'effet de la distance sur la transmission.

b) Résultats et Discussion

1. **Pour D=1m**

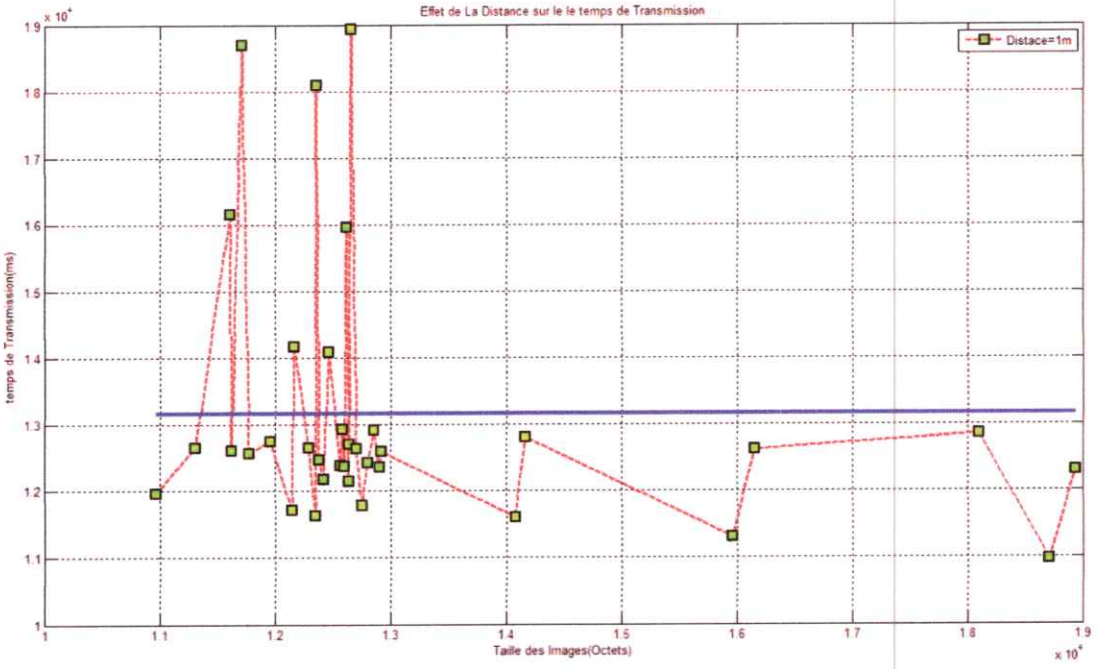

Figure 4.24 Effet de la taille des images sur le temps de transmission pour D=1m.

La figure 4.24 montre que le temps de transmission pour une distance de 1m tourne autour d'une valeur moyenne de 1.35ms.

2. **Pour D=20m**

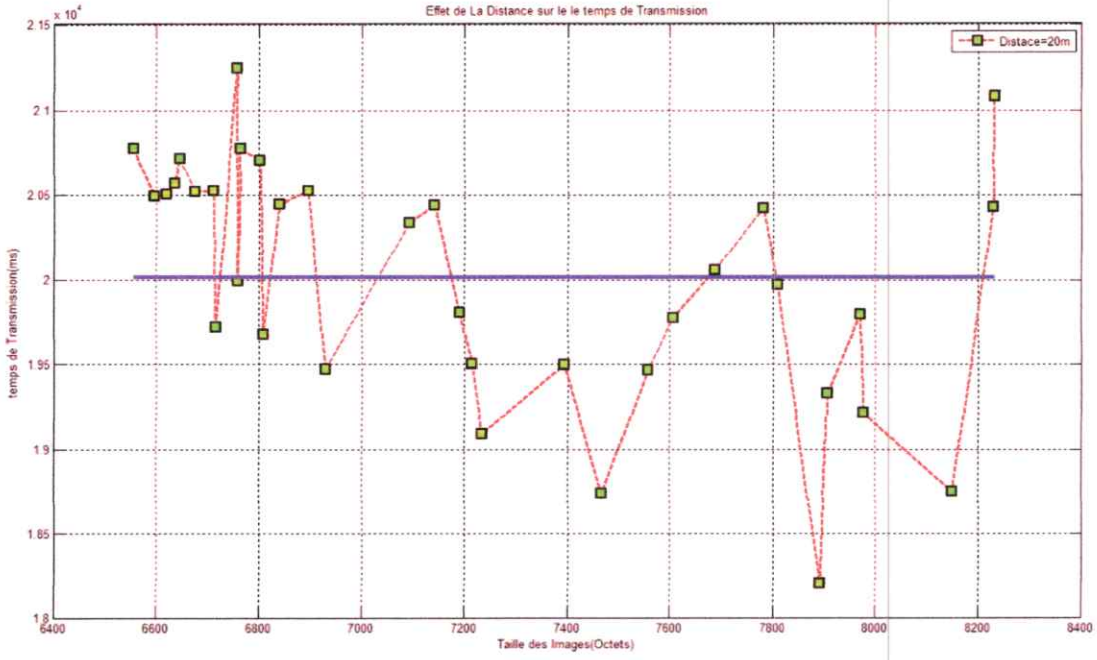

Figure 4.25 Effet de la taille des images sur le temps de transmission pour D=20m

La Figure 4.25 montre que le temps de transmission pour une distance de 20m tourne autour d'une valeur moyenne de 2ms.

c) Conclusion

Les résultats des deux distance 1 et 20 ms ont donne respectivement les temps de transmission de 1.35 et 2 ms, ce qui est logique vue la distance qui augmenter mais pas d'une manière très considérable (2-1.35=0.65ms de différence pour une multiplication de distance de 20 fois).

4.4.6 Fiabilité de la liaison

a) Experience

1. envoyer une vidéo d'une scène avec mouvement (tailles d'images variables) et comparer la taille des images émises avec celle reçues correspondantes,
2. Faire l'expérience pour différentes scènes et pour relativement de longues périodes.
3. Conclure sur la fiabilité de la transmission (quantité des images ou paquets perdus ou reçus en parties ou perdus).

b) Résultats et Discussions

1. **Essai N01**

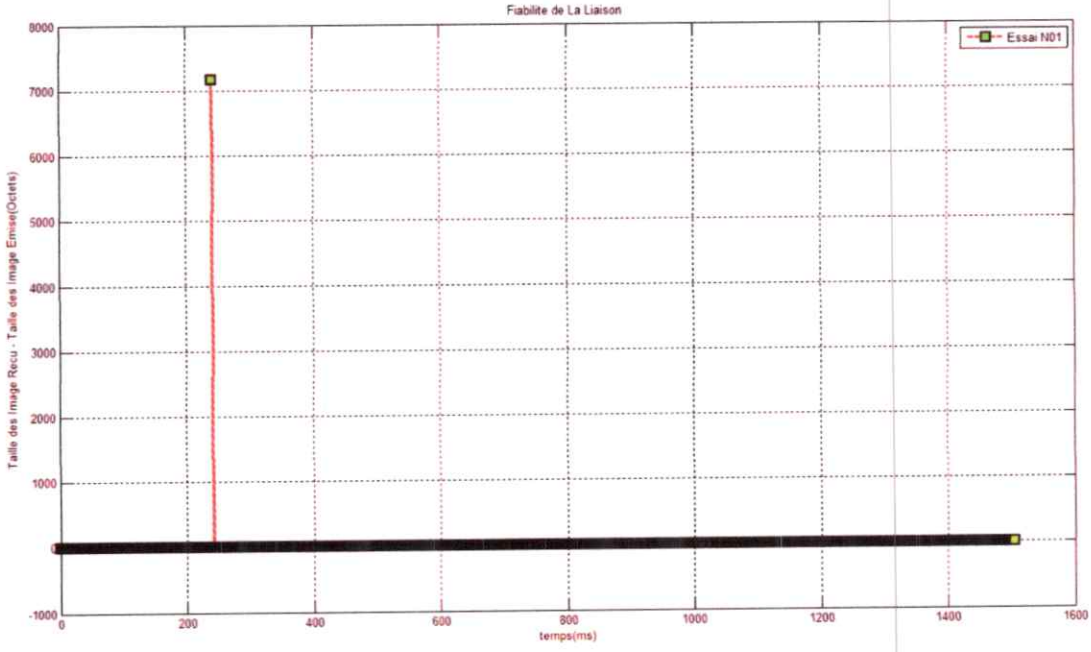

Figure 4.26 Teste de fiabilité de la liaison, Essai N01.

La Figure 4.26 montre une différence de 7000Octes après une durée de transmission de 230 ms et puis une transmission parfaite des données entre Ord1 et 2.

2. **Essai N02**

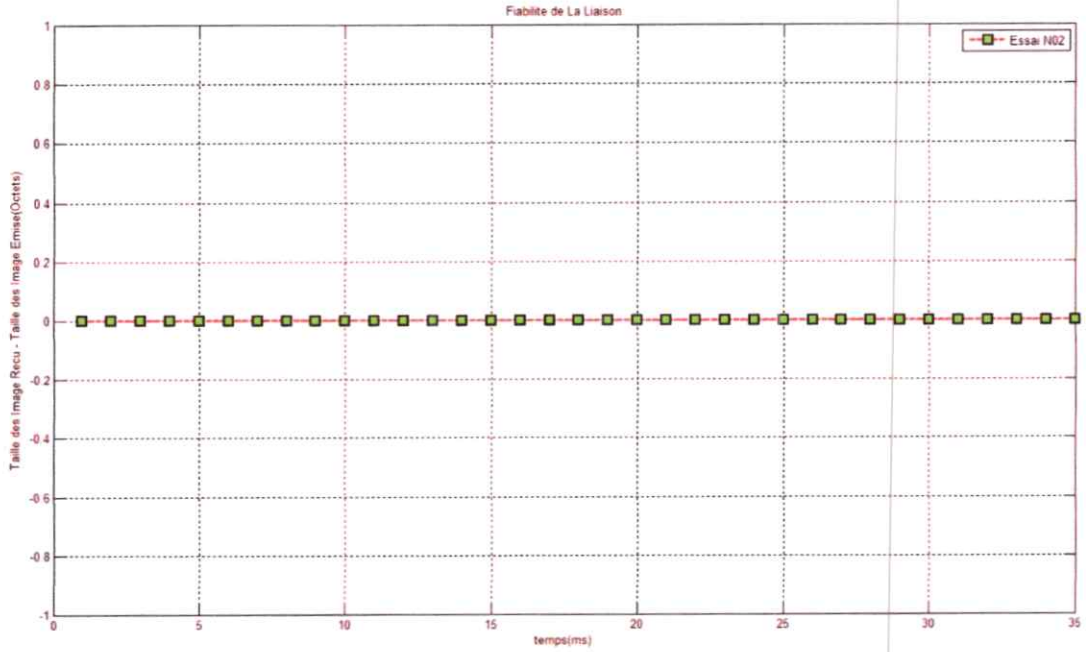

Figure 4.27 Teste de fiabilité de la liaison, Essai N02.

La Figure 4.27 montre une transmission complète à 100%.

3. **Essai N03 :**

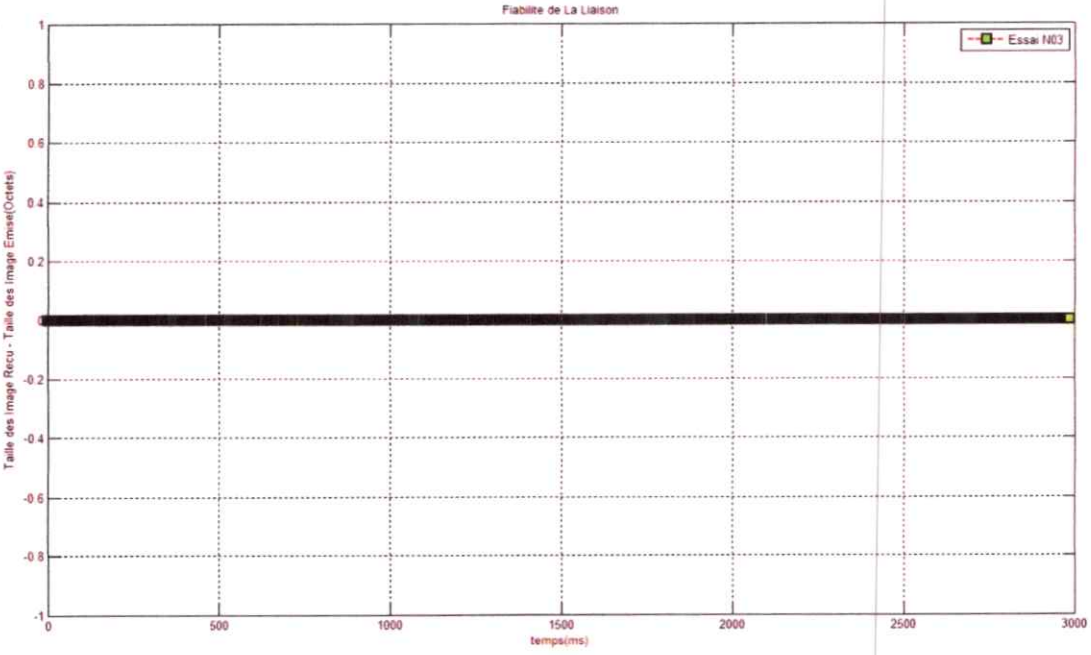

Figure 4.28 Teste de fiabilité de la liaison, Essai N03.

La Figure 4.28 montre une transmission complète à 100%.

4. **Essai N04**

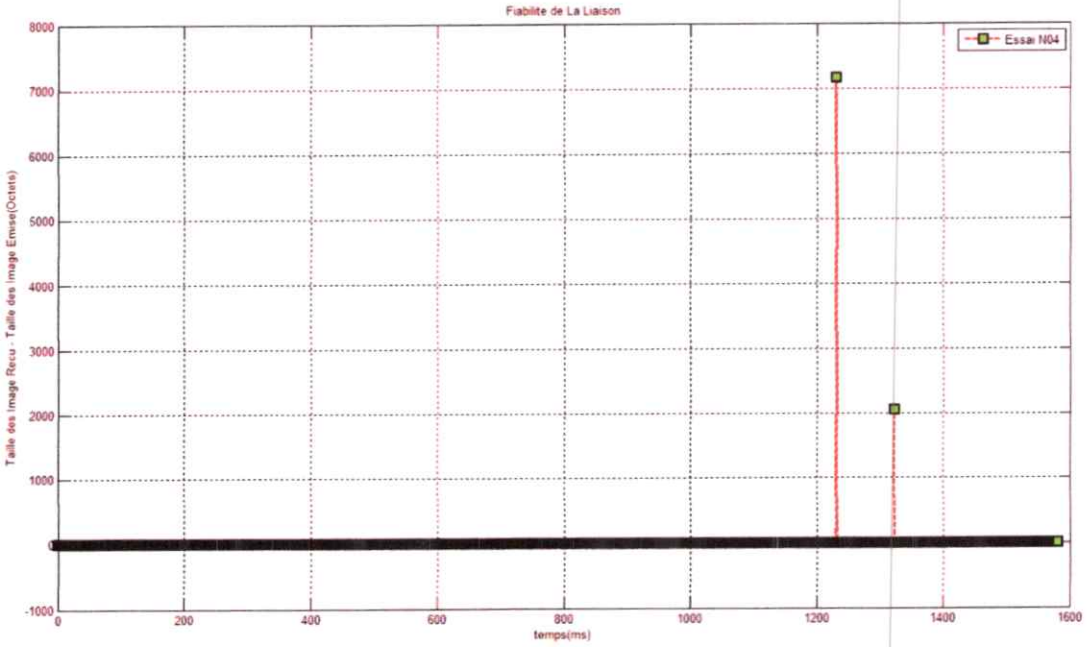

Figure 4.29 Teste de fiabilité de la liaison, Essai N04.

La Figure 4.29 montre qu'aux abscisses 1220 et 1350 ms de durée de transmission ; une perte double de 7000Octes et 2000Octes respectivement ont été enregistré, ce qui corresponds a la perte de 63% (taille typique des images est de 11000Octet) pour le premier paquet et de18% pour le deuxième.

5. Essai N05

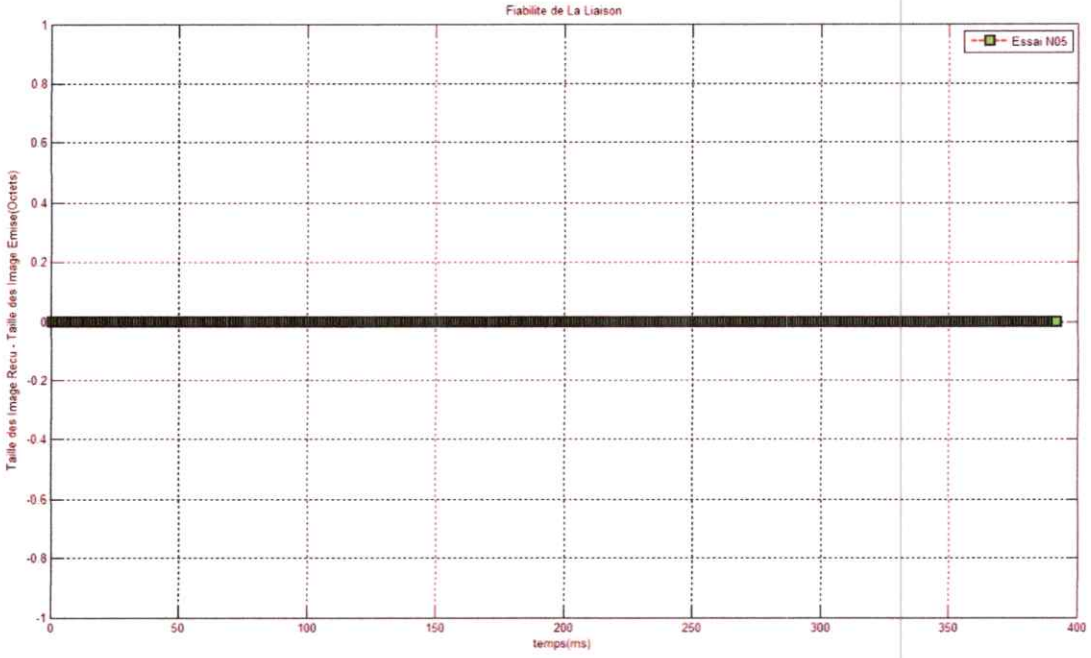

Figure 4.30 Teste de fiabilité de la liaison, Essai N05.

La Figure 4.30 montre une transmission complète à 100%.

6. Essais N06

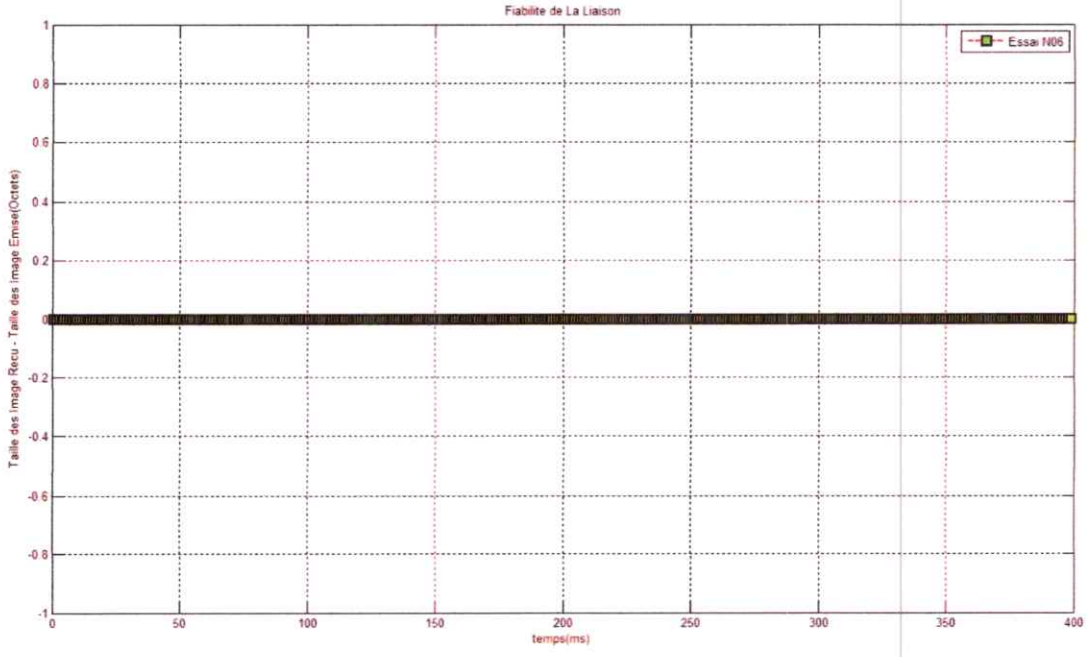

Figure 4.31 Teste de fiabilité de la liaison, Essai N06.

La Figure 4.31 montre une transmission complète à 100%.

7. Essai N07

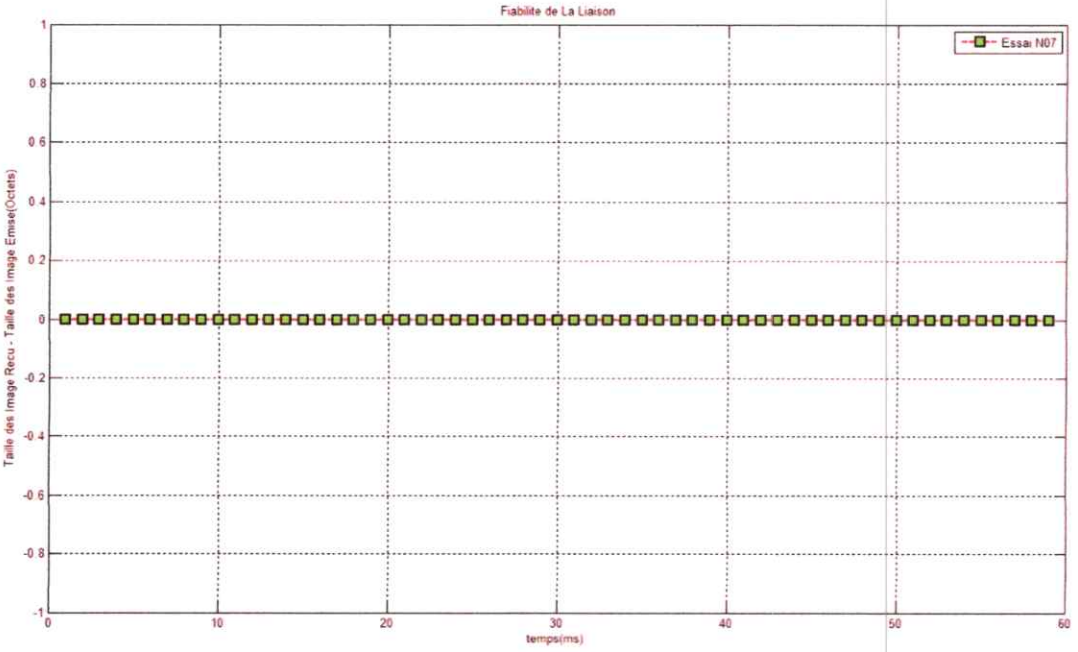

Figure 4.32 Teste de fiabilité de la liaison, Essai N07.

La Figure 4.32 montre une transmission complète à 100%.

c) Conclusion

Pour notre application, on peut considérer que c'est une transmission parfaitement fiable car la perte d'une image ou une dizaine est tolérable puisque elle ne sera pas remarquée par l'œil humain.

CONCLUSION GENERALE

Le projet de l'implémentation d'un algorithme de transmission réseau de la vidéo en temps réel a débuté par la réalisation d'une communication point à point entre deux ordinateurs et pour une petite distance de 1m, ensuite on est passé vers la vidéo en essayant de capturer des images vives à partir de la camera USB et les envoyer sur des distances pouvant atteindre 20m, il a été relevé la nécessité de l'application de la compression continue des images pour réduire la latence qui existait entre les images reçues et celle envoyées et les pertes des informations d'une part et pour répondre à l'exigence d'une transmission en temps réel d'autre part.

Pour cela, On a opté pour le standard de compression JPEG pour son efficacité et sa simplicité relative.

Le Protocol UDP, qui a été choisi en dépit du Protocol traditionnel TCP, nous a permis plus de flexibilité dans la programmation en admettant le fait de la non-garantie de livraison (pas d'accusé de réception),tolérable pour notre application vu que les images reçues ne faisaient pas apparaitre des pertes visibles.

Une multitude d'expériences et pour des durées de transmission qui ont atteint, pour certaines, une durée de 45 mn nous ont permis d'essayer de s'approcher de la réalité.

En outre, La robustesse de cette implémentation a été aussi testée en essayant de varier quelques paramètres essentiels tels que, la taille du tampon, la distance de transmission et la taille des images, les résultats obtenus ont été satisfaisants.

Il est a noté que certaines expériences ont été effectuées en transmettant des images à partir de notre camera braquée sur un autre écran diffusant une vidéo d'un film avec des scènes variables, ce qui donne une richesse des informations et donc valide notre implémentation pour une éventuelle utilisation commercial.

L'enregistrement continu des informations de l'ensemble des expériences a produit des données des images dont la taille n'a pas dépassé 2 GO sur le disque, ce qui constitue un autre point fort pour notre implémentation qui pourrait justifier son utilisation pour le développement d'autres applications sur cette base.

La dernière série des expériences ont prouvé sa fiabilité dans la comparaison de la taille des images reçues avec celle envoyées

Cette implémentation a prouvé La distance de 20 m est largement suffisante pour des applications réseau

Finalement, on peut conclure que les résultats obtenus, pour notre application, sont significatifs et démontre la robustesse de notre algorithme.

PERSPECTIVES

Nous avons, en final, éprouvé le besoin d'émettre un certain nombre de travail a faire pour des applications futurs comme perspective, on en peut citer :

- Utiliser le programme pour des applications de vidéo conférences ou surveillances et pour plusieurs récepteurs à la fois.
- Implémenter des algorithmes de traitement d'image ou de poursuite d'objets.
- Implémenter l'algorithme sur un système embarqué exemple Quad-rotor
- Des applications de communications au sein des entreprises en y ajoutant l'audio aux paquets contenant les données des images.
- Le fait que cette application à l'aspect temps-réel lui donne l'avantage d'être utilisé dans multitude de domaines.
- Implémentation des algorithmes de chiffrement et de déchiffrement pour la sécurité des données.

REFERENCES BIBLIOGRAPHIQUES

[1] http://www.adobe.com/fr/products/pdfs/dvprimer.pdf 15-08-2017

[2] http://support.logitech.com/fr_be/product/hd-webcam-c310/specs 29-08-2017

[3] RTP: A Transport Protocol for Real-Time Applications, Request For Comment (RFC) 3550, Jul. 2003.

[4] les protocols UDP et TCP Christian Bulfone licence MIASS

[5]http://be.soft.free.fr/reseaux/cours/UNICAST%20BROADCAST%20MULTICAST.pdf 26-07-2017

[6] CS4233 Network Programming Socket Programming – UDP Socket.

[7] http://abdelhamid-djeffal.net/web_documents/courssockets.pdf 11-07-2017

[8] Daniel Salles, " Éducation à l'image et aux médias : la liberté de la presse ", Centre de Ressources en éducation aux médias CREM, mars 2005.

[9] Daniel Salles, " Éducation à l'image et aux médias : la liberté de la presse ", Centre de Ressources en éducation aux médias CREM, mars 2005.

[10] http://www.axis.com/fr/products/video/about_networkvideo/compression.htm 23-05-2017

[11]http://help.adobe.com/fr_FR/mediaencoder/cs/using/WSb8e30982e628fbecc0e59e6131255b4dd2-8000.html

[12]http://perso.univrennes1.fr/pierre.nerzic/IN/Cours/S3P3%20%20Codages%20et%20compression.pdf

[13] http://www.spherevideo.com/Cyberbulletin/pdf/cyberbull_05_08.pdf 15-06-2017

[14] [EG0001] Edouard Gomez et Lionel Dufresne. Mémoire de Multimédia, MPEG4 et XviD.

[15] C. Perkins and E. Royer. Ad-hoc On-demand Distance Vector (AODV) routing. Workshop on Mobile Computing Systems and Applications, pp. 90-100, Feb. 1999.

[16] T. Clausen and P. Jacquet. Optimized Link State Routing protocol. Technical report,IETF (Internet Engineering Task Force), Request For Comment (RFC) 3626, Sep.2003.

[17] P. Jacquet, P. M uhlethaler, A. Qayyum, A. Laouiti, T. Clausen, and L. Viennot.Optimized link state routing protocol. IEEE INMIC, Dec. 2001.

[18] V. Kawadia and P. R. Kumar, A cautionary perspective on cross-layer design. IEEE Wireless Communications, vol. 12, no. 1, pp. 3-11, Feb. 2005.

[19] J. Huuskoa, J. Vehkaperaa, P. Amon, C. Lamy-Bergot, G. Panzad, J. Peltolaa, M. G. Martini, Cross-layer architecture for scalable video transmission in wireless network, Signal Processing : Image Communication, no. 22, pp. 317-330, 2007.

[20] Jean CHARLES AMEY, " codage des couleurs ", TPE 2001-2002.

[21] JPEG IMAGE COMPRESSION ON THE TEXAS INSTRUMENT VIDEO PROCESSING BOARD TMS320DM6437 ;Jaymin Jasoliya B.E., Electronics and Communication Engineering, South Gujarat University, India, 2006.

[22] D. A. Huffman, A method for the construction of minimum-redundancy codes, Proceedings of the Institute of audio Engineers, vol. 40, no. 9, pp. 1098-1101, 1952.

[23] Y. Q. Shi and H. Sun. Image and video compression for multimedia engineering : fundamentals, algorithms, and standards. CRC press, Boca raton, U.S.A, 2000.

[24] J. G. Proakis, Digital Communications. McGraw Hill International edition, second edition, 1989.

[25] D. A. Huffman, A method for the construction of minimum-redundancy codes, Proceedings of the Institute of audio Engineers, vol. 40, no. 9, pp. 1098-1101, 1952.

[26] S. C. Chan and K. L. Ho, A new two-dimensional cosine transform algorithm, IEEE Transactions on Signal Processing, vol. 39, no. 2, pp. 481-485, 1993.

[27] G. K. Wallace, The JPEG still picture compression standard, Communications of the Association for Computing Machinery (ACM), vol. 34 no. 4 pp. 30-44, Apr. 1991.

[28] W. B. Pennebaker and J. L. Mitchell, JPEG Still Image Data Compression Standard,Van Nostrand Reinhold, New York, NY, USA, 1993.

[29] ISO/IEC DIS 10918-1. Digital Compression and Coding of Continuous-Tone Still Image (JPEG). Technical report, CCITT Recommendation T.81, 1992.

[30] Image and Video Compression EE398A Bernd Girod InformationSystemsLaboratory Department of Electrical Engineering Stanford University.