Université de Québec en Outaouais
Département d'informatique et d'ingénierie

Rapport final

Projet Synthèse
INF4173
Hiver 2007

Application d'archivage et de recherche de séquences de vidéo surveillance

Document rédigé par
Nabila Solhi
Pierre Hugues Dessureault

Professeur superviseur
Madame Nadia Baaziz

Professeur coordonnateur
Monsieur Michal Iglewski

Table des matières

1- INTRODUCTION

Le but principal de l'installation des caméras de vidéosurveillance est la sécurité publique est privé des gens ainsi que la protection de leurs biens. Toutefois la mise en place d'un système de vidéosurveillance ne peut s'expliquer seulement qu'à cause de l'insécurité des individus. D'autres raisons plus importantes rentrent en jeu.
La mise en place d'un système de vidéosurveillance permet aussi la prévention des incidents ainsi que l'augmentation de la rapidité d'intervention. A titre d'exemple, dans la prévention du suicide ou les accidents qui peuvent se produire sur les routes. . Bref la vidéosurveillance permet d'assurer et de préserver la sûreté à une échelle convenable.

Les anciennes caméras de surveillance avaient des images de basses qualités, sans possibilité de zoomer, ni de changer l'angle de vue. Récemment, avec l'avènement de La vidéo intelligente, plusieurs fonctionnalités ont été intégrées.
Ces fonctionnalités consistent à convertir des données vidéo brutes en informations et renseignements exploitables par l'analyse grâce à des algorithmes qui analysent en temps réel les séquences vidéo filmées par des caméras. Ces algorithmes intègrent des capacités de calcul très importantes utilisables pour le traitement vidéo en temps réel sur plusieurs applications.

Depuis quelques années, la vidéo intelligente intègre la détection de mouvement : cette fonctionnalité permet de cueillir et d'enregistrer les images dans des cas spécifiques où un objet en mouvement entre dans une zone d'intérêt. Ce type de fonction permet le stockage d'images seulement en fonction de l'arrivée de certains types d'événements, en vue d'une analyse ultérieure. De nouvelles fonctions de vidéo intelligente commencent à s'étendre aujourd'hui ces possibilités d'analyse à la source. En ne sortant que les données judicieuses, elles viennent particulièrement facilité l'analyse de données vidéo cruciales.

Par ailleurs, la quantité de mouvements et des gestes faits même en une seule minute sont innombrables et par la suite difficile à analyser. Les utilisateurs n'ont donc pas assez de temps pour consulter l'ensemble des informations disponibles. Il est donc indispensable de développer des techniques d'organisation et de recherche de documents multimédias permettant de faciliter l'accès à ces informations.

2- OBJECTIFS DU PROJET

L'objectif principal de ce projet est de développer une application interactive entre un client et un serveur qui permet une aide à l'archivage, la recherche et la localisation des séquences de vidéo surveillance au travers d'images caractéristiques et de descriptifs se trouvant dans une base de données.

Différentes étapes intermédiaires sont nécessaires pour en arriver là, parmi lesquelles :

- Établir une connexion JDBC [1]entre l'interface java et la base de données;
- Trouver le(s) type(s) de donnée adéquat pour stocker les données multimédias dans une base de données;
- Chercher et comprendre les classes de Java qui manipulent les images;
- Trouver les algorithmes de compression d'images;
- Fournir une interface java facile à utiliser;
- Stocker les images rapidement et efficacement;
- Pouvoir chercher les images selon une date précise (avec l'heure), une date sans préciser l'heure ou/et entre deux dates ;
- Coder et comprendre l'algorithme de marquage des images (*watermarking*);
- Coder et comprendre l'algorithme de l'extraction du marquage.

[1] Java DataBase Connectivity; une API qui permet la connexion entre une base de données et une interface Java.

3- LES TECHNOLOGIES UTILISÉES

- Le langage Java (1.6) : possède une API [2] extrêmement riche. C'est en l'utilisant que le programmeur va pouvoir accéder à toutes les ressources de la machine, et celles du réseau Internet. Cette vaste librairie est divisée en packages (des ensembles de classes).
  En effet, pour gérer les images en Java, Il y a maintes façons de le faire, plusieurs modèles de gestion des images coexistent et peuvent semer la confusion lorsque l'on démarre et c'est ainsi qu'on trouve ici et là des exemples parfois plus compliqués que nécessaire. Avant de commencer, il est bon d'avoir les idées claires sur les modèles existants

  - **Le modèle push :** C'est le premier modèle de gestion d'images. Basé sur la classe unique ***java.awt.Image.***
    Les images sont chargées par la classe *java.awt.Toolkit* et leurs données sont traitées par une instance de *ImageConsumer* au fur et à mesure qu'elles arrivent.

  - **Le modèle en mode immédiat :** Basé sur la classe *BufferdImage* et ses dérivées et qui hérite de la classe *java.awt.Image.*
    Quand l'image est crée pour ce modèle, toutes les données sont disponibles dans *BufferedImage* d'où le terme mode immédiat.

  - **Mode pull :** Utile pour optimiser les traitements lourds sur des images de taille importante.
    Compatible avec les autres mais nécessite les classes de Java Advanced Imaging API(JAI).

    Dans notre projet nous avons utilisé le mode immédiat car c'est le plus simple à gérer.

- NetBeans IDE (version 5.5) : Cet IDE[3] contient des outils qui sont faciles à utiliser pour le développement des interfaces java.

- SQL serveur 2005 : Microsoft SQL Server 2005 étend les performances, la fiabilité, la disponibilité, les possibilités de programmation et la simplicité d'utilisation de SQL Server 2000. SQL Server 2005 comporte plusieurs nouvelles fonctionnalités qui en font une excellente plateforme de gestion de base de données pour des applications multimédias.

[2] Application Programming Interface
[3] IDE est un sigle, qui signifie : (Integrated Development Environment) environnement de développement logiciel, en informatique.

La dernière version de SQL Server contient un type de données s'appelant Image pour stocker les données binaires de longueur variable occupant de 0 à 2^31 - 1 (2 147 483 647) octets. Il offre aussi la possibilité de manipuler les images en Java grâce à un package *java.sql* qui fournit l'API pour accéder et traiter des données stockées dans un point d 'émission de données (habituellement une base de données relationnelle) employant le langage de programmation Java. Cet API inclut un cadre par lequel différents conducteurs peuvent être installés dynamiquement pour accéder à différents points d'émission. L'utilisation en est très simple, mais quelques bases en langage SQL (Structured Query Language) sont requises.

- JDBC (Java DataBase Connectivity) est une API qui est fournie avec Java permettant de se connecter à des bases de données.

- JAI 1.2 (Java Advanced Imaging) est une API qui fournit un ensemble d'interfaces orientées objectivement et qui permet à des images d'être manoeuvrées facilement dans des applications et des applet de Java.

- JMF (Java media framework). est une interface de programmation d'application pour l'acoustique, la vidéo et autre outils des médias basés-temps dans des applications et des applet de Java.

4- PRÉSENTATION DU SYSTÈME COMPLET

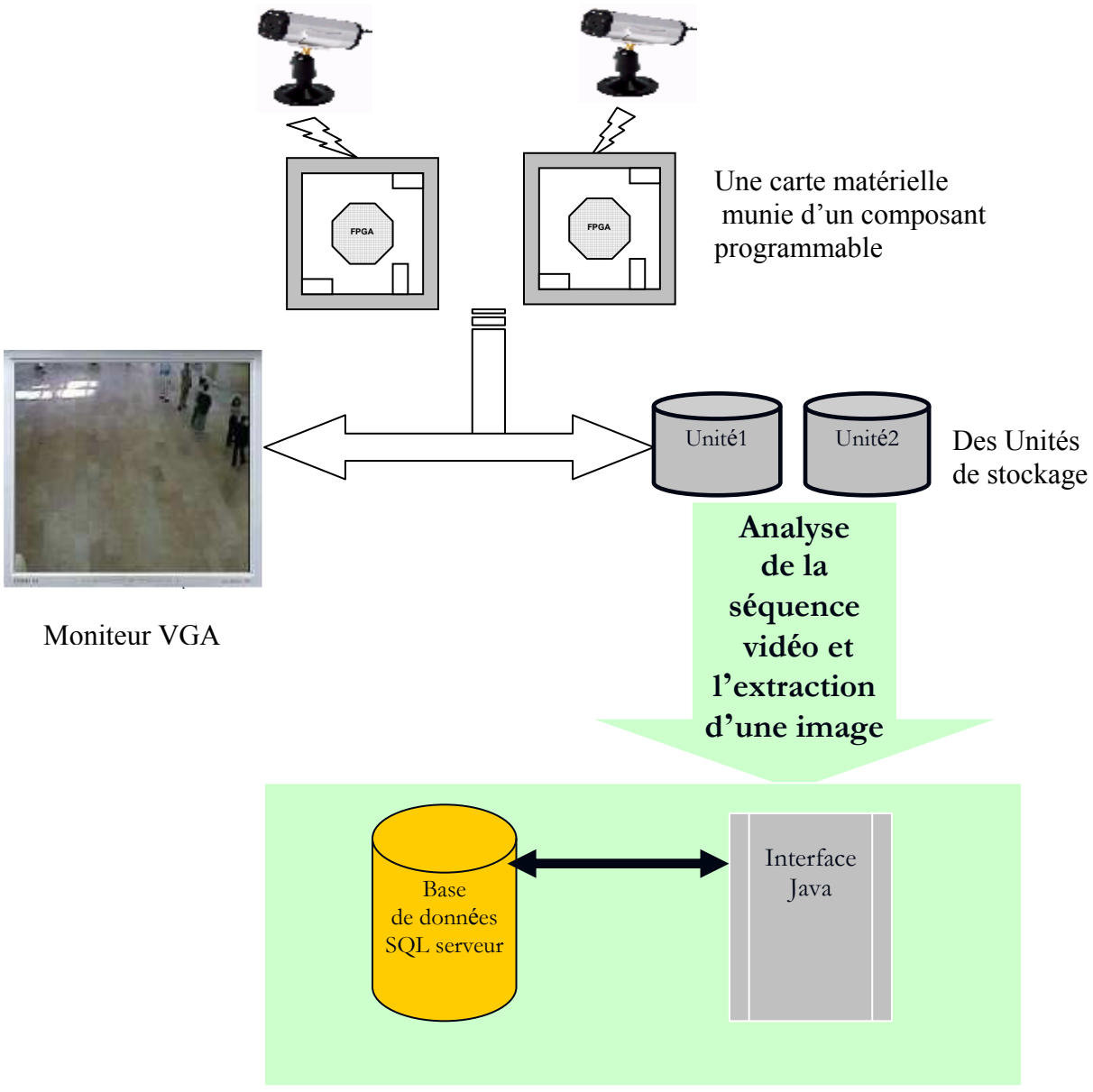


Figure 1: Système complet

Le système au complet est divisé en plusieurs modules :

Module d'acquisition : constitué d'une ou plusieurs caméras numériques capables d'envoyer pixel après pixel les données de chaque image de la séquence vidéo à une carte munie du composant programmable matériel de type FPGA[4]. Les données reçues sont temporairement stockées dans une mémoire RAM disponible sur la carte matérielle.
Le composant FPGA contient un encodeur/décodeur pour l'authentification des données enregistrées en mémoire RAM.

Le module d'affichage : ce module lit les données stockées en mémoire RAM et les affiche vers un moniteur VGA.
Le composant matériel est mis à profit afin de permettre la conversion des données numériques en format analogique requis pour l'affichage.

Le module de stockage : constitué de plusieurs unités de stockages (disques durs, bandes magnétiques…), ces dernières reçoivent les données qui proviennent du module d'acquisition.

Le module d'extraction : permet l'analyse des séquences vidéo et l'extraction des images selon une description (date, heure, localisation…).
Ce module n'a toujours pas été développé. Quelqu'un peut l'adopter comme projet dans les sessions qui viennent.

Le module d'archivage et de recherche : ce module a été développé dans le cadre de ce projet afin de réorganiser les données images vidéo issues du module d'extraction, et surtout faciliter l'accès à ces données.
Ce module contient trois sous modules :

- Archivage
- Recherche
- Traitement des images (authentification et compression)

[4] Field programmable gate array; est un élément fonctionnel de semi-conducteur et/ou un IC programmable avec des Configware.

5- ARCHITECTURE DE L'APPLICATION

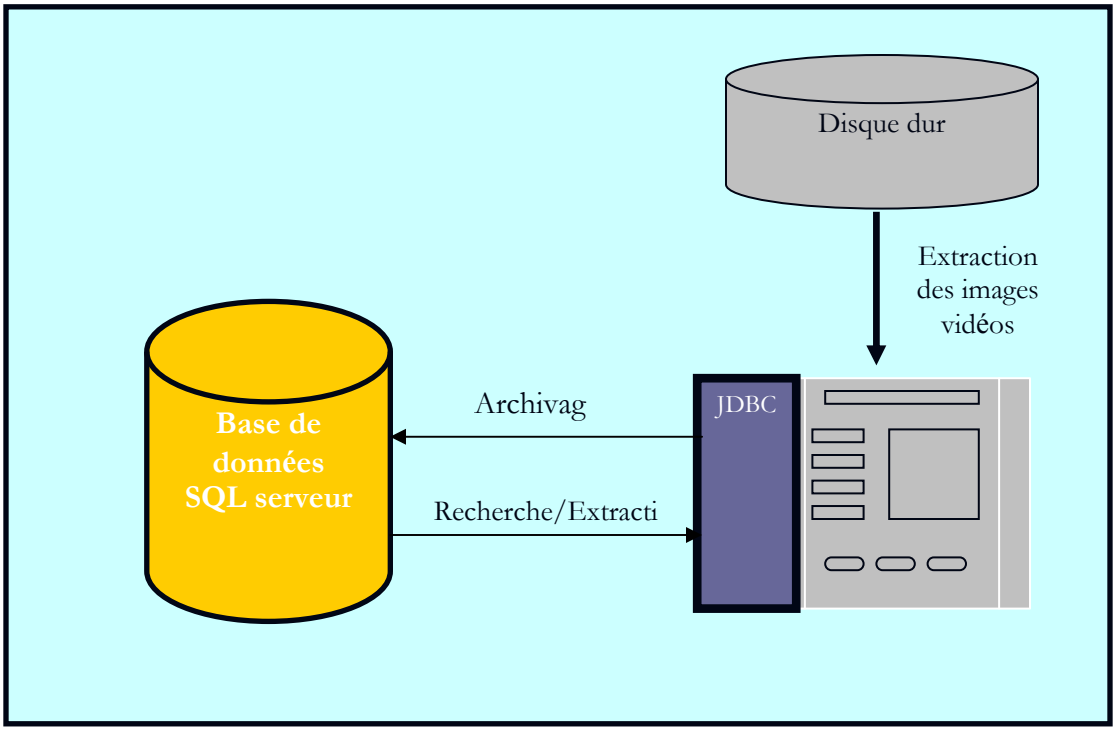


Figure 2: Architecture de l'application

Notre application est constituée d'une interface Java et une base de données SQL serveur qui permet l'archivage et la recherche des images caractéristiques résultantes du module d'extraction.
Un pilote JDBC est indispensable pour la connexion entre l'interface java et la base de données.

6- BESOINS FONCTIONNELS ET NON FONCTIONNELS DU PROJET

6.1 Les besoins fonctionnels

Cette application comprend plusieurs fonctionnalités et deux types d'utilisateurs (l'administrateur et l'usager).

Il y a 4 blocs de fonctionnalités :

1. Gestion des profils;
2. L'archivage;
3. La recherche qui inclut la visualisation des images vidéo, visualisation de la séquence vidéo, et l'affichage de la description qui correspond a chaque image;
4. Traitement des images inclut le marquage des images (watermarking), la détection de la marque et la compression des images sous le format JPEG2000[5].

[5] JPEG 2000 ou ISO/CEI 15444-1 est une norme commune de l'ISO et de l'UIT-T. C'est un standard de compression d'images défini par le comité Joint Photographic Experts Group. JPEG 2000 est capable de travailler avec ou sans pertes, utilisant une transformation en ondelettes (méthode d'analyse mathématique du signal). En compression irréversible, JPEG 2000 est plus performante que la méthode de compression JPEG ISO/CEI 10918-1 (le JPEG classique). On obtient donc des fichiers d'un poids inférieur pour une qualité d'image égale. De plus, les contours nets et contrastés sont mieux rendus.

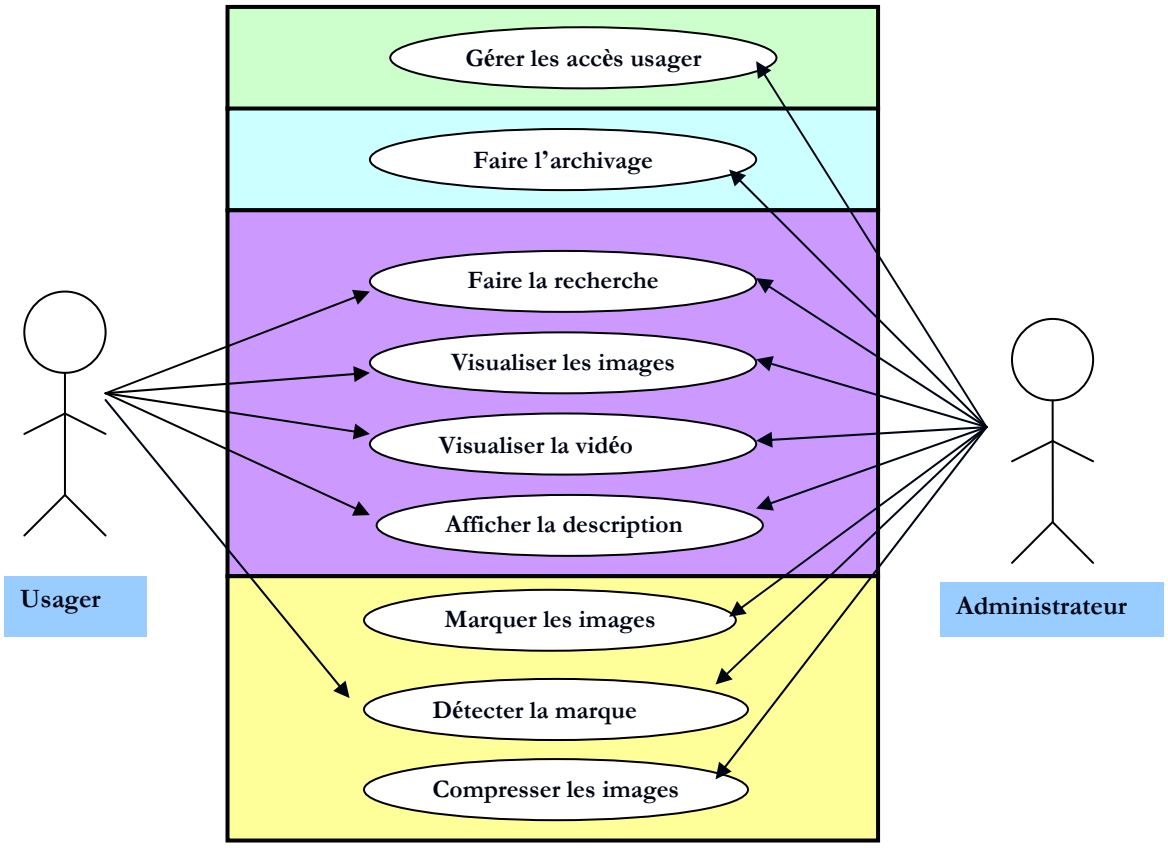


Figure 3: Fonctionnalités du système

L'administrateur a accès à toutes ces fonctionnalités, tandis que l'usager n'a accès qu'à la recherche, la visualisation des images, visualisation de la séquence vidéo, l'affichage de la description qui correspond à chaque image vidéo et la détection de type de marquage.

*6.*2 Les besoins non fonctionnels

Cette application sera très conviviale. Nous voulons que les utilisateurs aient du plaisir en utilisant notre application et qu'ils ne soient pas dépaysés. Vous allez voir que notre application sera très facile d'accès.

De plus, cette application sera performante. Grâce à notre système d'onglet, pas besoin d'ouvrir plusieurs écrans pour utiliser toutes les fonctionnalités de l'application. Aussi, l'usager aura de la facilité à retrouver les résultats de ses recherches.

Un critère important de l'application est l'aspect sécurité. Grâce à un nom d'usager et un mot de passe unique pour chaque utilisateur, nous

ajoutons de la protection si quelqu'un voudrait faire de la modification d'images ou bien accéder à des données secrètes.

Pour se rendre à un aspect performant et convivial, il y a un aspect ergonomique qui est très important. L'application contient de gros boutons qui sont faciles d'accès pour tous, l'application est facile à lire, noir sur fond gris, et aussi, les images sont de grandes tailles pour faciliter leur visionnement.

Tout cela rend l'application facile d'utilisation pour tous.

7- LES ÉTAPES EFFECTUÉES

7.1 La base de données

7.1.1 LE SCHÉMA RELATIONNEL DE LA BASE DE DONNÉES

Notre Base de données contient deux tables :

- **Image_Table**(Date, Localisation, Type_Image, Description, Img, Signature, Id_Sequence).
- **Usager**(Nom_Usager, mot_Passe, Restriction).

Image_Table
Date
Localisation
Type_Image
Description
Img
Signature
<1>Id_sequence

Usager
Nom_Usager
Mot_Passe
Restriction

Exemples:

- **Image_Table**(2002-20-20 23:30:20, 2_étage3, .jpeg, image bateaux, <Données binaires> ,1, C://image/video.avi).
- **Usager**(Nadia,uqo, Administrateur).

La table Image_Table permet le stockage des images ainsi que la description qui correspond à chaque image vidéo (type, description, signature et Date…)
Pour gérer les images et pouvoir les manipuler, un champ de type **image** est obligatoire dans notre table «Image_Table » qui les gère.

La table Usager permet la gestion des usagers. Trois champs ont été crées pour le stockage; le nom de l'usager, son mot de passe et la restriction qui représente le type de l'usager (administrateur ou utilisateur).

7.1.2 CRÉATION DES TABLES

Voici l'ordre SQL de la table Image_Table :

```
CREATE TABLE [dbo].[Image_Table](
      [Date] [datetime] NOT NULL,
      [Localisation] [varchar](20) COLLATE French_CI_AS NOT NULL,
      [Type_Image] [varchar](20) COLLATE French_CI_AS NOT NULL,
      [Description] [varchar](max) COLLATE French_CI_AS NULL,
      [Img] [image] NULL,
      [Signature] [int] NOT NULL,
      [Id_Sequence] [int] NOT NULL,
 CONSTRAINT [PK_Image_Table] PRIMARY KEY CLUSTERED
(
      [Date] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
```

Voici l'ordre SQL de la table Usager :

```
CREATE TABLE [dbo].[Usager](
      [Id_Usager] [nvarchar](20) COLLATE French_CI_AS NOT NULL,
      [mot_Passe] [int] NOT NULL,
      [Restriction] [nvarchar](20) COLLATE French_CI_AS NOT NULL,
 CONSTRAINT [PK_Usager] PRIMARY KEY CLUSTERED
(
      [Id_Usager] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

7.2 Établir la connexion

7.2.1 LE PILOTE JDBC

Définition

JDBC (Java DataBase Connectivity) est une API fournie avec Java permettant de se connecter à des bases de données, c'est à dire que JDBC constitue un ensemble de classes permettant de développer des applications capables de se connecter à des serveurs de base de données(SGBD).
JDBC bénéficie des avantages de java, dont la portabilité du code, ce qui lui vaut en plus d'être indépendant de la base de données et d'être indépendant de la plate-forme sur laquelle elle s'exécute.
L'API JDBC est représentée par le package **java.sql** dont les principales interfaces sont les suivantes :

- **Java.sql.DriverManager** : prend en charge le chargement des pilotes et permet de créer de nouvelles connexions à des bases de données. la liste principale des pilotes JDBC recensés du système est tenues à jour.
- **Java.sql.Connection** : représente une connexion à une base de données.
- **Java.sql.Statement** :c'est une classe que l'application emploie pour transmettre des instructions à la base. Elle représente une requête SQL.
- **Java.sql.ResultSet** : cette classe symbolise un ensemble de résultats dans la base de données et autorise l'accès aux résultats d'une requête rangée par rangée. Pendant le traitement de la requête, un ResultSet conserve un pointeur vers la rangée manipulée. L'application se déplace séquentiellement dans l'ensemble des résultats.

Architecture du driver JDBC

Dans un système client/serveur, l'accès aux bases de données avec JDBC peut s'effectuer selon un modèle à deux couches ou bien un modèle à trois couches.
Pour notre application, nous avons opté pour un modèle à deux couches, où l'application java est intimement liée avec notre base de données SQL par un pilote JDBC (com.microsoft.sqlserver.jdbc.SQLServerDriver). Donc les instructions SQL sont directement envoyées à la base de données, cette dernière renvoyant les résultats par un biais tout aussi direct.

Chargement du Pilote

Pour se connecter à une base de données, il est essentiel de charger dans un premier temps le pilote de la base de données à laquelle on désire se connecter grâce a un appel au *DriverManager.*

L'instruction suivante charge le pilote.

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

7.2.2 CONNEXION ENTRE LA BASE DE DONNÉES ET LE DRIVER JDBC

Une fois que le pilote est enregistré, il faut encore ouvrir une connexion vers la base de données grâce à un code analogue à l'exemple suivant :

Connection conn;
String url="jdbc :sqlserver://hostname :port;database=dbName"+"use","password"
conn = DriverManager.getConnection(url);

Le gestionnaire de pilote essaie alors de trouver un pilote pouvant utiliser le protocole spécifié dans le URL de la base de données.
Ensuite garce à la méthode *getConnection* de l'objet *DriverManger* on indique la base de données à charger à l'aide de son URL.

7.3 LES PROTOTYPES D'INTERFACE JAVA

Afin de fournir une application conviviale, nous avons conçu une interface claire et facile à utiliser.

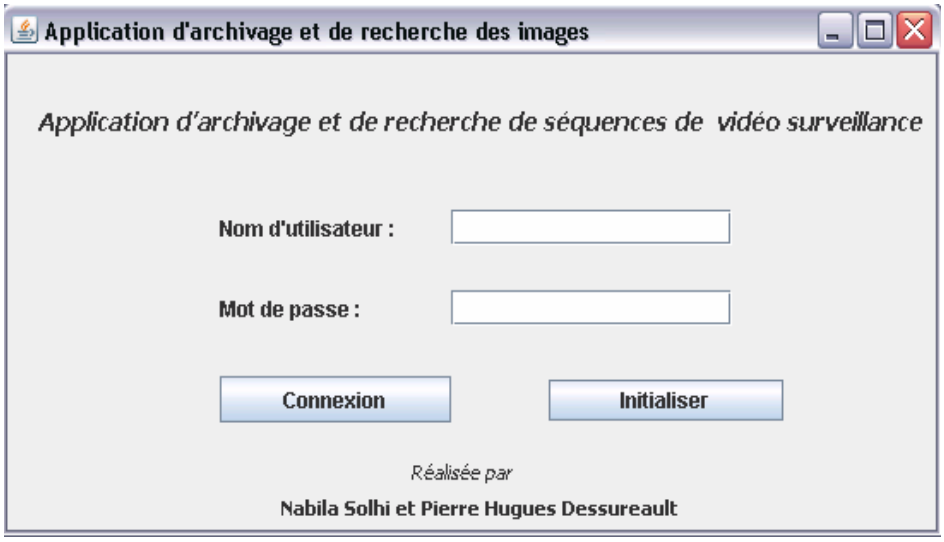
Notre interface englobe les fonctionnalités suivantes :

- Un écran de connexion
- L'insertion des images (Archivage)
- La recherche des images selon la date et l'heure
- Traitement des images (marquage, compression et détection de la marque)

Les prototypes suivants vont vous donner plus de détails sur notre interface.

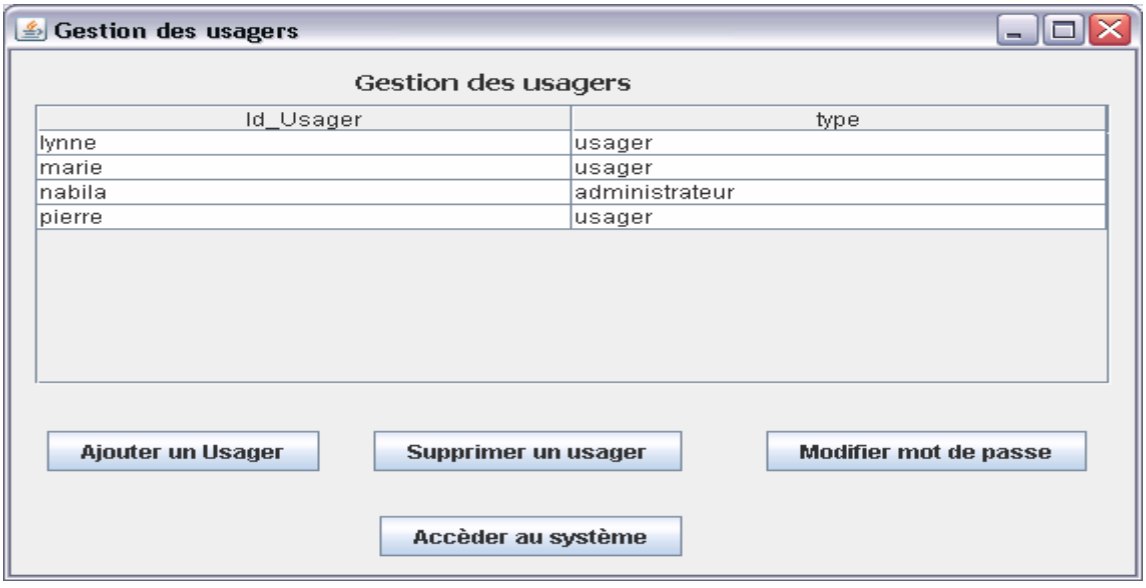
Écran de connexion

Dans un premier lieu nous avons crée un écran de connexion. Cette section contient deux champs; un pour entrer le nom d'usager et l'autre pour entrer le mot de passe.



Gestion des usagers

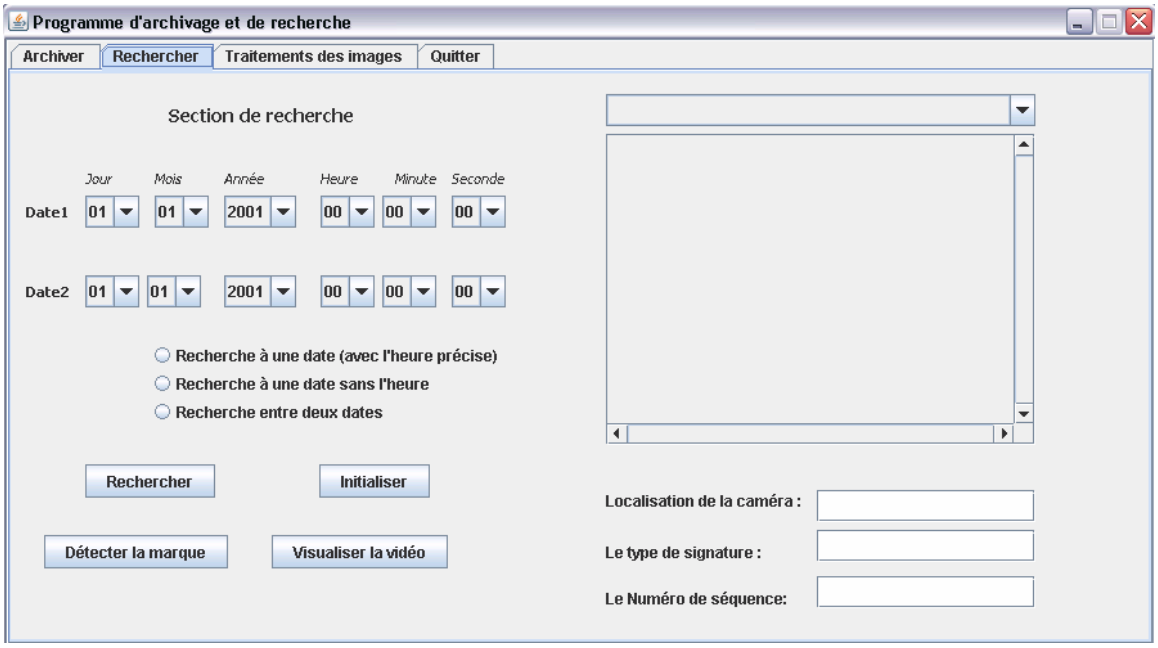
L'administrateur est le seul à avoir accès à la gestion des usagers. Il pourra ajouter un utilisateur en lui attribuant un nom et un mot de passe. Il peut aussi supprimer un utilisateur et modifier son mot de passe.



Section de recherche

Dans cette section, nous avons crée des menus déroulants pour choisir la date et l'heure de recherche.
Nous avons aussi crée trois boutons '*radio'* afin de choisir l'intervalle de temps, autrement, dés qu'un utilisateur choisit une date, il a trois options pour effectuer sa recherche. Soit :

Recherche à une date (avec l'heure précise) : le programme affiche une seule image dans le *Jpanel* a droite.
Recherche à une date sans l'heure : le programme affiche toutes les images qui ont été enregistrées durant la date choisie.
Recherche entre deux dates : dans ce cas le programme va afficher toutes les images qui ont été stockées entre les deux dates choisies.

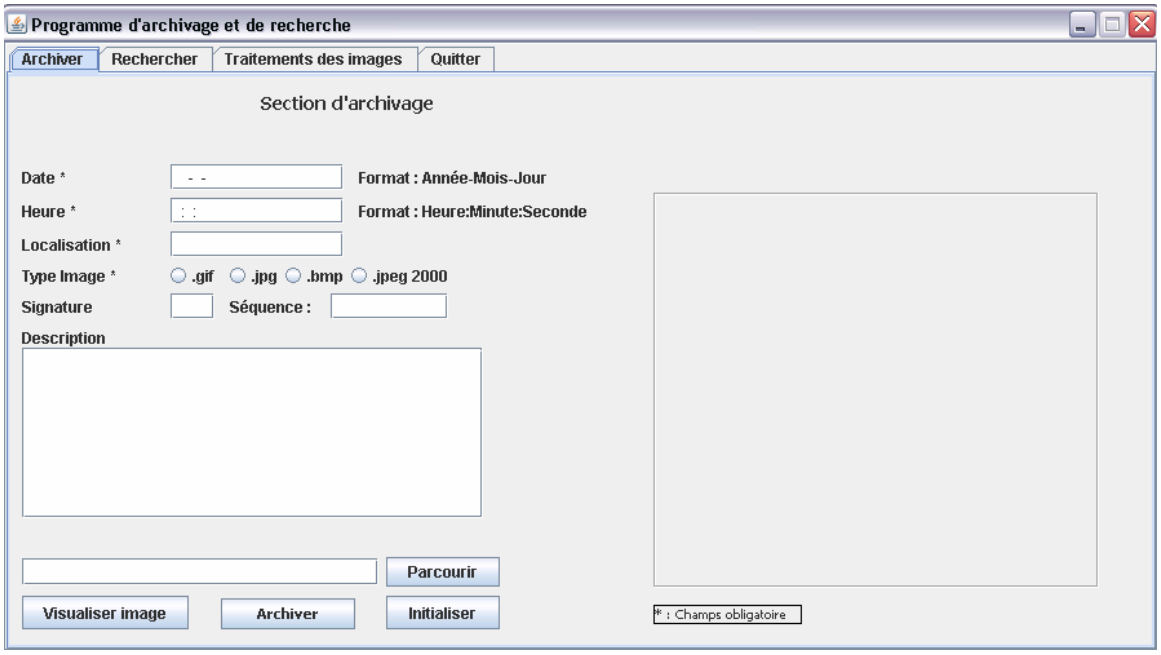


Section d'archivage

Dans cette section nous avons crée plusieurs champs et boutons :

Date : pour entrer la date de l'image sous le format (année, mois, jour)
Heure : pour entrer l'heure sous le format (heure : minute : seconde)
Localisation : pour décrire la séquence vidéo d'où provient l'image en question.

Type Image : pour cela nous avons crée des boutons '*radio*' qui représentent les quatre type d'images à insérer : gif[6], jpg, bmp[7], jpeg2000).
La signature : pour décrire le type de marquage.
Description : pour insérer plus d'informations sur l'image concernée.
Parcourir : ce bouton nous permet d'aller chercher l'image sur le disque dur.
Ouvrir image : permet de visualiser l'image dans le *Jpanel* à droite avant de la stocker dans la base de données.
Archiver : une fois tous les champs sont remplis, on clique sur ce bouton afin de stocker toutes les informations dans la base de données.



[6] Graphics Interchange Format, un format de données informatique d'image numérique.
[7] BMP, acronyme de *Bitmap*, est un format d'image numérique ouvert développé par Microsoft et IBM. C'est un des formats d'images les plus simples à développer et à utiliser pour programmer. Il est lisible par quasiment tous les visualiseurs et éditeurs d'images.

7.4 L'EXÉCUTION DES REQUÊTES SQL

Pour exécuter une requête SQL, il s'agit dans un premier temps de créer un objet *Statement*, pouvant être obtenu à partir de l'objet *Connection*.
Un objet *ResultSet* permettra de récupérer les données en provenance de l'objet *Statement*.

```
Statement stat = conn.createStatement();
ResultSet rs= stat.executeQuery(query);
```

La méthode *executeQuery* permet de récupérer les enregistrements sélectionnés dans l'objet *rs* de type *ResultSet*.

7.5 INSERTION DES IMAGES DANS LA BASE DE DONNÉES

Pour faire l'insertion des images et des données dans la base de données, nous avons utilisé dans notre table un type qui se nomme Image. Le type Image peut contenir n'importe quoi en format de byte, soit des images, du son et des séquences vidéo. Il est important de noter que la grandeur des objets qui sont possibles d'insérer dans la table peut être jusqu'à une grandeur maximale de 2 giga-octets.

Pour insérer l'image avec java dans la table, il suffit d'utiliser une requête SQL **insert** avec les données voulues.

```
String query1=("insert into Image_Table values (?,?,?,?,?,?)");
```

Chaque paramétre de notre requête préparée est spécifié par un **'?'**.
Dans notre code on a six points d'interrogations, ces derniers représentent le nombre de colonnes qui sont dans notre table 'Image_Table'.

Avant d'exécuter la requête 'query1', nous devons affecter les valeurs aux variables avec la méthode setXXX.

```
ps.setString(1, date + " " + heure);
 ps.setString(2,NomFichier);
 ps.setString(3,type);
 ps.setString(4,Desc);
 ps.setBinaryStream(5, fis, (int) file.length());
ps.setString(6,"0");
```

Le premier argument correspond à la variable que nous voulons définir. La position **1** représente le premier **'?'**. Le second argument est la valeur que nous voulons affecter à la variable.

Concernant la partie spécifique à l'insertion de l'image, on s'appuie sur des composants gérant un flux binaire. On commence donc par définir le fichier à consulter et on lui associe un flux entrant. Le flux est ensuite affecté à la colonne img de la table à l'aide de la méthode setBinaryStream dont le second et troisième paramètre correspond respectivement au flux entrant et à la taille à extraire.
Voici le code :

```
File file = new File(Fichier);
fis = new FileInputStream(file);

ps.setBinaryStream(5, fis, (int) file.length());
```

Par la suite, il nous reste à exécuter notre requête à l'aide des méthodes *prepareStatement() et executeUpdate().*

```
ps = conn.prepareStatement(query1);
ps.executeUpdate();
```

7.6 LA RECHERCHE DANS LA BASE DE DONNÉES

La phase de recherche a pour objectif de satisfaire les besoins d'un utilisateur en extrayant efficacement des données en réponse à une requête. Cette fonctionnalité est bien sûr dépendante de la qualité de l'organisation de la base de données.

Pour exécuter une commande SQL sous Java, il faut commencer par créer un objet *Statement* qui enverra des requêtes SQL à la base de données.

```
Statement stat = conn.createStatement();
```

Ensuite placer l'instruction à exécuter dans une chaîne dont voici le code :

```
String query=("select * from Image_Table where Date = '" + laDate + " " + lHeure
+ "'");
```

Appelons alors la méthode *executeQuery()* de la classe *Statement :*

```
rs = stat.executeQuery(query)
```

où *rs* représente Un objet *ResultSet. Ce dernier* permet de récupérer les données en provenance de l'objet *Statement*. Tandis que la méthode *executeQuery*() permet de récupérer les enregistrements sélectionnés dans un objet de type *ResultSet*.

Maintenant il reste le chargement de l'image dans fichier de type binaire, cela se fait avec la méthode statique *read* de la classe *ImageIO*. Cette dernière choisit un lecteur approprié, en fonction du type de fichier.

Dans le cas de notre projet le type du fichier est byte car on manipule des données binaires.

```
fileBytes = rs.getBytes(5);

BufferedImage monImage=ImageIO.read(img);
```

La méthode getBytes renvoie le nombre d'octets disponibles dans le champ cinq de notre base de données. Sachant que ce champ représente la colonne où les images sont stockées au niveau de la base de données.

Nous avons crée un nouveau fichier (image) sur le disque dur pour pouvoir y stocker les images provenant de la base de données.

```
OutputStream targetFile=  new FileOutputStream("c://image//new.gif");
```

Il ne nous reste par la suite qu'a d'écrire le contenu du fichier et de fermer le fichier.

```
targetFile.write(fileBytes);
targetFile.close();
```

Après que cela soit fait, nous cherchions à faire l'affichage de l'image dans le panel. Nous avons trouvé un moyen rapide. Nous lisons l'image avec un BufferedImage tel que vu plus haut.

```
File img = new File("c:/image/new.gif");
BufferedImage monImage=ImageIO.read(img);
```

Il ne nous reste qu'à modifier notre image pour qu'elle soit de la grandeur de notre panel et d'y faire son affichage. Étant donné que nous avons plusieurs résultats retournés par nos requêtes, nous utilisons un tableau pour y stocker les images et y faire l'affichage par la suite.

```
Image imageModifie =
Image.getScaledInstance(PanelRech.getWidth(),PanelRech.getHeight(),m
onImage.OPAQUE);
```

tableau[i] = imageModifie;

Notre affichage dans le panel se fait de la façon suivante. Les images sont affichées une après les autres dans le panel, et un scrollbar est présent pour descendre dans la liste d'image. De plus, il est possible de sélectionner une image parmi la liste grâce au combobox qui contient les dates de chacune des photos affichées. Nous avons eu de la difficulté à afficher les images une après l'autre, mais grâce à deux fonctions, soit repaint et revalidate, tout fonctionne parfaitement. Aussi, pour afficher une image dans un panel, il suffit de la transformer en icône, de l'ajouter à un label et d'ajouter le label au panel. Voici le code qui démontre tout cela.

```
ImageIcon icone = new ImageIcon(tableau[j]);
JLabel image = new JLabel(icone);
image.setBounds(0,0,PanelRech.getWidth(),PanelRech.getHeight());
PanelRech.add(image);
PanelRech.revalidate();
PanelRech.repaint();
```

7.7 TRAITEMENT DES IMAGES

7.7.1 LE WATERMARKING (LE MARQUAGE)

Les algorithmes de watermarking ont connu des évolutions importantes pour répondre à l'objectif de sécurité. L'idée générale consiste à introduire un biais dans la répartition statistique des données numériques des œuvres. Ce biais statistique sert à coder l'information que l'on veut dissimuler, tout en étant très peu visible. Les méthodes d'introduction du biais statistique sont variées et dépendent notamment de la nature des œuvres : images, flux audio ou vidéo.

Les performances d'un marquage sont appréciées sous les quatre critères suivants : Imperceptibilité, Robustesse, Complexité, Capacité

Imperceptibilité : Le tatouage doit être invisible à l'œil humain. Prenons deux exemples très simples pour souligner son importance. Imaginons une image en niveau de gris avec une large zone uniforme. Si l'on rajoute un peu de bruit, ceci va immédiatement se voir dans cette zone. Il faut plutôt mettre le tatouage dans des zones de fort gradient (contour de formes, zones fortement texturées,…) où l'œil est moins sensible.

Robustesse : On parle de robustesse pour définir la résistance du tatouage face à des transformations de l'image tatouée. Ces transformations peuvent être de type géométrique (rotation, zoom, découpage …). Elles peuvent modifier certaines caractéristiques de

l'image (histogramme des couleurs, saturation…). Il peut aussi s'agir de tous les types de dégradations fréquentielles de l'image (compression avec pertes, filtres passe haut ou passe bas, impression de l'image, etc.…). Ces attaques sont dénommées « attaques aveugles », car le pirate agit sans réellement savoir ce qu'il fait.

Complexité : Dans la pratique, la plupart des opérations de tatouage doivent pouvoir s'effectuer en temps réel. Ceci implique une contrainte supplémentaire sur la complexité des opérations utilisées pour le marquage et pour la détection.

Capacité : La capacité d'un système de tatouage numérique désigne le rapport : « nombre de données » à dissimuler sur « taille du document hôte ». Dans le cas du marquage, la capacité se limite souvent à 1 bit. De façon générale, plus la capacité est faible, plus la robustesse et l'imperceptibilité sont fortes.

L'algorithme de watermarking

Développé récemment en 2006 par des chercheurs coréens; Sang-Kwang Lee[8], Young-Ho Suh[8], et Yo-Sung Ho[9]. Cet algorithme nous propose une nouvelle technique réversible d'authentification pour les images, qui peuvent inclure une quantité significative de données tout en gardant la qualité visuelle élevée (les valeurs de PSNR[10] des images filigranées sont toujours plus hautes que le DB[11] 48.13).
Le watermarking incluant le procédé de l'arrangement proposé d'authentification est montré dans Figure 3.

[8] Electronics and Telecommunications Research Institute (ETRI) 161 Gajeong-Dong, Yuseung-Gu, Daejeon 305-700, KOREA.

[9] Gwangju Institute of Science and Technology (GIST) 1Oryong-Dong, Bul-Gu, Gwangju 500-712, KOREA {sklee,syh}@etri.re.kr, hoyo@gist.ac.kr.

[10] (Peaksignal-to-noise-ratio) Le rapport signal sur bruit désigne la qualité d'une transmission d'information par rapport aux parasites. Il mesure la qualité de reconstruction dans la compression etc. d'image.

[11] Le décibel (DB) est une unité de mesure logarithmique.

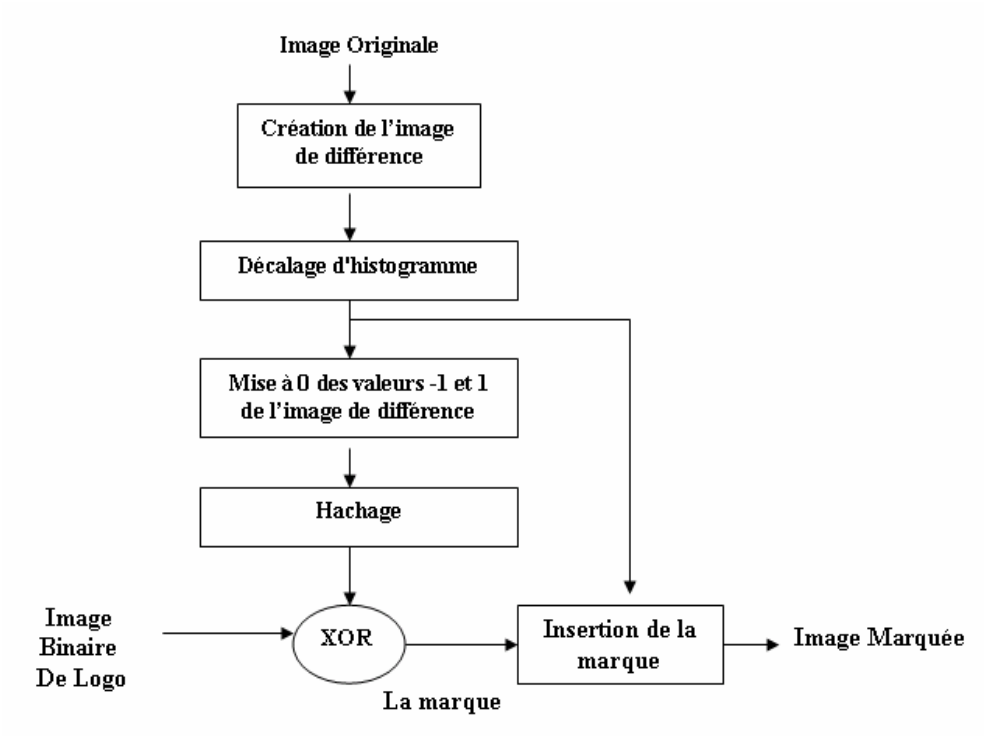


Figure 3: l'organigramme de watermarking

A partir d'une image originale I (i, j) de taille M*N pixels, nous construisons ce qu'on appelle une image de différence D (i, j) de taille M*N/2.

Pour 0<= i <= M et 0<=J<=N/2-1.

$$D(i, j) = I(i, 2j+1) - I(i, 2j)$$

Où I (i, 2j+1) et I (i, 2j) représentent la ligne impaire et la ligne paire de l'image pixel, respectivement.

Pour le filigrane, nous vidons les casiers d'histogramme de -2 et de 2 en décalant quelques valeurs de Pixel dans l'image de différence. Si la valeur de l'image de différence est supérieure ou égal à 2, nous additionnons un à la ligne impaire de l'image pixel. Si la valeur de différence est inférieur ou égal a -2, nous soustrayons un de la ligne paire de l'image pixel[12].
Donc l'image de différence tilde sera représentée comme suit :

[12] Le pixel est la plus petite division d'une image

$$\widetilde{D}(i,j) = \widetilde{I}(i,2j+1) - I(i,2j)$$

Où,

$$\widetilde{I}(i,2j+1) = \begin{cases} I(i,2j+1)+1 & \text{if } D(i,j) \geq 2 \\ I(i,2j+1)-1 & \text{if } D(i,j) \leq -2 \\ I(i,2j+1) & \text{otherwise} \end{cases}$$

Cet algorithme utilise la fonction d'hachage MD5 (Message Digest 5) qui est une fonction de hachage cryptographique qui permet d'obtenir pour chaque message une empreinte numérique (une séquence de 128 bits ou 32 caractères en notation hexadécimale). On conjoncture cette dernière pour produire deux messages ayant le même sommaire de message.

$$\mathbf{D_s} = \{D_s(i,j) : 0 \leq i \leq M-1; 0 \leq j \leq \tfrac{N}{2} - 1\}$$
$$\widetilde{\mathbf{D}}_\mathbf{s} = \{\widetilde{D}_s(i,j) : 0 \leq i \leq M-1; 0 \leq j \leq \tfrac{N}{2} - 1\}$$

Puis, nous calculons les informations parasites de la fonction d'hachage A(L) de longueur 128 comme suit :

$$H(K, M, N, \widetilde{\mathbf{D}}_\mathbf{s}) = \{A(l) : 0 \leq l \leq 127\}$$

Où,

K[13] est une clef d'utilisateur se composant de quelques bits.
Afin de produire un watermark W (m, n), nous combinons la fonction d'hachage A (L) avec l'image logo B (m, n) de taille P*Q en utilisant l'opération XOR.

$$W(m,n) = \widetilde{A}(l) \oplus \widetilde{B}(m,n)$$

A la fin nous insérons la marque pixel par pixel.

[13] Une clé est un secret nécessaire pour identifier une marque. Dans les principaux modèles de watermarking, elle permet aussi bien d'inscrire la marque que de la lire ou de l'enlever.

Image non signée **Image signée**

Extraction du marquage

La figure 4 montre les étapes de l'extraction et/ou détection de watermark.

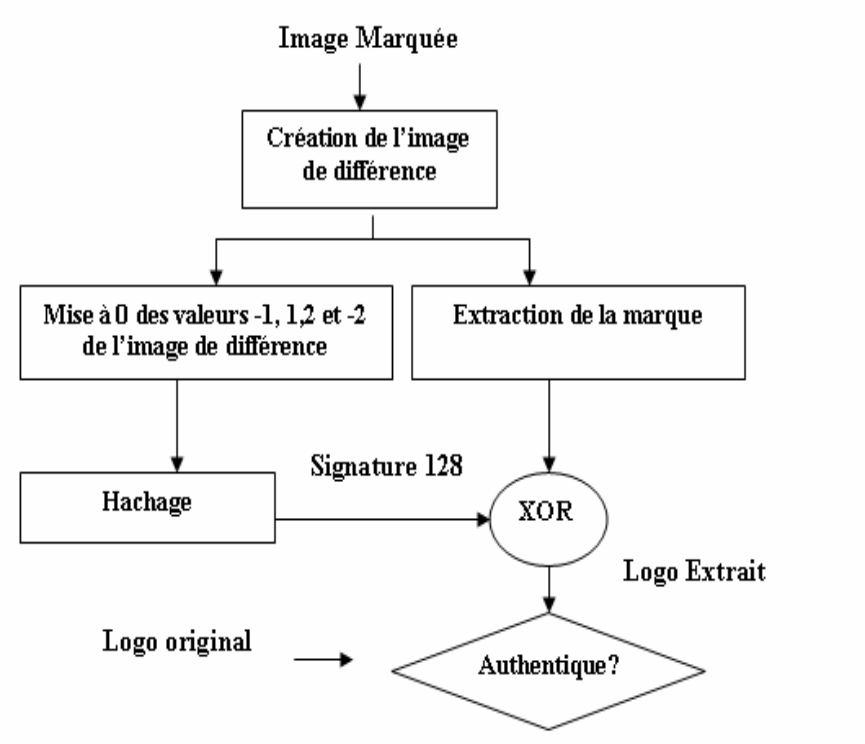


Figure 4: l'organigramme de l'extraction

Dans ce processus, nous vérifions que l'image n'a pas été altérée et si l'image est authentique, nous renversons l'image marquée de nouveau à l'image originale tout en suivant le chemin inverse décrit dans la section de watermark.

A partir de l'image marquée Ie(i, j) de taille Me*Ne pixels, on calcule l'image de différence De(i,j) de taille Me*Ne/2 pixels.

Ensuite nous passons à l'extraction du watermark We(m,n) comme suit :

$$W_e(m,n) = \begin{cases} 0 & \text{if } D_e(i,j) = -1 \text{ or } 1 \\ 1 & D_e(i,j) = -2 \text{ or } 2 \end{cases}$$

Simultanément, nous renversons l'image filigranée de nouveau à l'image originale en décalant quelques valeurs de Pixel dans l'image de différence.

$$I_r(i,2j+1) = \begin{cases} I_e(i,2j+1) - 1 & \text{if } D_e(i,j) \geq 2 \\ I_e(i,2j+1) + 1 & \text{if } D_e(i,j) \leq -2 \\ I_e(i,2j+1) & \text{otherwise} \end{cases}$$

Puis, nous calculons les informations parasites de la même façon qui est décrite en haut.

$$\mathbf{D_r} = \{D_r(i,j) : 0 \leq i \leq M-1; 0 \leq j \leq \tfrac{N}{2} - 1\}$$

$$\widetilde{\mathbf{D}}_\mathbf{r} = \{\widetilde{D}_r(i,j) : 0 \leq i \leq M-1; 0 \leq j \leq \tfrac{N}{2} - 1\}$$

$$H(K, M_e, N_e, \tilde{\mathbf{D}}_\mathbf{r}) = \{\tilde{A}_e(k) : 0 \leq k \leq 127\}$$

Où, Dr tilde égale à Dr excepté les valeurs de différences de -2,-1,1 et 2 qui sont remplacés par la valeur zéro.

Afin d'extraire le logo binaire, nous combinons la fonction d'hachage Ae(K) avec le watermark extrait We(m,n) de longueur C en utilisant la l'opération XOR.

$$\tilde{B}_e(m,n) = \tilde{A}_e(k) \oplus W_e(m,n)$$

A la fin, nous reconstruisons le watermark B(m,n) tilde par une duplication périodique jusque nous aurons B(m,n) de longueur C .

Ensuite nous vérifions l'intégrité de l'image en comparant les deux watermark Be(m,n) tilde avec B(m,n) tilde.

7.7.2 LA CONVERSION EN JPEG2000

Pour sauver de l'espace mémoire sur les disques durs pendant l'archivage, il est important d'utiliser un format de fichier qui a un bon taux de compression. Dans le cas de notre projet, il est possible d'utiliser quatre formats d'image. Des bmps, des gifs, des jpgs et un tout nouveau format qui est jpeg2000. La plus grande différence entre jpeg et jpeg2000 consiste au niveau de la compression. Par rapport à jpeg, l'image jpeg2000 est compressée à environ 25% de plus. Pour une image d'environ 200ko en jpg, elle sera de 150 ko en jpeg2000. De plus, la compression est sans perte, donc la qualité de l'image est pratiquement la même. Ceci est un avantage important quand nous voulons voir des détails provenant des caméras sur un système de vidéo surveillance.

Pour réussir à faire la conversion en jpeg2000, nous avons utilisé l'API JAI (Java Advanced Imaging) qui est une librairie qui contient les supports pour plusieurs types de fichiers en java. Dans la dernière version 1.2 de imageIO existe un support au format jpeg2000. Cela nous rend la tâche plus facile pour faire l'écriture et la lecture des fichiers jpeg2000.

Image bmp de taille 257 KB

Image compressée de taille 157 KB

En conclusion, il est important de noter que le format jpeg2000 est un format d'avenir, et que son utilisation est très simple en java.

8- CONCLUSION

En conclusion, L'évolution technologique a diminué le coût du matériel de vidéosurveillance et surtout a permis d'améliorer sa fiabilité et sa stabilité. Cependant beaucoup d'évolutions sont à venir pour le traitement (détection d'activité) et la transmission (Wifi, fibre optique,…) des données. De plus ces évolutions ont pour but de faciliter le passage de l'analogique au numérique et d'améliorer l'exploitation des données.

A la fin, il faut mentionner que ce projet nous a permis :

- d'approfondir nos connaissances dans le langage Java et la connexion aux bases de sonnées SQL serveur.
- de nous initier à la manipulation et aux traitements d'images en java.
- d'avoir une initiation au watermarking.
- d'explorer plusieurs packages et classes dont on ignorait l'existence.

9- REMERCIEMENTS

Nous voulons remercier Mme Nadia Baaziz notre superviseur, pour ses conseils précieux et pour le temps qu'elle nous a consacré.
Nous tenons à remercier aussi Monsieur Michal Iglewski, notre coordonnateur.

10- BIBLIOGRAPHIE

[1] *Image Processing in JAVA*, by Douglas A.Lyon, 1999, Upper Saddle River, NJ 07458

[2] *Au cœur de java 2 Fonction avancées*, Volume 2, Cay S.Horstmann et Gary Cornell, Publié par CampusPress, 2002 Sun Microsystems, Inc.

[3] *Joint near-lossless compression and watermarking of still images for authentication and tamper localization*
Robert Caldelli, Francesco Filippini, Mauro Barni, 2006

[4] *Reversible image authentication based on watermarking*
Sang-Kwang Lee, Young-Ho Suh, and Yo-Sung Ho, ICME 2006

[5] Les sites visités
http://www.netbeans.org/
http://www.netbeans.org/kb/50/using-netbeans/dbconn.html#pgfId-1156774
www.sun.com
http://msdn2.microsoft.com/en-us/library/ms950403.aspx
http://java.developpez.com/cours/
http://penserenjava.free.fr/pens_2.4/indexSom.html
https://jai-imageio.dev.java.net/