Projet PTrans Quai des Apps 2016-2017

Livrable Unique 4

Sommaire du rapport

Présentation de l'entreprise

Quais des apps est une jeune entreprise implantée sur Nantes depuis quelque années. Cette dernière travaille sur 3 domaines principaux :

- La réalité augmentée
- La réalité virtuelle
- La reconnaissance d'image

NB : La réalité virtuelle est le fait de générer entièrement des images et de les projeter à l'intérieur du dispositif de l'utilisateur. Tandis que la réalité augmentée consiste à superposer des éléments numériques sur ce que voit l'utilisateur.

L'équipe est composée de 3 personnes

- Vincent Marin : un chercheur en physique et en astrophysique
- Michaël Merlange : un ingénieur spécialisé en modélisation 3D7
- Gwen le Villoux : le designer de l'équipe

1) Contexte du projet :

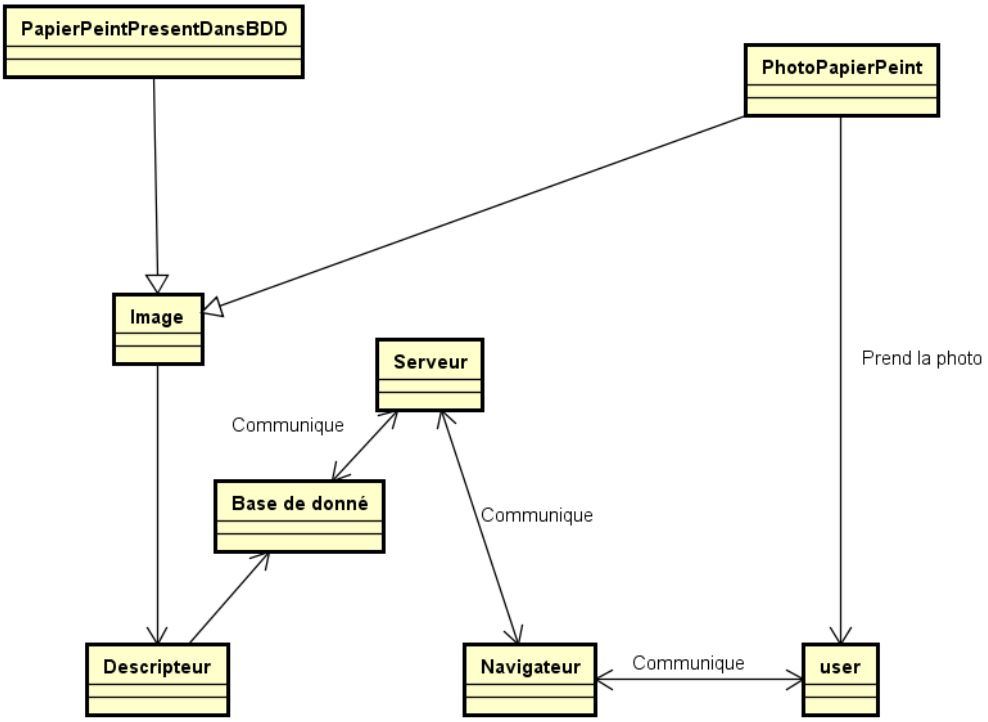
Le but de ce projet transversal est d'étudier les différentes possibilités quand à la reconnaissance d'images. Quai des apps aimerai faire une enquête afin d'avoir une idée relative de ce qui est possible de faire, et ce à quels coûts. Afin de simplifier le sujet, le domaine d'étude se restreint à des images de papiers peints et de carrelages. Le contexte d'utilisation serait lorsqu'un client souhaite retrouver les références d'un produit, il lui suffirait de photographier le carrelage / papier peint afin qu'une application lui ressorte la fiche des produits les plus ressemblant.

Il est important de noter que l'on travail avec l'idée de ressemblance et dont d'exactitude. Le but n'est pas de retrouver à tout le coup un model à partir d'une photo de ce dernier mais de faire ressortir tous les produits ayant des similarités visuelles avec la photo

A côté de cela, notre but n'est donc pas forcément de créer un prototype fonctionnel, mais de faire un état des lieux de ce qui se fait à l'heure actuelle.

2) Modèle du domaine

Nous ne savions pas dans quel cadre nous placer afin de faire le modèle de domaine nous avons donc choisis de nous mettre dans le cadre d'un produit fini.



Toutes les images sont transformées en descripteurs afin de les comparer entre elles.
Un certains nombre d'interrogations restent tout de même, notamment : Le descripteur d'une photo prise par l'utilisateur, est il calculé en local ou est ce que l'image est envoyée au serveur.

3) Les objectifs du projet :

Comme expliqué précédemment notre but n'est pas de créer prototype fini. Notre travail consiste surtout à faire un rapport précis sur la direction que devrait prendre Quai des apps si jamais ils voudraient se mettre à travailler sur un tel projet.
Nous travaillons donc sur les axes suivants :

- Étudier les différentes solutions existantes afin de répondre à la problématique de la reconnaissance de similarités d'images.

- Essayer d'implémenter une ou plusieurs des différentes solutions rencontrées
- Mettre en place un système de test afin comparer les performance de chaque solution
- Puis les comparer afin de mettre en avant les différents avantages et contraintes qui leur sont associés.

Néanmoins, il est possible qu'aucunes des solutions trouvées soient facilement implémentable par des 4ème années. Dans ce cas il nous faudra expliquer clairement quels étaient les obstacles qui nous ont empêchés d'aller jusqu'au bout de l'implémentation.

4) Analyse FURPSE

		T faible	Faible	Moyenne	Important	T Important
	Adéquations des fonctions					X
	Précision Fidélité des résultats					X
	Interopérabilité	X				
	Sécurité	X				
	Conformité			X		

Toute l'importance de ce projet est que soit notre prototype (ou au moins nos recherches si notre prototype n'est pas fonctionnel) correspondent aux attentes de l'entreprise. La mesure de résultats joue un grand facteur aussi car sans cela il n'est pas possible d'attester de la réussite de notre travail.
Nous N'avons pas besoin d'interopérabilité car il n'y a pas de système préexistant avec lequel nous devons être compatible.
De plus nous ne travaillons avec aucunes données sensibles donc la sécurité n'est pas un point important.
Enfin bien qu'il soit important que nos implémentations fonctionnent avec les formats d'images standards (JPG PNG GIF BITMAP) ce qui compte vraiment est l'algorithme derrière et non l'implémentation

		T faible	Faible	Moyenne	Important	T Important
	Compréhension					X
	Apprentissage				X	
	Exploitation		X			
	Ergonomie		X			

Il est indispensable que Quai des Apps soit en mesure de comprendre notre travail ou ce dernier n'aura aucune utilité pour eux.
A côté de ça, il avait été dit que même si notre implémentation est très bonne elle ne sera pas utilisée par la suite par l'entreprise. L'exploitation et l'ergonomie ne sont donc pas prioritaires

		T faible	Faible	Moyenne	Important	T Important
	Maturité		x			
	Tolérance aux pannes					
	Remise état de marche					

Notre but n'est pas de créer un produit juste prouver par un prototype que la possibilité d'un tel produit existe.
De plus il n'existe pas de "panne" à proprement dit pour notre système d'où l'absence de X

		T faible	Faible	Moyenne	Important	T Important
	Temps de réponse				x	
	Utilisation des ressources				x	

Comme le but est de tester chaque méthode de reconnaissance d'images il est important que chaque implémentation soit relativement efficace à la fois en complexité temporelle e en complexité mémoire, le but étant de ne pas fausser nos tests sur la qualité potentielle d'une méthode.

		T faible	Faible	Moyenne	Important	T Important
	Analyse des défaillances				x	
	Facilité de modification		x			
	Stabilité				x	
	Automaticité des tests				x	

Il est important de savoir si la performance d'une méthode sur un test donné est lié à notre implémentation ou à la nature même de cette méthode. La détection de défaillances est donc importantes
De plus il sera nécessaire de créer de nombreux jeux tests diversifiés mais utilisables pour chaque méthode afin de détecter ces possibles défaillances.

		T faible	Faible	Moyenne	Important	T Important
	Adaptation évolutive		x			
	Installation et modification	X				
	Cohabitation	X				
	Remplacement	X				

Notre système n’est pas sensé être installé de manière “durable” vu qu'il s’agit d’un simple prototype d'où la faible importance pour les critères suivants.
Néanmoins il pourrait être intéressant d’avoir un template d'implémentation afin d’être le plus efficace pour développer chaque prototype de méthode

5) Les contraintes imposées par l'entreprise :

Aucunes contraintes n'a été explicitement exprimée par l'entreprise. Si nous pouvons arriver au stade d'un prototype fonctionnel viable ce serait déjà une très bonne avancée. De ce fait, il nous a été conseillé de faire avec les technologies qui nous étaient les plus familières.

Néanmoins, il est clair que si nous nous voulons comparer correctement des images entre elles est nécessaire de travailler avec des fichiers "descripteurs" d'images.

Enfin il va sans dire que si un produit basé sur ce projet est un jour réalisé, il utilisera certainement des technologies WEB. Donc dans l'éventualité ou nous arrivons rapidement à pseudo produit fini il sera envisageable de le réimplémenter avec des langages tels que javascript.

6) Définition du premier sprint et avancement des premières semaines

Le premier sprint contient la totalité de ce rapport, la bibliographie de deux premières méthodes de reconnaissance d'image le Perceptual Hash par DCT et par histogramme RGB

Les premières semaines ont été utilisées afin de mettre en place les outils que nous avons choisis d'utiliser dans ce projet, faire la liste des tâches à faire ainsi que leur répartition entre les membres du groupe, et débuter la rédaction du rapport.

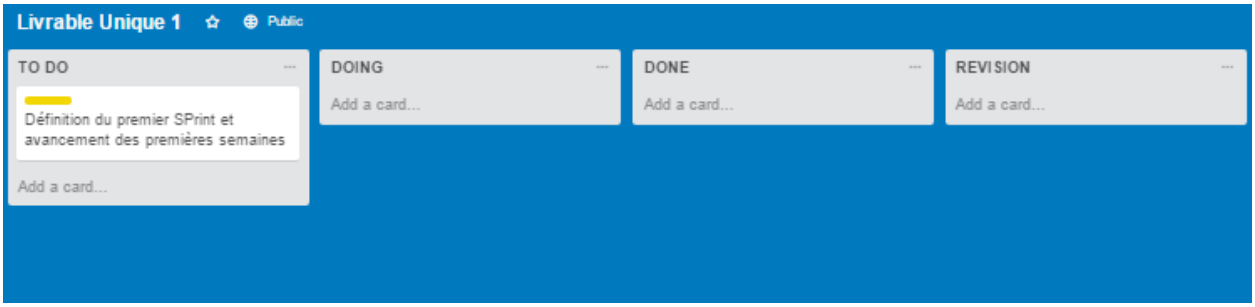
7) Les resultats des stands up meeting

Il y a 7 semaines entre la première réunion que nous avons eu avec Quai des Apps et le rendu de ce Livrable. Nous travaillons chaque semaine avec l'outil Trello afin de se répartir les tâches et de faire un mini-bilan tous les 7 jours.

Pour plus de simplicité nous avons fait des capture d'écran de notre Trello chaque semaine.

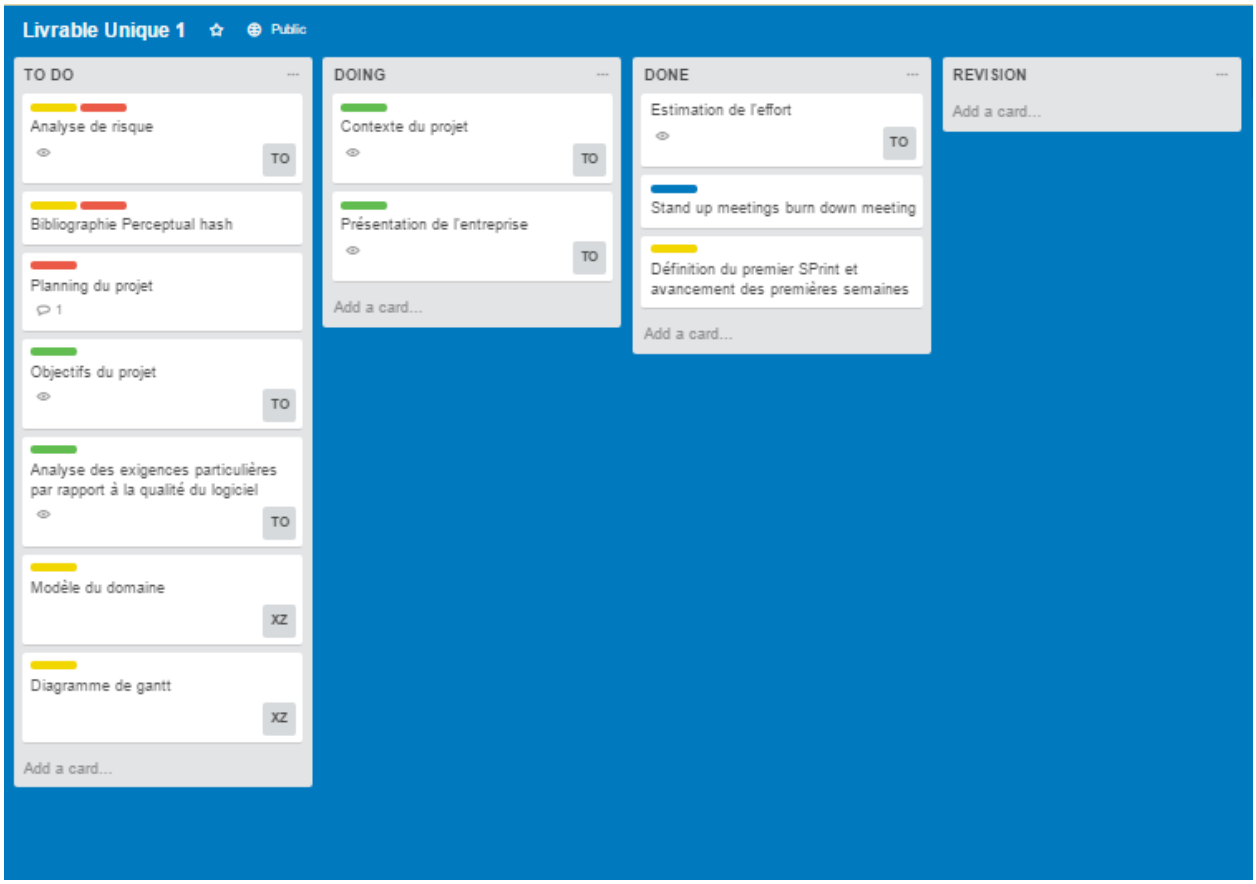
Semaine 1 :

Nous avons mis en place les différentes tâches à réaliser afin de compléter ce livrable



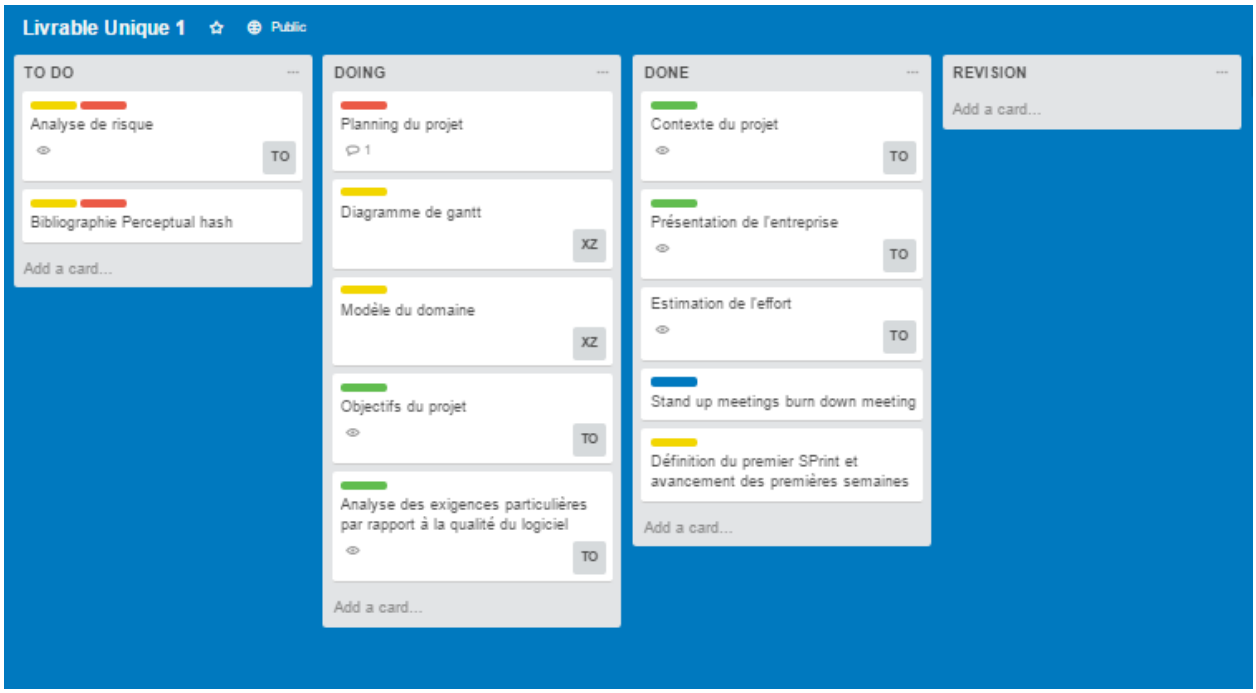
Semaine 2 :

Nous avons commencé les parties simples du Livrable

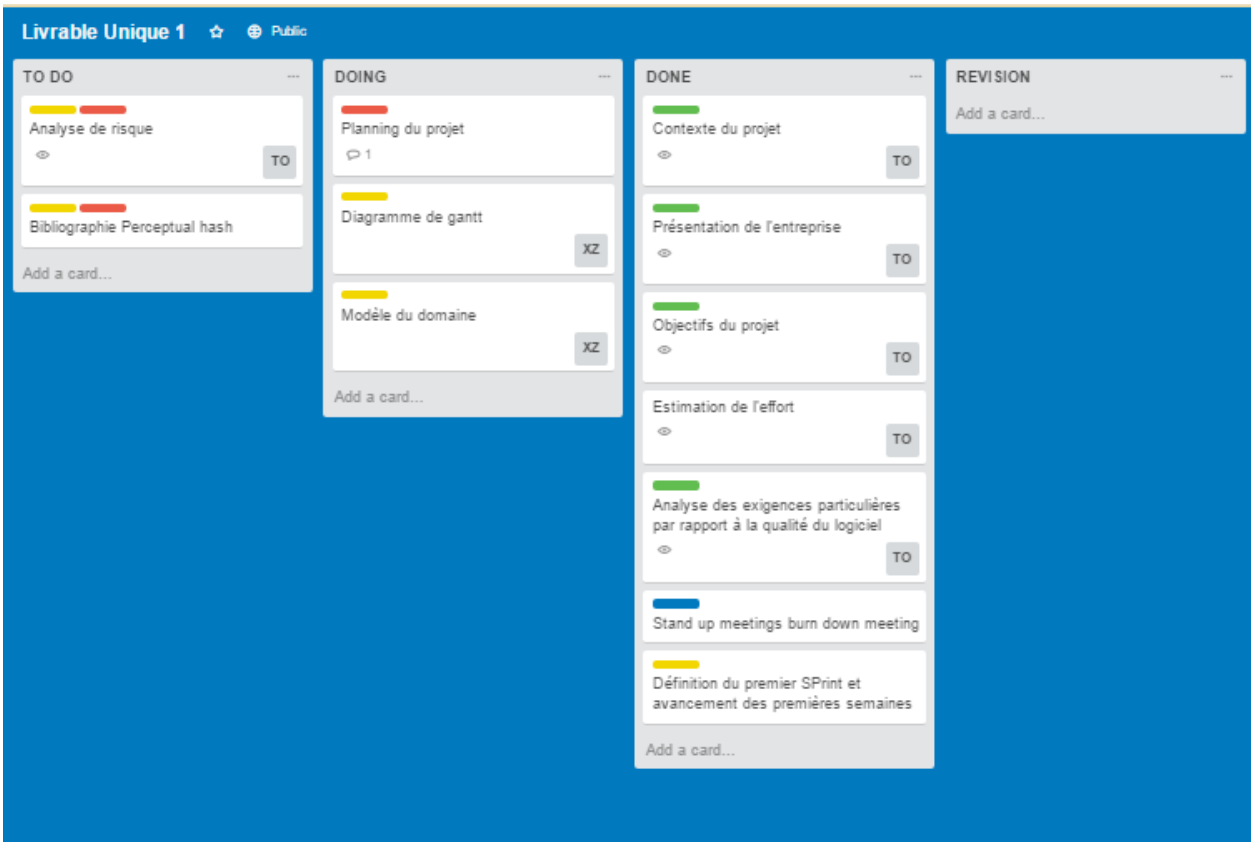


Semaine 3 :

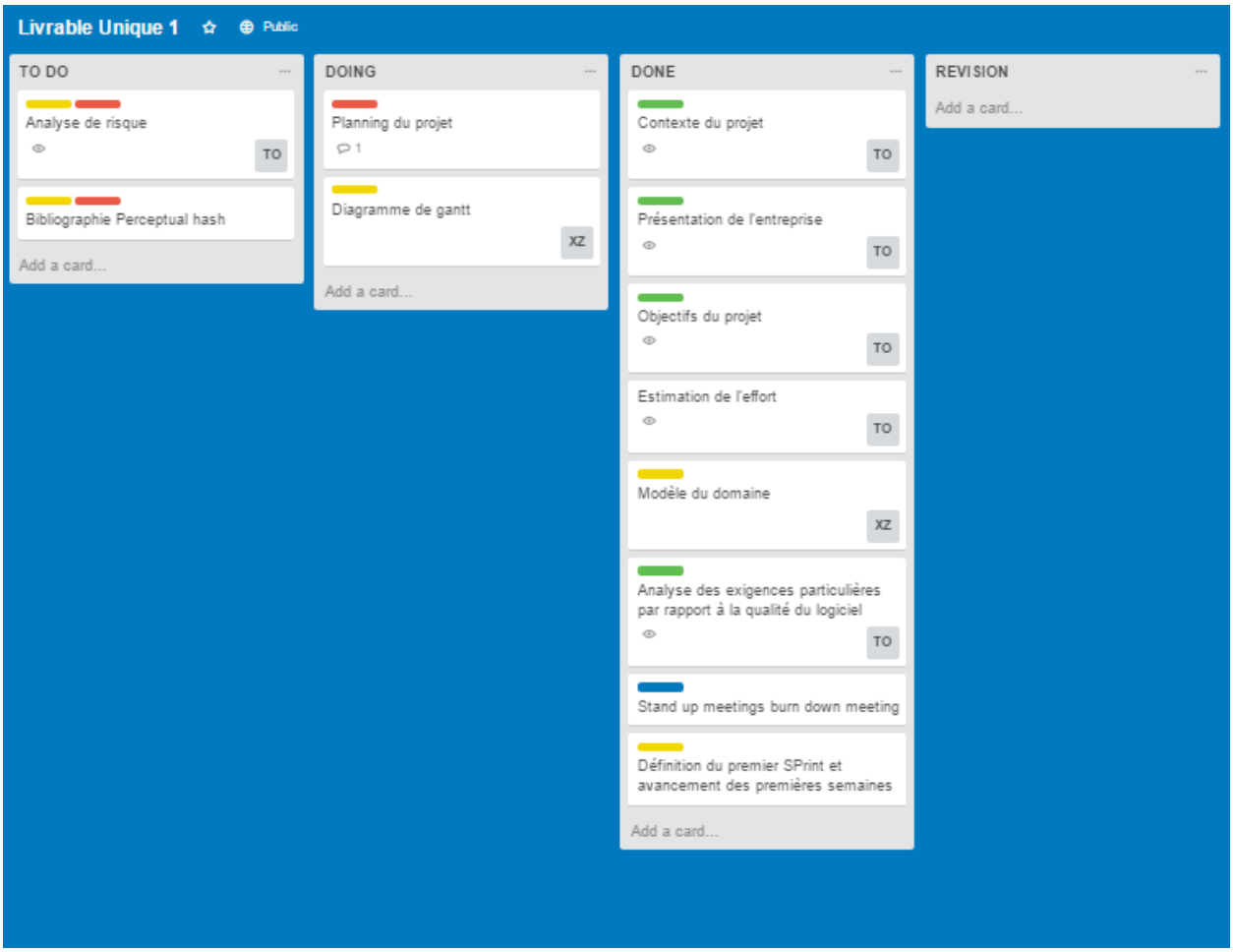
Les parties simples du livrables sont finies nous attaquons les diagrammes et planning



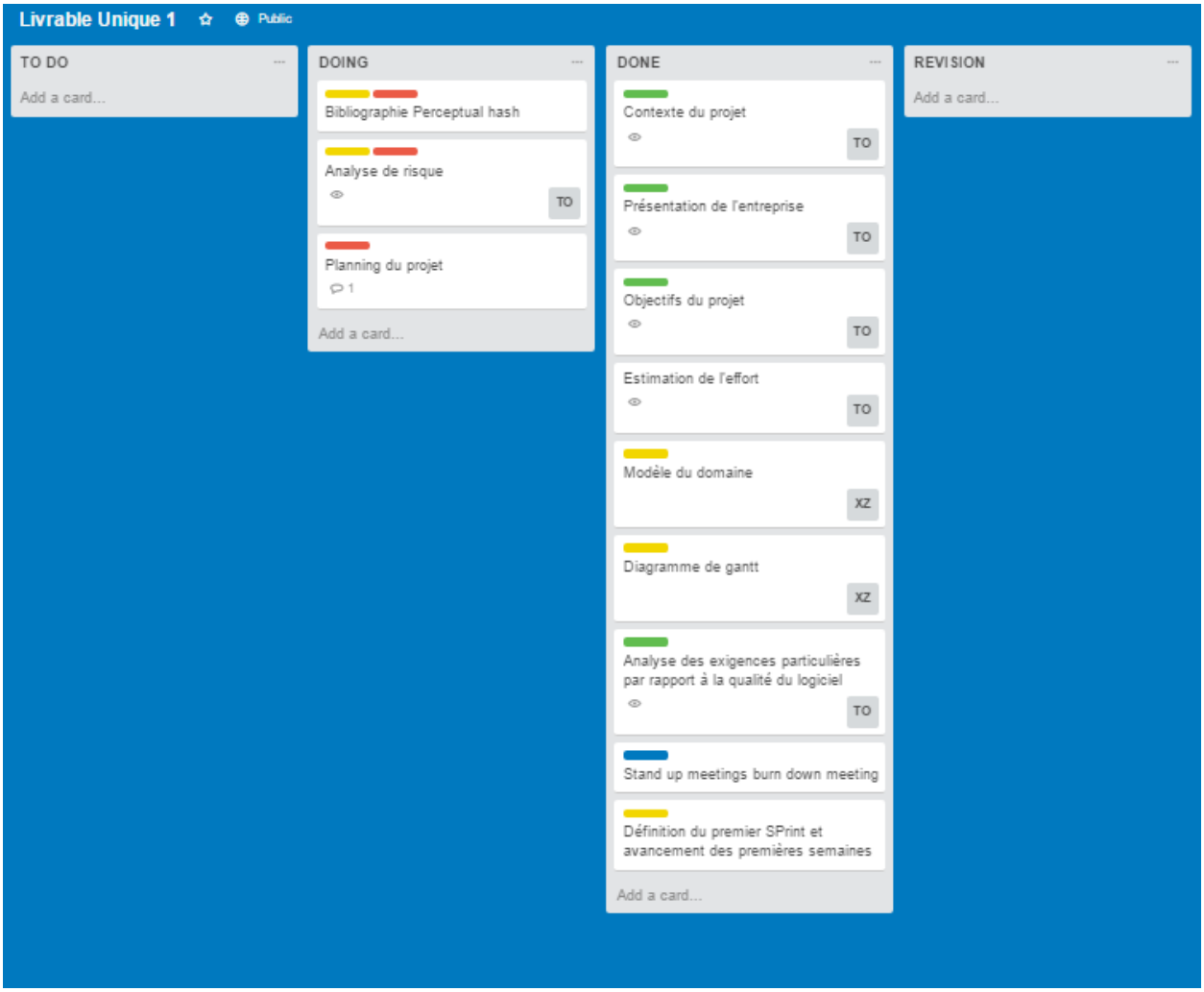
Semaine 4 :


Livrable Unique 1
Public
TO DO
Analyse de risque
TO
Bibliographie Perceptual hash
Add a card...
DOING
Planning du projet
1
Diagramme de gantt
XZ
Modèle du domaine
XZ
Add a card...
DONE
Contexte du projet
TO
Présentation de l'entreprise
TO
Objectifs du projet
TO
Estimation de l'effort
TO
Analyse des exigences particulières par rapport à la qualité du logiciel
TO
Stand up meetings burn down meeting
Définition du premier SPrint et avancement des premières semaines
Add a card...
REVISION
Add a card...

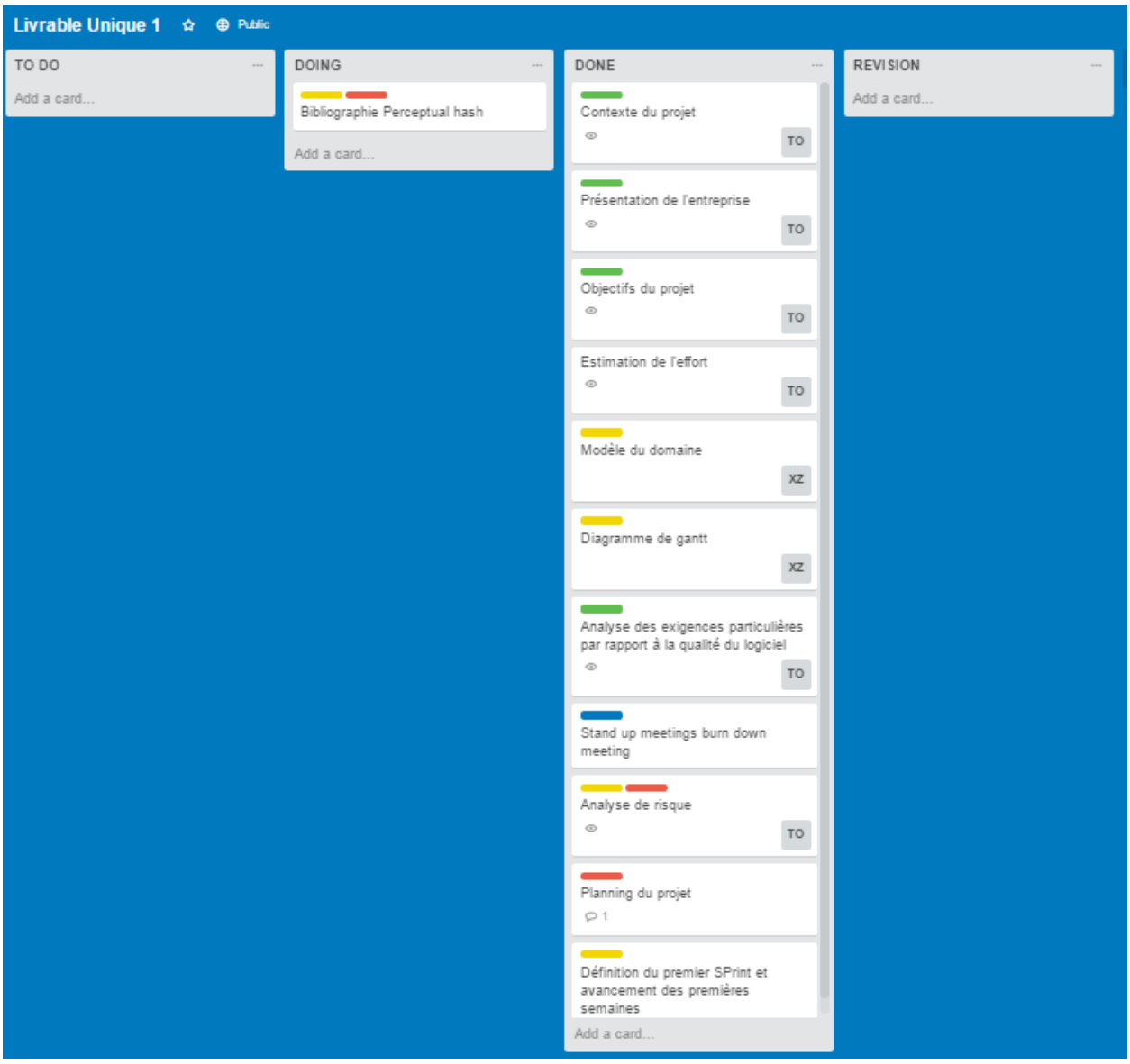
Semaine 5 :


Livrable Unique 1
Public
TO DO
Analyse de risque
TO
Bibliographie Perceptual hash
Add a card...
DOING
Planning du projet
1
Diagramme de gantt
XZ
Add a card...
DONE
Contexte du projet
TO
Présentation de l'entreprise
TO
Objectifs du projet
TO
Estimation de l'effort
TO
Modèle du domaine
XZ
Analyse des exigences particulières par rapport à la qualité du logiciel
TO
Stand up meetings burn down meeting
Définition du premier SPrint et avancement des premières semaines
Add a card...
REVISION
Add a card...

Semaine 6 :


Livrable Unique 1
Public
TO DO
Add a card...
DOING
Bibliographie Perceptual hash
Analyse de risque
TO
Planning du projet
1
Add a card...
DONE
Contexte du projet
TO
Présentation de l'entreprise
TO
Objectifs du projet
TO
Estimation de l'effort
TO
Modèle du domaine
XZ
Diagramme de gantt
XZ
Analyse des exigences particulières par rapport à la qualité du logiciel
TO
Stand up meetings burn down meeting
Définition du premier SPrint et avancement des premières semaines
Add a card...
REVISION
Add a card...

Semaine 7 :


Livrable Unique 1
Public
TO DO
Add a card...
DOING
Bibliographie Perceptual hash
Add a card...
DONE
Contexte du projet
TO
Présentation de l'entreprise
TO
Objectifs du projet
TO
Estimation de l'effort
TO
Modèle du domaine
XZ
Diagramme de gantt
XZ
Analyse des exigences particulières par rapport à la qualité du logiciel
TO
Stand up meetings burn down meeting
Analyse de risque
TO
Planning du projet
1
Définition du premier SPrint et avancement des premières semaines
Add a card...
REVISION
Add a card...

8) Planning du projet

Nous avons mis en place un planning de la manière suivante. :

- Le premier sprint nous sert à prendre une bonne connaissance du projet ainsi qu'a créer une base d'image qui nous serviront à créer une banque de tests. Ces images seront triés par la catégorie du test associés (test sur des images floutées, test sur des images avec rotation, test sur des images avec changement d'échelle…. La liste n'est pas exhaustive)

- Le deuxième troisième et quatrième sprint nous servent à implémenter des méthodes basique de PH et à les tester à partir de nos banques de tests.

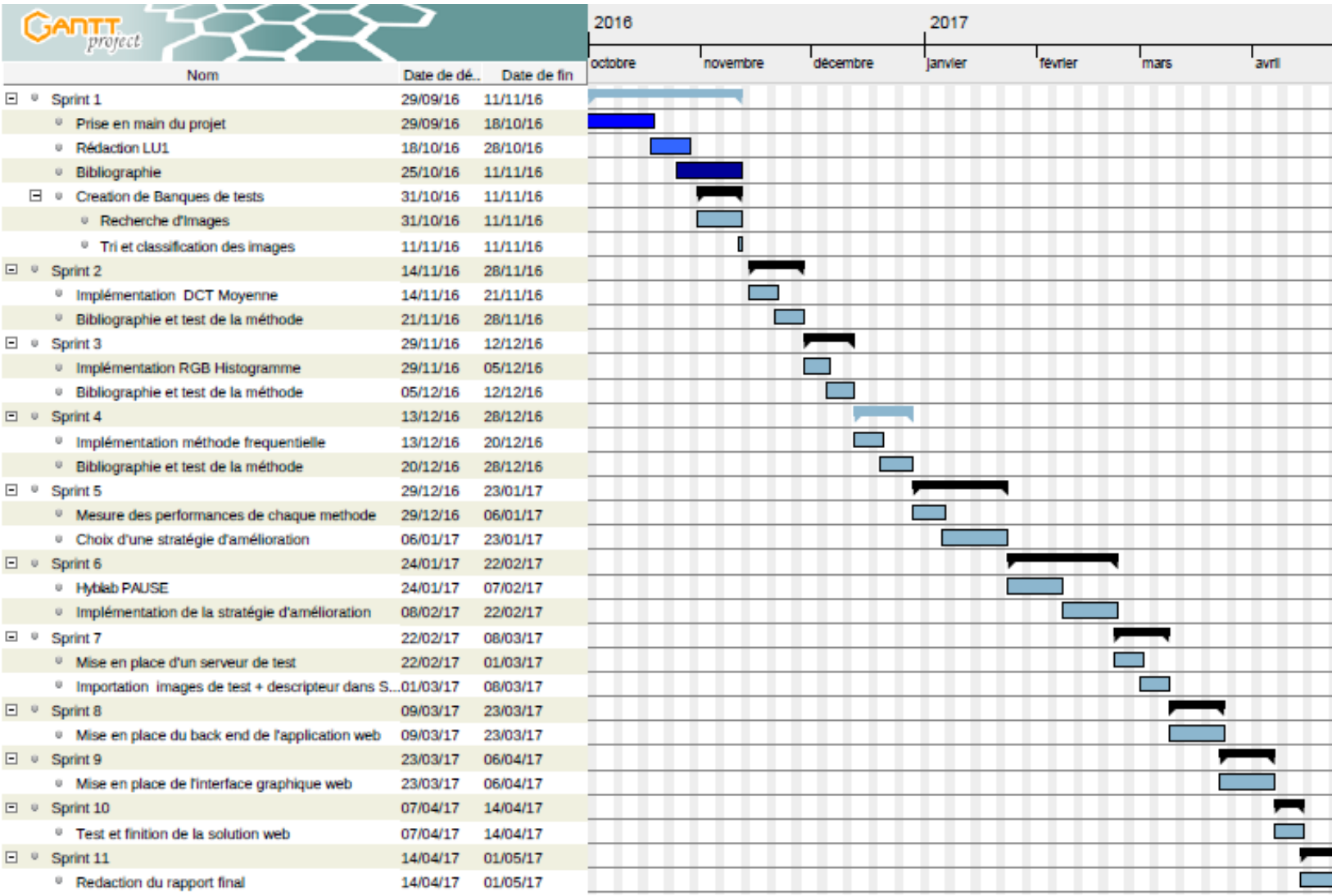
- Le cinquième sprint nous permettra de comparer les méthodes entre elles afin de choisir une stratégie pour la conception d'un prototype

- Le sixième nous permettra d'implémenter le prototype et de le tester

- Le 7eme 8eme 9eme et 10eme sprint nous permettront d'implémenter un prototype d'une application web utilisant le prototype développé durant le sprint 6

- Le dernier sprint est réservé à la finition de l'application web et à l'écriture du rapport

NB : il est probable que nous trouvions pas de méthode efficace pour répondre à la problématique dans les temps voulu. Dans ce cas les sprints consacrée au développement de l'application web serviront à la recherche et l'implémentation de méthode alternatives.



Nom	Date de dé..	Date de fin
Sprint 1	29/09/16	11/11/16
Prise en main du projet	29/09/16	18/10/16
Rédaction LU1	18/10/16	28/10/16
Bibliographie	25/10/16	11/11/16
Creation de Banques de tests	31/10/16	11/11/16
Recherche d'Images	31/10/16	11/11/16
Tri et classification des images	11/11/16	11/11/16
Sprint 2	14/11/16	28/11/16
Implémentation DCT Moyenne	14/11/16	21/11/16
Bibliographie et test de la méthode	21/11/16	28/11/16
Sprint 3	29/11/16	12/12/16
Implémentation RGB Histogramme	29/11/16	05/12/16
Bibliographie et test de la méthode	05/12/16	12/12/16
Sprint 4	13/12/16	28/12/16
Implémentation méthode frequentielle	13/12/16	20/12/16
Bibliographie et test de la méthode	20/12/16	28/12/16
Sprint 5	29/12/16	23/01/17
Mesure des performances de chaque methode	29/12/16	06/01/17
Choix d'une stratégie d'amélioration	06/01/17	23/01/17
Sprint 6	24/01/17	22/02/17
Hyblab PAUSE	24/01/17	07/02/17
Implémentation de la stratégie d'amélioration	08/02/17	22/02/17
Sprint 7	22/02/17	08/03/17
Mise en place d'un serveur de test	22/02/17	01/03/17
Importation images de test + descripteur dans S...	01/03/17	08/03/17
Sprint 8	09/03/17	23/03/17
Mise en place du back end de l'application web	09/03/17	23/03/17
Sprint 9	23/03/17	06/04/17
Mise en place de l'interface graphique web	23/03/17	06/04/17
Sprint 10	07/04/17	14/04/17
Test et finition de la solution web	07/04/17	14/04/17
Sprint 11	14/04/17	01/05/17
Redaction du rapport final	14/04/17	01/05/17

9) Analyse de Risque

Comme pour chaque projet quelque soit sa nature, ce Ptrans possède un certains nombre de risques qu'il nous est nécessaire de mettre au clair.
On parle de risque pour désigner un événement néfaste au déroulement du projet ainsi qu'à son aboutissement.

Nous pouvons classer ces risques en trois catégories distinctes. Les risques fonctionnels, les risques liés à la Technique et les autres risques.

Dans les risques fonctionnels, classés par gravité nous avons donc :

- Que les résultats apportés par nos recherches ainsi que le rapport généré ne conviennent pas au client. Si le risque arrive alors le travail fournit pendant l'année sera obsolète. Ce risque possède une probabilité non négligeable de se réaliser. Il sera donc nécessaire de consulter le client relativement souvent afin d'être sûr que les visions sur le projet ne divergent pas.

Probabilité	Gravité	Acceptabilité
Faible	Très importante	Quasi-nulle

- Le deuxième risque est que le système de test choisi afin de tester les différentes méthode de reconnaissance d'image ne soit pas pertinent. Dans cette situation les résultats fournis dans le rapport risque d'être faussé et ou mal interprété à l'intérieur du rapport de fin de projet. Pour éviter ce risque il sera nécessaire de faire des tests "d'étalonnages" afin de s'assurer que nos résultats restent cohérents entre eux

Probabilité	Gravité	Acceptabilité
Moyenne	Importante	Faible

Dans les risques associés à la technique nous avons :

- Que l'implémentation des solutions trouvées soient peu efficace. En effet il est possible que l'implémentation d'une ou des méthodes de reconnaissance d'images soient raté. Dans ce cas même si la méthode s'avère être très efficace nos résultat seront trompés par le manque de qualité de notre implémentation. Pour éviter que ce risque ne se produise il sera judicieux de développer et de tester plusieurs implémentations d'une seule méthode de reconnaissance d'image

Probabilité	Gravité	Acceptabilité
Moyenne	Moyenne	Faible

- Que l'équipe n'arrivent à implémenter aucunes des méthodes trouvées sur internet / dans les livres. Ce risque a une probabilité plutôt faible de se réaliser mais reste quand même non négligeable. L'impact n'est pas forcément très important compte tenu du fait que notre rôle est surtout de faire un rapport des technologies actuelles

Probabilité	Gravité	Acceptabilité
Faible	Moyenne	Moyenne

:

Enfin dans les risques de types autres nous avons :

- Un manque de temps de travail. Si chacun des membres du groupe ne fournit pas un travail conséquent dans le projet il est fort possible que le rendu soit en retard et ou rendu de manière incomplète. Cela signifierait un échec partiel / complet du projet. Il est donc impératif que chacun réalise les tâches qui lui sont attribué tout en respectant les deadline mise en place par le groupe.

Probabilité	Gravité	Acceptabilité
Forte	Importante	Variable

:

NB L'acceptabilité de ce risque est variable car il dépend très largement de la qualité du travail fournis.

- Mauvaises gestions de notre temps. Ce qui induirait un retard voir sur le rendu. Ce risque est très important d'où la nécessité de mettre en place des moyens de de répartitions des tâches.

Probabilité	Gravité	Acceptabilité
Moyenne	Moyenne	Moyenne

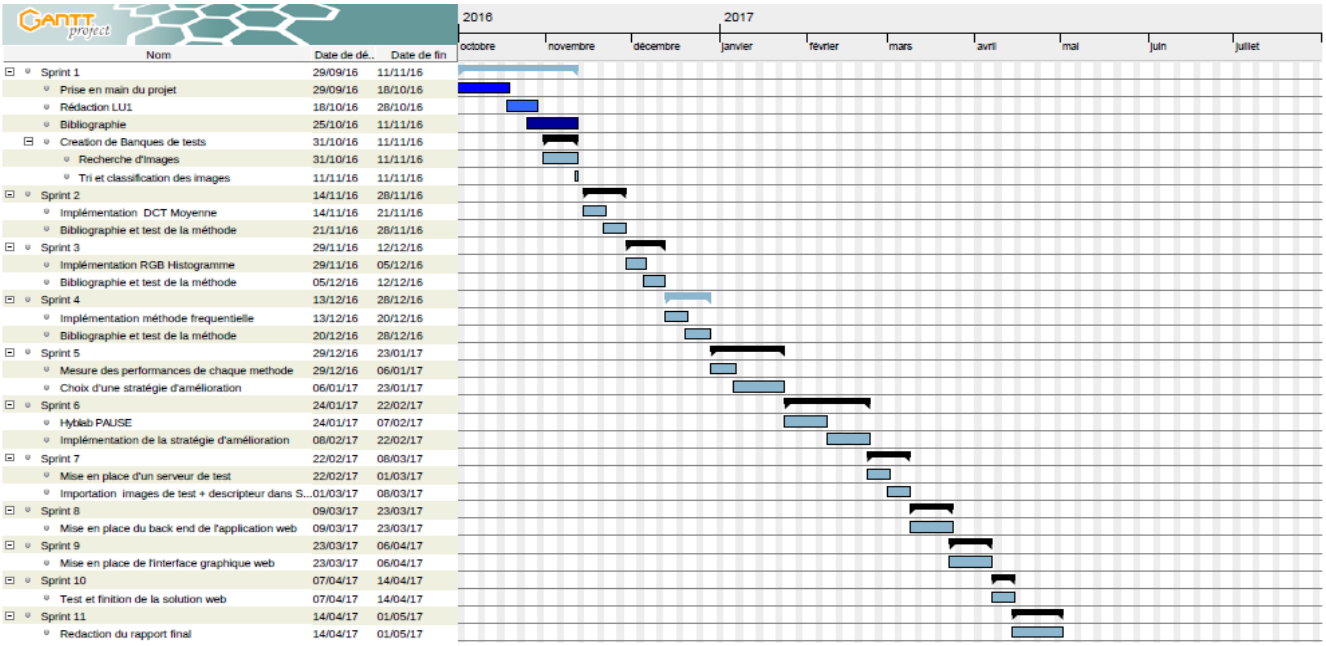
:

- Mauvaises communication au sein de l'équipe. En effet si tout le monde n'est pas à jour sur l'avancée du projet ainsi que la répartition des tâches qui a été mise en place cela induira une perte de temps considérable et augmentera le risque de retard du rendu

Probabilité	Gravité	Acceptabilité
Faible	Moyenne	Moyenne

10) Resultat du Sprint 1

Quand est arrivé la fin du premier sprint, nous avions commencé une implémentation de perceptual hash utilisant la DCT. Le problème était que nous avions sauté trop d'étapes. Nous n'avions concerté ni nos professeurs ni nos clients sur nos choix, nous n'avions pas de vrai système ni de base d'images permettant de tester proprement notre implémentation. De plus notre planning était extrêmement simpliste et naïf. Il s'agissait juste de trouver une nouvelle méthode de comparaison d'image toute les deux semaines et de la tester



Nom	Date de dé..	Date de fin
Sprint 1	29/09/16	11/11/16
Prise en main du projet	29/09/16	18/10/16
Rédaction LU1	18/10/16	28/10/16
Bibliographie	25/10/16	11/11/16
Creation de Banques de tests	31/10/16	11/11/16
Recherche d'Images	31/10/16	11/11/16
Tri et classification des images	11/11/16	11/11/16
Sprint 2	14/11/16	28/11/16
Implémentation DCT Moyenne	14/11/16	21/11/16
Bibliographie et test de la méthode	21/11/16	28/11/16
Sprint 3	29/11/16	12/12/16
Implémentation RGB Histogramme	29/11/16	05/12/16
Bibliographie et test de la méthode	05/12/16	12/12/16
Sprint 4	13/12/16	28/12/16
Implémentation méthode frequentielle	13/12/16	20/12/16
Bibliographie et test de la méthode	20/12/16	28/12/16
Sprint 5	29/12/16	23/01/17
Mesure des performances de chaque methode	29/12/16	06/01/17
Choix d'une stratégie d'amélioration	06/01/17	23/01/17
Sprint 6	24/01/17	22/02/17
Hybkab PAUSE	24/01/17	07/02/17
Implémentation de la stratégie d'amélioration	08/02/17	22/02/17
Sprint 7	22/02/17	08/03/17
Mise en place d'un serveur de test	22/02/17	01/03/17
Importation images de test + descripteur dans S...	01/03/17	08/03/17
Sprint 8	09/03/17	23/03/17
Mise en place du back end de l'application web	09/03/17	23/03/17
Sprint 9	23/03/17	06/04/17
Mise en place de l'interface graphique web	23/03/17	06/04/17
Sprint 10	07/04/17	14/04/17
Test et finition de la solution web	07/04/17	14/04/17
Sprint 11	14/04/17	01/05/17
Redaction du rapport final	14/04/17	01/05/17

Suite aux remarques que nous avons collecté lors des différentes réunions, nous avons décidé des choses suivantes avec notre entreprise et professeurs.

Tout d'abord, modifier notre approche avec l'idée de ressemblance et non d'exactitude. En effet, la méthode DCT nous permet facilement de savoir si deux images sont très proches l'une de l'autre, mais marche beaucoup moins bien quand ils s'agit de trouver des similarités entre deux images. Et cette méthode ne convenait pas aux besoins du client.

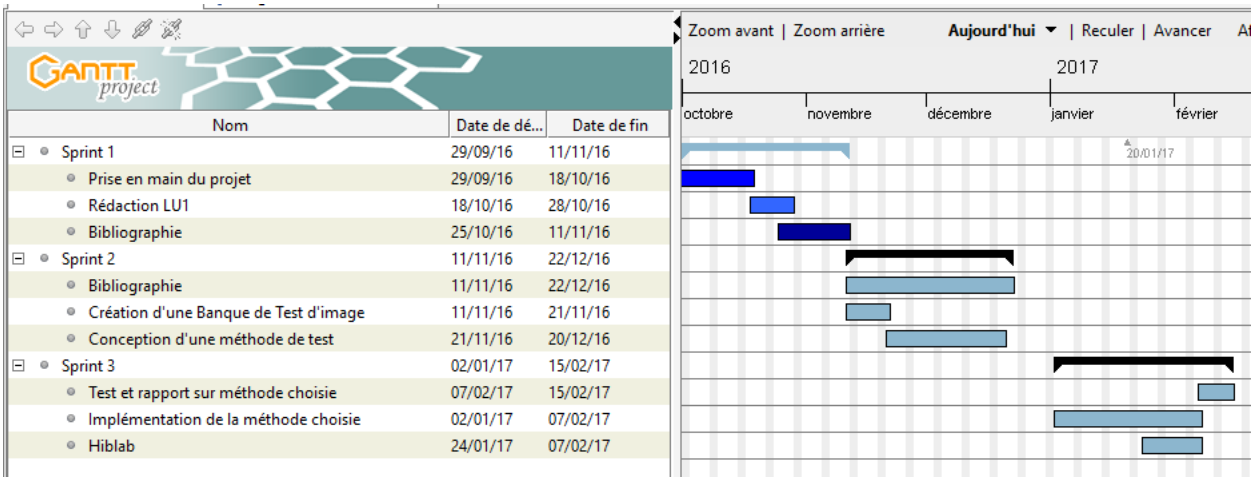
Le deuxième point était d'établir une vrai banque de test avec un nombre conséquent d'images à contrario de la dizaines d'image que nous utilisions pour tester la DCT. Car sans cela il n'est pas possible d'évaluer correctement les résultats des méthodes testées.

Une autre des remarques était de faire de la bibliographie et de trouver un certains nombre d'implémentations sur internet à tester avec la base d'image mise en place. Il nous était demandé de ne pas forcément coder nous même ces implémentations mais d'utiliser des programmes déjà codés afin de rendre plus rapide la recherche de solutions viables.

Enfin un planning plus réaliste et avec des résultats mieux définis sur le court / moyen terme devait être mis en place.

Le changement de planning

La première chose que nous avons changé a été notre planning en suivant les indications de nos professeurs nous sommes arrivés au résultat suivant.



Nom	Date de dé...	Date de fin
Sprint 1	29/09/16	11/11/16
Prise en main du projet	29/09/16	18/10/16
Rédaction LU1	18/10/16	28/10/16
Bibliographie	25/10/16	11/11/16
Sprint 2	11/11/16	22/12/16
Bibliographie	11/11/16	22/12/16
Création d'une Banque de Test d'image	11/11/16	21/11/16
Conception d'une méthode de test	21/11/16	20/12/16
Sprint 3	02/01/17	15/02/17
Test et rapport sur méthode choisie	07/02/17	15/02/17
Implémentation de la méthode choisie	02/01/17	07/02/17
Hiblab	24/01/17	07/02/17

L'idée est comme montré précédemment, de se recentrer sur les objectifs de court et moyens terme afin de simplifier l'organisation et d'avancer de manière plus régulée.

Le premier sprint qui suivait le LU2, consistait donc en la création d'une banque d'images ainsi qu'une méthode de test pour évaluer nos futures implémentations.
Dans le même temps il nous fallait faire de la bibliographie afin de trouver des méthodes de comparaison d'images qui conviennent mieux aux attente de Quai des Apps.

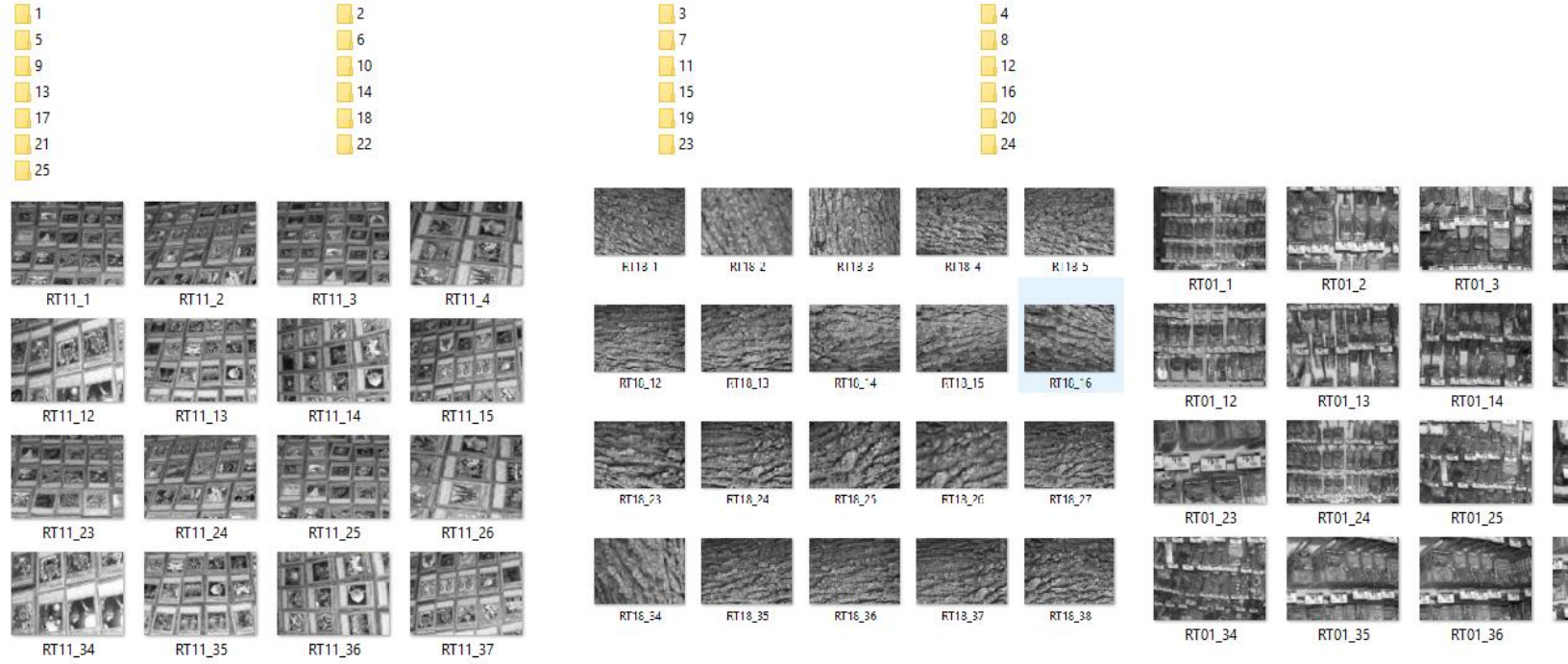
Le sprint suivant quand à lui était réservé à l'implémentation et aux tests de la méthode trouvée durant le sprint 2.

11) Resultat du Sprint 2

Dans ce sprint, nous avons donc constitué une base d'images suffisamment conséquente afin de pouvoir d'évaluer la qualité de nos implémentations

Nous nous sommes tournés vers une base déjà mise en place par le CFAR (Center for automation Research)

La base est structurée de la manière suivante



La collection est composée de 25 dossier chacun de ses dossier contient 40 photos d'un même type d'objet. Les objets représentés sont très aléatoires, buissons, étalage de fruits, de boissons de matériel de bricolage.

L’avantage principale de cette collection est qu’elle nous permet de voir comment se comporte une méthode de comparaison d’image face à deux contenus visuels similaire

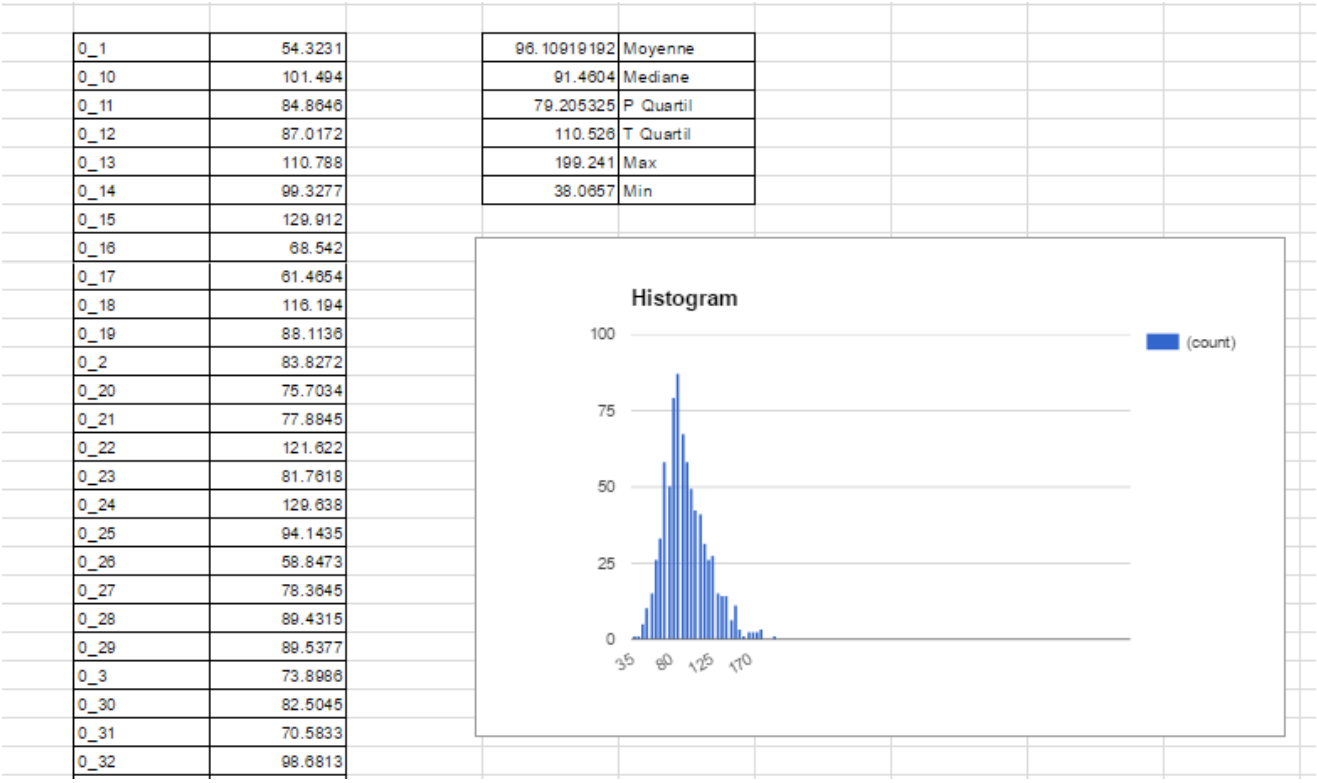
En contrepartie l’intégralité des images de la collection sont en niveaux de gris ce qui limite l’étendue des tests que nous pouvons effectuer. Notamment si la méthode de comparaison utilisée se base sur les couleurs présentes dans l’image.

La méthode de test marche donc de la manière suivante.

Tout d’abord nous partons du principe que nous travaillerons avec des distances.
Deux images avec de faibles distances doivent donc avoir plus de similarité visuelle que les autres.

Nous testons en premier toutes les images similaires entre elles.
Cad, si je prends les images du dossier 1 je vais comparer toutes les images du même dossier entre elles puis regrouper les résultats
Nous répétons ça sur tout les dossier

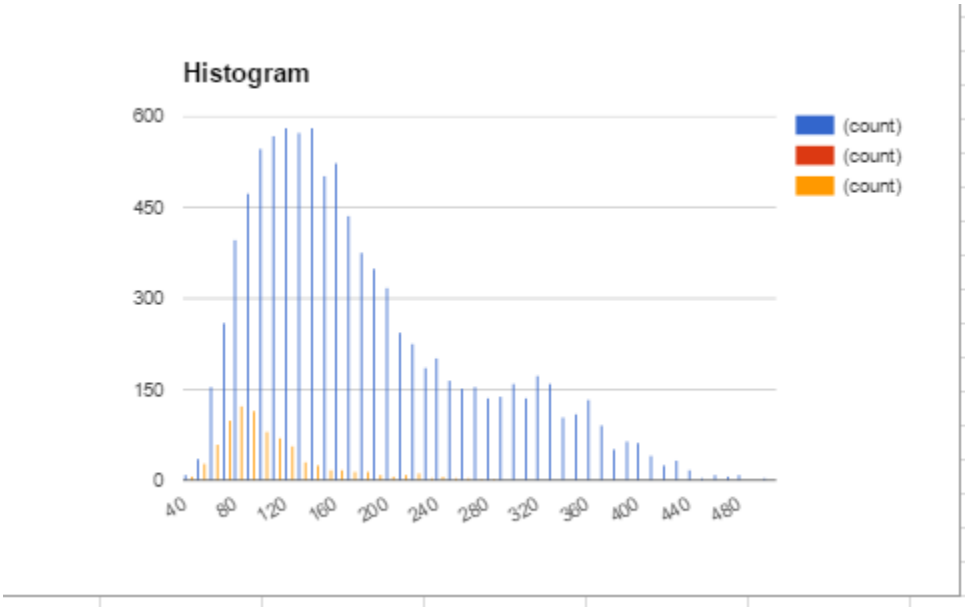
Cela nous donne une distribution des distances pour des images de contenu visuel similaire
Ci dessus un exemple de distribution obtenu lors du sprint suivant



0_1	54.3231
0_10	101.494
0_11	84.8646
0_12	87.0172
0_13	110.788
0_14	99.3277
0_15	129.912
0_16	68.542
0_17	61.4654
0_18	116.194
0_19	88.1136
0_2	83.8272
0_20	75.7034
0_21	77.8845
0_22	121.622
0_23	81.7618
0_24	129.638
0_25	94.1435
0_26	58.8473
0_27	78.3645
0_28	89.4315
0_29	89.5377
0_3	73.8986
0_30	82.5045
0_31	70.5833
0_32	98.6813

96.10919192	Moyenne
91.4604	Mediane
79.205325	P Quartil
110.526	T Quartil
199.241	Max
38.0657	Min

Une fois que nous avons la distributions pour des images similaires il nous faut celle pour des images de contenus différents pour ainsi comparer les deux distributions entre elles



En orange la répartition des distances pour des images similaires
En bleu la répartition pour des images différentes

A partir de la comparaison des deux distributions nous pouvons déterminer un seuil pour lequel nous considérons que deux images sont similaires ou non.

L'exemple ci dessus nous montre un cas problématique ou le seuil de différence n'est pas évident
En effet les deux répartitions se superposent
De ce fait le seuil que nous allons choisir va forcément admettre un certains nombre de non positif et/ou d'erreurs.

Plus la méthode de comparaison sera efficace plus la différence entre les deux distributions sera évidente.
Bien sur, la méthode parfaite n'existe pas mais en fonction du contexte d'utilisation nous pouvons optimiser cette dernière ainsi que le seuil choisis afin que les résultats nous conviennent au mieux.

Dans un même temps nous avons continué nos recherches sur les méthodes de comparaisons d'images.
Notre idée était de pousser nos recherches vers des descripteurs d'images utilisant les textures présentent dans les image.
En effet le principe d'une texture est qu'elle présente une certaine forme de répétition dans son contenu visuel.
Cette même idée de répétition se retrouve dans les papier peints.

Un descripteur qui nous semblait donc intéressant fut le Homogeneous Texture Descriptor
Ou descripteur de texture homogène en français
L'explication plus détaillée de son fonctionnement se trouve dans la partie bibliographie

12) Resultat du Sprint 3

Maintenant que nous avions une base et un système de test ainsi qu'une idée de la méthode que nous voulions tester il ne nous manquait plus qu'à trouver une implémentation valide sur internet.

Nous nous sommes rendus compte que la librairie openCV nous permettait très facilement de calculer le descripteur de texture homogène mais qu'en plus qu'elle intégrait d'autre descripteurs dans son code source qui marchaient avec le même principe.

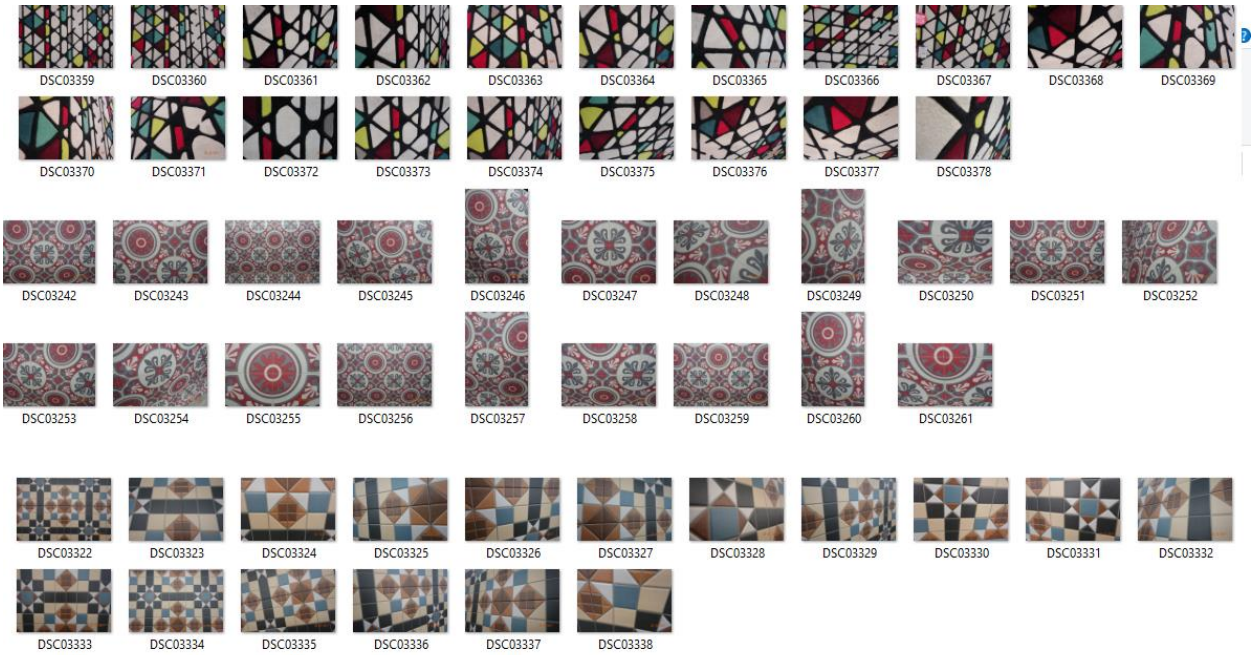
Pour faire simple il nous suffisait de changer quelque ligne de code pour passer d'un descripteur à un autre
Nous avons donc commencé à faire nos tests sur notre collections d'images.
Malheureusement nous n'avons pas encore fini d'interpréter les résultats pour établir de la réelle efficacité de cet algorithme par rapport à notre contexte d'évaluation.

Un autre problème s'est soulevé en même temps.
Nous avions prévu après l'interprétation des résultats de passer à l'étude d'un autre générateur de descripteurs d'images.
Néanmoins une grande partie des descripteurs d'OpenCV utilisent les couleurs des images pour fonctionner.
Et les images présentent dans notre base sont exclusivement en niveau de gris.

Nous avons alors changé d'approche. Plutôt que de chercher sur internet une base d'image qui pouvait possiblement nous servir, nous avons préféré créer la notre

Nous sommes donc partis dans des magasins spécialisés afin de prendre en photo des modèles de papier peints ou d'objets ayant les mêmes caractéristiques visuelles (couleur dans les mêmes tons, répétitions visuelles, formes simples).

Dans le même principe chaque dossier contient des images d'un seul et même objets mais sous différents angles et différents rapprochements.



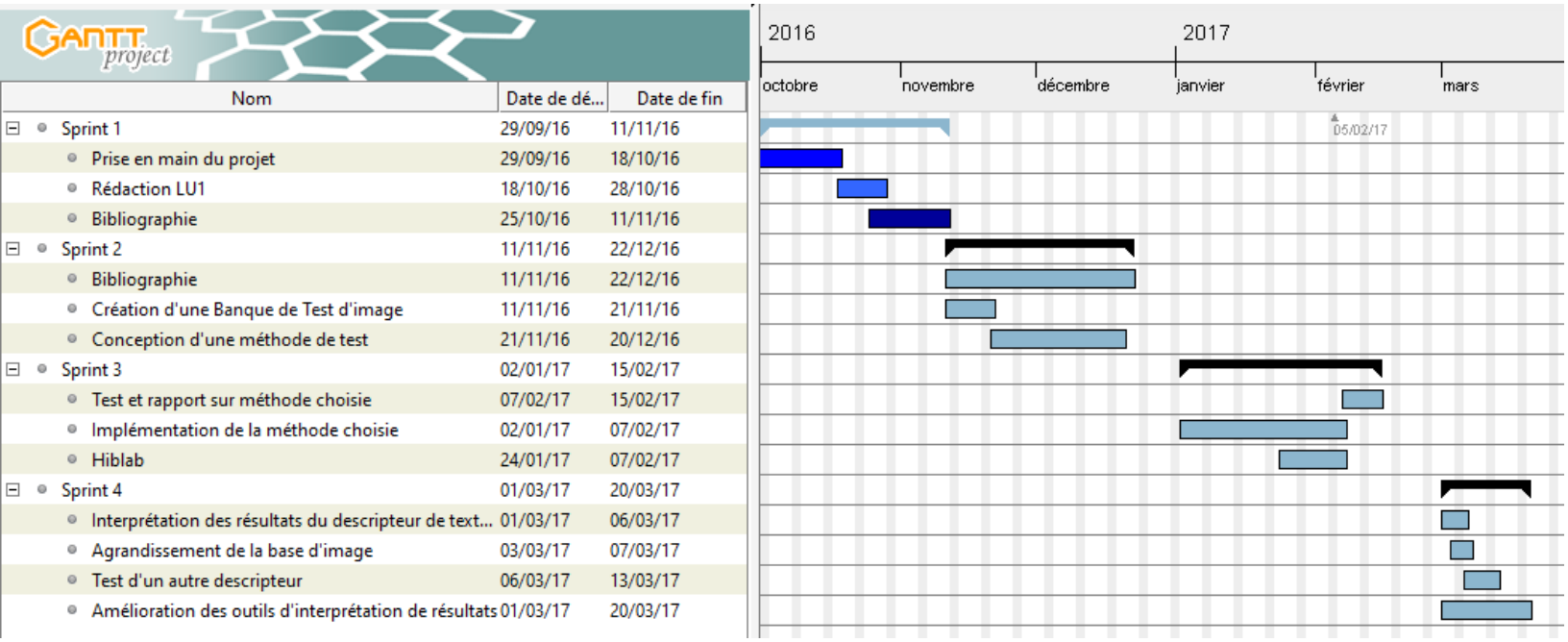
Ainsi nous pouvons garder la même méthode de test sur cette nouvelle collection.

Nous ne possédons pour l'instant que 175 photos pour 9 objets différents mais nous comptons continuer à élargir cette collection afin de renforcer la validité de nos tests

13) Comparaison estimation planning / réalité

Nous avons quasiment respecté le planning que nous nous étions fixé suite au remaniement des sprints.
Nous avons quand même pris quelque jours de retards notamment sur la partie interprétation des tests.
En effet il était long et laborieux de récupérer les milliers de distances entre toutes les images de notre collection et de faire de l'analyse de données en utilisant excel
Nous sommes en train de réfléchir sur de meilleurs moyens pour automatiser cette partie et gagner en efficacité.

14) Planning pour le sprint 4



Nom	Date de dé...	Date de fin
Sprint 1	29/09/16	11/11/16
Prise en main du projet	29/09/16	18/10/16
Rédaction LU1	18/10/16	28/10/16
Bibliographie	25/10/16	11/11/16
Sprint 2	11/11/16	22/12/16
Bibliographie	11/11/16	22/12/16
Création d'une Banque de Test d'image	11/11/16	21/11/16
Conception d'une méthode de test	21/11/16	20/12/16
Sprint 3	02/01/17	15/02/17
Test et rapport sur méthode choisie	07/02/17	15/02/17
Implémentation de la méthode choisie	02/01/17	07/02/17
Hiblab	24/01/17	07/02/17
Sprint 4	01/03/17	20/03/17
Interprétation des résultats du descripteur de text...	01/03/17	06/03/17
Agrandissement de la base d'image	03/03/17	07/03/17
Test d'un autre descripteur	06/03/17	13/03/17
Amélioration des outils d'interprétation de résultats	01/03/17	20/03/17

Avec le rendu de ce livrable s'achève la fin du sprint 3.
Nous comptons donc sur les semaines à venir, travailler sur les points suivant :

- Finir l'interprétation des résultats pour le descripteur de texture. Cad regrouper toutes les distributions d'images similaires et non similaires afin d'établir (dans la mesure du possible) un seuil efficace de distance pour ce descripteur - Estimation de l'effort 4-5 jours

- Agrandir la base d'image : probablement passer un samedi dans les magasins à photographier des papier peints des tapis et des carrelages - Estimation de l'effort 2 jours

- Tester un nouveau descripteur : utiliser la même méthode que pour le descripteur de texture afin de mettre en évidence la présence ou non d'un seuil de distance entre les images similaire et différentes - Estimation de l'effort entre 6 et 14 jours

- Améliorer les outils d'interprétation de résultats. Cette partie se fera au fur et à mesure du sprint. Nous pensons que plus nous passerons du temps à interpréter nos fichiers de distances plus nous trouverons des manières d'aller plus dans notre travail - Estimation de l'effort sur toute la durée du sprint

15) Résultats pour le sprint 4

Dès le début du sprint 4 nous avons du faire de grosses modifications de notre base d'images. Sans vraiment s'en rendre vraiment compte nous nous éloignions de notre but original.

En effet le but de ce projet est de concevoir un prototype de programme capable d'associer des images qui se ressemble et non de retrouver un objet de manière exacte. Or notre base était un regroupement de photos d'un seul et même objet. Ce qui ne convenait pas vraiment à notre cadre d'utilisation

La suggestion faite par le Client et les professeurs était de constituer une base d'images venant de sites vendeur de papiers peints, de créer une classification de cette dernière puis de continuer notre travail comme prévu dans le sprint.

Voilà un aperçu de la base sur laquelle nous avons travaillé durant ce sprint

Comme on peut le voir sur cette image nous avons un grand nombre de papiers peints possédant une même texture et avec de légères nuances de couleurs ce qui est plus adapté pour notre projet.

Une fois cette base construite nous avons créé une classification de 20 *10 images

Nous avons pris 20 images à part et nous avons regroupé les 10 images qui leur ressemblaient le plus dans notre base.
Cela reste léger mais ce travail est très laborieux et avec les examens / projets à rendre nous avions déjà perdu une semaine de travail ou nous n'avions pas travaillé notre projet.

Voici à quoi ressemble la classification en format texte

```
0 26 33 32 59 51 69 25 70 37 61
1 2 114 111 3 4 112 113 12 11 77 79
44 43 45 192 190 191 39 40 46 47 174
52 53 195 193 194 54 55 196 50 51
86 80 84 83 107 122 92 95 96 10 123
141 142 143 176 177 197 172 209 200 199
144 139 140 156 157 158 165 166 167 164
169 170 168 171 172 201 202 203 204 173
180 178 179 184 185 186 192 174 181 176
207 208 209 210 206 205 200 198 197 142
71 20 31 58 63 72 101 103 7 29 68
2 5 9 13 14 15 17 34 38 57 67
3 10 16 28 56 88 90 101 112 115
4 0 11 12 15 17 32 33 45 62 89
7 3 10 13 14 20 21 29 31 58 68
8 2 3 13 14 16 21 23 31 34 38
9 10 14 17 22 38 46 49 56 62 72
10 7 8 13 14 15 46 56 68 73 88
71 7 10 20 31 56 58 63 68 101 103
33 0 26 32 37 50 51 59 61 69 70
63 7 20 31 58 68 103 108 71 101 106
65 8 21 23 35 60 67 98 34 38 72
102 26 69 100 107 116 25 32 62 89 122
106 7 135 16 18 20 31 58 68 101 103 105
```

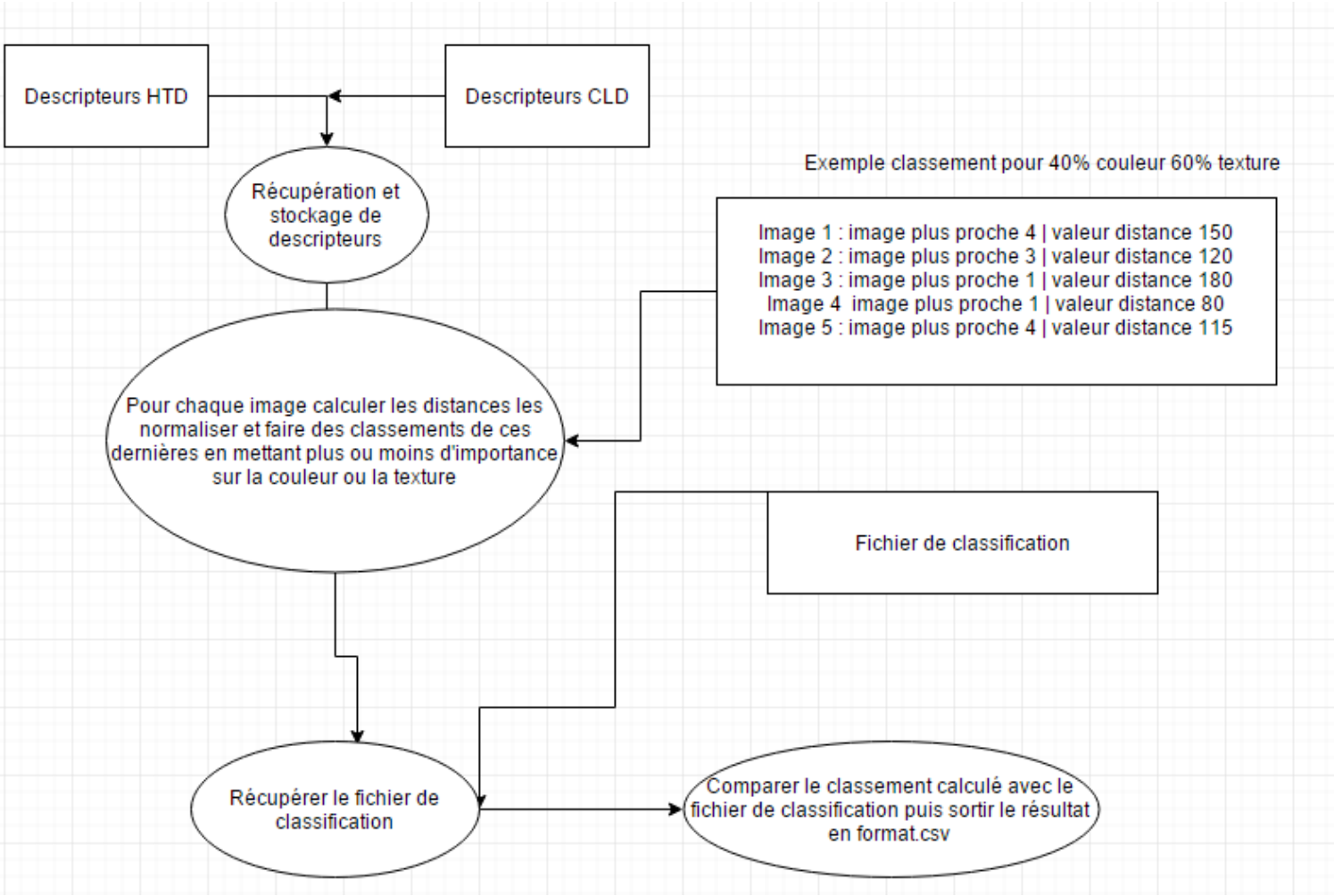
Le premier nombre correspond au numéro d'identification de l'image avec laquelle nous travaillons
Les nombre suivant est la liste de toutes les images qui ressemble à celle désignée par le premier id.

Il est important de noter que cette classification n’est pas un classement. Juste une liste des images les plus ressemblantes entre elles à l’intérieur de notre base.

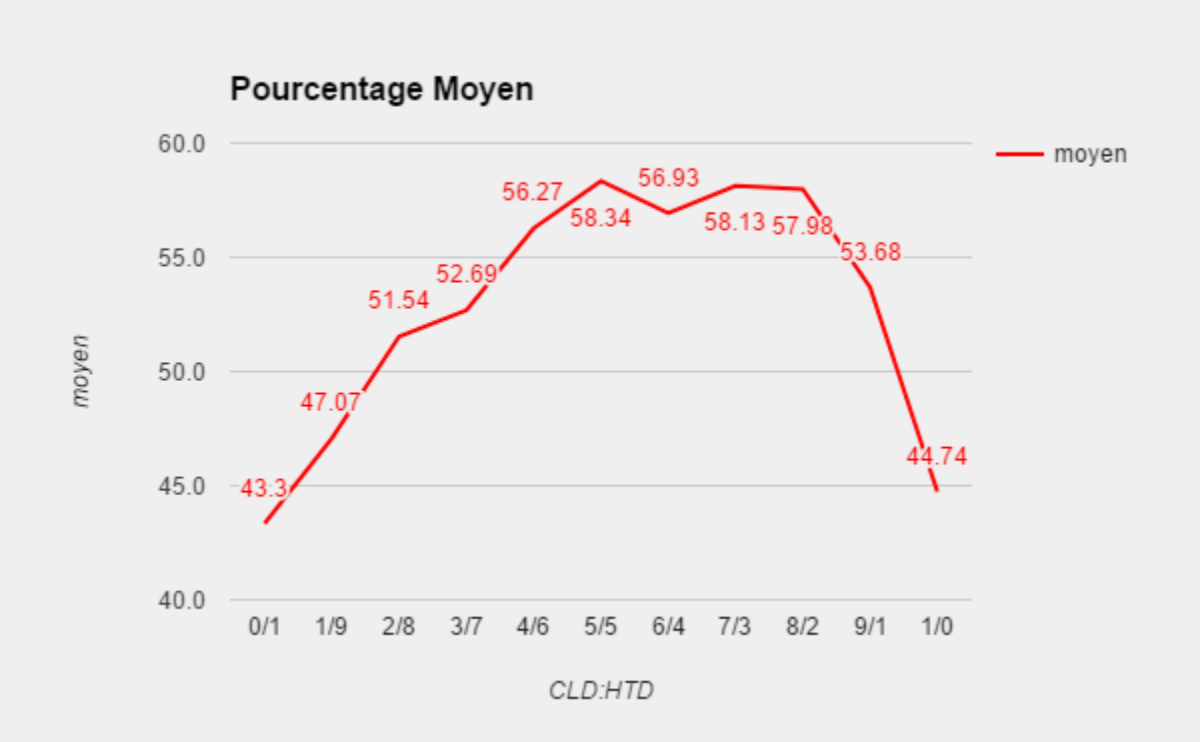
Notre système de test a donc été modifié par la même occasion. Le principe maintenant était de choisir une image A de la classification et demander à notre programme de sortir les N images ressemblants le plus à cette dernière.
Plus un grand nombre de ces N images se trouve à l’intérieur de notre classification plus l’algorithme utilisé est performant.
N étant un nombre à définir de manière expérimental.

Suite à ce changement de méthode de test et de classification il était nécessaire de développer un outil plus efficace pour l'interprétation de données.
En effet depuis le début du projet nous ne travaillons quasiment qu’avec excel pour trier les images par rapport à leurs distances. Cela nous faisait perdre un temps non négligeable surtout que seulement un de nous deux avait Excel tandis que l’autre devait se débrouiller avec open office.

Nous sommes donc partis du programme générant le fichier de distances inter-image pour en faire quelque chose de bien plus puissant.
Cette nouvelle version récupère les descripteurs à partir de fichier textes.
Calcule toutes les distances entre toutes les images.
Normalise les résultats pour pouvoir additionner des distances de descripteurs différents
Puis téléchargez le fichier contenant la classification humaine
Et compare cette dernière avec les résultats machines


Descripteurs HTD
Descripteurs CLD
Récupération et stockage de descripteurs
Exemple classement pour 40% couleur 60% texture
Image 1 : image plus proche 4 | valeur distance 150
Image 2 : image plus proche 3 | valeur distance 120
Image 3 : image plus proche 1 | valeur distance 180
Image 4 image plus proche 1 | valeur distance 80
Image 5 : image plus proche 4 | valeur distance 115
Pour chaque image calculer les distances les normaliser et faire des classements de ces dernières en mettant plus ou moins d'importance sur la couleur ou la texture
Fichier de classification
Récupérer le fichier de classification
Comparer le classement calculé avec le fichier de classification puis sortir le résultat en format.csv

Et voici les résultats obtenus avec ce programme :
CLD est l'importance donnée à l'information couleur de l'image
HTD est l'importance donnée à l'information de la texture
Le pourcentage moyen désigne le pourcentage d'image trouvée par le programme qui se trouvait dans la classification humaine.



Ratio	0/1	1/9	2/8	3/7	4/6	5/5	6/4	7/3	8/2	9/1	1/0
Score (%)	43.3	47.07	51.54	52.69	56.27	58.34	56.93	58.13	57.98	53.68	44.74

On peut donc voir que le système est le plus efficace lorsqu autant d'importance est donnée à la couleur quant à la texture.

Bien que ces résultats ces résultats restent légers quant à leur fiabilité.
En effet il faudrait une base d'image beaucoup plus grosse ainsi qu'une classification bien plus complexe si nous voulions avéré de l'efficacité de notre algorithme

Par rapport aux objectifs fixés sur ce sprint nous avons plutôt réussis les tâches que nous nous étions attribués. Nous n'avons juste pas pu prendre le temps de tester un autre système de descripteur d'images.

16) Tests Unitaires

Etant donné que notre rendu au client ne sera pas un programme mais plus un compte rendu de nos recherches nous n'avons pas fait beaucoup de tests unitaire.
Mais nous avons testé néanmoins un grande partie de notre programme traitant le calcul des distances entre les descripteurs

Le système marche de la manière suivante

1) On génère de faux descripteurs d'images contenant tous les même valeurs
2) On utilise notre système pour récupérer ses valeurs
3) On teste une par une chacune d'entre elle pour voir si elles sont conforme à celles d'origines
4) Vu que toutes les valeurs sont identiques dans chaque descripteur la somme des distances entre tous les descripteurs doit être égale à 0. On vérifie donc que cette somme fasse bien 0
5) Si toutes ces conditions sont remplies alors le test est validé.

```
bool testUnitaire() {

    std::ofstream HTDcreate, CLDcreate;
    std::vector<int> vecData;


    // création des fichiers de descripteurs

    HTDcreate.open("testHTD.txt");
    CLDcreate.open("testCLD.txt");

    // remplissage des fichiers

    for (int i = 0; i < 5; i++)
        HTDcreate << "10 20 30 40" << std::endl;

    for (int i = 0; i < 5; i++)
        for (size_t i = 0; i < 3; i++)
            CLDcreate << "10 20 30 40 " << std::endl;

    HTDcreate.close();
    CLDcreate.close();

    // extraction des descripteurs

    std::vector<DescripteurHTD> vecHTD = extractDescripteurHTD("testHTD.txt");
    std::vector<DescripteurCLD> vecCLD = extractDescripteurCLD("testCLD.txt");

    // création d'un vecteur à comparer avec le résultat de l'extraction

    for (size_t i = 0; i < 4; i++)
        vecData.push_back(10 * (i + 1));

    for (int i = 0; i < 5; i++) {

        if (!vecHTD[i].compare(vecData)) {
            std::cout << "step 1 " << std::endl;
            return false;
        for (size_t j = 0; j < 3; j++)
            if (!vecCLD[i].compare(vecData, j)) {
                std::cout << "step 2 " << std::endl;
                return false;
            }
    }

    // calcul des distances

    RankPerImg rankHTD = outputRank(vecHTD);
    RankPerImg rankCLD = outputRank(vecCLD);

    // verification de la somme des distances soit bien égale à 0

    if (rankCLD.getSumDistance() != 0 || rankHTD.getSumDistance() != 0) {
        std::cout << "step 3 " << std::endl;
        return false;
    }

    return true;
```

17) Partie Bibliographie

Introduction aux fonctions de hachages

Une fonction de hachage est donc une fonction qui en prenant en entrée une grande quantitée de données et de nature très diversifiées est capable de renvoyer un condensé d'informations manière optimisée par rapport à usage définie.

On appelle empreinte la sortie de taille fixe de la fonction de hachage. Le gros avantage de ce système est qu'il est capable d'identifier un objet de manière univoque. A la base les fonctions de hachage étaient utilisées afin de simplifier la gestion des grosses bases de données. En effet les système de hachages permettent de convertir des données de grandes tailles variables en empreinte bien plus petite en mémoire. Cela réduit de manière considérable le temps mis par des opération telles que le tri, l'insertion ou l'accès à un élément

Beaucoup de fonctions de hachages sont dites à "sens unique" cad qu'il est très facile de prendre un objet et de lui appliquer la fonction mais il est très difficile voir infaisable de partir d'un objet retournée par la fonction de hachage pour retrouver son état originel.

Dans ce genre de système une infime modification de la valeur d'entrée change radicalement les données qui apparaîtront en sortie.
Il arrive cependant que deux entrées de même nature arrive à donner la même sortie par la fonction de hachage. On parle alors de collisions.

Les fonctions de hachages cryptographique ne rentre pas dans la catégorie des systèmes de chiffrement. Notamment à cause de l'absence de clef dans le système ainsi que la non réversibilité de l'algorithme, mentionnée plus haut

exemple fonction de

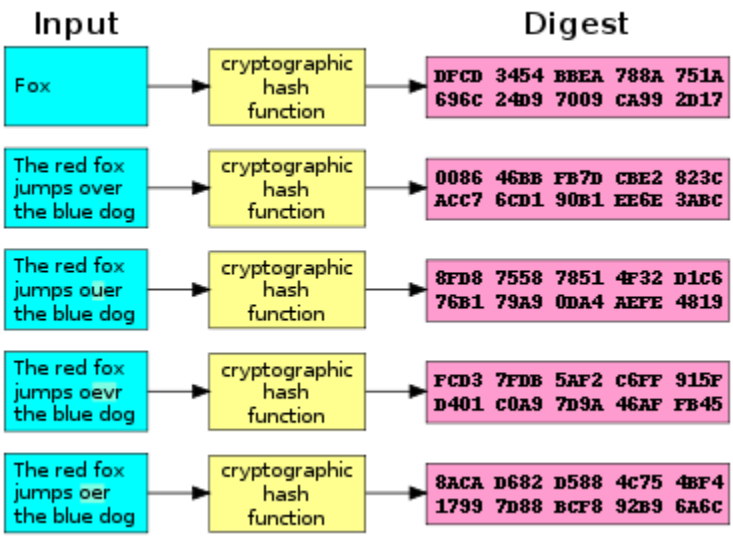


Figure A d'utilisation de hachage

Une fonction de hachage doit respecter deux critères de base :

- La compression : quelle que soit la taille du message d'entrée il faut que l'empreinte de ce message soit de taille réduite.

- L'efficacité : L'empreinte d'un message doit être rapide / facile à calculer.

En plus de ces deux critères de bases il existe d'autre éléments que les fonctions de hachages doivent vérifiées afin de fonctionner correctement.

- La resistance aux collisions. Une fonction de hachage est considérée comme résistante aux collisions si il est quasiment impossible de trouver deux messages différents M et M' tel que h(M) = h(M') avec une complexité meilleure que O(^n/2) opérations

- La résistance au calcul de préimages : Soit K une empreinte de la fonction de hachage H. H est considérée comme résistante au calcul de préimages si il est impossible de trouver un message L tel que L = H(K) avec une complexité meilleure que O(2^n)

Les fonctions de hachages dans le cadre de la reconnaissance d'images

Le but de notre projet est de détecter des similitudes entre deux images. Ce qui revient à prendre deux objets contenant beaucoup de données et à les comparer entre eux.
Les fonctions de hachages peuvent donc être d'une très grande utilité.
A contrario des fonctions de hachages cryptographiques nous travaillerons alors avec des fonctions de hachages perceptuelles.

Le terme perceptuel vient du fait que ce qui est encodée n'est pas les informations numériques mais les variables qui jouent sur la perception humaine de l'image (luminosité, histogramme des couleurs, structures des formes ….etc). On parle alors de correspondance perceptuell

Ces fonctions de hachages ne sont pas autant affectés par des changements légers sur la structure de l'image. De ce fait il serait possible de détecter des similitudes entre une image et une copie légèrement altérée.
Il s'agira donc de faire la différence entre des altérations "acceptables" (compressions, légères altération des couleurs, flou) et celles qui ne le sont pas

Figure B image d'origine

Figure C image avec modifications acceptables

Figure D Image avec modifications non acceptables

Il est donc nécessaire de faire une liste non exhaustive des modifications considérées acceptables et malveillantes

Modifications acceptables	Modifications malveillants
• Erreurs de transmission • Ajout de bruit • Compression et Quantification • Mise à l'échelle • Réduction de résolution • Rotation • Filtrage • Conversion de couleurs • Réglage de contraste • Changements de luminosité, teinte et de saturation • etc	• Suppression des objets sur l'image • Déplacement des éléments de l'image pour changer leurs positions • Ajout de nouveaux objets • Changements des caractéristiques de l'image : couleur, textures, structure, impression, etc . . . • Modifications du contexte de l'image : de jour ou d'emplacement • Les changements de conditions d'éclairage : manipulations d'ombre • Dégradation de la qualité • etc . . ..

Fonctionnement Global d'une fonction de hachage perceptuelle

Pour une grande majorité des système de hachage perceptuel il existe 4 grandes étapes afin de transformer une image en une empreinte perceptuelle. La transformation, l'extractions de valeurs utiles, la quantification et la crypto-compression

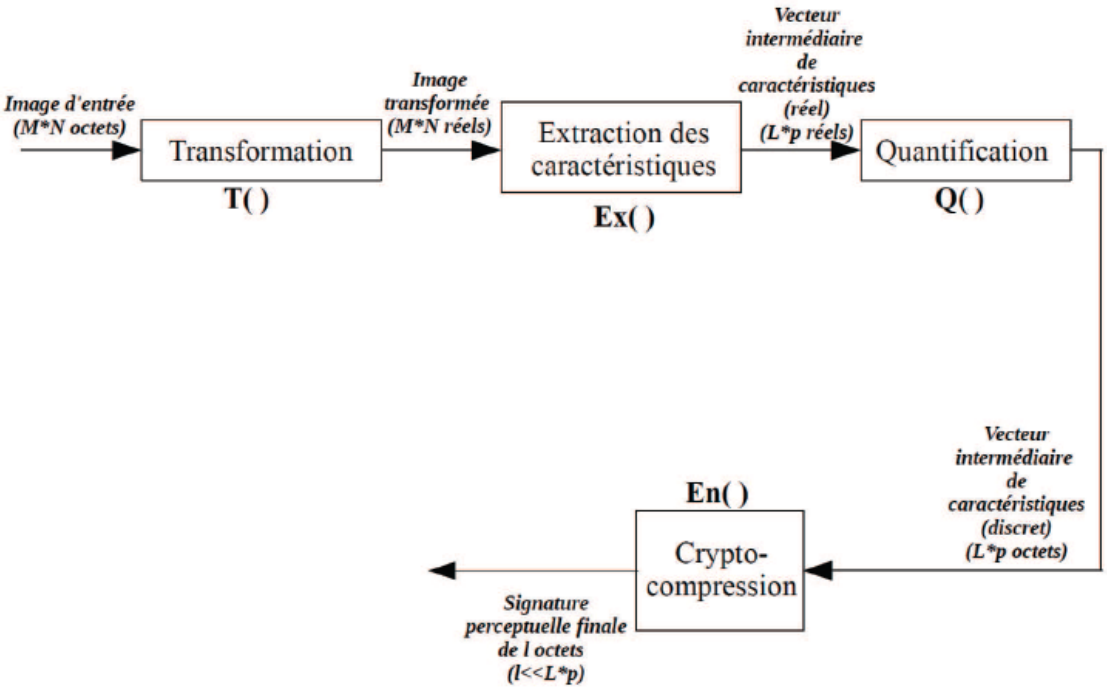


Image d'entrée
(M*N octets)
Transformation
T( )
Image
transformée
(M*N réels)
Extraction des
caractéristiques
Ex( )
Vecteur
intermédiaire
de
caractéristiques
(réel)
(L*p réels)
Quantification
Q( )
Vecteur
intermédiaire
de
caractéristiques
(discret)
(L*p octets)
En( )
Crypto-
compression
Signature
perceptuelle finale
de l octets
(l<<L*p)

Le but de la transformation est d'appliquer un certains nombre d'opérations sur l'image originale (altération des couleurs, lissage transformations fréquentielles réduction de la tailles) afin de rendre les valeurs extraites à l'étape suivante dépendante des valeurs des pixels de l'image ou des coefficients fréquentiels.

Une méthode courante en hachage perceptuel est d'appliquer une transformée en ondelette DWT ou une transformée en sinus discrète DCT.

NB : quand une DWT est appliquée il n'est souvent gardé que la bande LL. En effet, cette dernière est une version extrêmement grossière de l'image et possède souvent une grosse partie des informations perceptuelles de l'image

La deuxième étape consiste simplement à extraire ces fameuses caractéristiques visuelles et de les inclure dans un vecteur de réel. Il arrive qu'une caractéristique soit composée elle même d'un vecteur de valeurs réelles. De plus certains système perceptuel effectue parfois une seconde étapes d'extractions des valeurs pour ne garder que les plus pertinentes (cf méthodes sur la préservation grossière de l'image)

L'étape suivante nous permet alors de transformer ces réel en entiers par un processus de quantification. Bien qu'il soit possible de garder des systèmes de quantifications uniformes, les méthodes de quantifications adaptatives sont très largement utilisées dans les techniques de hachage perceptuel.

Enfin nous utilisons ces valeurs discrètes pour générer une courte signature perceptuelle.

Bien qu'une grande partie des techniques de hachages perceptuels sont basés sur le même principe d'extraction de caractéristiques visuelles il est possible des les distinguer en plusieurs catégories :

- Méthodes basées sur les statistiques d'image.
- Méthodes basées sur la relation.
- Méthodes basées sur la préservation grossière de l'image.

NB : Beaucoup de méthodes traitées dans la partie suivantes pourraient suffire dans le cadre d'utilisation de notre projet. La raison pour laquelle une grosse partie de ces techniques sont considérées incomplètes (dans un cadre d'utilisation plus général) vient de la problématique de la sécurité. En effet les créateurs de ces méthodes voulaient s'assurer qu'il ne soit pas possible pour une personnes tier d'arriver à trouver des collisions ou des préimages. Cela impacte souvent la robustesse d'une méthode à des opérations telles que la rotation ou le redimensionnement.

Voici donc quelque explications et exemple pour chaque catégorie.

Méthodes basées sur les statistiques d'intensités d'image

Dans le domaine de l'imagerie il existe un certains nombre de statistiques qui varient très peu lorsque de faibles opérations sont appliqués sur l'image. L'une des premières approches ces méthodes furent celles de Schneider et Chang. Leur idée était d'utiliser des valeurs d'intensité telles que la moyenne ou même la variances à partir de bloc découpés de l'image d'origine. Il se base alors sur les données extraites pour créer leur empreinte de l'image. Le vrai problème de cette méthode vient du fait qu'il est n'est pas difficile de modifier une image de telle sorte que son histogramme reste le même. D'autres méthodes ont été mise en place afin de combler cette faille. Notamment celle de El Ghoneimy et Tewfik, basée sur la normalisation du moment. Mais malgré l'amélioration par rapport à la méthode plus simpliste des statistiques d'intensités, ces dernières restent tout de même vulnérable aux attaque lié au recadrage.

Méthodes basées sur la relation

Cette catégorie de méthode est basée sur des relations invariantes des coefficients de transformée discrète ou transformée en ondelette. Lin and Chang ont notamment proposé en 2001 une méthode de hachage perceptuel basée sur un bloc de DCT pour authentifier Une image. Cette méthode est extrêmement robuste quand il s'agit de détecter une modification malveillantes d'une compression JPEG. En effet même après plusieur compression JPEG de suite cette technique considérera toujours que l'image n'a pas été modifiée. L'inconvénient de cette méthode, est par contre la très faible tolérance à toutes autres modifications telle que la rotation, redimensionnement ...etc .

Méthodes basées sur la préservation grossière de l'image

Ces méthodes consistent à ignorer une partie ou une composante de l'image afin de tolérer plus d'opérations non malveillantes. Par exemple Fridrich et Goljan ont proposé en 2000 une méthode qui basé sur la DCT qui au lieu de s'appliquer sur l'intégralité du domaine de fréquence travaillait uniquement avec les fréquences les plus basses.

La méthode DCT et distance de Hamming

Comme la transformée Fourier, la DCT est une méthode qui permet de passer du domaine temporel au domaine fréquentiel. Pour traiter une image, nous devons tout d'abord la diviser en plusieurs blocs de taille 8x8 ou 16x16.

NB Il est intéressant de noter que la taille du bloc choisie augmente de manière considérable le coût de calcul de la DCT.

Nous appliquons ensuite la DCT dans chaque bloc et pour chaque "couleur". Les valeurs contenues dans chaque bloc ne contiendront plus les valeurs de couleurs pour chaque pixel mais une carte des fréquences et de leurs amplitude à l'intérieur du même bloc.

Mathématiquement parlant voici comment s'exprime la DCT

$$DCT(i,j) = \frac{2}{N}C(i)C(j)\sum_{x=0}^{N-1}\sum_{y=0}^{N-2} pixel(x,y)cos[\frac{(2x+1)i\pi}{2N}]cos[\frac{(2y+1)j\pi}{2N}]$$

Tandis que la transformée DCT inverse s'exprime de la manière suivante

$$pixel(x,y) = \frac{2}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} C(i)C(j)DCT(i,j)cos[\frac{(2x+1)i\pi}{2N}]cos[\frac{(2y+1)j\pi}{2N}]$$

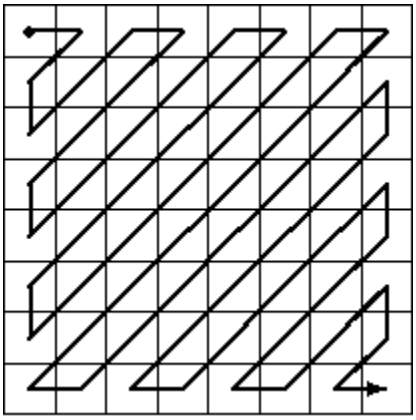
Quelque soit la formule que nous utilisons la constante C sera

$$C(x) = \begin{cases} \frac{1}{\sqrt{2}} & \text{pour } x = 0 \\ 1 & \text{pour } x > 0 \end{cases}$$

DCT(i,j) est un coefficient exprimé dans le domaine fréquentiel pour un pixel à la position y,x dans un bloc donné.

Une fois la DCT appliquée sur tous nos blocs nous rentrons ces derniers à l'intérieurs d'un tableau 1D.

Afin que les comparaison de signature marche correctement il est important que les blocs soient toujours rentré dans le même ordre.
L'ordre le plus courant étant l'ordre dit par zigzag (cf figure ci contre)



Ordre "zigzag"

Pour simplifier la comparaison, nous devons quantifier notre résultat. La façon la plus simple de faire est de convertir nos coefficients en valeur binaire.
Nous choisissons d'abord une valeur qui nous servira de seuil, par exemple la valeur moyenne des coefficient DCT dans un bloc. Les pixels qui sont inférieur à cette valeur moyenne prendront la valeur 0, tandis que les valeurs supérieurs seront égales à 1..
Une fois ces étapes terminées nous pouvons faire la comparaison entre deux empreintes d'images.

La méthode de la distance de hamming est une solution simple mais efficace pour répondre à ce problème.
Elle n'est néanmoins applicable que si les signatures comparées sont en binaire et possède la même taille.
Elle consiste simplement à faire une comparaison bit à bit des différentes signatures.
Par exemple, la distance de Hamming entre 1100 et 1011 et 3.

Les Descripteurs MEPG7 et leurs fonctionnements

Dans cette partie nous allons expliciter le fonctionnement des différents descripteurs que nous a souhaitons tester.

Avant de parler des descripteurs, nous devons d'abord présenter MPEG 7(Multimédia Contenu Description Interface).

Parce que la rapide augmentation des informations audiovisuelles sur Internet, la recherche informations intéressantes parmi la masse de données devient un travail pénible. Donc MPEG 7 a donc été créé pour répondre à cette problématique

MPEG 7 est le dernier membre de la famille MPEG, Le but principale de MPEG 7 est d'offrir un ensemble de méthodes et outils qui pourraient décrire différentes classes de contenus multimédias. MPEG 7 propose un standard pour mesurer les audiovisuel caractères. On appelle ce standard comme Descripteur(D). Et Descripteur Schéma(DS) décrit la structure de descripteur et la relation entre eux.

Descripteur de textures homogènes :

L'idée de ce descripteur : Si une image peut être partitionnée en un ensemble de régions qui ont une texture homogène, il est alors possible d'utiliser les caractéristiques de ces textures pour indexer les différentes régions d'une image.

Cette texture caractéristique basée sur le HVS (Système visuel humain)[1] correspond bien à quelques résultats d'expériences psychophysiques.

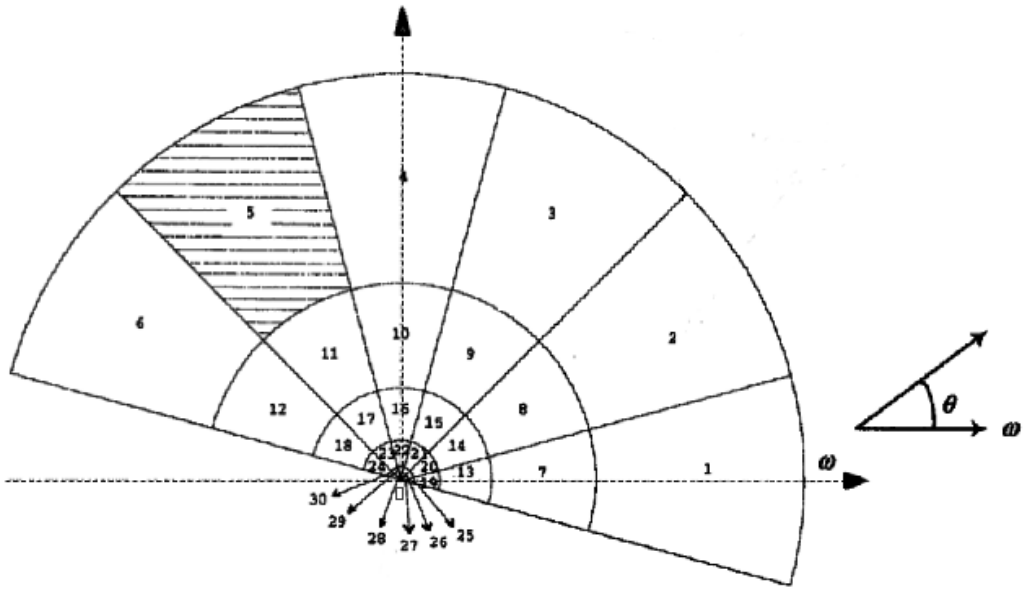


Figure 1 Répartition de la région de fréquence avec filtre HVS[2]

Comme figure 1, la meilleure représentation de sous-bans HVS pour la texture est une division du domaine de fréquence à octave-bandes(5 division) au long de la direction radiale et en largeur égale au long de la direction angulaire. Cette disposition des fréquences permet d'associer des textures extraites et le système de perception humaine.

Pour obtenir le descripteur, l'image est filtrée avec un ensemble de filtres de Gabor ces derniers étant sensible à l'orientation et à l'échelle.

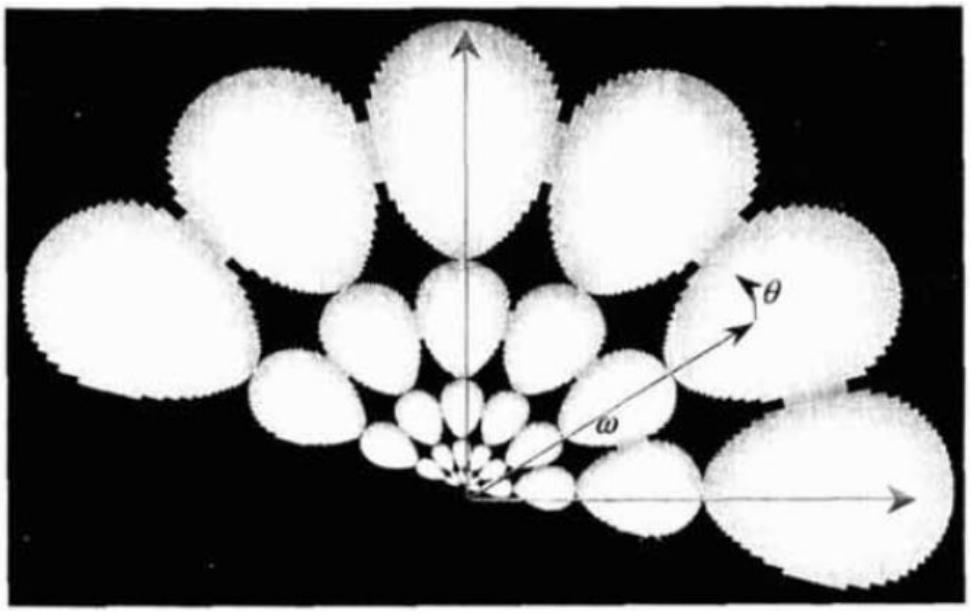


Figure 2 filtre de Gabor à coordonnées polaires[2]

Cet ensemble de filtres de Gabor s'écrit de la manière suivante :

$$\Gamma_{mk}(\omega,\theta) = \exp\left[-\frac{1}{2}\left(\frac{(\omega-\omega_m)^2}{\sigma_{\omega,m}^2}+\frac{(\theta-\theta_k)^2}{\sigma_{\theta,k}^2}\right)\right]$$ [3]

Paramètres des octave-bandes en direction radiale[2]

Index radial	0	1	2	3	4
Fréquence centrale(Après normalisation)	0.75	0.375	0.1875	0.09375	0.046875
Bande longeur	0.5	0.25	0.125	0.0625	0.03125

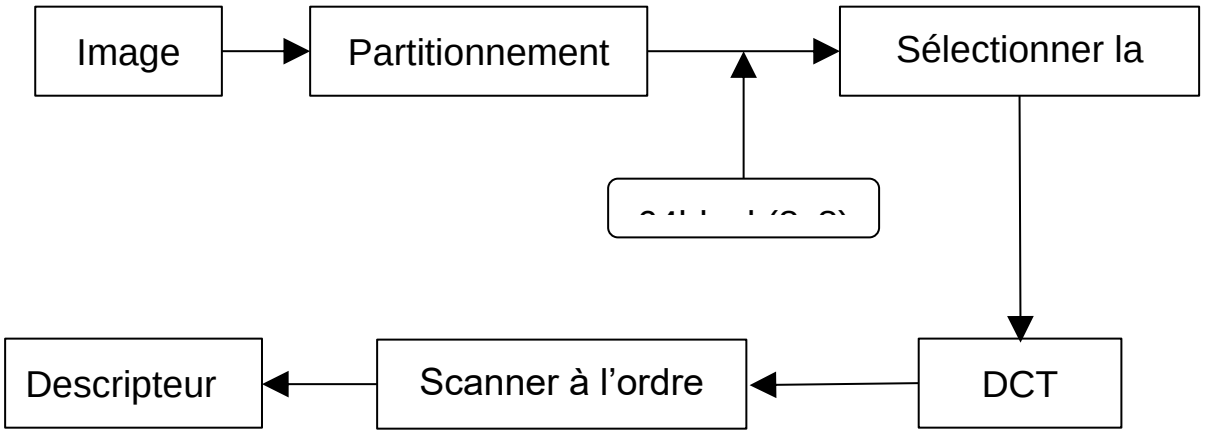
Enfin, nous calculons le descripteur par les sorties de ces filtres de Gabor. Le descripteur est composé de 62 nombres de 8bits pour chaque image. Pour ces 62 nombres, les deux premier sont la valeur moyenne de niveau de gris et écart-type d'image. Après pour chacun de ces 30 bandes, il y a la valeur moyenne Mi et l'écart-type Ei.

Donc Descripteur = [M,E,M1,...,M30,E1,...,E30]

Descripteur par disposition de couleur (color layout descriptor):

Ce descripteur est calculé à la base de espace YCrCb.
La transformation entre RGB et YCrCb :

Y = 0.299R+0.587G+0.114B
Cb = -0.169R-0.331G+0.500B
Cr = 0.500R-0.419G-0.081B



Sources :

A. HADMI : Protection des données visuelles Analyse des fonctions de hachage perceptuel (2012)

N. MERABET M. MAHLIA : Recherche d'images par le contenu (2011)

C. ZAUNER : Implementation and Benchmarking of Perceptual Image Hash Functions (2010)

Dr. N. Krawetz : Looks like it (2011)

http://www.cfar.umd.edu/~fer/High-resolution-data-base/hr_database.htm

Frank Nack , Adam T. Lindsay ．Everything You Wanted to Know about MPEG-7,part1[J]．IEEE Trans. On Multimedia,2000,2:7-13

I. Fogel and D. Sagi, "Gabor filters as texture discriminator" Biol. Cybern., vol. 61, pp. 103–113, 1989

Yong Man Ro and Munchurl Kim, "MPEG-7 Homogeneous Texture Descriptor", ETRI Journal, Volume 23, Number 2, June 2001

https://www.cs.auckland.ac.nz/courses/compsci708s1c/lectures/Glect-html/topic4c708FSC.htm#homogen

Les sources suivantes proviennent de site chinois

http://blog.csdn.net/newchenxf/article/details/51719597

http://blog.csdn.net/jubincn/article/details/6882179

http://blog.csdn.net/jubincn/article/details/6882179