Rapport Projet ACP : Livrable 1

ING1 GI Groupe 11

23 avril 2025

MONDINA Antonin
VALENTE Mathias
GINESTE Thomas
CHOSSON Clément
THIEBAUT Mattias

Table des matières

Contexte général du projet :

Le traitement d'images est un domaine clé en informatique, avec des applications dans la photographie numérique, le médical ou la sécurité. Dans le cadre de ce projet, nous nous intéressons à une problématique classique mais toujours d'actualité : le débruitage d'images numériques.

Une image acquise par un capteur, comme une caméra ou un scanner, est souvent altérée par différents types de bruit. Le bruit que nous considérons ici est le bruit gaussien additif, modélisé comme un signal aléatoire centré, généralement issu de perturbations électroniques internes au capteur ou de conditions d'acquisition difficiles (faible lumière, température, etc.). Ce bruit est non structuré, souvent peu visible à l'œil nu, mais peut considérablement nuire à la qualité de traitement et d'analyse de l'image.

L'objectif principal de ce projet est de restaurer une image dégradée par du bruit gaussien en estimant l'image originale la plus probable. Pour cela, nous utilisons une méthode statistique puissante : l'analyse en composantes principales (ACP) ou, en anglais, Principal Component Analysis (PCA). L'ACP permet de transformer un ensemble de données corrélées (ici, les petits blocs d'image appelés patchs) en un espace où les données sont représentées de façon plus compacte et moins redondante.

Nous étudierons et comparerons deux variantes principales de cette méthode : une approche globale (PGPCA), où l'analyse est faite sur l'ensemble des patchs de l'image, et une approche locale (PLPCA), qui applique l'ACP sur des zones restreintes.

Étapes principales du projet :

La première phase consiste à simuler la dégradation d'une image propre par l'ajout d'un bruit gaussien additif centré. Pour cela, une fonction noising(X0, σ) sera implémentée afin de générer plusieurs versions bruitées d'une image d'origine, avec différentes valeurs d'écart-type du bruit (typiquement σ = 10, 20, 30). Cette étape permettra de constituer une base de données d'images bruitées à traiter par la suite.

La deuxième étape porte sur l'extraction et la vectorisation de patchs. À partir de chaque image bruitée, de petits blocs carrés (patchs) de taille s × s sont extraits à l'aide de la fonction ExtractPatches(Xb, s), puis vectorisés avec VectorPatchs(Ypatchs) pour être analysés mathématiquement. Deux approches sont à considérer : une approche globale, dans laquelle l'ACP sera appliquée à tous les patchs extraits de l'image entière, et une approche locale, qui consiste à découper l'image en imagettes W × W, puis à appliquer l'ACP sur les patchs extraits localement.

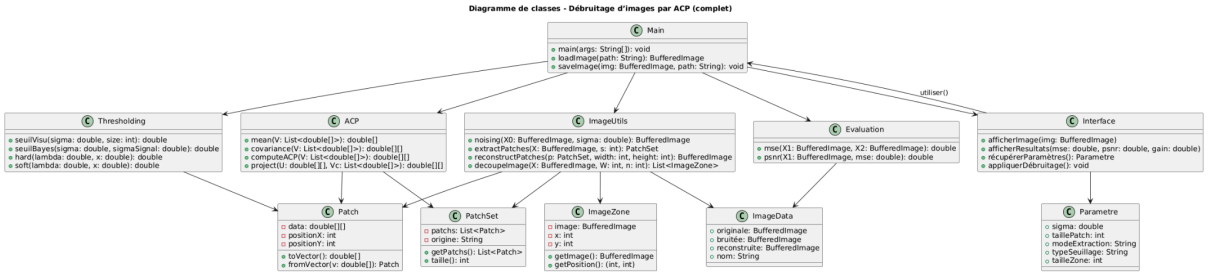
Une fois les patchs préparés, l'étape suivante consiste à effectuer l'Analyse en Composantes Principales. Cette méthode permet d'identifier une base orthonormale qui met en évidence les directions principales du signal et atténue le bruit. Les fonctions MoyCov(V) et ACP(V) permettront de calculer le vecteur moyen, la matrice de covariance, et d'extraire les composantes principales par décomposition.

La quatrième phase repose sur la projection et le seuillage. Les patchs sont projetés dans la base obtenue via Proj(U, Vc) puis leurs coefficients sont filtrés pour supprimer les valeurs considérées comme du bruit. Deux méthodes de seuillage seront comparées : le seuillage dur (SeuillageDur) et le seuillage doux (SeuillageDoux), avec des seuils déterminés selon les approches VisuShrink (SeuilV) ou BayesShrink (SeuilB). L'objectif est de réduire le bruit sans dégrader l'information utile.

Une fois les patchs nettoyés, l'étape de reconstruction de l'image débruitée est amorcée. La fonction ReconstructPatches(...) permet de réassembler les patchs filtrés pour former une image cohérente, tandis que ImageDen(...) regroupe l'ensemble du pipeline de débruitage en une seule fonction. Le but est de produire une estimation de l'image d'origine, notée $\hat{X}$, la plus fidèle possible.

Enfin, une évaluation des performances est nécessaire pour valider l'efficacité de la méthode. À cette fin, deux indicateurs seront mis en œuvre : le MSE (Mean Squared Error) et le PSNR (Peak Signal-to-Noise Ratio). Ces métriques permettront de comparer les différentes stratégies de traitement (choix du seuil, type de seuillage, ACP locale vs globale) et de justifier les décisions prises au cours du projet.

Diagrammes :



Le diagramme de classes ci-dessus présente l'architecture logicielle du projet de débruitage d'images par Analyse en Composantes Principales (ACP). Il met en évidence les principales classes développées, leurs responsabilités, ainsi que les interactions entre elles.
La classe Main constitue le point d'entrée du programme. Elle joue un rôle central en orchestrant les différentes étapes du traitement d'image, depuis le chargement de l'image jusqu'à la visualisation du résultat final.

La classe ImageUtils regroupe toutes les fonctions liées à la manipulation des images, notamment l'ajout de bruit, l'extraction et la reconstruction des patchs, ainsi que la gestion du découpage en zones (dans le cas d'un traitement local).
Les patchs extraits sont modélisés par la classe Patch, qui stocke les données d'un petit bloc d'image ainsi que sa position. Ces patchs peuvent être vectorisés afin d'être traités mathématiquement.

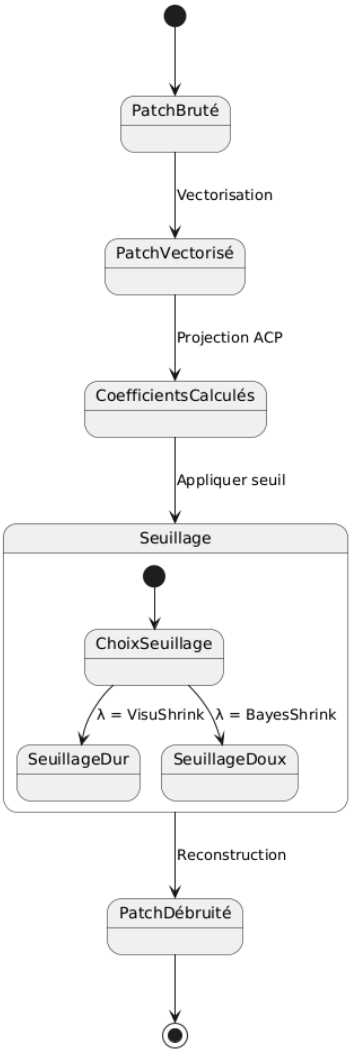
La classe ACP gère l'analyse en composantes principales : elle calcule la moyenne des vecteurs, leur matrice de covariance, puis extrait une base de vecteurs propres à l'aide d'une décomposition en valeurs singulières.
Les coefficients projetés sont ensuite traités par la classe Thresholding, qui applique un seuillage (dur ou doux) en fonction du type de filtre choisi (VisuShrink ou BayesShrink).

L'évaluation de la qualité du débruitage est assurée par la classe Evaluation, qui calcule des indicateurs standards comme le MSE (Mean Squared Error) et le PSNR (Peak Signal to Noise Ratio).
Enfin, la classe ImageZone est utilisée dans le cas d'un découpage local : elle permet d'associer une zone de l'image à ses coordonnées. Les relations entre les classes expriment les dépendances fonctionnelles du système, avec Main qui pilote l'ensemble des modules.
La classe PatchSet regroupe une collection de patchs extraits de l'image. Elle facilite la gestion des ensembles de patchs pour les traitements collectifs comme l'ACP. Elle propose des méthodes pour accéder à la liste de patchs et connaître leur nombre total.

ImageData regroupe, dans une même structure, les trois versions successives d'une image : l'image originale, la version bruitée et la version reconstruite après traitement. Elle stocke également un nom d'identification, utile pour suivre les résultats de plusieurs essais ou images.

La classe Interface gère l'interaction avec l'utilisateur. Elle permet de charger une image, de définir les paramètres de traitement, de lancer l'algorithme de débruitage, et d'afficher les résultats à l'écran (image débruitée, valeurs de PSNR et MSE, gain). Elle constitue la passerelle entre la logique métier et l'utilisateur.

La classe Paramètre encapsule tous les paramètres de traitement saisis par l'utilisateur via l'interface graphique : niveau de bruit (σ), taille des patchs, méthode d'extraction (globale, locale, hiérarchique), type de seuillage (dur ou doux), et taille des zones locales. Elle permet de centraliser et transmettre facilement ces réglages aux différents modules.

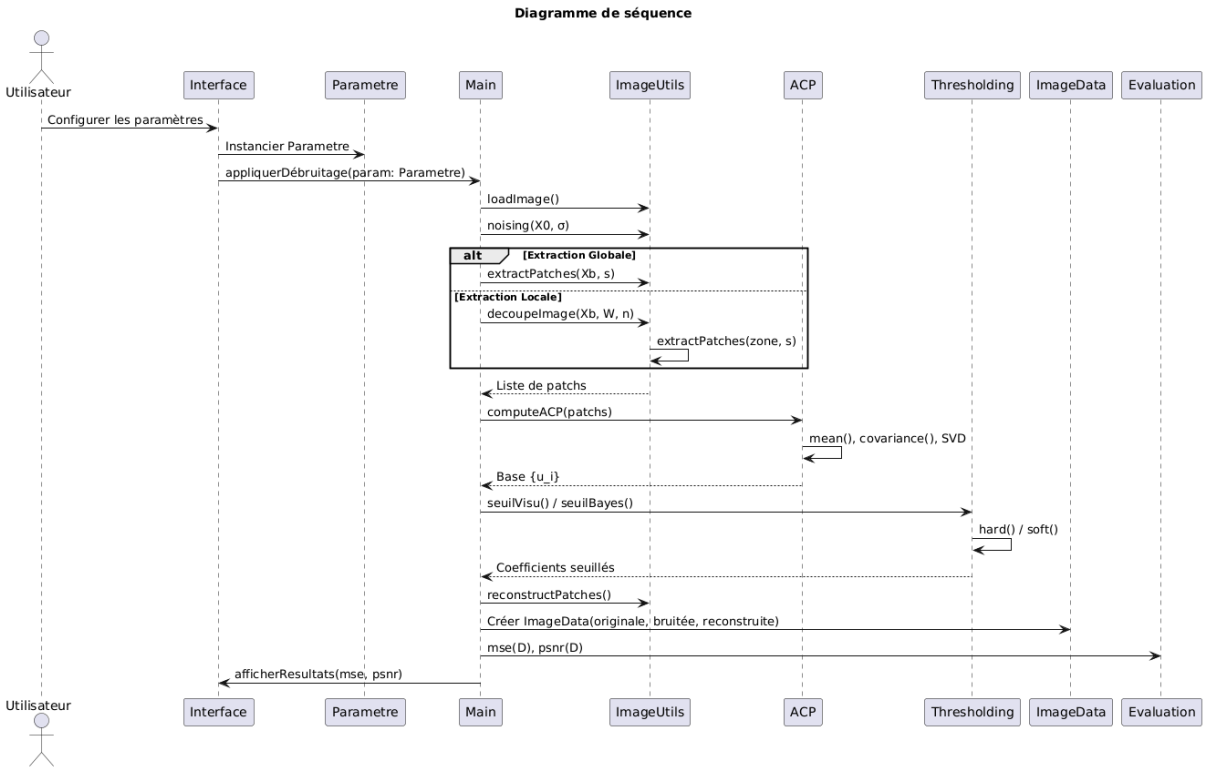


Le diagramme d'états représente le cycle de vie d'un patch tout au long du processus de traitement. Il illustre la manière dont un bloc d'image bruité évolue progressivement vers une version débruitée à travers les différentes étapes analytiques.

Le patch initial est extrait de l'image bruitée et se trouve dans l'état PatchBruté. Ce patch est ensuite vectorisé, ce qui lui permet d'entrer dans l'état PatchVectorisé. À partir de là, une projection dans l'espace des composantes principales est effectuée : le patch atteint alors l'état CoefficientsCalculés, où il est représenté par un ensemble de coefficients projetés.

Une étape de seuillage est ensuite déclenchée. Celle-ci peut suivre deux branches : le SeuillageDur, basé sur un seuil déterministe (VisuShrink), ou le SeuillageDoux, utilisant un seuil bayésien (BayesShrink). Ces deux traitements sont modélisés dans un état composite nommé Seuillage, avec une transition conditionnelle en fonction de l'algorithme choisi.

Enfin, une fois le seuillage terminé, le patch est reconstruit dans sa version débruitée, correspondant à l'état final PatchDébruité. Ce diagramme offre une vue claire du cheminement d'un patch à travers les transformations successives, du bruité vers le propre.

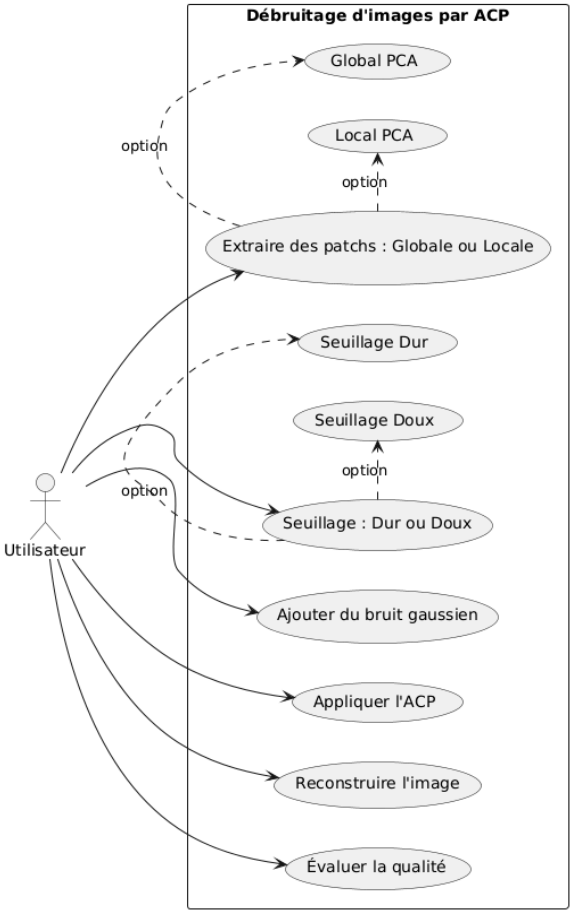


Le diagramme de séquence illustre l'enchaînement temporel des interactions entre les différents composants du système lors du traitement d'une image. Il s'agit d'un scénario dynamique mettant en évidence les appels de méthodes et les dépendances logiques entre les objets.

Le processus débute avec l'intervention de l'utilisateur, qui déclenche l'exécution via la classe Interface. Cette dernière fait appel à la classe Main qui fait appel à ImageUtils pour appliquer un bruit gaussien à l'image d'origine (noising). Ensuite, une phase d'extraction des patchs est déclenchée. Deux cas sont possibles : soit une extraction globale sur toute l'image, soit un découpage local en zones, avec extraction individuelle dans chaque sous-zone.

Une fois les patchs extraits, Main envoie les vecteurs à la classe ACP, qui effectue l'analyse : calcul de la moyenne, de la covariance, puis décomposition en vecteurs propres.
Les coefficients ainsi obtenus sont ensuite soumis à un seuillage, qui peut être dur ou doux selon le paramétrage choisi par l'utilisateur. Ce seuillage est réalisé via la classe Thresholding.
Enfin, Main utilise ImageUtils pour reconstruire les patchs filtrés en une image complète, puis appelle Evaluation pour calculer les métriques de qualité. Les résultats sont ensuite affichés à l'utilisateur via un module graphique.

Ce diagramme permet donc de visualiser l'intégralité du flux de traitement de manière chronologique, en explicitant les échanges entre les entités logicielles.



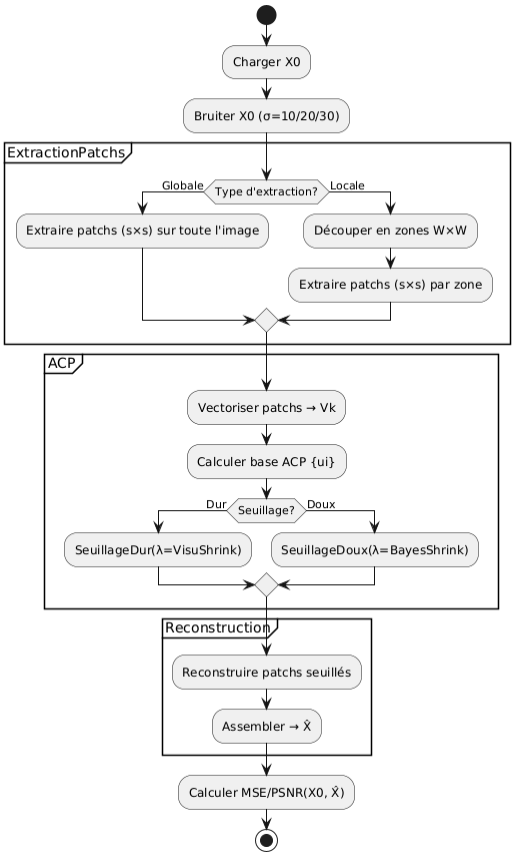
Le diagramme de cas d'utilisation met en évidence les fonctionnalités principales offertes par notre application de débruitage d'images, vues du point de vue de l'utilisateur final. L'acteur principal ici est l'utilisateur, qui interagit avec le système pour déclencher les différentes phases du traitement.

Parmi les cas d'utilisation principaux, on retrouve l'ajout de bruit gaussien à une image d'origine, l'extraction de patchs, l'application de l'Analyse en Composantes Principales (ACP), le seuillage des coefficients projetés (avec un choix entre les méthodes dur ou doux), la reconstruction de l'image débruitée, ainsi que l'évaluation de la qualité de cette reconstruction par les métriques MSE et PSNR.

Certaines actions sont enrichies par des cas d'utilisation secondaires. Par exemple, l'extraction de patchs peut être réalisée selon deux modes : extraction globale, où les patchs sont prélevés sur l'ensemble de l'image, ou extraction locale, où l'image est d'abord découpée en zones.

De même, l'application de l'ACP implique le calcul d'une base orthonormale issue de la covariance des vecteurs patchs. Enfin, le seuillage demande un choix du seuil à appliquer, selon deux méthodes classiques : VisuShrink et BayesShrink.

Ce diagramme résume donc clairement les fonctions attendues du programme et leurs variantes possibles, en plaçant l'utilisateur au cœur du système.



Le diagramme d'activité décrit de manière séquentielle les étapes du processus de débruitage, depuis l'importation de l'image jusqu'à l'obtention des résultats évalués. Il illustre le workflow fonctionnel que suit le programme.

Le processus commence par le chargement d'une image d'origine (notée X_0), à laquelle est appliqué un bruit gaussien d'écart-type σ, généralement fixé à 10, 20 ou 30 selon les scénarios à tester.

Ensuite, une phase d'extraction de patchs est initiée, organisée en partition spécifique dans le diagramme. L'utilisateur peut choisir entre une extraction globale, où l'image est découpée en patchs s×s uniformément, ou locale, qui consiste à découper l'image en zones W×W avant d'extraire les patchs dans chaque sous-zone. Ce choix permet de tester des stratégies de traitement différentes (notamment dans l'évaluation finale).

La seconde grande étape est dédiée à l'ACP. Elle commence par la vectorisation des patchs, suivie du calcul d'une base ACP. Après cela, un seuillage est appliqué sur les coefficients obtenus : soit un seuillage dur, basé sur la méthode VisuShrink, soit un seuillage doux, avec un seuil calculé selon BayesShrink.

Enfin, une phase de reconstruction assemble les patchs filtrés pour reconstituer l'image débruitée $\hat{X}$. L'image finale est comparée à l'image d'origine X_0, à l'aide des métriques MSE et PSNR, afin de quantifier la qualité du débruitage.

Ce diagramme d'activité est précieux car il synthétise le flux logique du traitement, avec des alternatives claires et des partitions qui séparent proprement les étapes conceptuelles du pipeline.

Organisation de la base d’images :

Afin de garantir une gestion claire, évolutive et reproductible des données utilisées dans le projet, nous avons mis en place une architecture de dossiers structurée pour stocker les différentes images nécessaires au traitement.
Cette base est divisée en trois répertoires principaux :

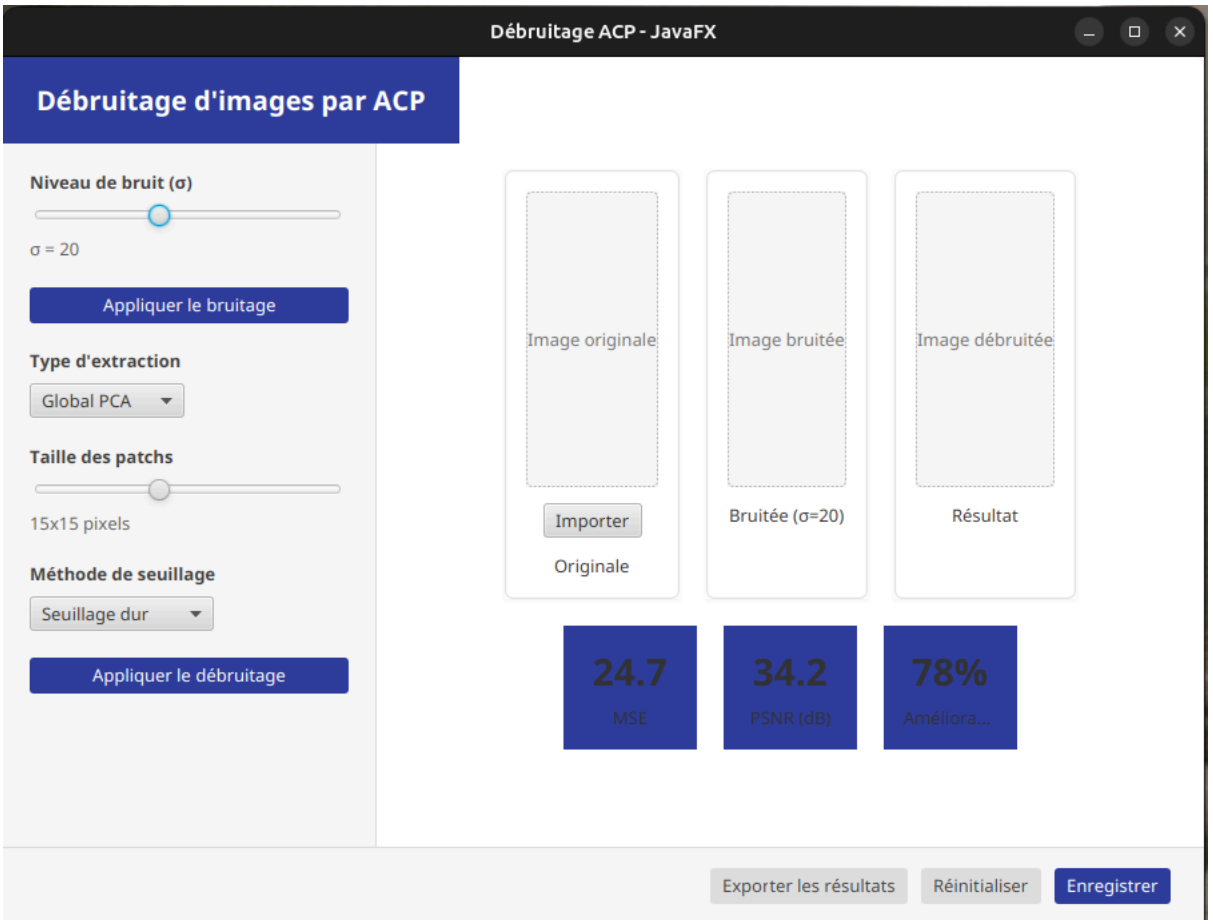
- Le dossier /images/sources/ contient les images sources non modifiées, servant de référence pour le bruitage. Ces images sont utilisées comme point de départ pour générer des versions bruitées.

- Le dossier /images/bruitees/ regroupe les versions bruitées des images originales. Chaque image y est associée à un niveau de bruit spécifique (par exemple : lena_noisy_sigma10.png, lena_noisy_sigma20.png, etc.), afin de tester les performances du débruitage dans différents cas de figure.

- Le dossier /images/results/ contient les images débruitées reconstruites après application de l’ACP et du seuillage. Ces fichiers permettent de comparer visuellement et numériquement les effets des différentes méthodes utilisées.

Cette arborescence permet non seulement de faciliter le traitement par lot, mais également d’assurer une séparation claire entre les données d’entrée, intermédiaires et de sortie.
Elle est également compatible avec l’automatisation du traitement dans l’environnement Java, les chemins étant prédéfinis dans les fonctions d’import/export du programme.

Analyse de l’IHM :

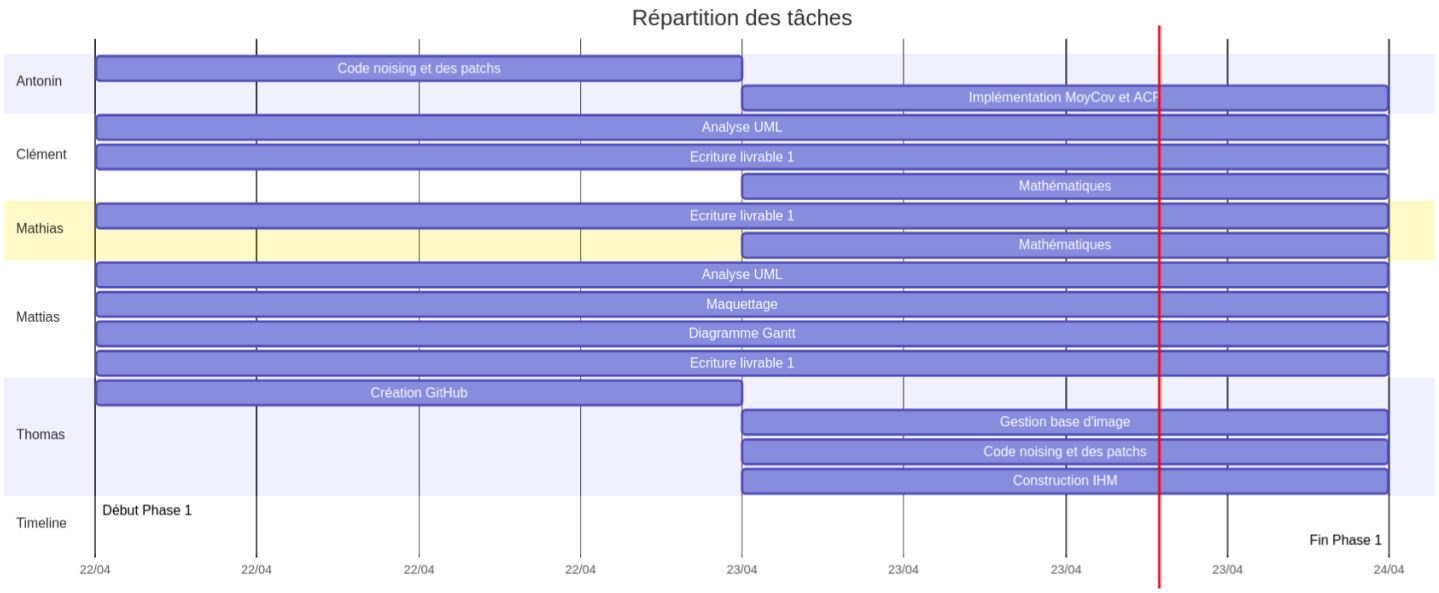


La maquette de l’interface utilisateur (IHM) est conçue pour offrir une navigation simple et efficace. Sur la partie gauche de l’écran, un panneau latéral regroupe les principaux paramètres de traitement d’image. L'utilisateur peut y ajuster le niveau de bruit (σ) à l’aide d’un slider, sélectionner le type d'extraction (comme le PCA global), définir la taille des patches avec un autre slider, et choisir la méthode de seuillage (dur ou doux). Un bouton intitulé « Appliquer le bruitage » permet d’ajouter du bruit à l’image selon les paramètres définis, tandis que le bouton « Appliquer le débruitage » lance le processus de suppression du bruit.

Au centre de l’interface, trois emplacements verticaux sont réservés à l’affichage visuel des images : l’image originale, l’image bruitée (avec indication du niveau de bruit, ici σ = 20), et l’image débruitée. Chacune est accompagnée d’un titre clair. Sous ces images, trois indicateurs de performance sont affichés : la MSE (Mean Squared Error), le PSNR (Peak Signal-to-Noise Ratio), et un pourcentage indiquant l’amélioration obtenue après le traitement.

Enfin, en bas à droite, trois boutons permettent de gérer l’ensemble des opérations : « Exporter les résultats », « Réinitialiser » les paramètres, et « Enregistrer » les configurations ou les résultats.

Répartition des tâches (phase 1) :



Lien GitHub :

https://github.com/Thomasgin/Projet-Java.git