



Ecole Polytechnique de l'Université de Tours
Département Informatique
64 avenue Jean Portalis
37200 Tours, France
Tél. +33 (0)2 47 36 14 14
polytech.univ-tours.fr

Projet Recherche & Développement 2021-2022

Détection des usures d'outil de coupe par vision assistée par ordinateur



POLYTECH[®]
TOURS

Entreprise

Polytech



Tuteur entreprise

Guillaume ALTMAYER

Étudiant

Alexis CASSAGNAUD (DI5)

Tuteur académique

Patrick MARTINEAU

Liste des intervenants

Entreprise

Polytech
64 avenue Jean Portalis
37200 Tours, France
polytech.univ-tours.fr



Nom	Email	Qualité
Alexis CASSAGNAUD	alexis.cassagnaud@etu.univ-tours.fr	Étudiant DI5
Patrick MARTINEAU	patrick.martineau@univ-tours.fr	Tuteur académique, Département Informatique
Guillaume ALTMAYER	guillaume.altmeyer@univ-tours.fr	Tuteur entreprise



Avertissement

Ce document a été rédigé par Alexis Cassagnaud susnommé l'auteur.

L'entreprise Polytech est représentée par Guillaume Altmeyer susnommé le tuteur entreprise.

L'Ecole Polytechnique de l'Université de Tours est représentée par Patrick Martineau susnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assume l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



Pour citer ce document

Alexis Cassagnaud, *Détection des usures d'outil de coupe par vision assistée par ordinateur*,
Projet Recherche & Développement, Ecole Polytechnique de l'Université de Tours, Tours,
France, 2021-2022.

```
@mastersthesis{
  author={Cassagnaud, Alexis},
  title={Détection des usures d'outil de coupe par vision assistée par ordinateur},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université de Tours},
  address={Tours, France},
  year={2021-2022}
}
```

Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
Table des figures	v
1 Introduction	1
1 Acteurs, enjeux et contexte	1
2 Objectifs.....	1
3 Hypothèses	2
4 Bases méthodologiques.....	2
2 Description générale	3
1 Environnement du projet	3
2 Caractéristiques des utilisateurs	3
3 Fonctionnalités du système	3
4 Structure générale du système.....	4
3 État de l'art / Veille technologique	5
1 Utilisation de l'apprentissage profond pour la segmentation d'images avec l'utilisation de l'architecture U-net	5
Apprentissage profond.....	5
Opérations des réseaux de neurones.....	6
Choix du modèle du réseau de neurones.....	6

	Limites.....	7
2	Solutions hors apprentissage automatique	7
	Détection de contours.....	7
	Transformée de Hough.....	8
4	Analyse et conception	10
1	Analyse	10
1.1	Hypothèses utilisées	10
1.2	Spécifications.....	11
2	Modélisation proposée.....	11
5	Mise en oeuvre	12
1	Introduction	12
2	Outils et librairie utilisés.....	12
3	Éléments d'implémentation, choix techniques	12
3.1	Analyse par contours et détection de lignes.....	12
3.2	Analyse par histogrammes	13
3.3	Analyse par couleurs.....	13
4	Analyse des résultats, évaluation, qualité	14
4.1	Résultats de la détection de contours	14
4.2	Résultats de l'analyse par histogrammes	14
4.3	Résultats de l'analyse par couleurs.....	15
4.4	Améliorations à prévoir.....	16
6	Bilan et conclusion	18
1	Bilan du semestre 9	18
2	Bilan du semestre 10.....	18
3	Bilan sur la qualité	19
4	Bilan auto-critique.....	19
	Annexes	20
A	Planification, gestion de projet	21
1	Evolution du projet	21
2	Description des tâches.....	22
	Tâche 1 : Analyse des travaux pré-existants.....	22
	Tâche 2 : Recherche de solutions	22
	Tâche 3 : Questions pour le client.....	22
	Tâche 4 : Etat de l'art.....	22
	Tâche5 : Rédaction rapport	22

Tâche 6 : Préparation à la soutenance	22
Tâche 7 : Soutenance.....	23
Tâche 8 : développement du processus de la détection de contour	23
Tâche 9 : Développement du processus de transformée de Hough.....	23
Tâche 10 : Tests et retour du client	23
Tâche 11 : Ré-ajustements et amélioration.....	23
Tâche 12 : Développement de la fonction pour la mesure du Vb	23
Tâche 13 : Développement du programme de gestion des données	23
Tâche 14 : Développement du programme d'entraînement du modèle de réseau de neuronne.....	23
Tâche 15 : Développement du programme d'analyse d'image	23
Tâche 16 : Ajout d'un modèle pour la segmentation d'image.....	24
B Description des interfaces	25
1 Interfaces homme/machine.....	25
C Cahier de Spécifications	26
1 spécifications Fonctionnelles.....	26
1.1 Fonctionnalités à développer	26
Description de la fonction de traitement d'image :.....	26
Description de la fonction de gestion des données :.....	26
Description de la fonction d'entraînement :.....	26
Description de la fonction d'analyse :	27
2 Spécifications non fonctionnelles	27
2.1 Contraintes de développement et conception	27
2.2 Contraintes de fonctionnement et d'exploitation.....	27
2.2.1 Performances.....	27
2.2.2 Capacités	27
2.2.3 Contrôlabilité	27
2.2.4 Sécurité.....	28
2.2.5 Evolution du système	28
D Cahier du développeur	29
1 Introduction	29
2 Organisation des sources	29
3 Descriptions détaillées des données exploitées.....	29
4 Descriptions détaillées des programmes	30
4.0.1 Analyse_contours_hough.....	30
4.0.2 analyse_histogrammes.....	30
4.0.3 analyse_couleurs	30

E	Document d'installation	32
F	Document d'utilisation	33

Table des figures

3 État de l'art / Veille technologique

3.1	Illustration apprentissage profond	5
3.2	réseau de neurones	6
3.3	Architecture U-net	7
3.4	Filtre gaussien	8
3.5	Transformée de Hough	8

4 Analyse et conception

4.1	Diagramme d'utilisation de la solution	11
-----	--	----

5 Mise en oeuvre

5.1	Résultat de la détection de contours	14
5.2	Résultat de la détection de lignes	15
5.3	Résultat de l'histogramme	15
5.4	Image originale	16
5.5	Extraction de la couleur	16
5.6	Masque de sortie	16
5.7	Résultat de l'histogramme	17

A Planification, gestion de projet

A.1	Le diagramme de Gantt initial	21
A.2	Le diagramme de Gantt revisité	21
A.3	Le diagramme de Gantt de la mise en oeuvre	22

1

Introduction

1 Acteurs, enjeux et contexte

Ce projet est issu d'une collaboration entre Polytech Tours et le CEROC (Centre d'Etudes et de Recherches sur les Outils Coupants) qui a émis un besoin lié au domaine de l'usinage. En effet, l'usinage est un procédé de fabrication soustractive permettant de produire des pièces mécaniques dans lequel est spécialisé le CEROC. Les besoins liés à l'analyse d'image pour la détection d'usure sont très importants car cette analyse à partir d'images est pour l'instant faite à la main et très régulièrement. La raison à cela est que l'usure des outils de coupe amène de lourds enjeux pour la qualité des pièces usinées : un outil a une durée de vie, et s'il est trop endommagé, l'usinage perd en qualité, voire peut causer des problèmes techniques. C'est pour cette raison que les pièces sont contrôlées régulièrement, ce qui implique le démontage, l'analyse et le remontage, afin de procéder ou non au remplacement de la pièce. Ce processus répété s'avère long et coûteux.

Le passage à l'industrie 4.0 consiste à automatiser l'analyse de l'usure des outils coupant afin de procéder au remplacement de ces outils uniquement lorsque cela est nécessaire. Cela conduira à un gain de temps considérable pour le CEROC, voire pour d'autres acteurs du monde de l'usinage.

Acteurs :

- Client : CEROC (Centre d'Etudes et de Recherches sur les Outils Coupants)
- MOA : M. Patrick Martineau comme encadrant et M. Guillaume Altmeyer comme représentant CEROC
- MOE : M. Alexis Cassagnaud

2 Objectifs

L'objectif du projet est de concevoir et de réaliser une solution d'analyse d'image capable d'apporter une aide à la détection d'usure des outils coupants afin de gagner en performances. On peut segmenter cette notion en trois niveaux d'analyse, les deux premiers pouvant répondre partiellement au besoin, sans remplacer une intervention humaine régulière. Il existe plusieurs types d'usures qu'on peut retrouver sur les outils coupants, le plus largement répandu étant

l'usure en dépouille, notamment sur les arêtes droites, on se focalisera sur celui-ci pour les deux premiers niveaux d'analyse.

Tout d'abord être en mesure de déterminer si l'outil est neuf ou bien usé. Ce premier niveau constituera l'objectif minimum à atteindre. Cette analyse minimaliste sera en mesure de répondre en partie au besoin du client et sera une première avancée vers l'automatisation de la détection d'usure.

Être capable de quantifier l'usure présente sur les outils via des mesures constituera le deuxième niveau à atteindre. Ces mesures permettront aux utilisateurs d'accéder à des informations précises quant à l'usure détectée et pourront prendre des décisions en fonction de ces informations.

Finalement, être capable de détecter tous les types d'usure sera la fonctionnalité qui complètera la solution d'analyse d'usure. Cela implique donc de revoir les deux premiers niveaux pour chacun des autres types d'usures existants. Finalement, la solution serait capable d'indiquer tous les types d'usures présents sur un outil, et ce, avec les mesures correspondantes.

3 Hypothèses

Les choix de techniques à utiliser pour répondre au besoin via l'analyse d'image sont fortement dépendants des données dont dispose le client. Avant tout, il est impératif d'avoir un accès à ces données pour développer une solution adaptée à ces dernières. Un échantillon d'au moins une centaine de données devra être livré au plus tard en décembre 2021, puis le reste des données au plus tard en février 2021.

La qualité est importante pour évaluer s'il est possible de les exploiter directement pour de l'apprentissage profond. Si la quantité de données est suffisante (de l'ordre d'un millier d'images pour obtenir des résultats fiables, et de l'ordre d'une centaine d'images pour obtenir des résultats approximatifs) et si une information relative à l'usure est associée à chaque image, alors on peut considérer que l'utilisation d'apprentissage profond est possible. Selon la nature de cette information, différentes méthodes peuvent être envisagées directement en rapport avec le besoin énoncé dans la section 1.2. Une information textuelle contenant une mesure ou bien une simple chaîne de caractère indiquant si l'image correspond à un outil usé ou neuf rendra possible une classification binaire. S'il existe une information graphique (masque binaire indiquant les zones d'usures sur l'image) alors une classification multi-classe ainsi qu'une quantification de cette usure sera possible.

Si ces conditions sur les données ne sont pas respectées, alors des techniques hors apprentissage automatique seront abordées. On se concentrera sur les deux premiers paliers d'analyse fixés dans la section 1.2, à savoir la détection et la quantification d'usure en dépouille (la plus fréquente dans le domaine de l'usinage).

4 Bases méthodologiques

Le déroulement du projet se fera selon un cycle en spirale afin de développer une première base de la solution, puis après des tests et des vérifications de ce premier prototype, le développement se poursuivra sur un nouveau prototype pour répondre un peu plus au besoin du client.

2

Description générale

1 Environnement du projet

Ce projet ne dépend d'aucun projet parallèle ni projet existant. Toutefois, il fait suite à un stage réalisé au CEROC durant l'été 2021 et qui a réalisé une première approche de la problématique de l'automatisation de l'analyse d'usure. Dans un premier temps, les données - des images d'outils de coupe - dont dispose les bases de données du CEROC ont été réorganisées et classées, ce qui permet d'avoir des informations relativement claires sur ces données : type d'outil, type d'usure, quantification de l'usure. Ces données seront la composante essentielle de la solution d'analyse d'image.

Une première approche de l'analyse d'image a été faite au cours de ce stage, toutefois les résultats n'ont pas été exploitables. Ce projet consiste donc à concevoir une solution aboutissant à des analyses satisfaisantes grâce à une utilisation plus poussée des méthodes d'analyse d'image.

La solution sera réalisée en langage de programmation python, ce qui la rendra portable et simple d'utilisation.

2 Caractéristiques des utilisateurs

On pourra identifier un type unique d'utilisateur de la solution que sont les ingénieurs du CEROC. On considérera donc que l'utilisateur possède une connaissance minimale de l'informatique qui lui permettra d'utiliser la solution sous toute ses fonctionnalités, voire qui lui permettra de modifier des paramètres. La compréhension des algorithmes mis en œuvre ne sera pas requise. L'utilisation de la solution ne nécessitera pas d'expérience préalable et se prêtera autant aux utilisateurs réguliers qu'à ceux occasionnels. Aucun système de droits d'accès d'utilisateur particulier ne sera mis en place.

3 Fonctionnalités du système

La solution est constituée de plusieurs programmes répartis en deux modules distincts. Le premier consiste en l'utilisation de techniques de traitements d'image tandis que le second consiste en l'utilisation d'apprentissage profond.

La première fonctionnalité de la solution via le premier module est

- L'analyse d'une image pour détecter la présence d'usure

Les fonctionnalités suivantes font appel au second module et permettent :

- L'analyse d'une image via un modèle de réseau de neurones pour en tirer des prédictions sur l'usure
- Le paramétrage de l'entraînement du modèle selon une base d'apprentissage, de validation et d'entraînement
- La réorganisation de données dans différents répertoires pour constituer les bases nécessaires au modèle (entraînement, validation et test)

4 Structure générale du système

La solution s'articule autour de plusieurs programmes, réalisés en langage de programmation python, qu'il est possible de lancer indépendamment.

- Un programme pour lancer une analyse d'image via un traitement d'image basé sur la détection de contours et l'opération de transformée de Hough. Il prend une image en entrée et retourne une nouvelle image en sortie où les zones d'usure sont visibles.
- Un programme pour lancer une analyse d'image par le modèle de réseau de neurones entraîné. Il prend une image en entrée et retourne une prédiction en sortie.
- Un programme pour entraîner le modèle à partir de plusieurs lots de données (entraînement, validation, test). Les hyperparamètres peuvent être ajustés en modifiant le programme. Il ne prend rien en entrée et retourne une sauvegarde du modèle ainsi qu'un fichier contenant une description des paramètres utilisés pour le modèle.
- Un programme pour restructurer ou réorganiser les répertoires des lots de données. Il permet, à partir de toutes les données disponibles, de les répartir dans trois répertoires différents qui correspondent aux données d'entraînement, de validation et de test.

3

État de l'art / Veille technologique

1

Utilisation de l'apprentissage profond pour la segmentation d'images avec l'utilisation de l'architecture U-net

Apprentissage profond

[L'apprentissage profond avec Python - Les meilleures pratiques / François Chollet / machine learning .fr]

L'apprentissage profond (Deep Learning) est un sous-domaine de l'apprentissage automatique (Machine Learning) qui consiste à interpréter des données selon des variables pour en extraire des informations utiles, et les catégoriser. Ce processus se base sur :

- Des points de données d'entrées (input data points)
- Des données de sortie attendus (expected output)
- Une manière de mesurer les performances de l'algorithme (ou feedback)

L'apprentissage automatique repose sur l'identification de variables issues de données. Un apprentissage est réalisé grâce à des données fournies en exemple puis l'algorithme est testé sur un nouveau lot (set) de données fournies, afin de pouvoir mesurer la performance de l'algorithme sur les données d'apprentissage et de test.

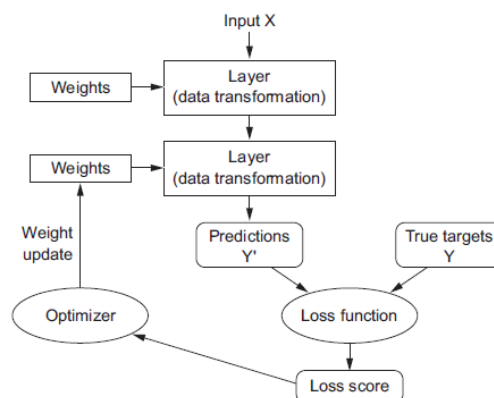


Figure 3.1 – Illustration apprentissage profond

L'apprentissage profond est une nouvelle approche basée sur l'utilisation de couches (layers) successives, la profondeur faisant référence au nombre de couches du modèle. On parle alors de réseaux de neurones. Les couches successives agissent à la manière de filtres qui vont, par le biais de petites opérations, transformer l'information d'origine en une représentation de plus en plus différente, et surtout de plus en plus utile pour interpréter l'information.

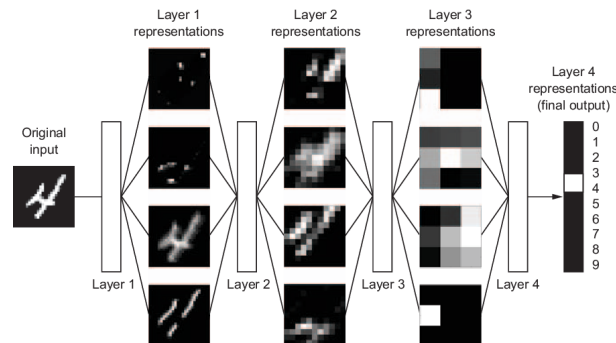


Figure 3.2 – réseau de neurones

Pour chaque couche du modèle, les différents poids constituent les paramètres de la couche. C'est ce paramétrage qui permet au réseau de neurones de trouver la bonne interprétation des données. C'est lors de l'apprentissage du modèle que le paramétrage est ajusté, notamment grâce à une fonction de perte (loss function) ou fonction objective qui évalue la distance entre la prédiction du modèle et le résultat attendu, et l'optimiseur (optimizer) qui modifie les poids des différentes couches. Finalement, c'est cette particularité de l'apprentissage profond qui en fait un outil très puissant : le réseau de neurones ajuste lui-même les représentations des données utiles. C'est pourquoi de grandes avancées ont été faites dans les problèmes de perceptions tels que la classification d'image, la reconnaissance de voix, etc.

Opérations des réseaux de neurones

Plusieurs traitements d'image sont effectués par un réseau de neurones. La convolution est une opération qui donne en sortie une image modifiée et qui, après de nombreuses boucles du réseau de neurone, pourra faire ressortir des informations de l'image initiale. On parle d'extraire des cartes de caractéristiques. Il existe plusieurs types de convolution, la convolution 2D est celle qui nous intéressera ici. On parle alors de réseaux de neurones convolutifs (CNN pour convolutional neural network). Une fonction d'activation doit être choisie pour accompagner l'opération de convolution, et c'est généralement la fonction « relu » qui est utilisée et qui applique une fonction $\max(0, x)$ où x est une valeur de la matrice d'image. L'opération MaxPooling consiste à diminuer la taille de l'image en retenant les informations les plus significatives. On applique généralement cette opération avec des strides de taille (2x2), on retient donc la valeur maximale de chaque matrice (2x2) de l'image, ce qui conduit à diminuer la taille de l'image par 2.

Choix du modèle du réseau de neurones

Afin de développer une solution de détection des usures d'outils de coupe par vision assistée par ordinateur (ou computeur vision), il est nécessaire de s'intéresser aux méthodes de Machine Learning et en quoi elles permettent de répondre au besoin. Ici, le besoin du client en termes d'analyse d'image correspond précisément à de la segmentation d'image. L'ordinateur doit être en mesure de reconnaître la surface d'usure de l'outil, c'est-à-dire déterminer la zone (ou tous les pixels) de l'image correspondant à l'usure et la classifier selon son type. A partir de cette segmentation, des traitements pourront être effectués pour quantifier l'usure via une méthode pour chaque type d'usure existant. Différentes architectures de modèles de Deep Learning existent pour réaliser cette segmentation d'image. Celle proposée ici est l'architecture u-net qui possède des avantages intéressants : d'une part elle a

obtenus de meilleurs résultats que d'autres architectures au moment de sa parution (notamment l'architecture VGG16), d'autre part c'est cette architecture qui a été employée dans l'étude du laboratoire WZL qui a réussi à obtenir des résultats qui semblent assez satisfaisants. [Digital image processing with deep learning for automated cutting tool wear detection]

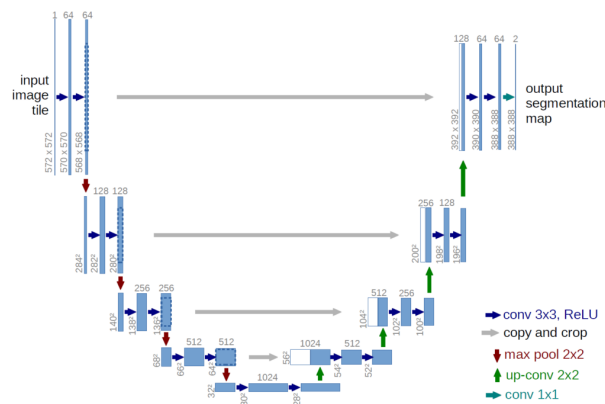


Figure 3.3 – Architecture U-net

Le schéma ci-dessus décrit le processus d'analyse grâce à plusieurs types d'opérations réalisés de manière successive : convolutions, max pooling, transposées de convolutions et skip connections. La particularité de cette architecture est qu'elle utilise une notion de « décodage » pour rétablir la dimension de la prédiction à la taille de l'image d'entrée et ainsi obtenir un masque de prédiction en sortie.

Limites

Pour utiliser cette méthode de segmentation d'image il est nécessaire de fournir les données d'entrées adaptés à l'apprentissage du modèle. Le nombre d'images doit être assez conséquent pour que le modèle puisse être suffisamment robuste face à une nouvelle image. Les étiquettes associées aux images sont tout autant importantes. La segmentation d'image nécessite au préalable qu'un masque soit dessiné humainement et précisément pour chacune des images constituant le set de données d'entraînement, ce qui peut représenter un travail conséquent.

2 Solutions hors apprentissage automatique

[Cours d'analyse d'image - Chapitre 2 : transformations locales : filtrage, détection de contours / Jean-Yves

Il existe des solutions de traitement d'images qui permettent d'extraire des informations tels que des contours, des zones ou des formes particulières. Ces solutions doivent être sélectionnées selon leurs particularités, les images d'entrées à analyser et les formes que l'on veut analyser.

Détection de contours

On parle de contours dans une image pour désigner les frontières entre un objet et un fond, ou entre deux objets. Pour caractériser les zones de contours, on recherche les changements brutaux d'intensité dans l'image, ou les discontinuités. Toutefois, il faut rester vigilant car une discontinuité dans l'image n'appartient pas forcément au contour d'un objet. Une ombre par exemple, ou un changement de couleur peut rendre la détection de contour difficile. L'objectif est donc, au sein d'une image, de détecter les pixels appartenant à un contour en se basant sur un changement d'intensité. Les résultats obtenus restent des probabilités que chacun des pixels désignés fasse partie d'un contour.

Ici, on s'intéressera à l'utilisation du filtre Canny pour la détection de contours. Celui-ci met en œuvre plusieurs opérations successives.

- Application d'un filtre gaussien pour enlever le bruit. Le filtre gaussien est utilisé pour lisser une image et réduire le bruit.

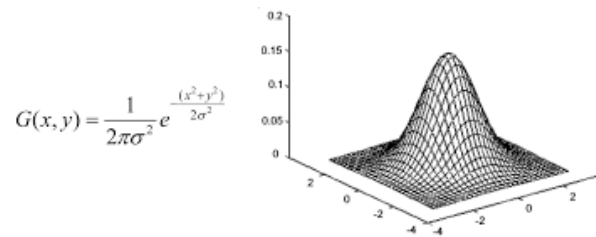


Figure 3.4 – Filtre gaussien

- Calcul de l'intensité du gradient dans l'image via un filtre de Sobel. On calcule d'une part la composante horizontale du gradient G_x : On calcule d'autre part la composante verticale du gradient G_y : La norme du gradient $|G|$ étant la somme des normes des composantes $|G_x|$ et $|G_y|$
- Calcul des directions du gradient. Cette direction s'obtient via l'opération

$$\theta = \arctan(G_y/G_x)$$

. Puis on arrondit les valeurs des directions par les multiples de $\pi/4$.

- Suppression des non-maxima. Lorsque la norme du gradient en un pixel de l'image de coordonnées $(x; y)$ est inférieur à la norme du gradient de deux de ses voisins le long de la direction du gradient, on change la valeur de la norme de ce pixel à $(x; y)$ à 0.
- Seuillage des contours. Cette dernière étape consiste à décider si chaque pixel appartient ou non à un contour. On utilise pour ça un seuil haut et un seuil bas. Pour chaque pixel de l'image de coordonnées $(x; y)$, si la norme de son gradient est inférieur au seuil bas, le pixel n'appartient pas à un contour, si elle est supérieure au seuil haut, le pixel appartient à un contour, et si elle est comprise entre les deux seuils, le pixel appartient à un contour s'il est connecté à un autre pixel déjà accepté comme contour.

Transformée de Hough

La transformée de Hough est une méthode permettant de faire de la reconnaissance de formes dans une image. Bien qu'elle soit utilisée à l'origine pour détecter les lignes dans une image, de nombreuses variantes ont été développées, par exemple pour la détection de cercles ou d'ellipse.

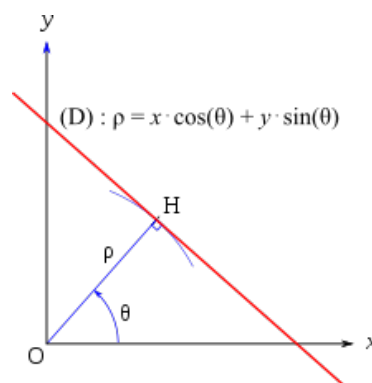


Figure 3.5 – Transformée de Hough

Cette méthode se base sur le paramétrage d'une droite dont l'équation est la suivante :

$$\rho = x \cos \theta + y \sin \theta$$

avec θ comme angle et ρ comme distance.

Cette méthode est basée sur une droite paramétrée par des coordonnées polaires $(\rho; \theta)$. ρ est la distance de la droite à l'origine du repère et θ est l'angle que fait la perpendiculaire à la droite de l'axe x . Les coordonnées des points de cette droite vérifient l'équation de la droite

$$\rho = x \cos \theta + y \sin \theta$$

. L'algorithme de Hough utilise une matrice accumulatrice qui représente le plan (ρ, θ) où ρ est le nombre de valeurs possibles de ρ et θ les valeurs possibles de θ . Pour chaque point (ou pixel) de l'image traitée, chaque droite de coordonnées (ρ, θ) incrémente l'élément correspondant à la matrice accumulatrice A . À la fin du traitement, les points de la matrice dont la valeur est la plus élevée correspondent à un grand nombre de points alignés sur l'image.

4

Analyse et conception

1 Analyse

1.1 Hypothèses utilisées

Etant donné que les images dont dispose le client n'ont pas de masque associé pour indiquer les zones des différents types d'usures présents, l'utilisation de segmentation d'image via l'apprentissage profond n'est pas envisageable dans l'immédiat. On pourra imaginer une structure de modèle pour réaliser cette section d'image, mais il ne sera pas possible d'entraîner le modèle, et donc on ne pourra pas obtenir de résultats corrects.

D'autre part, étant donné la présence d'une information textuelle (une mesure) relative à l'usure, incrustée de manière graphique sur chaque image, une classification binaire pour déterminer si l'outil est usé ou neuf peut sembler envisageable. Toutefois, il est à noter qu'une extraction de cette information pour chaque image est nécessaire, et qu'il faudra ensuite déterminer un seuil à partir duquel ces mesures indiquent si l'outil est usé ou neuf. Cette classification binaire ne permettra pas de quantifier une l'usure lors de l'analyse d'une nouvelle image, ni de quantifier quels types d'usures seront présents. Ainsi, il est préférable d'aborder cette solution de manière secondaire lors du développement de la solution.

Finalement, la technique privilégiée pour développer la solution d'analyse d'usure est le traitement d'image via la détection de contour et la transformée de Hough. Lors du développement, des retours du client seront nécessaires afin d'évaluer les résultats produits par les traitements d'image. Si le client indique que la reconnaissance n'est pas satisfaisante, des améliorations, voire des changements complets seront apportés afin de tenter d'améliorer la détection d'usure, et ce jusqu'à ce que l'analyse soit jugée satisfaisante.

Si l'analyse par traitement d'image est jugée satisfaisante, et que l'amélioration de cette analyse devient laborieuse, alors l'aspect de développement d'une solution d'analyse d'image basé sur l'apprentissage profond pourra être abordée. Si aucun retard n'a été pris, il s'agira de développer la solution en s'orientant vers une classification binaire ("outil usé" ou "outil neuf"). Ce qui impliquera de procéder à l'extraction de l'information des mesures présentes dans les images. Par la suite, si les résultats sont jugés satisfaisants par le client, le développement d'un modèle d'apprentissage profond de segmentation d'image pourra être développé pour s'orienter vers une classification multiclasse, afin de répondre au troisième pallier du besoin selon la section 1.2).

En revanche, si le planning n'est pas respecté et qu'on constate du retard pour la fin du développement de la solution de traitement d'image, alors il sera possible de passer directement au développement du modèle d'apprentissage profond pour la classification multi-classe.

1.2 Spécifications

La solution à développer se compose de plusieurs programmes python.

Un premier programme constituera une première tentative de résolution du problème en faisant appel à des techniques de traitement d'image, à savoir un premier traitement basé sur de la détection de contour suivi d'un second traitement basé sur la transformée de Hough, le tout afin de détecter l'usure de l'image fournie en entrée. A cela pourra venir s'ajouter une fonctionnalité de mesure pour quantifier l'usure détectée.

Les autres programmes auront pour but de réaliser une seconde approche pour répondre au besoin via l'utilisation d'apprentissage profond. A cette fin, un premier programme sera conçu pour gérer et compacter les données d'entrée à sélectionner pour constituer les bases d'apprentissage, de validation et de test pour le modèle de réseau de neurones à utiliser. Un deuxième programme sera utilisé pour la construction d'un modèle via un apprentissage en s'appuyant sur les différentes bases précédemment citées. Pour finir, un troisième programme sera sollicité pour réaliser l'analyse d'une image et répondre au besoin de l'utilisateur.

2 Modélisation proposée

Le diagramme d'utilisation ci-dessous illustre la manière dont l'utilisateur pourra interagir avec les différents programmes. Un programme spécifique existe pour chacun des cas d'utilisation.

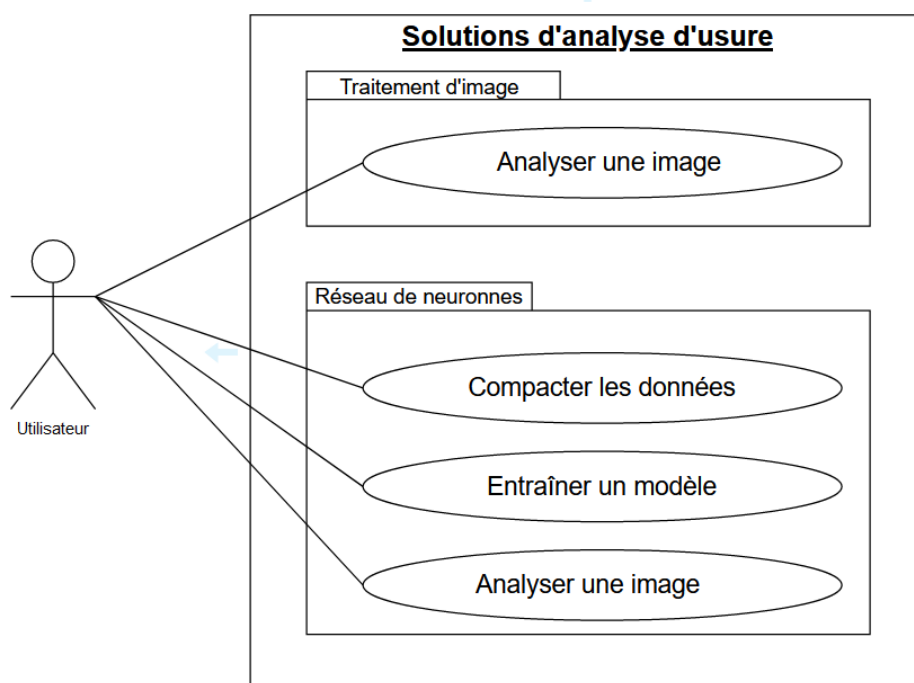


Figure 4.1 – Diagramme d'utilisation de la solution

5

Mise en oeuvre

1 Introduction

Lors de la phase de mise en œuvre, le traitement par apprentissage profond n'a pas été abordé. A la place, le développement basé sur le traitement d'image algorithmique a été davantage approfondi afin d'aboutir à des conclusions plus solides et pouvoir mieux rendre compte des possibilités qui existent en traitement d'image dit classique pour répondre au besoin qu'est l'analyse d'usure présente sur les outils de coupe. D'autres programmes que ceux initialement prévus ont alors été développés afin de travailler sur différentes approches pour atteindre l'objectif fixé.

2 Outils et librairie utilisés

Comme indiqué dans les spécifications, la solution est réalisée en python. Différentes librairies sont utilisées par les programmes de la solution.

1. Opencv est une librairie qui met à disposition de nombreuses fonctions de traitement d'image. C'est la plus importante du projet car c'est sur ces fonctions de traitement d'image que s'appuie les programmes pour procéder à la détection de l'usure.
2. numpy est une librairie de calcul matriciel qui est utilisée par les programmes pour traiter des images comme des matrices, et d'y apporter des modifications.

3 Éléments d'implémentation, choix techniques

3.1 Analyse par contours et détection de lignes

L'implémentation du premier programme est basé sur deux traitements d'images : la détection de contours d'une part, et la détection de lignes d'autre part. L'idée consiste à obtenir une première image binaire qui afficherait l'essentiel des contours présents dans l'image. Parmi ceux là, on s'attend à retrouver les bords de l'outil de coupe qui marquent un contraste avec le fond de l'image, les bords de l'usure repérés par contraste avec le revêtement de l'outil, les zones d'usure internes qui créent des contrastes en elles, les résidus sur le revêtement de l'outil issus

des phénomènes physiques lors de l'usinage qui ne correspondent pas à l'usure, et les autres éléments de l'image qui sont inutiles pour notre analyse, tel que les données chiffrées incrustées dans l'image (vb, échelle) ou divers éléments de l'image.

Suite à cette détection de contours, c'est la détection de lignes qui est utilisée dans le but de différencier les contours entre eux. En effet, l'arête supérieure de l'outil de coupe est rectiligne, ce qui veut dire qu'on peut le détecter en cherchant une ligne dans l'image. L'usure en dépouille quant à elle s'étend généralement de manière perpendiculaire sur la longueur de l'arête. On peut approximer le bord inférieur de l'usure comme une ligne qu'il est possible de détecter. Il est alors envisageable d'obtenir deux lignes correspondant à l'arête de l'outil et au bord inférieur de l'usure en dépouille.

Pour la détection de contours, différents filtres ont été essayés avec des convolutions tels que Sobel, Prewitt ou Laplacien. Une détection de contours via le filtre Canny Edge a également été mise en œuvre. La détection de lignes a été réalisée grâce à la méthode de la transformée de Hough utilisée par les fonctions HoughLines et HoughLinesP. La première permet de détecter les grandes lignes présentes dans l'image, tandis que la deuxième est plus adaptée pour détecter des segments dans l'image. C'est donc la deuxième qui nous s'avère la plus intéressante étant donné que l'usure sur l'outil ne concerne qu'une partie de l'image.

3.2 Analyse par histogrammes

Ce programme a pour but de détecter l'usure grâce aux variations des nuances de gris dans l'image en se basant sur les points suivants. D'une part l'usure en dépouille présente sur un outil est constituée de plusieurs régions différentes qui induisent un changement de couleurs selon l'état physique du matériau de l'outil de coupe. D'autre part que ces régions s'étendent le long de la longueur de l'arête de l'outil, un aspect représentatif de l'usure en dépouille. La démarche consiste alors à faire la somme des intensités des pixels sur la longueur de l'usure afin d'observer grâce à un histogramme un changement brutal qui correspond à un changement entre la surface de l'usure et la surface de l'outil.

Pour ce faire, il est d'abord nécessaire de localiser l'arête de l'outil de coupe sur l'image. Une fonction assure cette fonctionnalité et permet d'identifier les coordonnées de l'arête sur l'image. Cela se fait d'abord par une détection de contours en utilisant le filtre Canny Edge, puis en parcourant les colonnes de pixel de l'image binaire qui résulte de la précédente opération, et en notant la position du premier pixel non nul, qui correspond théoriquement à l'arête.

Une fois la courbe de l'arête localisée, la suite du programme consiste à réaliser la somme des niveaux de gris le long de cette courbe, et à boucler cette opération un certain nombre de fois selon la hauteur recherchée, en descendant d'un pixel à chaque fois pour constituer un histogramme.

3.3 Analyse par couleurs

Ce programme vise à détecter l'usure en se basant sur les nuances de couleurs de l'image. Sur un même outil, on peut parfois trouver une caractéristique récurrente de l'usure en dépouille au niveau de la couleur. En effet, sur le bord inférieur de l'usure on peut parfois remarquer que la fine frontière entre le revêtement de l'outil et l'usure possède la même couleur. Le but ici est d'essayer de détecter l'usure en recherchant les parties de l'image correspondant précisément à cette couleur. Ainsi, le programme consiste à appliquer un filtre sur l'image initiale et en ressortir un masque qui correspond à la partie la plus inférieure de la zone d'usure. Puis, en récupérant la courbe de l'arête de l'outil sur l'image, il est possible d'établir un masque qui englobe uniquement la forme de l'usure présente sur l'image.

Pour ce faire, le code couleur HSV a été adopté du fait que les variations de couleurs des pixels était plus importantes en RGB. Du fait que les pixels sont de couleurs proches mais pas identiques, il est nécessaire d'établir une plage de couleur pour extraire entièrement la zone ciblée. Une fois la plage de couleurs à extraire fixée, on extrait tous les pixels de l'image appartenant à cette plage pour obtenir un nouveau masque.

Afin de détecter la zone d'usure, on utilise la fonctionnalité implémentée dans le programme précédent pour détecter l'arête supérieure de l'outil. A partir de la courbe de l'arête supérieure de l'outil, on peut balayer chaque colonne pour détecter si la couleur ciblée est présente grâce au masque précédemment extrait, et si oui, on garde la hauteur du pixel afin de former une courbe qui correspondra à la partie inférieure de l'usure.

Une fois que l'on a la courbe du haut de l'outil et la courbe de la bordure inférieure de l'usure, on est en mesure de remplir la forme. Pour ce faire, on sélectionne chacun des points des deux courbes et on remplit un polygone sur une nouvelle image qui sera l'image finale de sortie.

4 Analyse des résultats, évaluation, qualité

Les résultats obtenus m'ont permis tout au long du projet de cibler les points clés quant à la nature des images à exploiter qui rendait l'analyse difficile. C'est la recherche de nouvelles solutions pour pallier ces difficultés qui ont abouti sur un programme final avec de meilleurs résultats.

4.1 Résultats de la détection de contours

Dans un premier temps, la détection d'usure via la détection de contours et la détection de lignes a soulevé une première difficulté. Lors de la détection de contours, on obtient bien les contours attendus mais l'image contient aussi une multitude de petits contours qui remplissent l'image là où on n'attend du vide. Cela s'explique par le fait que le revêtement de l'outil n'est pas homogène, mais présente de nombreuses pollutions visuelles (poussières, tâches, reflets, etc). Cela implique donc des variations d'intensité de couleurs fortes mais très petites qui sont interprétées comme des contours. Flouter l'image pour tenter de gommer ces défauts visuels entraîne une perte d'information au niveau de l'usure, il n'y a donc pas de moyen simple de supprimer ces contours parasites. La détection de ligne est faussée par les contours parasites et ne donne pas de résultats exploitables pour quantifier l'usure.

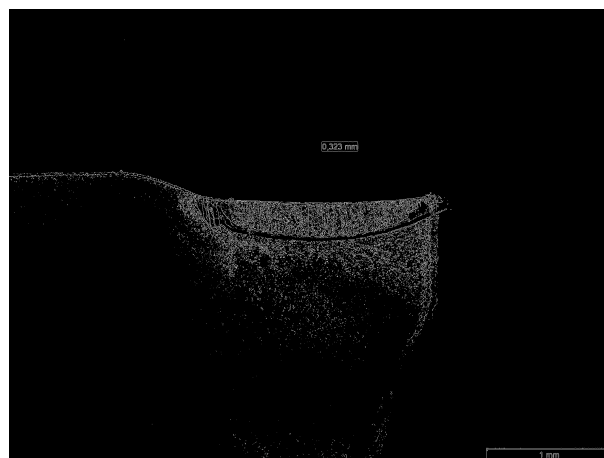


Figure 5.1 – Résultat de la détection de contours

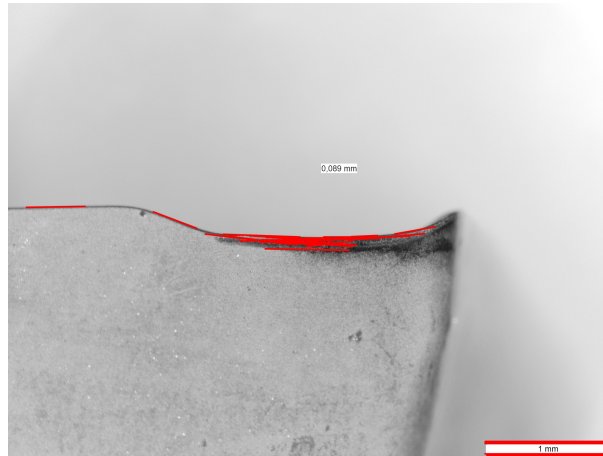


Figure 5.2 – Résultat de la détection de lignes

4.2 Résultats de l'analyse par histogrammes

Dans un second temps, l'analyse par histogrammes sur les niveaux de gris de l'image n'a pas donné de résultats exploitables. Si la fonctionnalité développée sur la localisation de l'arête de l'outil est efficace, à condition que l'image ne soit pas dégradée dans la partie supérieur gauche, c'est l'analyse de l'histogramme qui s'est révélée difficile à réaliser. En effet, si découper de sorte à traiter uniquement la zone d'usure est faisable, c'est l'irrégularité de l'usure qui pose problème. D'une part les couleurs varient sur une même ligne au sein de l'usure, ce qui perturbe les valeurs pour l'histogramme. D'autre part, le bord inférieur de l'usure est lui aussi irrégulier et la distance entre ce dernier et l'arête de l'outil varie, ce qui vient fausser par endroit l'analyse.

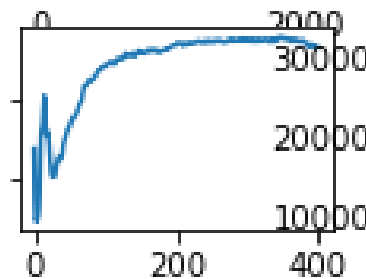


Figure 5.3 – Résultat de l'histogramme

4.3 Résultats de l'analyse par couleurs

Finalement, le dernier programme réalisé est celui qui aboutit aux résultats les plus proches de ceux attendus. Sur certaines images, on observe que le masque obtenu est proche de la zone d'usure attendue. La plage de couleur ciblée détermine le résultat obtenu, cela nécessite donc de connaître la couleur de l'image qui correspond à la bordure inférieure de l'usure. Une plage de couleur peut permettre de cibler l'usure sur une image et pas sur une autre. Il faut donc adapter la plage de couleur ciblée pour chaque image.

Parmi les images avec des résultats proches de ceux attendus, on peut observer des imprécisions. En effet, la courbe analysée comme étant la bordure inférieure de l'usure n'est généralement pas continue.

Il peut y avoir une absence de pixels sur certains segments de la courbe, ce qui est une source d'imprécision. Dans d'autres cas, certains pixels plus hauts ou plus bas dans l'image peuvent

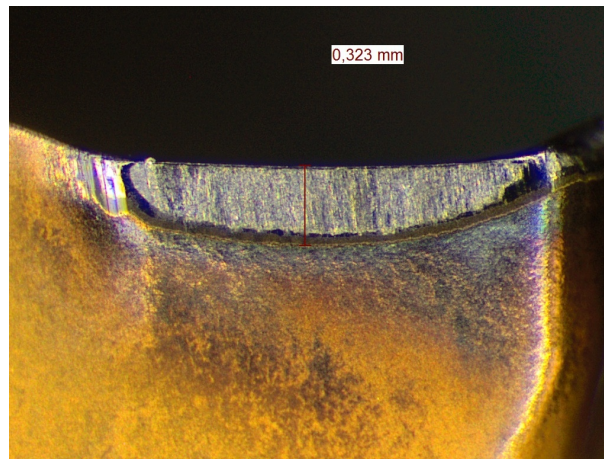


Figure 5.4 – Image originale



Figure 5.5 – Extraction de la couleur



Figure 5.6 – Masque de sortie

avoir été détectés, auquel cas il y a un changement brutal de hauteur, formant un pic sur la courbe. Il s'agit alors d'une erreur, les pixels détectés ne faisant pas partie de l'usure de l'outil.



Figure 5.7 – Résultat de l'histogramme

4.4 Améliorations à prévoir

Ce programme nécessite un paramètre connu à l'avance pour extraire l'usure. Toutefois, il est possible que la frontière entre l'usure et le revêtement de l'outil corresponde à des couleurs différentes selon les régions de l'image. De ce fait, il serait intéressant d'apporter une amélioration au programme avec la possibilité de combiner plusieurs masques issus d'extractions de couleurs différentes, afin d'améliorer la détection de l'usure.

De plus, il serait possible d'améliorer le programme avec une fonction pour supprimer les pics observés dans le masque de sortie qui sont des erreurs, comme expliqué à la section précédente.

La précision étant très importante, il sera également nécessaire d'améliorer le programme en s'assurant que la bordure de l'usure soit correctement prise en compte pour le masque de sortie final. Par exemple, une opération de dilatation est réalisée après extraction de la couleur afin de former un contour plus continu, or, il faut prendre en compte cette transformation pour bien détecter le bas de l'usure.

Ces améliorations sont une étape importante, car la mesure du VB sur l'usure en dépouille se fait par rapport à la hauteur maximale. Si le masque est incorrect, la mesure a de fortes chances d'être incorrecte également.

6

Bilan et conclusion

1 Bilan du semestre 9

Lors du semestre 9, un état de l'art a été réalisé et une liste de solutions potentielles pour répondre au besoin a été dressée. Des spécifications ont été rédigées et un diagramme de Gantt a été réalisé pour planifier les fonctionnalités de l'application à développer.

2 Bilan du semestre 10

Lors du semestre 10, la planification des tâches a été réorganisée. Après discussion avec le CEROC, et après confirmation de M. Martineau, il a été établi que le développement portera uniquement sur l'aspect traitement d'image classique. Le développement de modèle de réseau de neurones pour de l'apprentissage profond est avorté dans le but de fournir une approche plus qualitative et plus complète quant à l'utilisation de traitement d'image classique, et ainsi avoir une meilleure idée des possibilités et des limites offertes par cette approche.

Les deux premiers programmes n'ont pas abouti à de bons résultats, mais j'ai su innover en mettant en œuvre une nouvelle approche, et réutiliser certaines fonctionnalités précédemment implémentées. Finalement, les résultats obtenus montrent qu'il est possible de cibler une zone d'usure sur un outil de coupe, à condition de respecter certains critères comme évoqué dans la partie 5.3.3. Ces résultats montrent que la détection d'usure, et notamment l'usure en dépouille, par traitement d'image est faisable dès lors qu'on décide de spécialiser le programme pour correspondre à un type d'outil très précis. La solution requiert des améliorations pour généraliser le programme et le rendre performant sur un plus grand ensemble d'aspects d'usures.

En parallèle, l'analyse d'image par apprentissage profond reste une solution envisageable. Si elle a déjà fait ses preuves à travers quelques études, c'est une solution prometteuse mais qui reste coûteuse étant donné que des masques devront être réalisés à la main. Si cette piste est envisagée par la suite, il faudra insister sur l'utilisation de modèles pré-entraînés, ainsi que sur l'augmentation de données pour obtenir des résultats sur très faible échantillon d'images. L'augmentation de données devra être bien maîtrisée pour ne pas déformer l'usure présente sur les images initiales.

3 Bilan sur la qualité

Lors de la réalisation de mon travail, j'ai fait au mieux pour garantir la réutilisabilité du code. Pour cela j'ai veillé à commenter l'ensemble de mes programmes, et ce de manière régulière, afin de faciliter la compréhension de chacune des fonctions. Lors du développement, beaucoup de variables ont été initialisées avec valeurs codées en dur. Cela s'est fait pour gagner du temps, mais ces variables devront être initialisées autrement si la solution venait à évoluer, et ce dans une optique de réutilisabilité.

4 Bilan auto-critique

Cette phase de mise en œuvre a été pour moi très intéressante car j'ai pu travaillé sur un projet qui s'approchait d'un travail de recherche. Bien que la planification des tâches était prévu autrement à la fin de la phase de spécifications et d'analyse, j'ai réussi à me recentrer sur l'aspect analyse par traitement d'image à travers plusieurs approches différentes. J'ai également essayé de prendre du recul sur les conclusions à tirer de mon travail : efficacité, limites, possibilités d'amélioration.

Il est dommage que je n'ai pu réaliser les améliorations suggérées dans la section 4.4 car cela m'aurait permis de passer à une étape importante pour le rendu du projet : la quantification de l'usure. De plus, les solutions algorithmiques pour réaliser ce programme ayant été relativement laborieuses, le développement a été retardé et j'ai négligé la communication avec mes encadrants sur la fin du projet. Une meilleure communication aurait permis, d'une part, de donner retour à mes encadrants pour qu'ils aient conscience de mon avancement, et d'autre part, de m'assurer que mes choix de développement étaient appropriés à leurs attentes.

Annexes

Planification, gestion de projet

1

Le diagramme de Gantt initial du projet.

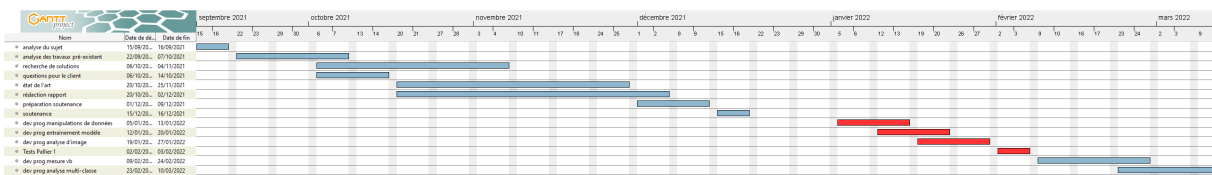


Figure A.1 – *Le diagramme de Gantt initial*

Finalement les données dont disposait le CEROC n'étaient pas adaptées aux tâches initialement prévues. De nouvelles fonctionnalités ont donc été ajoutés et le planning du S10 a complètement changé. Ci-dessous le diagramme de Gantt revisité.

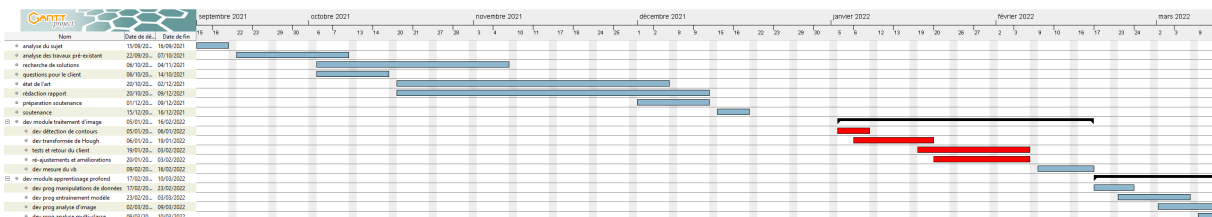


Figure A.2 – *Le diagramme de Gantt revisité*

Suite à la réorganisation des tâches à la demande du client, le diagramme de Gantt a évolué pour devenir celui si-dessous.

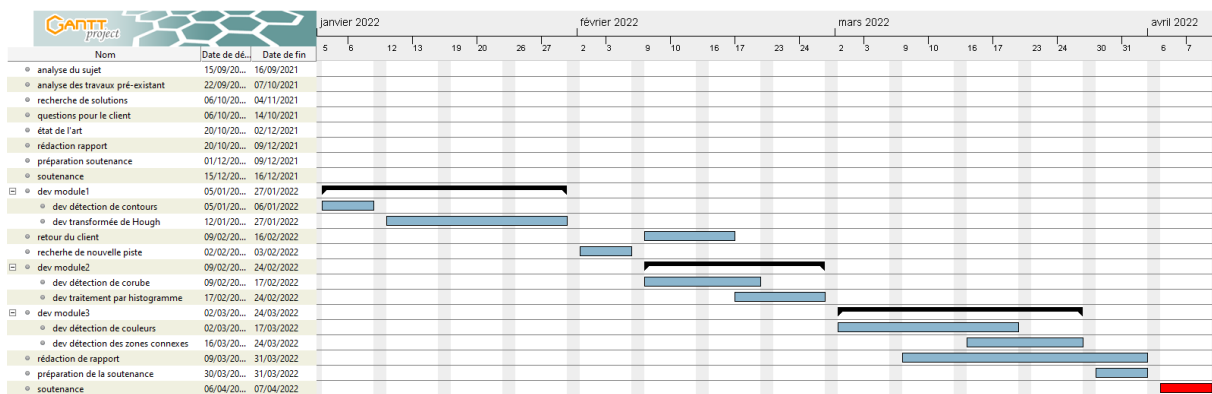


Figure A.3 – Le diagramme de Gantt de la mise en oeuvre

2 Description des tâches

Tâche 1 : Analyse des travaux pré-existants

- Date de début : 22/09/2021
- Date de fin : 07/10/2021
- Description : Lecture du rapport de stage de l'étudiant stagiaire au CEROC durant l'été 2020. Recherches sur les éléments techniques évoqués : usures, traitements d'image effectués

Tâche 2 : Recherche de solutions

- Date de début : 06/10/2021
- Date de fin : 04/11/2021
- Description : Analyse des solutions possibles pour répondre au besoin.

Tâche 3 : Questions pour le client

- Date de début : 06/10/2021
- Date de fin : 14/10/2021
- Description : Visite du CEROC, séries de questions-réponses avec le client pour définir les aspects techniques, la qualité des données et les précisions sur les fonctionnalités attendues

Tâche 4 : Etat de l'art

- Date de début : 20/10/2021
- Date de fin : 02/12/2021
- Description : Rédaction de l'état de l'art du rapport, apprentissage des techniques existantes voire auto formation pour les utiliser

Tâche5 : Rédaction rapport

- Date de début : 20/10/2021
- Date de fin : 09/12/2021
- Description : Rédaction du rapport et des spécifications du projet

Tâche 6 : Préparation à la soutenance

- Date de début : 01/12/2021
- Date de fin : 09/12/2021
- Description : Préparation du transparent et du contenu de la présentation pour la soutenance

Tâche 7 : Soutenance

- Date de début : 15/12/2021
- Date de fin : 16/12/2021
- Description : Soutenance du projet - S9

Tâche 8 : développement du processus de la détection de contour

- Date de début : 05/01/2022
- Date de fin : 16/02/2022
- Description : Développer la fonction du programme d'analyse d'image du module de traitement d'image permettant de détecter les contour d'usure présente sur une image.

Tâche 9 : Développement du processus de transformée de Hough

- Date de début : 06/01/2022
- Date de fin : 19/01/2022
- Description : Développer la fonction du programme permettant d'appliquer la transformée de Hough sur une image.

Tâche 10 : Tests et retour du client

- Date de début : 19/01/2022
- Date de fin : 03/02/2022
- Description : Retour du client sur la reconnaissance d'usure, et sur le niveau de satisfaction

Tâche 11 : Ré-ajustements et amélioration

- Date de début : 20/01/2022
- Date de fin : 03/02/2022
- Description : Correction et amélioration du processus de traitement d'image pour la détection d'usure

Tâche 12 : Développement de la fonction pour la mesure du Vb

- Date de début : 09/02/2022
- Date de fin : 16/02/2022
- Description : Développer la fonction du programme de mesure du Vb

Tâche 13 : Développement du programme de gestion des données

- Date de début : 17/02/2022
- Date de fin : 23/02/2022
- Description : Développer le programme de gestion des base d'apprentissage, de validation et de test.

Tâche 14 : Développement du programme d'entraînement du modèle de réseau de neuronne

- Date de début : 23/02/2022
- Date de fin : 03/03/2022
- Description :

Tâche 15 : Développement du programme d'analyse d'image

- Date de début : 02/03/2022
- Date de fin : 09/03/2022
- Description : Développement du modèle de réseau de neurones pour la classification binaire d'outils d'usure selon les sorties suivantes : "outil usé" ou "outil neuf"

Tâche 16 : Ajout d'un modèle pour la segmentation d'image

- Date de début : 09/03/2022
- Date de fin : 10/03/2022
- Description : Ajout d'un modèle de réseau de neurones pour la classification multi-classe basé sur la segmentation d'image.

B

Description des interfaces

1 Interfaces homme/machine

La solution développée ne disposera pas d'interface graphique. Il s'agira de plusieurs programmes à exécuter sous forme de scripts python. La plupart des paramètres qui interviendront dans les processus de ces programmes devront être inscrits en dur dans les programmes. D'autres seront transmis en paramètres au moment du lancement du programme en ligne de commande. Pour en savoir plus, se référer au document d'utilisation et au cahier du développeur.



Cahier de Spécifications

1 spécifications Fonctionnelles

1.1 Fonctionnalités à développer

Description de la fonction de traitement d'image :

Cette fonction est utilisée pour appliquer un traitement d'image sur une image d'entrée afin de produire une image binaire en sortie indiquant l'usure présente sur l'image.

Analyse_Hough

Entrée : L'image à analyser

Sortie : Une nouvelle image binaire

Préconditions : L'image d'entrée n'est pas vide

Postconditions : Néant

Description de la fonction de gestion des données :

Ce processus a pour but de faciliter la sélection des données qui constitueront les bases du modèle de réseaux de neurones. Il permet de compacter les données des répertoires indiqués en entrée sous forme de tenseur utilisable par le modèle.

Manage_data

Entrée : Liste des répertoires d'images et d'étiquettes à ajouter dans les bases d'apprentissage, de validation et de test

Sortie : Trois tenseurs de données et d'étiquette correspondant aux bases d'entraînement, de validation et de test

Préconditions : Néant

Postconditions : Néant

Description de la fonction d'entraînement :

Il s'agit du processus pendant lequel le modèle est créé et ajuste les poids entre ses différentes couches dans le but d'être en mesure d'analyser au mieux un certain type d'image.

Entraînement

Entrée : Néant

Sortie : Modèle entraîné

Préconditions : Les données d'entraînement, de validation et de test existent dans les répertoires prévus à cet effet

Postconditions : Néant

Description de la fonction d'analyse :

Cette fonction est sollicitée pour chaque image à analyser via le modèle de réseau de neurones. Elle retourne une prédiction faite par le modèle de l'usure présente sur l'image d'entrée.

Analyse

Entrée : L'image à analyser

Sortie : Prédiction du modèle

Préconditions : L'image d'entrée n'est pas vide

Postconditions : Néant

2 Spécifications non fonctionnelles**2.1 Contraintes de développement et conception**

Le langage de programmation utilisé sera le python.

L'environnement nécessaire sera un système d'exploitation sous une distribution linux, ou bien windows.

2.2 Contraintes de fonctionnement et d'exploitation**2.2.1 Performances**

Le temps de traitement d'une image via le programme principal est de l'ordre de la seconde pour une image d'une taille de quelques centaines de pixels. Le temps croît de manière exponentielle lorsque la taille de l'image à analyser augmente.

Le temps d'entraînement du modèle de réseau de neurones dépend de la taille de la base d'entraînement et du matériel utilisé, notamment le GPU. Pour une base de plusieurs milliers d'images avec un GPU à faible capacité, ou sans GPU, on peut supposer que le temps d'entraînement du modèle soit de l'ordre de l'heure.

Le temps d'analyse d'image du modèle de réseau de neurones, pour une image d'une taille de quelques centaines de pixels, sera de l'ordre de la seconde.

2.2.2 Capacités

Les différents programmes d'analyse d'image seront fait pour traiter une image à la fois. Aucune contrainte de taille maximum des images n'est présente.

2.2.3 Contrôlabilité

Le suivi d'exécution des différents programme pourra se faire via la console ou des messages seront affichés pour indiquer l'avancement des différents processus.

2.2.4 Sécurité

Etant donné qu'il n'existe pas de contrôle d'accès utilisateur, aucun aspect de sécurité particulier n'est relevé pour l'application.

2.2.5 Evolution du système

Il est prévu que le système soit intégré dans une application pour créer une interface utilisateur visuelle et pratique afin de manipuler facilement les différents paramètres de traitement d'image lors de son utilisation.

De plus, des améliorations pourront sans doute être apportées aux différents algorithmes de traitement d'image afin d'améliorer les résultats d'analyse.

D

Cahier du développeur

1 Introduction

Ce document présente le développement de la solution dans le détail et a pour but de faciliter la reprise du projet. Dans un premier temps l'organisation des fichiers sources sera abordée. Les données exploitées seront ensuite détaillées, et finalement une description détaillée de certaines fonctions sera présentée.

2 Organisation des sources

Chaque programme de la solution se présente sous la forme d'une fonction principale dans un fichier dédié et est complètement indépendant. On en compte trois :

1. analyse_contours_hough.py
2. analyse_hitogrammes
3. analyse_couleurs

Un fichier constants.py est présent pour y renseigner les variables constantes à utiliser pour la solution. Parmi celles-ci, on peut citer le nom du répertoire dans lequel les images d'entrées sont stockées, ou encore l'extension des images d'entrées. Les autres fichiers répertorient des fonctions qui sont appelées dans les programmes principaux de manière à gagner en lisibilité.

3 Descriptions détaillées des données exploitées

Chaque programme exploite un ensemble d'images présent dans un répertoire placé dans le répertoire courant du programme python, et dont le nom correspond à la variable constante présente dans le fichier constants.py. Une méthode présente dans le fichier data permet de retourner la liste des noms des fichiers présents dans le répertoire. Chaque image doit présenter la face d'un outil de coupe sur laquelle on veut détecter l'usure présente le long de son arête supérieure droite. Les images doivent être en couleurs, de mêmes dimensions, et toutes représentatives d'un seul et même type d'outil de coupe.

4 Descriptions détaillées des programmes

4.0.1 Analyse_contours_hough

Dans ce programme on va réaliser de la détection de contours et de la détection de ligne. On récupère au début de la fonction différents noyaux convolution dans le but de faire de la détection de contours par convolution. Un noyau de convolution se présente sous la forme d'une matrice, et ceux utilisés ici contiennent des valeurs particulières qui correspondent à un filtre connu (Prewitt, Sobel ou Laplace). Pour chacun d'entre eux, trois noyaux existent : un pour filtrer selon l'axe des abscisses, un autre selon l'axe des ordonnées, et un dernier (la somme des deux, pour filtrer selon les deux axes). La suite du programme consiste à récupérer la liste des images à traiter et faire une boucle pour traiter toutes les images. Pour chaque traitement, on affiche les résultats via la librairie matplotlib. Le filtre Canny Edge utilisé est basé sur l'utilisation de deux seuils : un faible et un fort. Ces seuils déterminent les contours de l'image à extraire. Pour déterminer au mieux quels seuils sont les mieux adaptés pour extraire les contours de l'usure sur les outils de coupe, la fonction `cannyRangeOperation` a été réalisée. Elle génère un ensemble d'images en balayant les seuils possibles afin de donner un aperçu des seuils les mieux adaptés. De la même manière, la fonction de la détection de lignes via la transformée de Hough est basée sur des seuils. Des fonctions ont été réalisées afin de balayer les résultats possibles sur l'ensemble des seuils, et ainsi tenter de cibler les meilleurs seuils pour détecter les lignes correspondant aux contours de l'usure en dépouille.

4.0.2 analyse_histogrammes

La fonction `topEdgeDetection` permet de retourner une courbe correspondant à l'arête supérieure de l'outil de coupe. Elle prend en entrée une image obtenue après l'application d'un filtre Canny Edge. L'opération consiste à balayer chaque colonne de pixels de l'image et de retenir la position du premier pixel blanc. La fonction retourne un tableau contenant la hauteur de chaque pixel, le pixel du coin supérieur gauche de l'image étant la position (0,0). La fonction `histDetection` consiste à séparer l'image en plusieurs colonnes et à générer des histogrammes sur les niveaux de gris pour tenter d'observer une variation brutale correspondant à la hauteur de l'usure.

4.0.3 analyse_couleurs

La fonction principale de ce programme consiste à appliquer un traitement sur chacune des images d'entrées localisées dans le répertoire `data`. Dans un premier temps on converti l'image d'entrée au format HSV qui sera utilisée pour le traitement d'image. Un affichage via matplotlib est géré, toutefois il n'a que peu d'utilité, les images de sorties seront sauvegardées dans les répertoires créés au début de la fonction. Une liste de couleurs a été codée en dur lors du développement, elle est vouée à être modifiée, notamment si une interface doit être associée à la solution. La fonction `getThres` est utilisée pour générer une plage de couleurs à partir d'une couleur et d'un tuple qui indique la tolérance pour extraire les couleurs. La fonction `getMasks` est utilisée pour extraire chacune des plages de couleurs de la liste données en paramètre. La fonction `wearEdgeDetection` est une fonction réalisée pour récupérer uniquement les premiers pixels en partant du haut de l'image. L'image à lui donner en paramètre doit être un masque de sortie de la fonction `getMasks`. Finalement, la fonction `getShape` permet, à partir des courbes de l'arête supérieure de l'outil de coupe et de la courbe formée par le bord inférieur de l'usure, de reconstituer la zone d'usure de l'image. Pour vérifier que l'usure détectée est cohérente, un seuil est défini avant la boucle. Si la

hauteur de l'usure détectée (entre la position de l'arête de l'outil et la position du pixel détectée) est supérieure à ce seuil, alors on considérera que la hauteur détectée est aberrante et qu'il ne peut pas s'agir d'usure. On ne retiendra donc pas ce pixel pour le masque de sortie. Finalement, avec les contours retenus, un polygone est rempli sur l'image de sortie.

E

Document d'installation

1. Installer un environnement python ainsi qu'un IDE pour programmer en python.
2. Installer la librairie openCV.
3. Installer la librairie numpy.
4. Installer la librairie Matplotlib.

F

Document d'utilisation

Ajouter les images à traiter dans le répertoire "data" situé dans le répertoire du programme python à exécuter. Il est possible de changer le nom de ce répertoire via le fichier constants.py. Exécuter le programme et observer les images de sorties dans les différents répertoires créés dans "./data".

Détection des usures d'outil de coupe par vision assistée par ordinateur

Alexis Cassagnaud

Encadrement : Patrick Martineau



En collaboration avec Polytech

Objectifs

- Déterminer si l'outil est neuf ou usé
- Quantifier l'usure en dépouille présente sur un outil
- Déterminer les différents types d'usure présents sur un outil et les quantifier

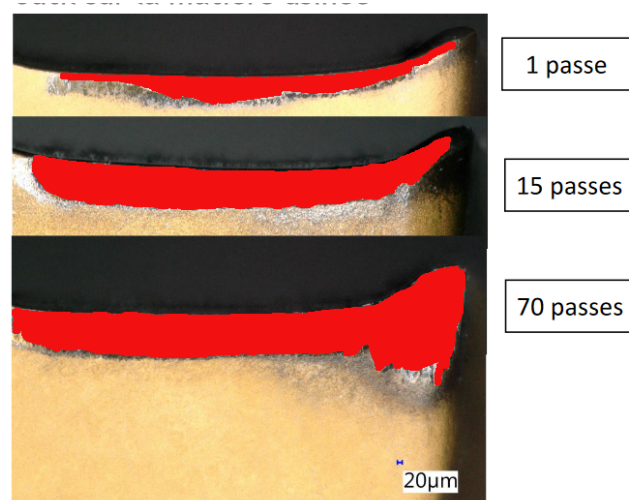
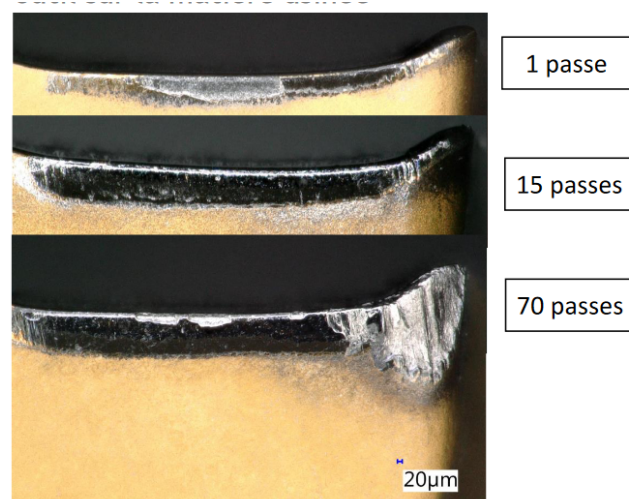


Mise en œuvre

1. Développement d'un premier programme basé sur la détection de contours et la détection de lignes
2. Développement d'un autre programme pour détecter l'usure via les variations de niveaux de gris de l'image
3. Développement d'un dernier programme basé sur les variations de couleurs de l'image

Résultats attendus

On souhaite appliquer un traitement d'image sur des photos d'outils de coupe afin d'obtenir un masque de l'image correspondant à l'usure présente sur l'outil.



Détection des usures d'outil de coupe par vision assistée par ordinateur

Alexis Cassagnaud

Encadrement : Patrick Martineau

Objectifs

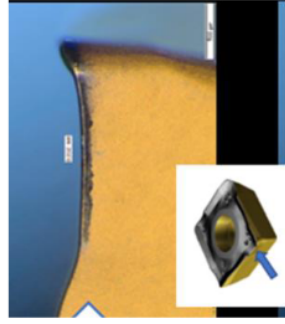
- Déterminer si l'outil est neuf ou usé
- Quantifier l'usure en dépouille présente sur un outil
- Déterminer les différents types d'usure présents sur un outil et les quantifier



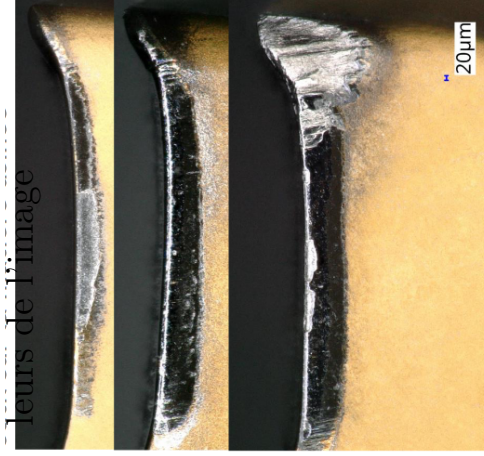
En collaboration avec Polytech

Mise en œuvre

1. Développement d'un premier programme basé sur la détection de contours et la détection de lignes
2. Développement d'un autre programme pour détecter l'usure via les variations de niveaux de gris de l'image
3. Développement d'un dernier programme basé sur les variations de contours de l'image



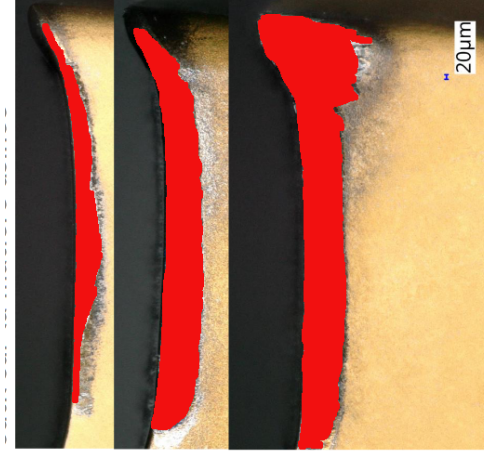
Face de dépouille
secondaire



1 passe

15 passes

70 passes



1 passe

15 passes

70 passes

Résultats attendus

On souhaite appliquer un traitement d'image sur des photos d'outils de coupe afin d'obtenir un masque de l'image correspondant à l'usure présente sur l'outil.



Détection des usures d'outil de coupe par vision assistée par ordinateur

Résumé

Ce projet s'inscrit dans le cadre de l'usine du futur (industrie 4.0) et consiste à mettre au point une solution d'analyse d'image pour la détection d'usures d'outils de coupe. Il s'agit de développer une solution d'analyse d'image capable d'identifier les zones d'usure sur une image d'outil coupant dans le but de tendre vers l'automatisation des mesures de l'usure au fur et à mesure de l'usinage.

Mots-clés

usinage, vision assistée par ordinateur, apprentissage profond, unet

Abstract

This project is about the conception of a solution of wear analysis on cutting tools based on computer vision. It consists in programming a process enabling the automatisisation of the detection and the measure of wear areas.

Keywords

Computer vision¹, Deep learning, unet

Entreprise

Polytech



Tuteur entreprise

Guillaume ALTMAYER

Étudiant

Alexis CASSAGNAUD (DI5)

Tuteur académique

Patrick MARTINEAU