

ISCAN

ICR - reconnaissance intelligente de caractères

SOUBRAND Florentin, VATON Thomas, LEMPEREUR Thibault, CHARDON

2 février 2018 - 29 mai 2018

<p>aubay.png</p>	<p>epita.jpg</p>
------------------	------------------

Table des matières

1 Introduction

IntroductionIntroductionIntroductionIntroductionIntroductionIntroductionIntroductionIntro
IntroductionIntroductionIntroductionIntroductionIntroductionIntroductionIntroductionIntroduc
IntroductionIntroductionIntroductionIntroductionIntroductionIntroductionIntroduction

2 Présentation de l'entreprise

2.1 Le secteur d'activité

Aubay est une Entreprise de Services du Numérique (ESN). Une ESN (anciennement SSII) est une société de service spécialisée dans l'informatique et les nouvelles technologies. L'entreprise peut compter sur ses experts pour accompagner voire réaliser les sociétés clientes dans leur projet. Pour cela, elle peut compter sur ses nombreux experts pour répondre aux différents besoins des missions d'une ESN (conseil, conception et réalisation d'outils, maintenance ou encore formation).

Les ESNs répondent aux besoins d'externalisation des expertises et des projets informatiques de certaines entreprises.

Aubay intervient, depuis sa constitution, uniquement auprès d'acteurs « Grands Comptes » et s'est tout particulièrement imposé sur le marché de la banque et de l'assurance. Aubay dispose de références majeurs chez les plus gros acteurs de ces secteurs. En effet, le secteur bancaire représente 42% du chiffre d'affaire du groupe, et les assurances près de 25%. A cela s'ajoute les services et l'industrie qui en représentent un dixième chacun, le secteur public et la télécommunication quant à eux correspondent à plus de 5% chacun, finalement le reste du chiffre d'affaire provient du e-commerce.

Ainsi parmi les principaux clients de l'entreprise, on peut citer : BNP Paribas, Groupe Crédit Agricole, Allianz, AXA, Groupe BPCE, Société Générale, Poste Italiane, Orange, SNCF, Intesa SAN PAOLO, Vodafone, Grupo Santander, Generali, ENGIE et BBVA.

2.2 L'entreprise

Présente depuis toujours dans plusieurs pays européens, Aubay ne cesse de grandir et a su s'implémenter dans de plus en plus de pays étranger.

Du Conseil à tout type de projet technologique, Aubay accompagne la transformation et la modernisation des systèmes d'information.

En outre le nombre de collaborateurs entre les années 2012 et 2016 a presque doublé. Aubay compte aujourd'hui près de 5 500 consultants dont plus de 2 200 en France. En 2017, l'entreprise a recruté 1200 collaborateurs en Europe dont 700 en France issus des meilleures écoles d'ingénieur, de commerce et des grandes universités.

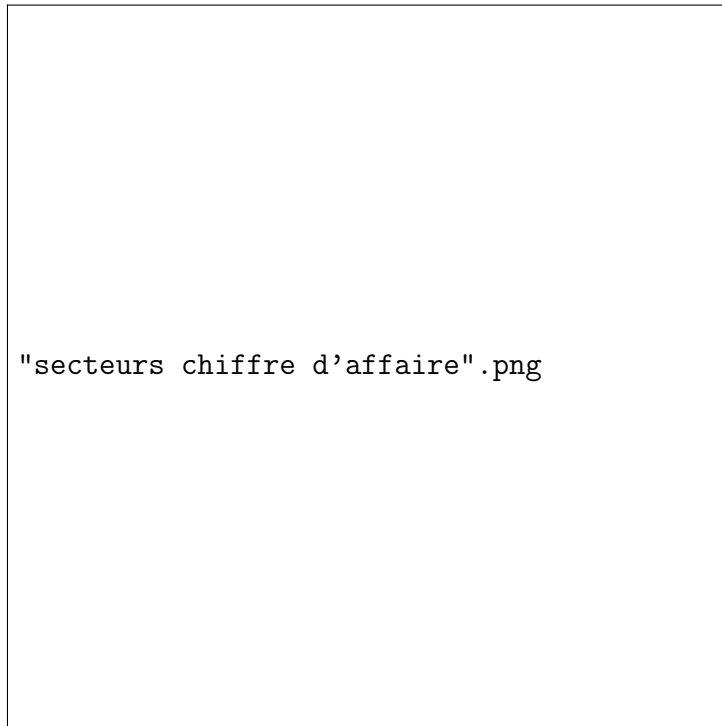


FIGURE 2 – Répartition du chiffre d'affaires du groupe Aubay par secteur

On constate que la croissance du nombre de collaborateurs dans l'entreprise est liée à celle du chiffre d'affaire, en effet la croissance de ce dernier depuis 2012 n'est jamais passée sous les 10% et a connu en 2014 un pic à 15%. Ainsi en 2012, le chiffre d'affaire s'estimait à environ 190.5 millions d'euros, puis en 2013 il est passé à 211.1 millions d'euros, pour atteindre en 2014 près de 243.3 millions d'euros. S'en est suivi en 2015 une augmentation jusqu'à 273.8 millions d'euros, pour en 2016 atteindre plus de 300 millions d'euros, soit une augmentation de 12% par rapport à l'année précédente.

2.3 La cellule Innovation

Une des grandes forces d'Aubay est que les solutions que l'entreprise apporte à ses clients sont des services innovants s'adaptant à leurs demandes. Ceci est aujourd'hui possible grâce aux projets menés au sein de la Aubay Innov'.

Cette cellule Innovation réunit des architectes et de experts en charge de l'organisation et de la conduite de travaux innovants autour des technologies digitales de demain. Créée en 2013, la cellule est aujourd'hui dirigée par Xavier

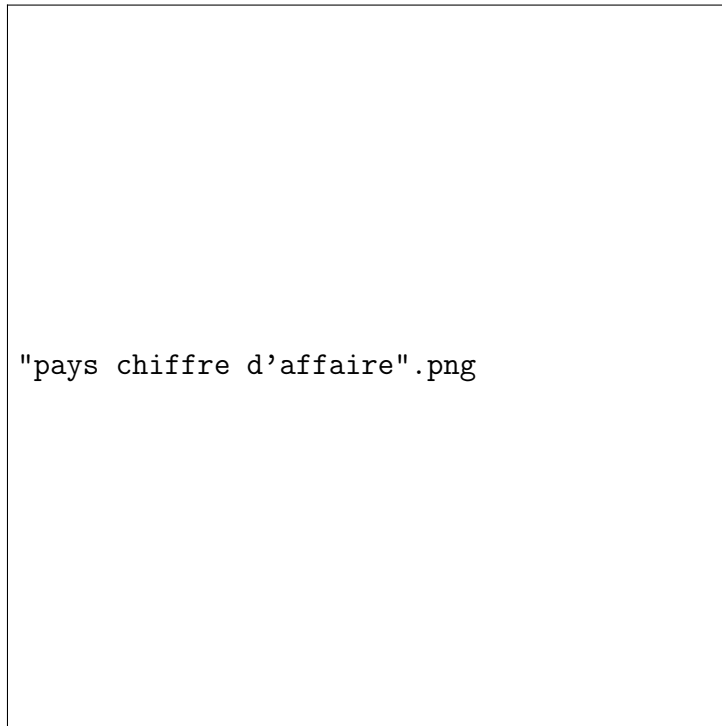


FIGURE 3 – Répartition du chiffre d'affaires du groupe Aubay par filiale

PRINCE, mon maître de stage. C'est dans cette partie de l'entreprise que s'est déroulé mon stage. On y retrouve plusieurs équipes menant des projets aux thématiques différentes dans un même environnement de travail.

Lors de sa création, cette dernière a reçu comme mission « d'acquérir la connaissance et le savoir-faire pour bâtir des solutions innovantes pérennes et adaptées aux besoins futurs ». Grâce à l'incubation et la réalisation des idées des membres d'Aubay Innov' en partenariat avec des experts, des laboratoires et les écoles, la cellule Innovation peut remplir son rôle.

Les projets en cours cherchent à appliquer des concepts comme le Big Data, la réalité augmentée, l'analyse de données de parcours clients sur site web (Web Analytics) mais encore l'IoT (Internet of Things) et le Blockchain.

C'est parmi ces nouvelles applications que se trouve le projet I'Scan, le sujet de mon stage.

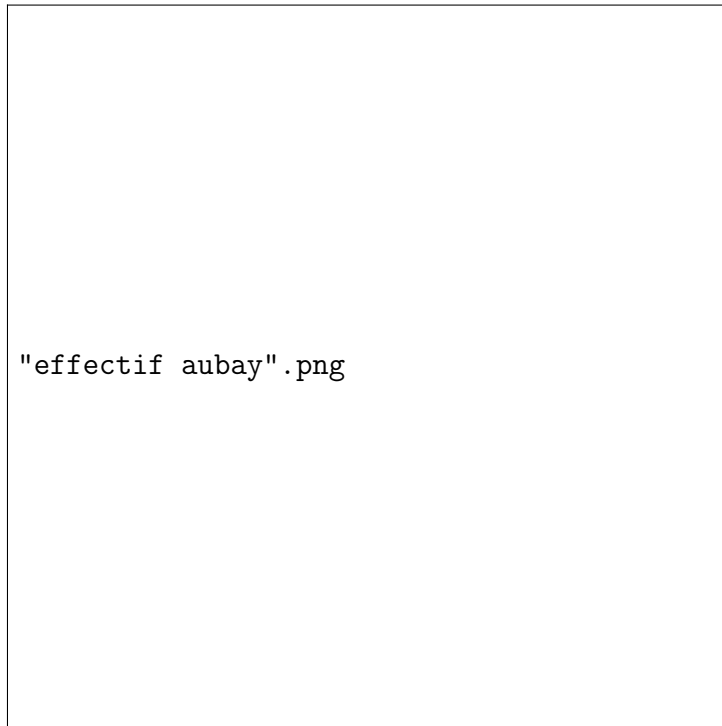


FIGURE 4 – Évolution des effectifs du groupe Aubay

2.4 Le projet I'Scan au sein de la cellule

Le projet I'Scan était mené par 4 autres stagiaires de différentes écoles quand je l'ai rejoint.

Le projet I'Scan est l'un de nombreux sujets de recherche de la cellule Innovation. Présent depuis plusieurs années, les techniques utilisées ont pu progresser au cours du temps. Les travaux sur ce projet permettent de rester à jour sur plusieurs domaines et de connaître les dernières méthodes utilisées.

Le premier est le traitement de l'image, plus précisément pour la reconnaissance de texte dans l'image. De nouveaux articles scientifiques continuent d'être publiés par exemple sur le découpage du texte en ligne, notamment avec de nouvelles techniques pour identifier des lignes qui se retrouvent très serrées et qui se chevauchent. Les méthodes les plus avancées permettent de reconnaître des lignes de textes orientés différemment sur l'image (jusqu'à des lignes de textes parfaitement perpendiculaires) ou du texte écrit en arc-de-cercle. Le projet I'Scan se concentre pour l'instant sur des textes écrits clairement et avec des lignes bien séparées et horizontales. L'un des objectifs

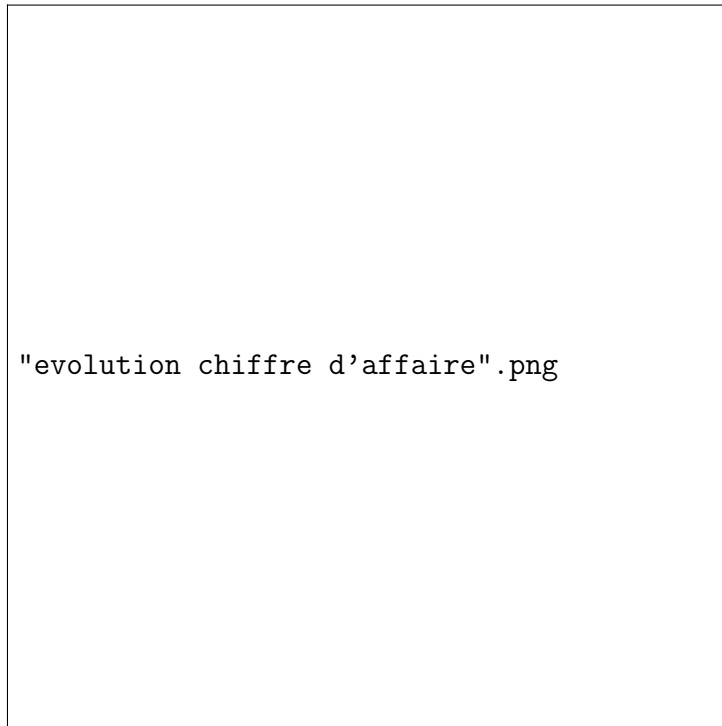


FIGURE 5 – Évolution du chiffre d'affaires du groupe Aubay

à court termes étant d'obtenir de très bons résultats sur ce type de textes avant d'essayer de traiter des textes où les auteurs ont été moins appliqués.

Le second est les réseaux de neurones. Le *machine learning* a trouvé parmi ses premières applications l'identification d'objets dans une image. l'Scan est la première utilisation d'une méthode de *machine learning* au sein d'Aubay avec l'implémentation d'un algorithme de k-NN (*k Nearest Neighbors*). L'utilisation d'une méthode de *deep learning* orientée traitement de l'image avec un CNN (*Convolutional Neural Network* ou Réseaux de neurones à convolution) est actuellement en étude. Le projet permet donc de rester à l'écoute des nouveautés dans ce milieu riche, jeune et qui vit de grosses évolutions régulièrement.

3 Le travail effectué

3.1 Le cahier des charges

Le but est d'améliorer et de transformer la façon dont les tableaux blancs sont utilisés et cela dans n'importe quel environnement (écoles et cours en ligne, salles de réunions et réunions télétransmise, etc.).

Le cœur du projet est un logiciel capable de convertir une écriture manuscrite et des schémas en une version numérisée et modifiable. Cela implique le découpage de l'image capturée en formes simples (carrés, triangles, cercles, lignes, etc.) et la reconnaissance du texte (sous plusieurs formes typographiques comme du texte tapé sur ordinateur, l'écriture manuscrite, majuscules et minuscules, etc.) puis la reconstruction de ceci en un fichier numérisé sous différents formats.

Plusieurs améliorations futures sont envisageables :

- Exporter l'image sous plusieurs formats numériques
- Détection et restitutions des couleurs

Il a été décidé d'implémenter le projet avec une structure «client lourd». Cela requiert l'installation du logiciel par l'utilisateur (à l'opposé d'un « client léger », utilisable depuis un navigateur Web).

Le logiciel est développé en C et C++. Ce choix a été fait pour des raisons de performances. Python a été utilisé pour réaliser des tests de réseau de neurones à convolution avec la bibliothèque Tensorflow. Si l'efficacité de cette nouvelle méthode est vérifiée, le réseau de neurones sera implémenté en C++ avec lui aussi la bibliothèque TensorFlow.

3.2 Fonctionnement global du projet

Actuellement, le projet se concentre sur la partie reconnaissance de caractères du cahier des charges, le gros du travail étant de reconnaître les caractères de l'écriture manuscrite.

Ce schéma montre les différentes étapes du projet et l'ordre dans lequel elles sont exécutées. Lors de la description de chacune des tâches que j'ai effectuées, je me référerai à la numérotation présente ici afin de pouvoir mieux situer la tâche dans le projet.

3.3 Affichage clair du traitement des espaces

Lors du découpage des lignes en mots (étape 4, figure ??), la méthode utilisée implique de trouver les blobs des lettres. Un blob est un groupe de pixel connecté qui partage une propriété commune. Dans notre cas, l'image est binarisée. C'est-à-dire que les pixels ne prennent que des valeurs binaires (0 le pixel est noir, 1 le pixel est blanc). La fonction de blob de la bibliothèque OpenCV va donc suivre les pixels noirs connectés et va ainsi tracer des zones autour de chaque lettre (figure ??).

La méthode utilisée regroupait ensuite les blobs proches ou qui se superposaient pour créer des blobs autour des mots (figure ??).

Cette méthode des blobs n'a pas donné les résultats attendus. Il fallait donc trouver une autre solution à ce problème.

La nouvelle méthode est une méthode en phase de recherche sur laquelle travaillait les membres de l'équipe à mon arrivée sur le projet. Cette méthode utilise un histogramme horizontal sur toute la ligne. Un histogramme représente la somme des pixels de chaque colonne. Il peut être visualisé ainsi :

Chaque espace vide, où il n'y a aucun pixel sur la colonne, situé entre 2 pics de valeurs correspond à un espace entre 2 caractères (3 espaces dans la figure 10). Le but est ensuite de classer ces espaces en 2 catégories : espaces inter-mots et espaces inter-lettres. Pour cela, on utilise la largeur de ces espaces puis on calcule 2 barycentres où celui regroupant les plus grands espaces est le barycentre des espaces inter-mots et l'autre celui des espaces inter-lettres.

La première solution était de calculer la moyenne des tailles des espaces afin de séparer les deux groupes. Malheureusement, la classification des espaces n'est pas si simple. En effet, il arrive que des espaces entre caractères soit plus grand que des espaces mots. A cause de cela, certains espaces inter-lettres se retrouvent de tailles supérieures à la moyenne et certains espaces inter-mots se retrouvent de tailles inférieures à la moyenne.

Il a donc été choisi de travailler avec une zone d'incertitude pour les valeurs centrales.

Autour de chaque barycentre, nous calculons un espace qui a pour centre la position de ce barycentre et pour largeur, deux fois l'écart-type de ce barycentre. Les espaces positionnés dans l'une de ces zones est rattaché au barycentre de cette zone. Tant qu'il reste plus de 20% des espaces qui ne sont pas rattachés nous doublons la taille de chaque zones en vérifiant qu'elles ne se croisent jamais.

Le développement de la méthode de colorisation permet une analyse visuelle rapide de la classification des zones d'incertitudes. Cette affichage n'intervient en rien dans le processus de traitement des images. Cela nous offre l'opportunité de savoir si il y a beaucoup de zones d'incertitudes ou pas et ainsi décider si nous gardons cette nouvelle méthode.

- Les zones jaunes sont les espaces inter-mots
- Les zones rouges sont les espaces inclassables (espaces incertains)
- Les zones bleues sont les espaces inter-lettres

On obtient ici le résultat sur une ligne. Après avoir observé les résultats sur 24 textes, les résultats étaient plutôt satisfaisants. On a ainsi pu remarquer que les ponctuations étaient une situation récurrente qui donnait des espaces incertains. Mais beaucoup d'espaces incertains étaient encore assez distincts pour être classé.

Nous avons donc choisi de diviser la zone d'incertitude en 3 parties :

- Zone d'espaces incertain-lettres (espaces d'abord considérés comme incertains mais de taille proche des espaces inter-lettres)

- Zone d'incertitude
- Zones d'espaces incertain-mot (espaces d'abord considérés comme incertains mais de taille proche des espaces inter-mots)

On obtient ainsi :

J'ai donc ensuite participé à l'amélioration de la fonction de colorisation afin d'obtenir un résultat correspondant aux 5 zones maintenant délimitée.

Comme nous travaillons sur le découpage de ligne en mots, le traitement est réalisé sur les images ligne par ligne. La fonction de colorisation nous donnait ses résultats ligne par ligne. Pour obtenir une sortie plus proche du texte de départ, j'ai ajouté une nouvelle partie à la fonction permettant d'avoir toutes les lignes sur une même image. La comparaison entre le texte d'origine et sa version colorée est donc plus simple (cf. annexe figure ??) et permet de rapidement identifier les erreurs de classification des espaces (cf. annexe figure ??).

Cette fonction d'analyse est utilisée uniquement si l'on souhaite étudier les résultats. Elle est alors appelée lors de l'étape 4 du projet (figure ??) et permet de mieux comprendre les erreurs potentielles car cette catégorisation est importante pour la reconnaissance des mots (étape 7 de la figure ??).

3.3.1 Interprétation et critiques des résultats

Grâce à l'analyse des résultats, nous avons gardé la méthode de séparation de mots par histogramme. En effet, sur la totalité de nos textes, il restait que très peu de zones rouges (complètement incertaine). Les autres membres du groupes ont ainsi pu utiliser ces résultats pour répercuter le calcul de ces différents types d'espaces sur la suite du traitement et obtenir une amélioration de la reconnaissance globale. Les zones ont permis de préférer l'appel de certaines fonctions dans la suite du traitement et mettre en avant leurs résultats par rapport aux autres. Dans le cas des zones rouges, aucune fonction ne peut être avantagée.

3.4 Amélioration de la segmentation des lignes

3.4.1 Description du sujet de recherche

Lors de l'étude des textes colorisés grâce à la fonction expliquée précédemment, nous avons pour observer plusieurs textes donnant des résultats complètement inattendus et inexploitable. On peut observer 2 problèmes différents :

- 2 lignes sont considérées comme une seule et ne sont pas segmentée (cela peut ne concerner qu'une partie de la ligne)
- Une ligne est considérée comme 2 lignes et est découpée alors qu'il ne fallait pas

Dans cet exemple extrait de l'annexe figure ??, on peut voir 3 lignes mises en avant par un découpage rose fait à la main sur l'image sortie par le programme.

Le premier type de problème entre des erreurs comme le mot « C'est » de la ligne 1 qui empêche de détecter un espace inter-mots entre le mot « sans » et le mot « cesse » juste en dessous sur la ligne 2.

Le second type de problème entraine des espaces inter-mots beaucoup trop grand, comme entre le mot « MAGNIFIQUE » de la ligne 3 et le bout de mot « passes » de la ligne 2, considérés sur la même ligne par le logiciel à cause de l'erreur de segmentation.

Cet espace anormalement grand engendre des résultats erronés dans le calcul de barycentre, faisant de cet espace le seul compris dans l'intervalle des espaces inter-mots sur le texte (annexe figure ??). Pour contourner rapidement cette erreur, il a été choisi d'implémenter une fonction de suppression des valeurs aberrantes pour calculer les barycentres sans ces espaces qui ne devraient pas exister. La fonction supprime le plus grand espace tant que ce dernier est 2 fois plus large que l'un des 3 plus grands espaces excepté lui-même et continue jusqu'à qu'elle ne supprime aucun espace. Les valeurs aberrantes n'étant que trop grandes, il n'y a pas besoin d'appliquer un traitement semblable aux petites valeurs.

Cette altération est souvent utilisée en statistique (enlever les valeurs extrêmes pour ne garder que les valeurs correspondant à la majorité des réponses) mais ne corrige pas les erreurs de segmentation. Cela permet néanmoins de pouvoir traiter les autres lignes avec des espaces classés proche de ce qui est attendu et de pouvoir continuer la reconnaissance sur une partie du texte.

3.4.2 Méthode dichotomique

Recherche et travail effectué

Pour essayer de corriger cette segmentation, nous avons pensé à une nouvelle méthode. Avec cette nouvelle méthode, nous voulons nous assurer que la segmentation ne coupe aucun caractère. En effet, plusieurs erreurs provenaient de segmentation découpant une lettre en 2 car pour elle, traverser quelques pixels de texte n'était pas gênant s'il n'y avait pas de meilleure solution. La première étape est de travailler avec les blobs des mots au lieu du texte directement. Ainsi nous obtenons une forme grossière du texte permettant de bien distinguer les zones vides.

Notre algorithme fonctionne de manière récursive sur des portions de l'image et est séparé en 2 parties et qui s'arrêtera lorsque la partie traitée ne contient qu'une seule ligne.

Jusqu'ici, cette fonction obtenait son résultat en calculant un ratio entre la moyenne des hauteurs des blobs compris dans l'image et le nombre de rangée de pixels sur l'image. Ce ratio devait être inférieur à une valeur choisie arbitrairement, affinée après plusieurs tests.

Cependant, avoir une valeur entrée « en dure » dans le code est souvent source d'erreur sur des cas spécifiques. De ce fait, lorsque que nous sommes en bout de ligne et seul un bout de ligne est présent sur notre image, la fonction se trompe.

C'est pour cela qu'une nouvelle version de cette fonction en cours de développement avec l'utilisation d'histogrammes verticaux.

Sur l'image de la figure ??, il est facile d'identifier qu'il y a plusieurs ligne : nous observons une zone nulle entre les deux piques. Mais il arrive souvent que cela ne soit pas le cas. Lorsque des caractères de lignes différentes sont sur la même rangée de pixels, ils se confondent sur l'histogramme vertical et nous n'avons plus de zones nulles entre les deux pics.

Il faut donc être capable de détecter les maximums locaux correspondant aux/à la ligne(s) présente(s) sur l'image. La méthode était encore en cours de développement. Ici, le marquage noir vertical qui apparaît environ au milieu de chaque histogramme correspond à la moyenne des valeurs de cet histogramme. Nous considérons qu'une portion d'image contient deux lignes si il y a plus deux zones distinctes qui dépassent la moyenne.

L'objectif est d'obtenir à chaque fois des portions ne comportant qu'une ligne.

Pour arriver à ce résultat, nous évitons les lignes qui se chevauchent en découpant verticalement en 2 parties égales des images comportant plusieurs lignes. Nous pratiquerons ensuite le même traitement sur chaque moitié pour arriver à segmenter notre texte.

Notre méthode se sépare en deux fonctions : la première qui va trouver les séparations horizontales vides de texte entre les lignes et la seconde qui reliera ces segments.

L'algorithme de recherche fonctionne ainsi :

- **Si** l'image ne contient qu'une ligne :
 - On dessine un segment qui correspond au bord bas de l'image
- **Sinon** :
 - On cherche toutes les rangées horizontales de pixels vides dans l'image
 - **Si** on en trouve aucune :
 - On sépare en 2 parties verticales notre image et on rappelle cette fonction avec chacune des moitiés
 - On relie les lignes résultantes de l'appel sur la moitié gauche avec les lignes résultantes de l'appel sur la moitié droite (appelle de la fonction décrite ensuite)
 - **Sinon** :
 - On découpe notre image selon les rangées horizontales vides trouvées et on rappelle notre fonction sur chacune des portions (une portion est la zone entre 2 rangées de pixels vides)

Les images évoquées ci-dessous sont les 2 moitiés verticales créées lorsque nous avons plusieurs lignes et qu'on ne peut tracer de rangée de pixels vides. La fonction de liaison fonctionne ainsi :

- **Si** l'image de droite ne donne aucun segment :
 - On continue les séparations du côté gauche pour qu'elles atteignent le bord droit du côté droit et on ajoute ces segments agrandis à la liste de résultat
- **Sinon** :
 - **Si** il y a plus de ligne d'un côté :
 - on remplit la liste des rangées vides du côté ayant le moins de rangées vide avec des rangées correspondant au bas de l'image
 - **Tant que** il reste des lignes à relier :

- On prend le premier segment de chaque côté et on les relie
- On ajoute les trois segments (le segment de gauche, la liaison, le segment de droite) comme un seul segment dans la liste de résultat
- On retourne la liste de résultat

Interprétation et critiques des résultats

Cette méthode a cependant été abandonnée.

En effet, cette méthode est efficace pour la séparation de 2 lignes de textes différentes puisqu'elle s'arrange pour ne découper que des zones sans textes écrits. Malheureusement si un texte est penché (les lignes du textes ne sont pas parallèles avec le bord de la page), la méthode a beaucoup de mal à relier les segments entre eux.

La modification apporté à la fonction `isMultipleLines()` n'a pas donné les résultats attendus et requis.

La totalité de cette méthode repose sur la décision prise par cette fonction mais cette dernière n'arrive pas à de bons résultats.

A cause de ce blocage, la segmentation a toujours trouvé des lignes fausses, coupant des lignes de textes, créant des lignes vides, etc.

L'amélioration de la segmentation des lignes reste un axe de recherche qui pourrait permettre d'avoir de meilleurs résultats. Durant la réunion qui a conclu l'abandon de cette méthode, une réflexion commune a abouti au développement de la méthode Histogramme-A*, documentée ci-après.

3.4.3 Méthode Histogramme-A*

Description du sujet de recherche et expérimentations

Le principe est d'utiliser au maximum les histogrammes verticaux pour continuer de tracer notre segmentation entre les lignes. Ainsi, lors du calcul d'un histogramme vertical local, le programme continuera sa segmentation sur la première ligne blanche située entre 2 pics sur l'histogramme. Le tracé trouvé continu de progresser tout droit (à l'horizontal) jusqu'à qu'il entre en collision avec un pixel de texte. Afin de contourner ce caractère, la fonction calculera un nouvel histogramme local.

Si ce dernier ne permet pas de trouver un espace blanc entre 2 sommets, le programme utilisera l'algorithme A* (cf. Choix par A*) entre le dernier point enregistré et un point calculé plus loin, qui prendra en compte la pente du texte. L'algorithme A* permettra de contourner les lettres qui se chevauchent (empêchant l'histogramme de trouver une solution) et reliera ces 2 points.

Séparation du texte en paragraphes

La première étape est la séparation du texte entier en différent paragraphe. Pour cela, le texte est découpé dès qu'une rangée de pixels blancs est trouvée.

L'algorithme applique la suite de son traitement sur chacun des paragraphes. Le découpage se fait sur la première rangée trouvée (dans le cas où plusieurs rangées sont présentes à la suite).

Début de la segmentation (début des lignes)

Cette fonction va déterminer si le paragraphe actuellement traité contient une ou plusieurs lignes. Il utilisera d'abord deux histogrammes verticaux sur les premiers 2% et 10% de l'image (depuis le bord gauche). Pour que ce résultat soit cohérent, on considère qu'une personne écrivant de gauche à droite (comme en français) commencera toujours ses lignes avec un espace entre elles, même si elles ont tendances par la suite à se rapprocher.

On obtient donc un début de ligne pour chaque espace blanc entre deux sommets de l'histogramme. Pour commencer, le début de ces lignes commencera au plus haut de chaque espace blanc (juste en dessous du texte détecté). Par la suite, l'algorithme sera modifié pour commencer au milieu de cet espace blanc. Nous comparerons ensuite les résultats et garderons la meilleure méthode.

Une fois nos débuts de segmentations trouvés, le but est de les faire traverser le texte jusqu'au bord droit. Ainsi, chaque début de ligne sera d'abord continué de manière horizontale jusqu'à que le tracé rencontre un pixel appartenant au texte.

Choix par histogramme

Lorsque la segmentation est arrêtée par un pixel de caractère, un histogramme vertical local est calculé pour trouver un espace blanc proche.

Si un pixel de texte est trouvé entre le point de collision et le point donné comme solution par l'histogramme, le résultat est considéré comme erroné et la fonction renverra une valeur particulière notifiant de cette erreur. Cette vérification permet d'éviter de couper des caractères ou des mots lors du découpage en ligne.

Si l'histogramme ne trouve aucun espace blanc (dans le cas où les lignes se chevauchent), la fonction de calcul par histogramme renverra aussi la valeur d'erreur.

Dans le cas où nous réceptionnerons la valeur d'erreur, nous utiliserons l'algorithme A* pour prendre une décision

L'algorithme A*

L'algorithme de recherche A* (prononcé « A étoile » ou « A star » en anglais) est algorithme de recherche de chemin dans un graphe entre un nœud initial et un nœud final tous deux donnés. A* calcul le meilleur chemin entre les 2 nœuds. Le meilleur chemin dépend d'une fonction de coût, le résultat de l'algorithme étant le chemin au coût le plus faible.

L'idée est très simple : à chaque itération, on va tenter de se rapprocher de la destination, on va donc privilégier les possibilités directement plus proches de la destination, en mettant de côté toutes les autres.

Toutes les possibilités ne permettant pas de se rapprocher de la destination sont mises de côté, mais pas supprimées. Elles sont simplement mises dans une liste de possibilités à explorer si jamais la solution explorée actuellement s'avère mauvaise. En effet, on ne peut pas savoir à l'avance si un chemin va aboutir ou sera le plus court. Il suffit que ce chemin amène à une impasse pour que cette solution devienne inexploitable.

L'algorithme va donc d'abord se diriger vers les chemins les plus directs. Et si ces chemins n'aboutissent pas ou bien s'avèrent mauvais par la suite, il examinera les solutions mises de côté. C'est ce retour en arrière pour examiner les solutions mises de côté qui nous garantit que l'algorithme nous trouvera toujours une solution (si tenté qu'elle existe, bien sûr).

On peut donc lui donner un terrain avec autant d'obstacles qu'on veut, aussi tordus soient-ils, s'il y a une solution, A* la trouvera.

Choix par A*

Afin de bien comprendre cette partie, il est nécessaire de connaître le fonctionnement de l'algorithme A*. Il est expliqué dans la sous-section précédente (cf. algorithme A*).

L'algorithme A* utilise 3 éléments pour déterminer son choix :

- Une position de départ qui sera le dernier point représentant une variation de hauteur (Point de Variation de Hauteur) dans le tracé de segmentation actuel. Le premier PVH est toujours le point le plus à gauche de notre début de ligne.
- Un point d'arrivée qui sera calculé en fonction de la pente du texte par rapport au précédent PVH.

- Une matrice de pixel qui représentera l'image du paragraphe actuellement traitée qui sera utilisé comme un graphe, où seuls les pixels qui n'appartiennent pas à du texte seront des nœuds du graphe.

La méthode A* renverra une liste de PVH qui sera le chemin à suivre par le tracé de segmentation entre le point de départ et le point d'arrivée. Dans le cas où A* ne trouve pas de solution, le paragraphe est considéré comme ne contenant qu'une ligne de texte.

Le résultat ne présente que les PVHs de la solution pour ne garder que les points importants de la segmentation.

L'histogramme ne trouvera pas de solution (espace entre le 'g' et le 'd' est sur la même ligne que le 'l'). On lance alors l'algorithme A* entre le point de départ de la ligne rouge et le point sur la même ligne 10% de la largeur de l'image plus loin, après vérification d'avoir trouvé un espace blanc.

Interprétation et critiques des résultats

Cette méthode a aussi été arrêté pendant son développement pour être ré-adaptée. En effet, en parallèle aux premiers résultats, il a été pensé de travailler avec cette algorithme dans une zone de sûreté, où l'on pourrait faire avancer la segmentation ainsi que l'algorithme A*, en s'assurant qu'il ne couperait aucune ligne.

Cette nouvelle idée a donné lieu à notre dernière méthode, aujourd'hui en cours d'implémentation.

3.4.4 Méthode Histogramme-Rectangle-A*

Cette méthode étant une évolution de la méthode précédente (Histogramme-A*), elle reprend certaines étapes de cette dernière.

Ainsi, le découpage de la totalité du texte en paragraphes, sur lesquels nous appliquerons notre traitement a été repris (cf ??).

Comme cité dans la partie précédente, nous cherchons ici à trouver une zone sécurisée où faire progresser notre segmentation.

Histogrammes séparés

Pour parvenir à trouver des zones sécurisées sur la totalité de la largeur de nos paragraphes, nous travaillons avec seulement un dixième de du paragraphe (sur la totalité de la hauteur). Nous nous retrouvons donc avec 10 colonnes de la même hauteur que notre paragraphe et largeur équivalente à un dixième de la largeur de notre paragraphe

Sur chacune de ces colonnes, le programme calcul un histogramme vertical sur la totalité de cette partie du paragraphe. Il cherche ensuite les valeurs nulles de notre histogramme. Grâce à cela, il trouvera des zones vides de caractères écrits et pourra déclarer cette zone comme sécurisée.

Pour obtenir un résultat visuel sur ces calculs, nous traçons des rectangles colorés correspondant à ces zones de blancs trouvées par les histogrammes. Ces rectangles ont pour largeur la même que la colonne, comme hauteur le nombre de rangées consécutives blanches trouvées par l'histogramme et se commençant au niveau de la première rangée de pixels blancs de l'histogramme.

Après les premiers résultats, nous avons choisi de retirer un maximum de rectangle inutile. Les premiers que nous avons enlevé sont les rectangles en contact avec le haut ou le bas de l'image. En effet, ces rectangles ne donnent aucune information pour séparer 2 lignes du texte écrit. Cela nous permet aussi de n'avoir aucun rectangle sur les paragraphes ne contenant qu'une seule ligne de texte écrit.

Dans la figure ??, nous pouvons identifier un autre problème. Ce dernier est créé par les accents, les points des « i » ou des « j », par certaines ponctuations, etc. Les histogrammes verticaux trouvent souvent un espace blanc entre eux et le mot le plus proche. Pour résoudre ceci, dès qu'un pic fin est identifié sur l'histogramme (moins de 30 valeurs non nulles à la suite), nous essayons de rattacher cet élément au gros pic le plus proche.

Pour cela, nous partons depuis le pic identifié. Nous mesurons ensuite les espaces blancs au dessus et en dessous de notre pic. Si un des espaces blancs arrive jusqu'au bord de l'image, il n'est pas pris en compte. Une fois ces mesures réalisées, on considère le plus petit des deux comme inexistant. Notre élément est alors rattaché au gros pic le plus proche.

Liaisons des rectangles

La prochaine étape est d'analyser la position des rectangles pour déterminer ceux qui montre une séparation entre deux lignes de textes écrits.

Les rectangles ont été sauvegardés dans cet ordre (dû au traitement par histogramme) : d'abord tous les rectangles des premiers 10%, du plus haut au plus bas puis du les rectangles entre les 10% et 20% de la largeur de l'image de haut en bas, etc. jusqu'aux rectangles des derniers 10% de haut en bas.

Pour relier les rectangles et former des lignes, notre algorithme commence par chercher le prochain rectangle non-relié et commence une nouvelle ligne de segmentation avec ce dernier. Il va ensuite connecter tous les rectangles qui ont au moins une rangée de pixels en commun avec l'un des trois derniers rectangles de notre début de ligne (ou tous les rectangles de notre début de segmentation s'ils sont moins de trois).

Le rectangle testé actuellement doit aussi avoir au plus 2 fois 10% de l'image d'écart avec le dernier rectangle du début de segmentation.

Un rectangle ne peut donc être relié qu'avec un rectangle situé à sa gauche.

Après avoir réussi à attacher tous nos rectangles, nous avons modifié notre fonction de colorisation des espaces pour qu'elle nous montre une couleur différente par ligne et ainsi mieux observer les résultats obtenus.

Grâce à cet exemple, il est possible de remarquer plusieurs comportements à utiliser ou à corriger.

Le premier concerne le rectangle rose. Il est le seul de sa couleur car il ne possède aucune rangée de pixels commune avec le rectangle vert à sa gauche. Le rectangle seul comme celui-ci ne fournissant que peu d'informations, nous avons choisi de les supprimer pour la suite de notre traitement

Le second se situe sur la ligne bleue à la troisième colonne. Deux rectangles sont attachés à la même ligne. Or deux rectangles de la même colonne ne peuvent et ne doivent pas être reliés à la même ligne sinon la segmentation ne saura pas lequel est le choix à suivre.

Nous gardons donc le premier qui se présente, celui du haut.

Cependant, ce rectangle du haut est ensuite supprimé pour une autre raison remarquée durant nos tests.

Un rectangle n'ayant aucune rangée en commun avec son prédécesseur ni son successeur est un rectangle donnant de mauvaises informations. Le plus souvent, le rectangle isolé se situe de l'autre côté de la ligne écrite et donnerait donc une segmentation traversant la ligne écrite et la découpant en plusieurs parties.

La dernière fonction de sélection des rectangles enlève les rectangles, donc les espaces blancs trop fins. Ils sont souvent causés par la séparation en 10 colonnes qui coupe des caractères en deux.

C'est le cas dans notre exemple (figure ??) : le premier rectangle bleu de la troisième colonne est alors supprimé. Ainsi, le second rectangle bleu de la 3ème colonne (figure ??) est le seul rectangle relié à la ligne bleue sur cette colonne. Il réapparaît donc dans nos résultats suivants (figure ??).

De plus, le rectangle vert de la cinquième colonne (figure ??) est maintenant considéré comme trop petit et disparaît aussi (figure ??).

C'est ici que se termine le traitement sur nos rectangles. Ils vont maintenant être transformés pour la prochaine étape.

Des rectangles vers le tracé de segmentation

Sur la figure ??, la sélection des rectangles est terminée et nous nous retrouvons avec des colonnes vides au milieu des groupes de rectangles.

C'est à ce moment que l'algorithme A* (cf partie algorithme A*) trouvera son utilité. Mais pour travailler avec cet algorithme, nous devons le fournir

deux points. Après plusieurs tests, nous avons choisi de sauvegarder 2 point de chaque rectangle : le point au milieu de son côté gauche et le point au milieu de son côté droit. Grâce à ce choix, la segmentation s'écarte au maximum des 2 lignes écrites.

Cette distance donne plus de sûreté à la solution de l'algorithme A*. Sur la figure ??, nous avons tracé 2 exemples de solution que pourrait nous donné l'algorithme de plus cours chemin : en rouge si nous découpons selon nos rectangles au plus proche du texte et en bleu si nous gardons un marge de sécurité.

On remarque que si nous restons au plus près du texte, l'algorithme A* peut choisir comme solution de passer par dessus un mot et créer de mauvaise segmentation.

Les points que l'algorithme A* devra relier ont été choisi. Dès que deux rectangles consécutives dans nos groupes de rectangles sont à plus de 10% de la largeur de l'image de distance (c'est-à-dire qu'il y a au moins une colonne vide entre les deux rectangles), cette méthode sera utilisée.

Les limites de l'utilisation d'A*

Après observations des résultats de l'étape précédente, nous avons trouvé deux situations à cause desquelles nous obtenions de mauvaises segmentations :

- lorsqu'il n'y a pas de rectangle entre le côté gauche de l'image et le début de notre groupe de rectangle
- lorsqu'il n'y a pas de rectangle entre la fin notre groupe de rectangle et le côté droit de l'image

La ligne rouge (première ligne de la figure ??) possède les 2 situations :

La solution trouvée pour corriger ces deux situations est une nouvelle fois l'utilisation de l'algorithme de plus court chemin A^* .

Malheureusement, dans ces deux situations nous n'avons qu'un seul point à utiliser pour notre algorithme : le point du bout de notre groupe de rectangles.

Pour obtenir un second point et relier notre groupe de rectangle aux bords de l'image, un point fictif est nécessaire. Nous devons donc extrapoler la position de ce point à partir des données que nous avons. Pour obtenir les coordonnées de ce point fictif, nous avons implémenté un modèle de régression linéaire en utilisant comme données les points déjà identifiés sur notre ligne.

Nous utiliserons le plus simple qui est l'ajustement affine. Cela consiste à rechercher une droite permettant d'expliquer le comportement d'une variable statistique Y comme étant une fonction affine d'une autre variable statistique X . Il s'agit donc de trouver deux réels a et b , tels que l'on ait, à quelques erreurs près :

$$Y = aX + b \quad (1)$$

Un ajustement affine est envisageable lorsque le nuage de points associé au couple de variables statistiques (X, Y) semble s'organiser autour d'une droite. C'est notre cas : nous utiliserons les PHVs (exemple sur l'image de la figure ??) comme donnée. Les coordonnées des PHVs détermineront leurs valeurs sur les deux axes pour le calcul de notre fonction affine.

L'ajustement affine permet de procéder à des prévisions sur un comportement futur proche ou à des interpolations entre deux mesures effectuées. Il existe plusieurs méthodes d'ajustement affine dont la plus usitée est l'ajustement par la méthode des moindres carrés, exemple de régression linéaire simple.

Une fois que nous connaissons l'équation correspondant à notre fonction (ses facteurs a et b , cf équation ??), il sera facile de récupérer les coordonnées de nos point factices.

Leur abscisse (valeur X) correspond au côté où nous le cherchons (0 pour le côté gauche et « largeur de l'image » pour le côté droit). Leur ordonnée en remplaçant X par leur abscisse dans notre fonction affine.

$$\hat{a} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \quad (2)$$

$$\hat{b} = \bar{y} - a\bar{x} \quad (3)$$

Où \bar{x} est la moyenne des abscisses des points donnés :

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n} \quad (4)$$

Où \bar{y} est la moyenne des ordonnées des points donnés :

$$\bar{y} = \frac{\sum_{i=1}^n y_i}{n} \quad (5)$$

\hat{a} et \hat{b} sont les estimateurs respectifs de a et b . Ce sont des estimations des valeurs et non leurs valeurs exacts.

Afin de réaliser tout ces calculs, nous avons besoins de connaître les coordonnées d'un maximum de point de notre groupe tracé de segmentation. C'est pour cela que la gestion de ces deux cas se fera après en dernier.

Cependant, il nous faut un maximum de point pour calculer cette pente. Si nous avons trop peu de rectangle, L'extrapolation d'un nouveau point pourrait donner de mauvais résultats.

Interprétations et critiques des résultats

L'analyse visuelle des derniers résultats rend cette solution très prometteuse (figure ??).

Sur les 24 textes de notre base de tests 2017, seuls 2 textes présentes des soucis de segmentation contre 9 avec la méthode présente à mon arrivée.

Ces résultats sont vrais si l'implémentation de l'algorithme de plus court chemin se fait sans soucis et donnent les résultats attendus.

En effet, son intégration étant en court, nous ne pouvons pas obtenir de résultats chiffrés sur une amélioration de la reconnaissance.

4 Conclusion

4.1 Avancée du projet

Le but de ce stage était l'amélioration du projet IScan, un logiciel de reconnaissance intelligente de caractères manuscrits développé au sein de la cellule innovation d'Aubay. Cette amélioration se traduit par une augmentation du taux de reconnaissance de l'application.

J'ai tout d'abord travaillé sur une fonction d'affichage qui a permis aux autres membres du projet d'analyser plus rapidement le résultat de traitement sur lequel ils travaillaient. En effet, la fonction de colorisation que j'ai développé avec l'aide d'un collègue a permis d'avoir un retour visuel sur la façon dont le programme classait les espaces. De plus, cette fonction de colorisation a permis de mettre en avant quelques soucis de la segmentation du texte en lignes, ce qui a orienté la suite de mon travail sur ce projet.

Reprendre la segmentation des lignes a occupé la majorité de mon temps de travail sur le projet.

Notre premier axe de recherche, la méthode dichotomique, n'a pas abouti mais m'a permis de comprendre les situations qui pouvait poser problème lors de ce traitement. Elle était efficace sur les textes où les lignes étaient presque horizontales et bien séparées. Or ceci n'est pas le cas de la majorité des textes et c'est en ajoutant cette nouvelle problématique que nous avons abordé notre réflexion sur la méthode suivante.

La seconde méthode a relancé nos recherches. C'est à partir de ce moment où nous avons pensé à l'intégration d'un algorithme de plus court chemin pour trouver la meilleure segmentation dans les zones difficiles (lettres mal formées, lettres de différentes lignes très proches l'une de l'autre, etc). La documentation avait complètement été écrite avant son développement ce qui nous a permis de trouver la troisième solution.

Cette dernière méthode applique un premier traitement pour déterminer la plus grande zone de sécurité possible où une séparation entre deux lignes doit intervenir. Cela permet d'avoir une plus grande confiance des résultats obtenus par l'algorithme de plus court chemin.

Les résultats obtenus sont prometteurs. Cependant, l'intégration de l'algorithme de plus court chemin n'ayant pas encore était terminée nous n'avons pas pu obtenir de résultats chiffrés.

4.2 Conclusion personnelle

Participer au développement d'un projet d'entreprise m'a donné un nouveau regard sur la méthodologie de travail.

Bien que la méthode dichotomique ait été abandonnée, cela m'a montré que l'organisation d'un projet en entreprise est bien différente de celle des projets réalisés en tronc commun à EPITA. Il est important de prendre le temps avant de se lancer dans le développement d'une solution. En effet, cette méthode a commencé à être développée avant la rédaction d'un quelconque document. Si nous avions respecté cet ordre, nous aurions sûrement passé moins de temps dessus. Écrire un document permet de bien réfléchir à la façon dont sera codée la solution pensée et de s'assurer de ne pas dévier de l'idée de départ.

Cependant, mon stage m'a aussi fait vivre le cycle de travail habituel dans les projets de recherche et développement. Lorsque nous cherchons des solutions, nous commençons par poser des hypothèses. Cela permet d'y réfléchir avec l'équipe, de poser l'idée sur papier et de penser aux cas particuliers. Après les premiers résultats, il est important de tester le plus de cas possible et de bien analyser les résultats. Parfois, développer quelques fonctionnalités annexes pour faciliter l'étude des résultats, comme la fonction de colorisation, peut être très utile.

Lors de nos recherches, nous devons essayer des choses et nous avons le droit à l'échec, comme ce fut le cas avec la méthode dichotomique.

Ce projet m'a donné l'occasion de travailler avec des personnes de formations différentes et ainsi échanger sur nos connaissances et expériences diverses. Nous avons ainsi pu apprendre les uns des autres.

Le travail proposé m'a donné l'occasion de confirmer la voie vers laquelle je me dirige pour la fin de mes études ainsi que pour avenir professionnel.

Annexes

Sommaire des annexes

Pré-traitement et colorisation des espaces	1
Segmentation avec la méthode dichotomique	3
Segmentation avec la méthode dichotomique ratée	4
Coloration : observation et correction	5

Pré-traitement et colorisation des espaces

Segmentation avec la méthode dichotomique

Segmentation avec la méthode dichotomique ratée

Coloration : observation et correction



FIGURE 6 – Organisation d'Aubay France

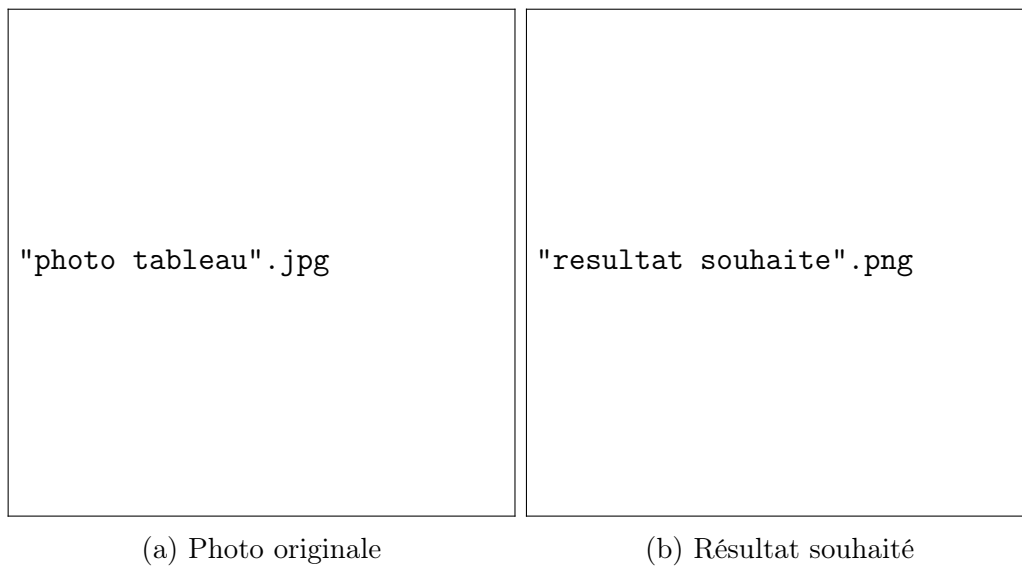


FIGURE 7 – Objectif du projet I'Scan



FIGURE 8 – Les différentes étapes du projet I'Scan

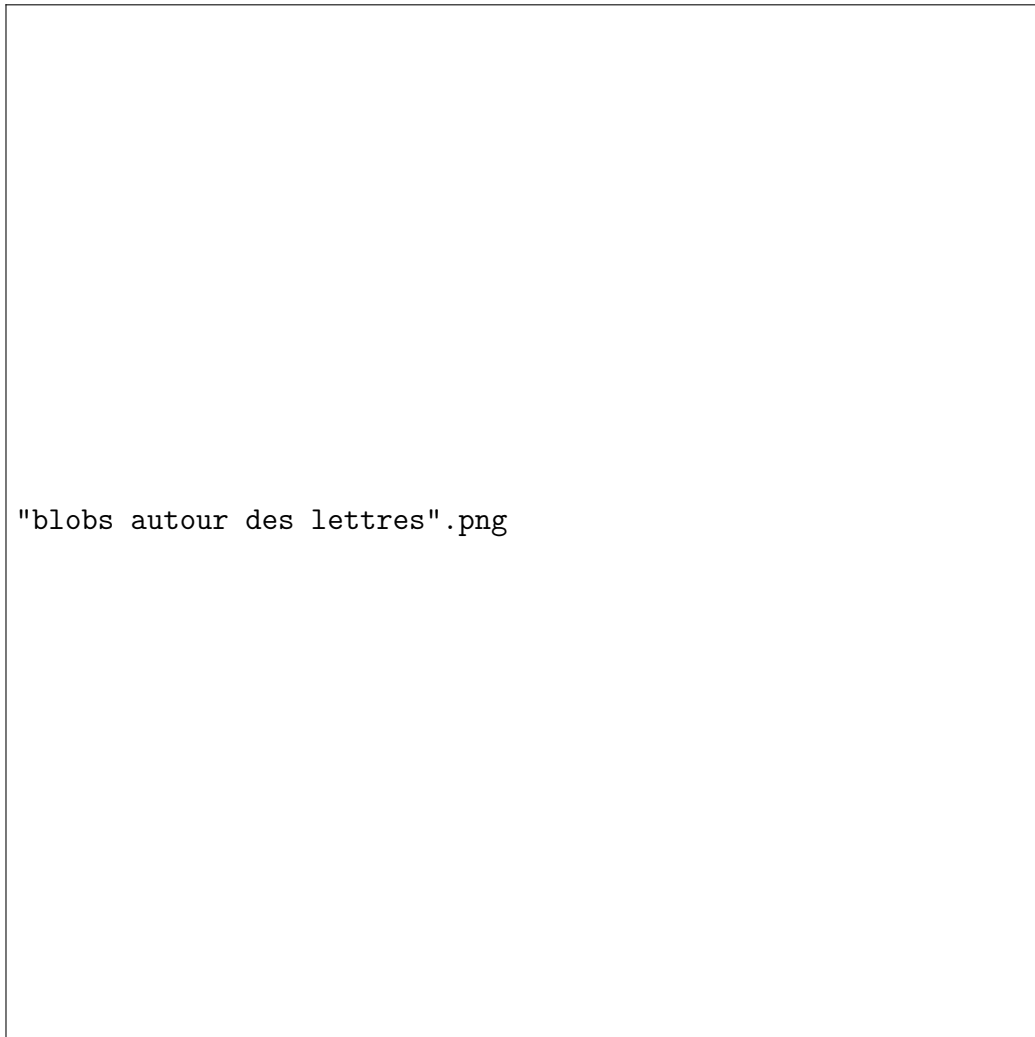


FIGURE 9 – Blobs autour des lettres

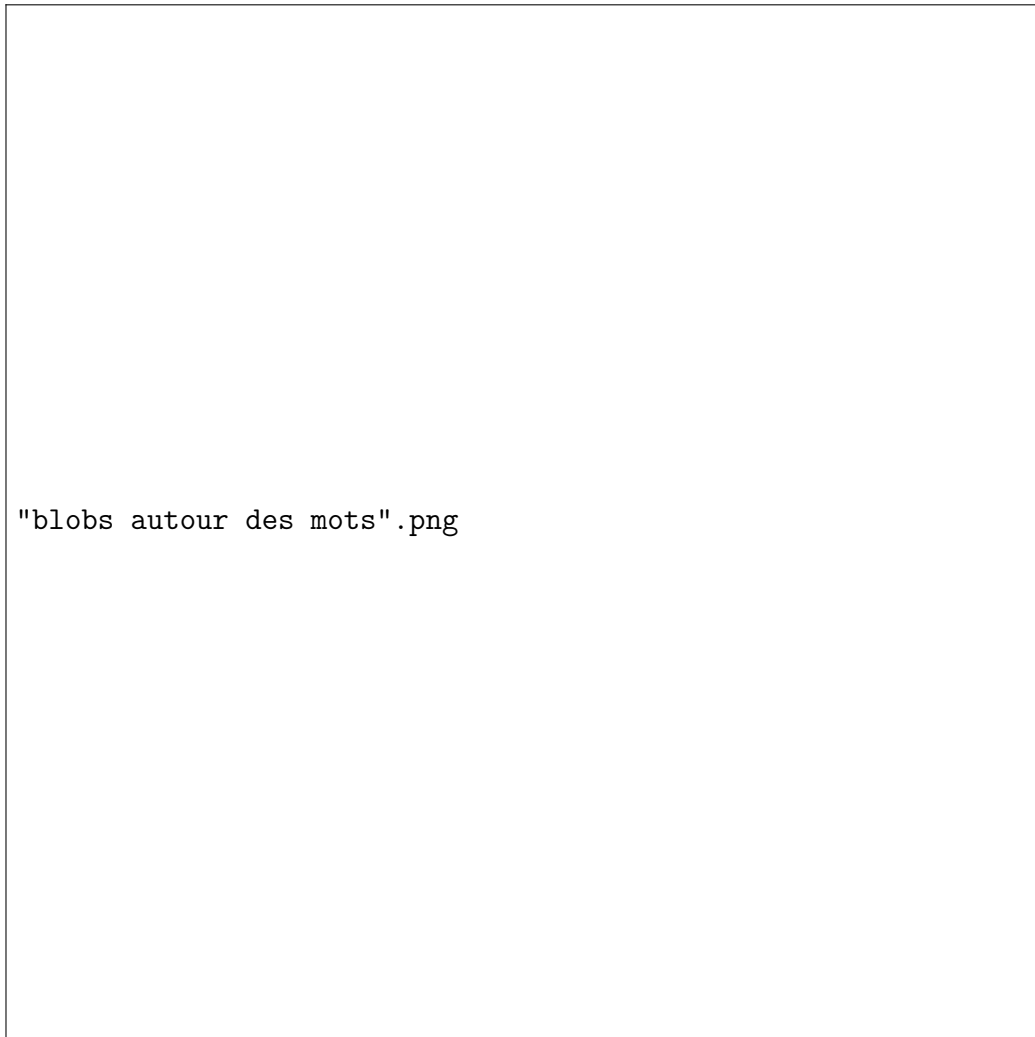
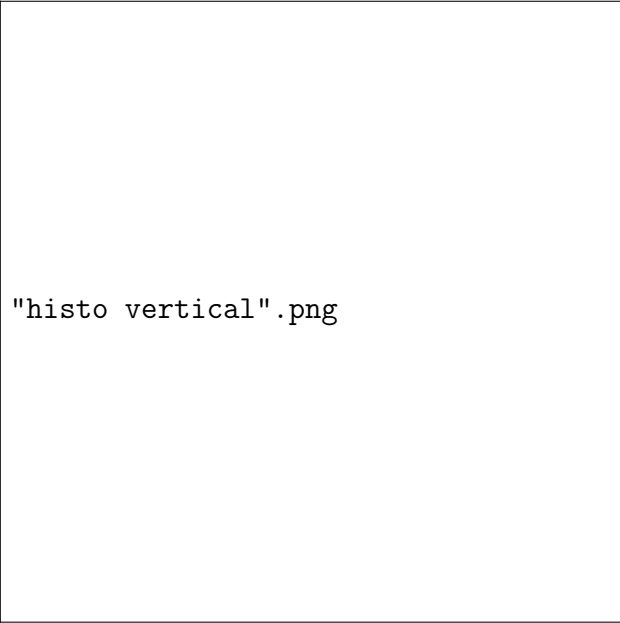


FIGURE 10 – Blobs autour des mots



"histo vertical".png

FIGURE 11 – Histogramme vertical



FIGURE 12 – Représentation des espaces et des barycentres



FIGURE 13 – Résultat de la colorisation sur une ligne



FIGURE 14 – Codes couleurs de la fonction de colorisation



FIGURE 15 – Résultat de la colorisation sur une ligne avec 5 couleurs



FIGURE 16 – Exemple d’erreur de segmentation posant problème dans le texte de l’annexe figure ??

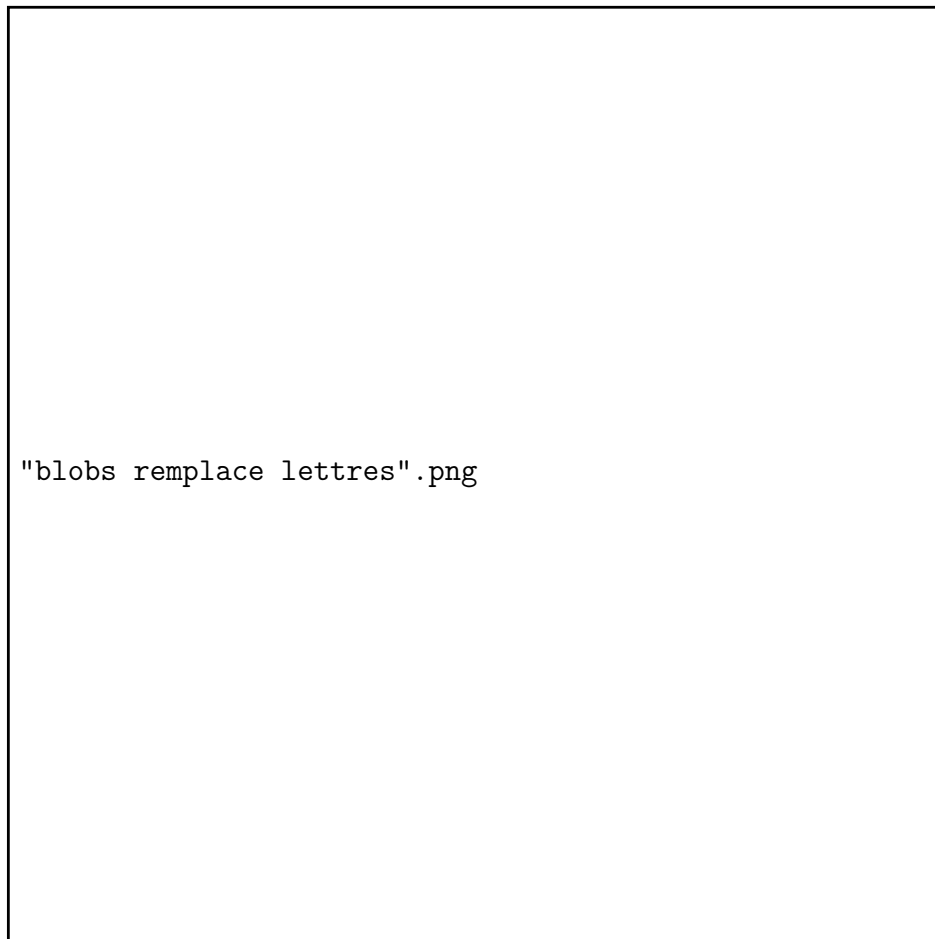
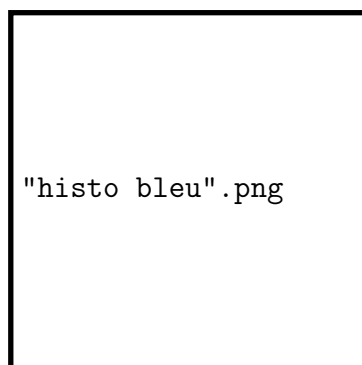
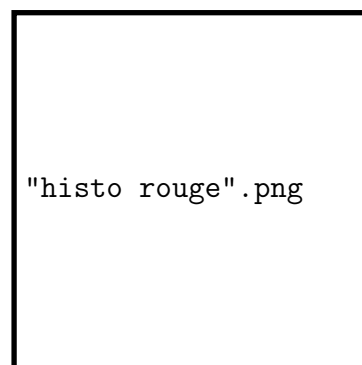


FIGURE 17 – Les blobs remplacent les lettres (extrait de l'annexe figure ??)



(a) Histogramme de plusieurs lignes



(b) Histogramme d'une ligne

FIGURE 18 – Affichage de l'histogramme vertical avec représentation de la moyenne

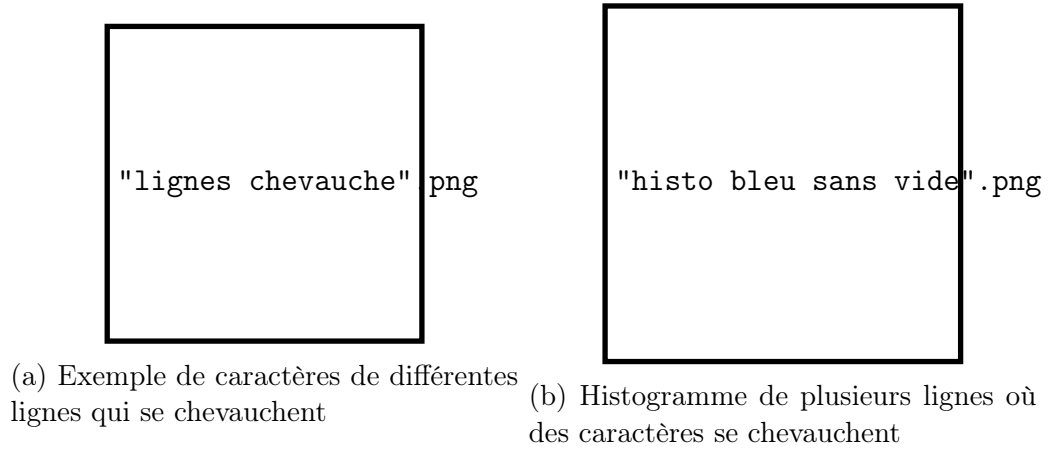


FIGURE 19 – Exemple de cas avec chevauchement des lignes

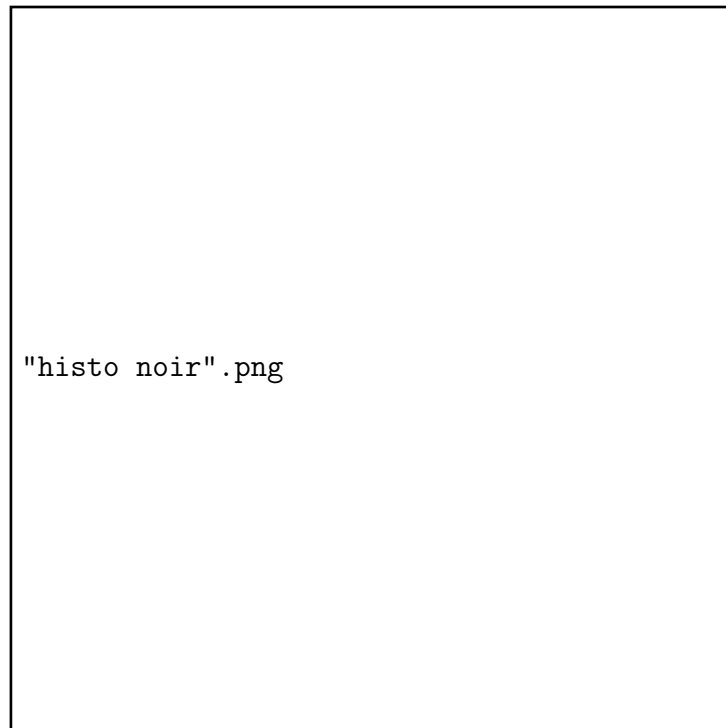


FIGURE 20 – Exemple d'espaces entre 2 lignes sur un histogramme vertical

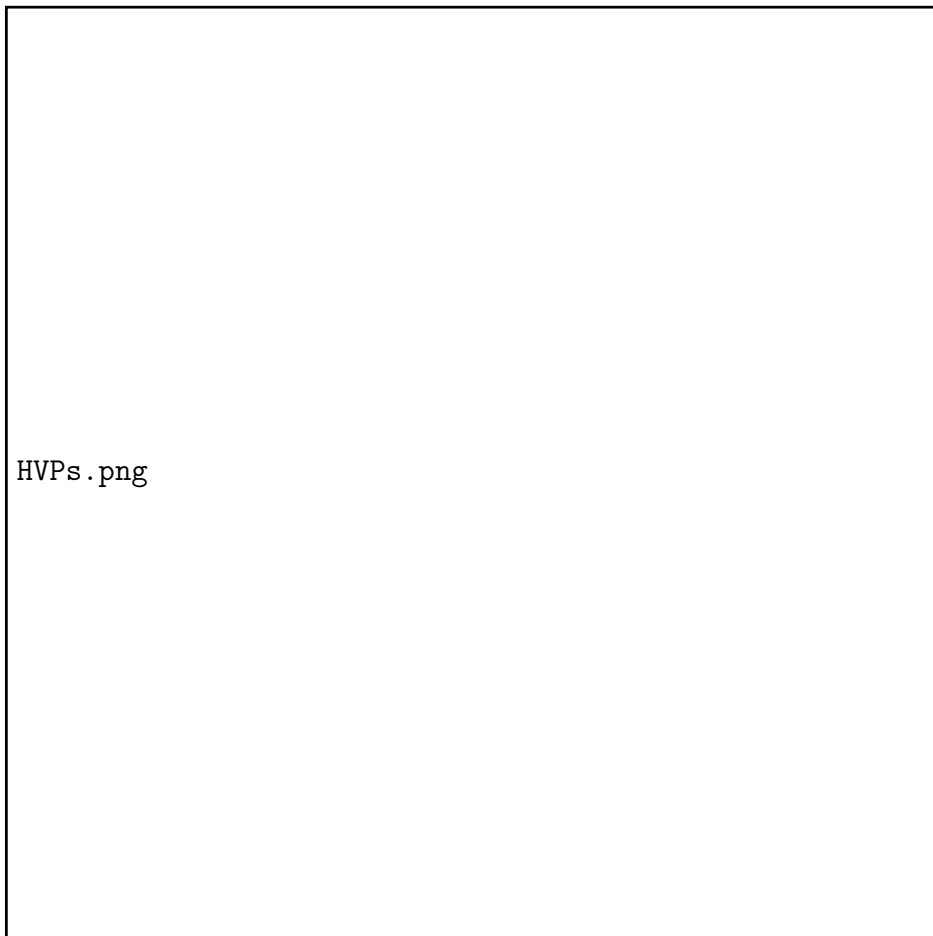


FIGURE 21 – Exemple de Points de Variation de Hauteur (PVH)

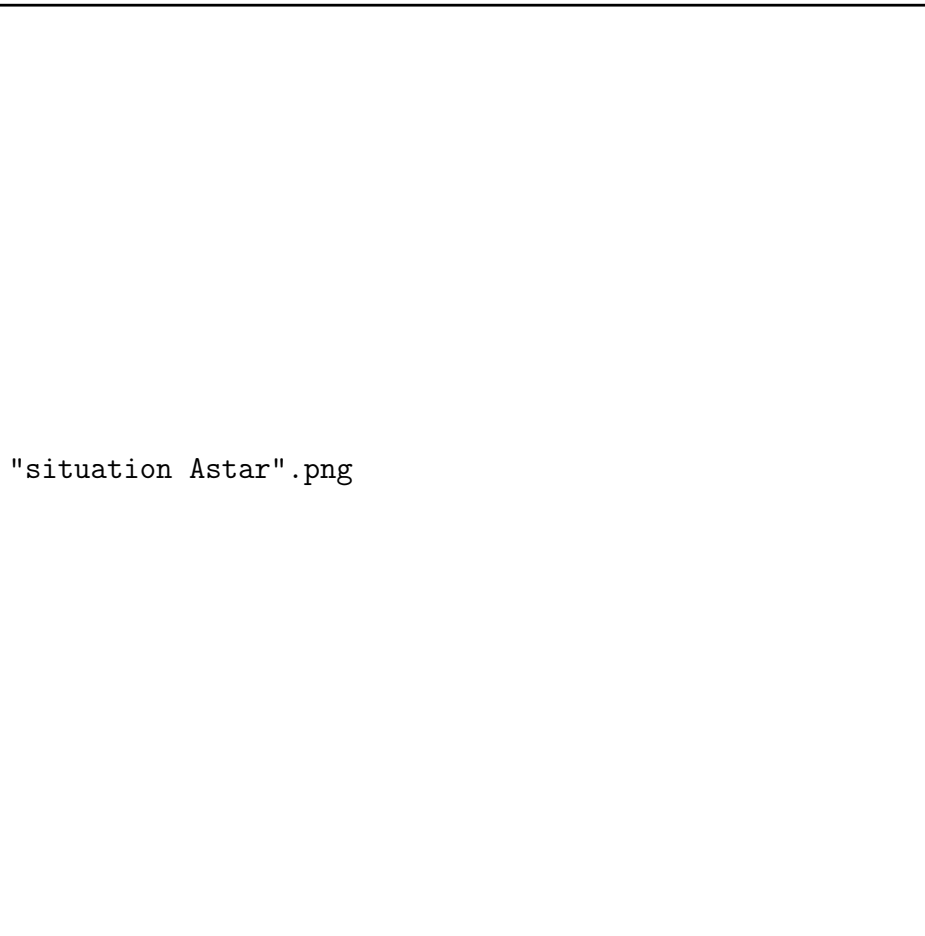
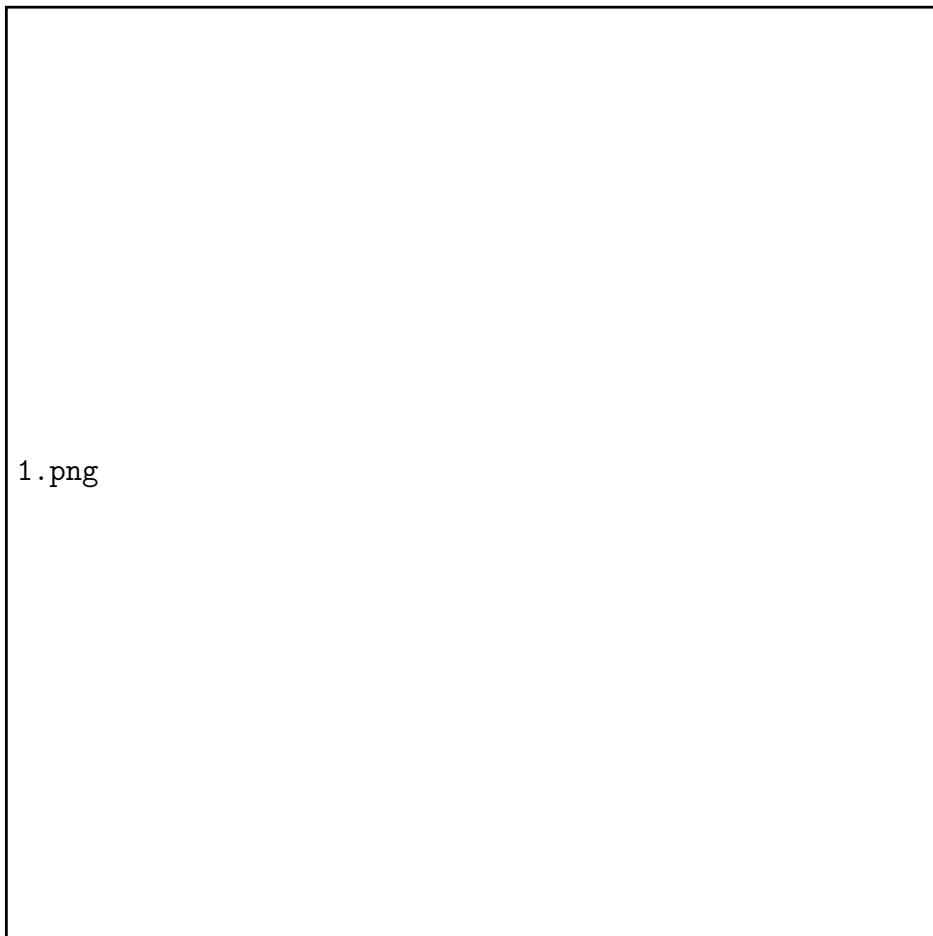


FIGURE 22 – Exemple de situation d'utilisation de l'algorithme A*



1.png

FIGURE 23 – Premier résultat

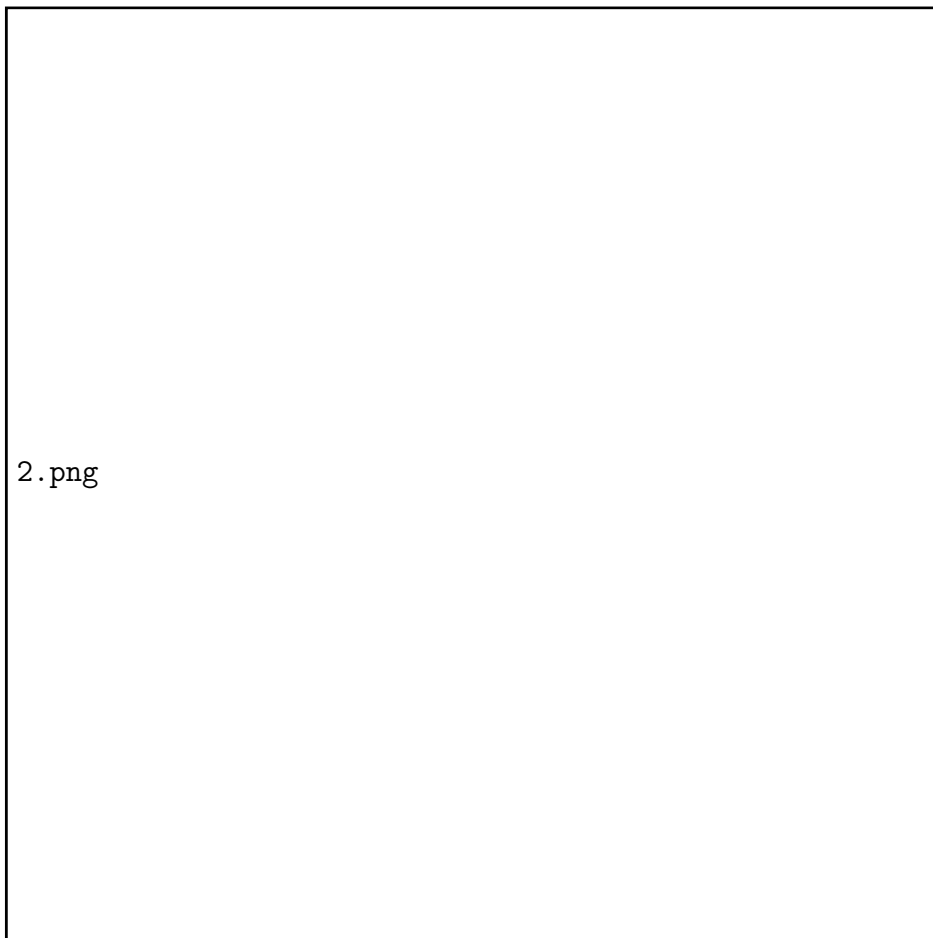


FIGURE 24 – Les rectangles atteignant le haut ou le bas de l'image ont été retirés

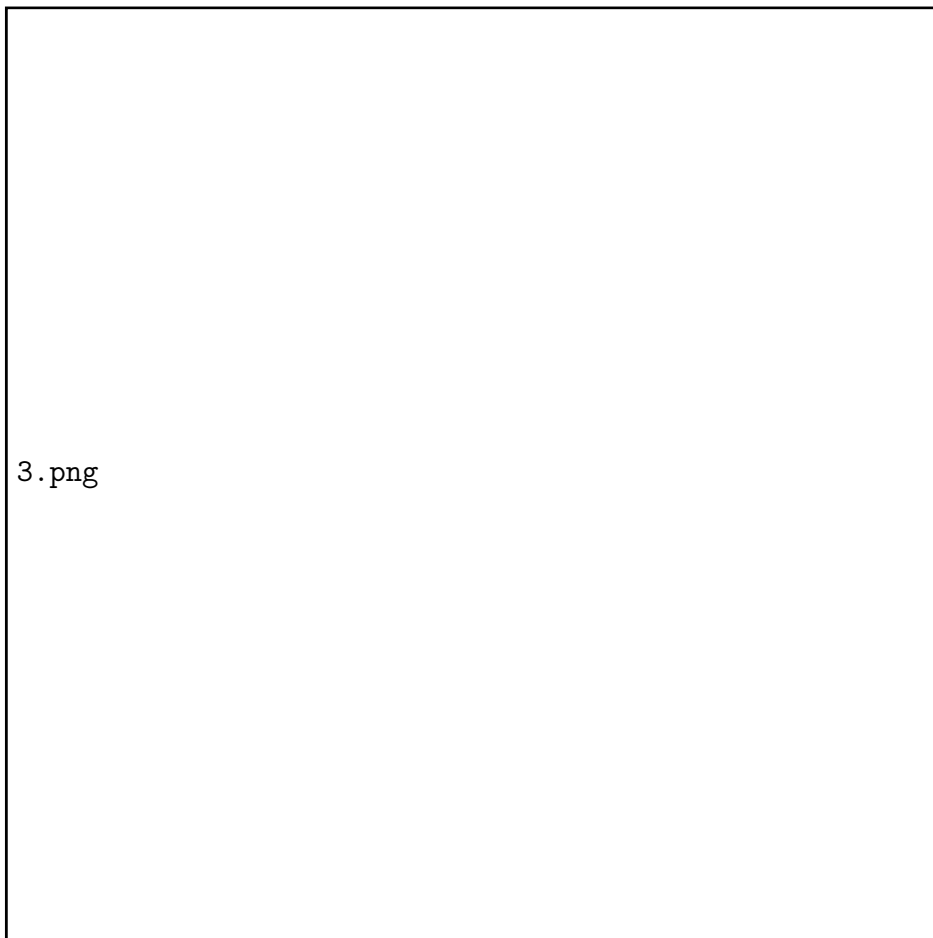


FIGURE 25 – Les accents et les ponctuations ont été regroupé au mot correspondant

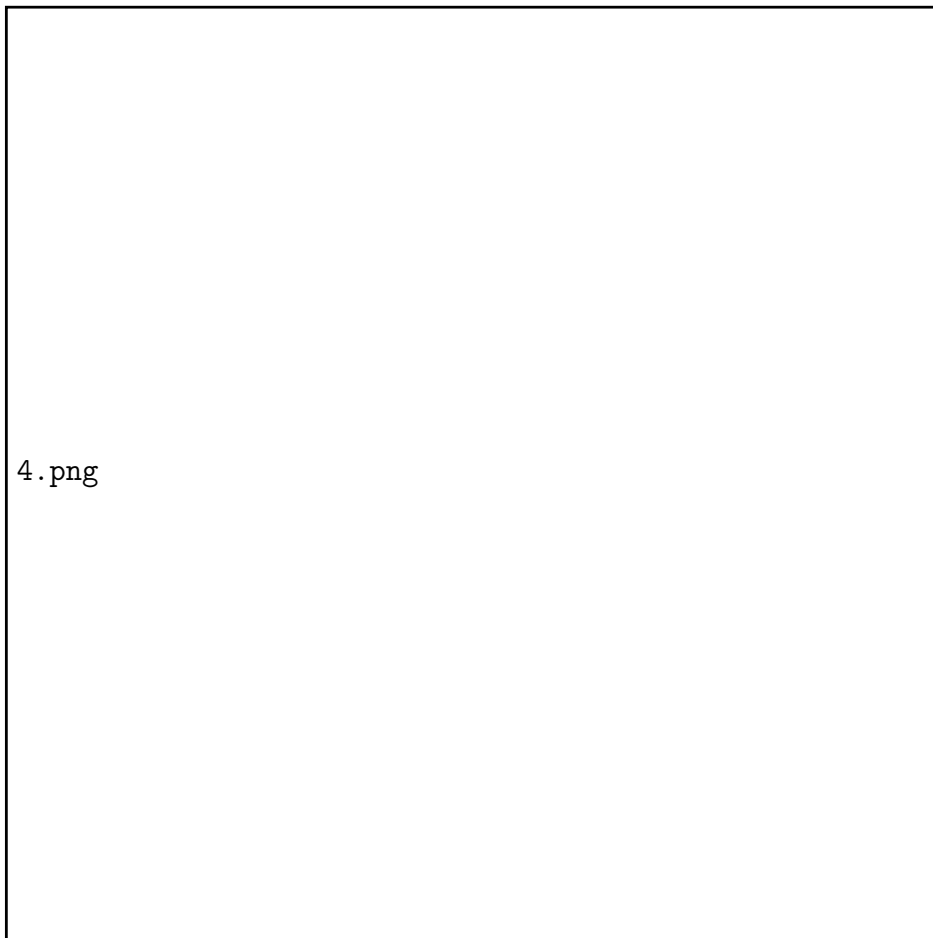
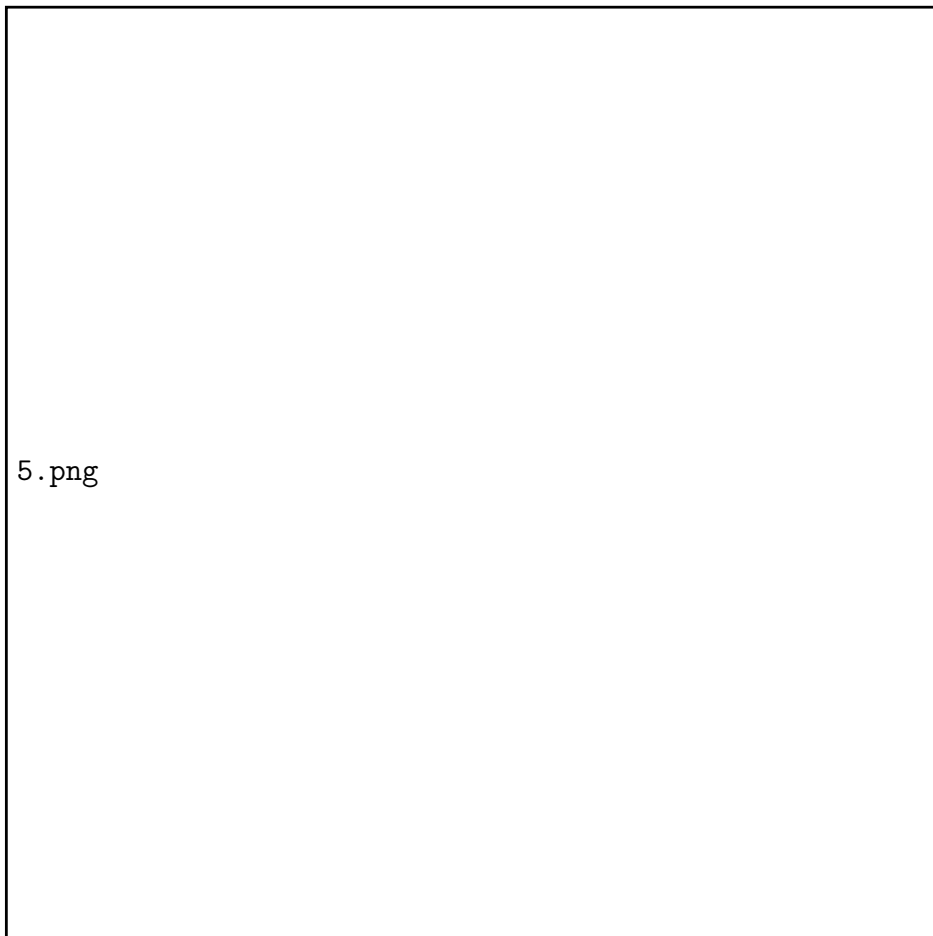
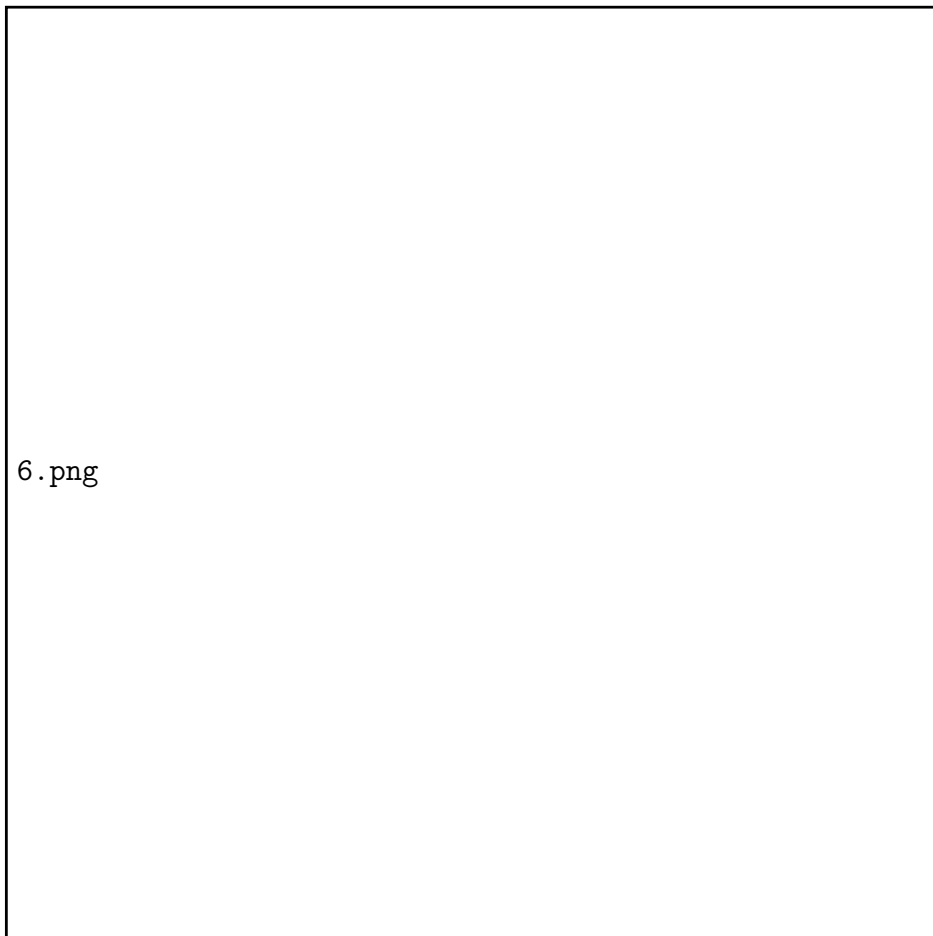


FIGURE 26 – Segmentation colorée



5.png

FIGURE 27 – Résultat obtenu après ces divers améliorations



6.png

FIGURE 28 – Résultat obtenu après ces divers améliorations

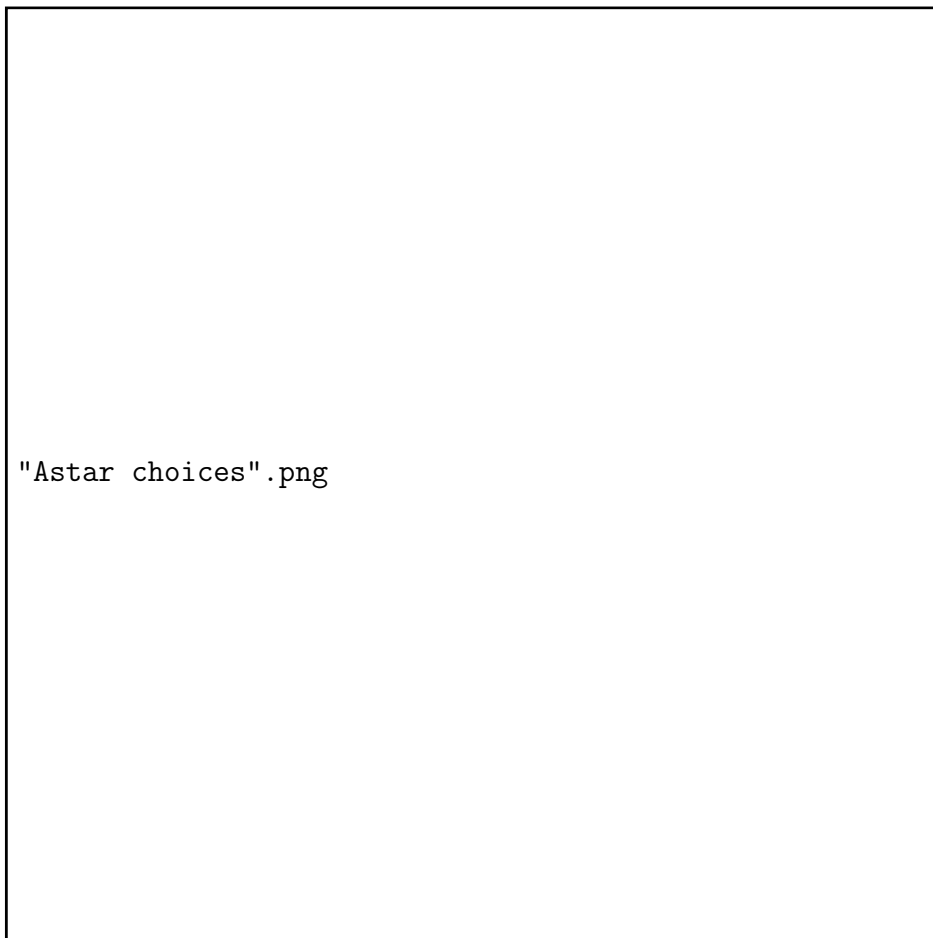
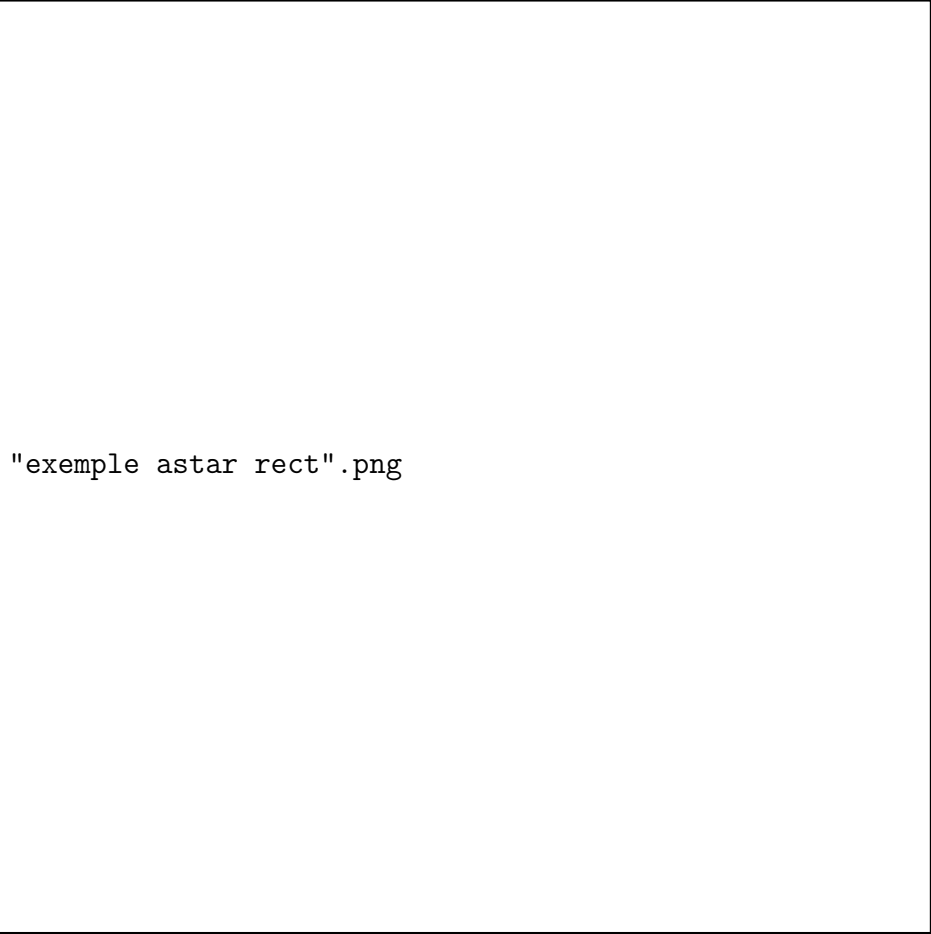


FIGURE 29 – Choix d'A* possible en fonction des points des rectangles sau-
vegardés




FIGURE 30 – Découpage en ligne avant l'implémentation d'A*



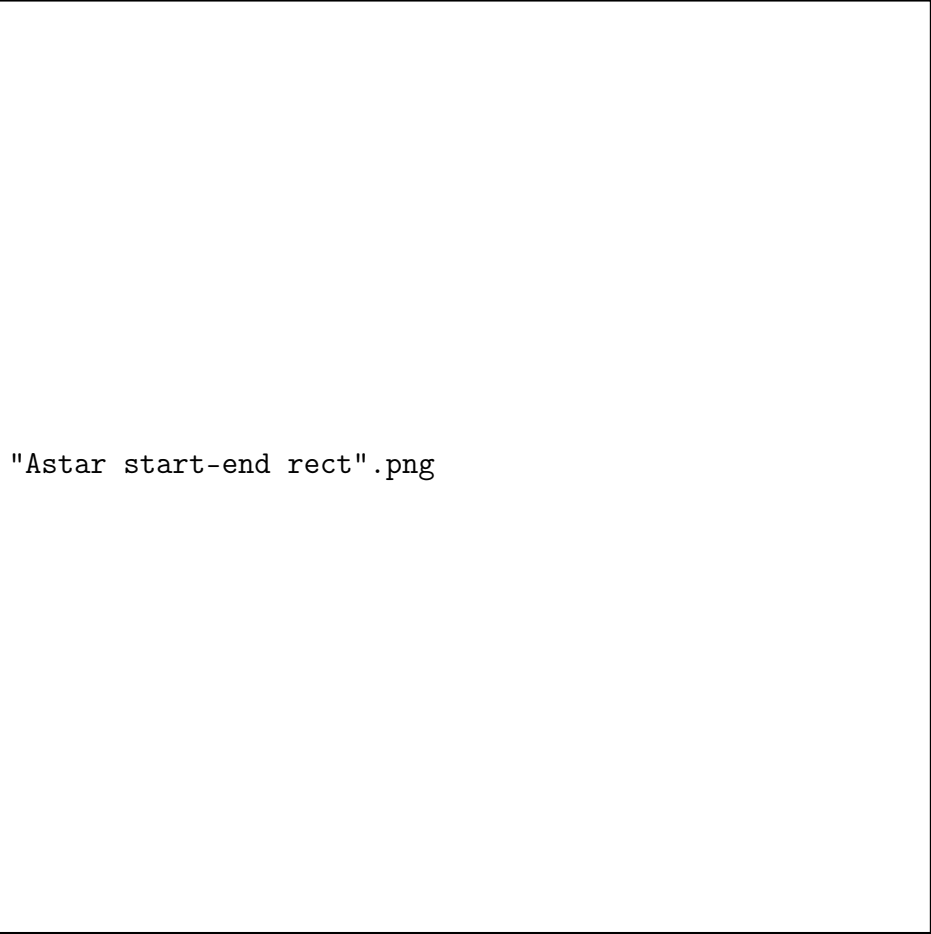
`"exemple astar rect".png`

FIGURE 31 – Exemple avec situations où A^* sera appelé



"exemple astar ligne".png

FIGURE 32 – Découpage en ligne avec A^*



"Astar start-end rect".png

FIGURE 33 – Ligne(s) avec un début et/ou une fin manquant(e)

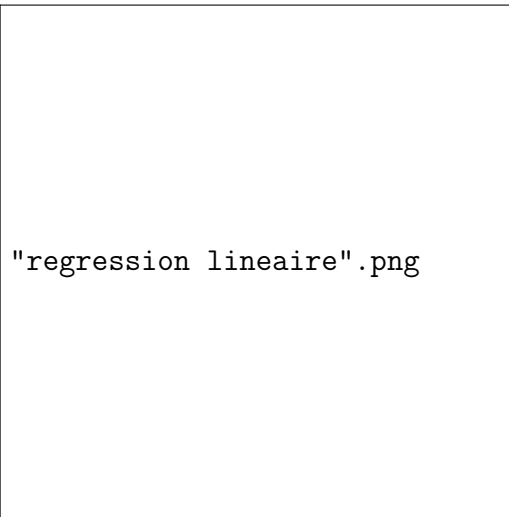



FIGURE 34 – Exemple d'un modèle de régression linéaire



"Astar start-end ligne".png

FIGURE 35 – Affichage des tracés de segmentation calculés avec les fonctions affines obtenues par la régression linéaire (A^* non utilisé pour montrer clairement la pente obtenue avec l'ajustement affine)

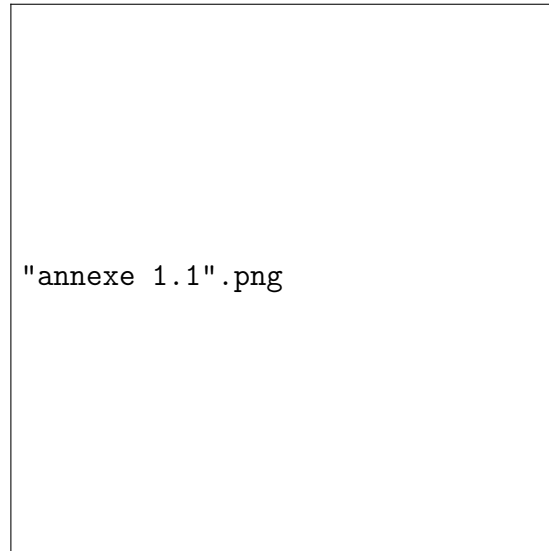


FIGURE 36 – Exemple de photo originale de texte

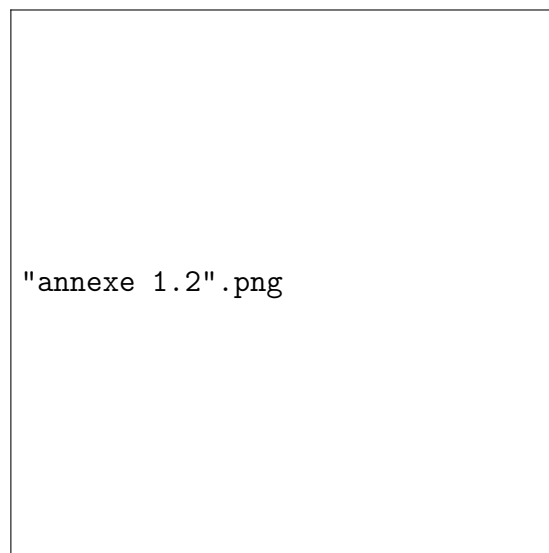


FIGURE 37 – Texte après binarisation : les pixels ont pour valeurs 0 (noir) ou 1 (blanc)

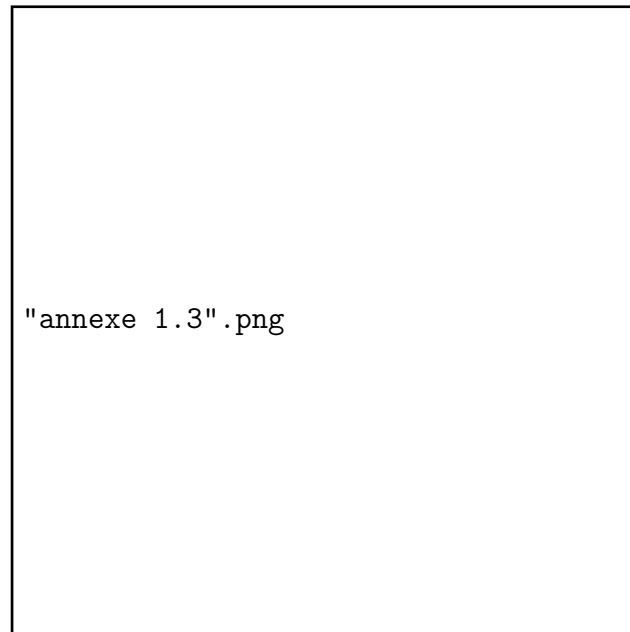


FIGURE 38 – Résultat de toutes les lignes colorisées mise sur une même page (segmentées avec l'ancienne méthode)

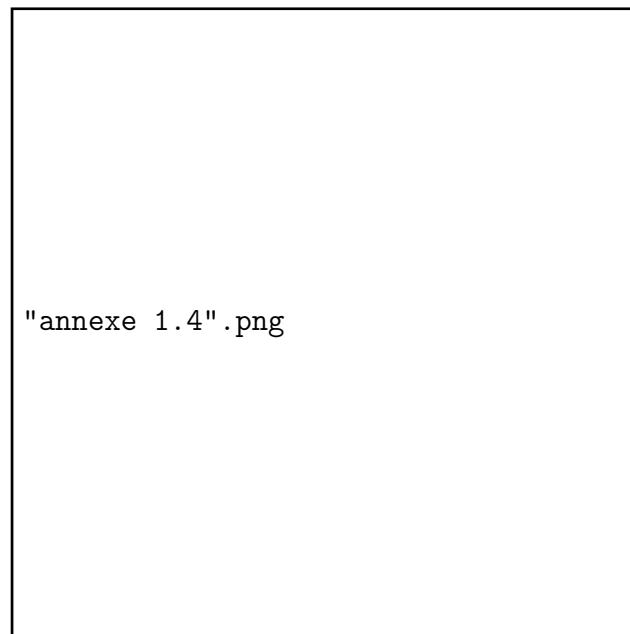


FIGURE 39 – Résultat prétraitement du traçage des blobs avant application de la méthode dichotomique

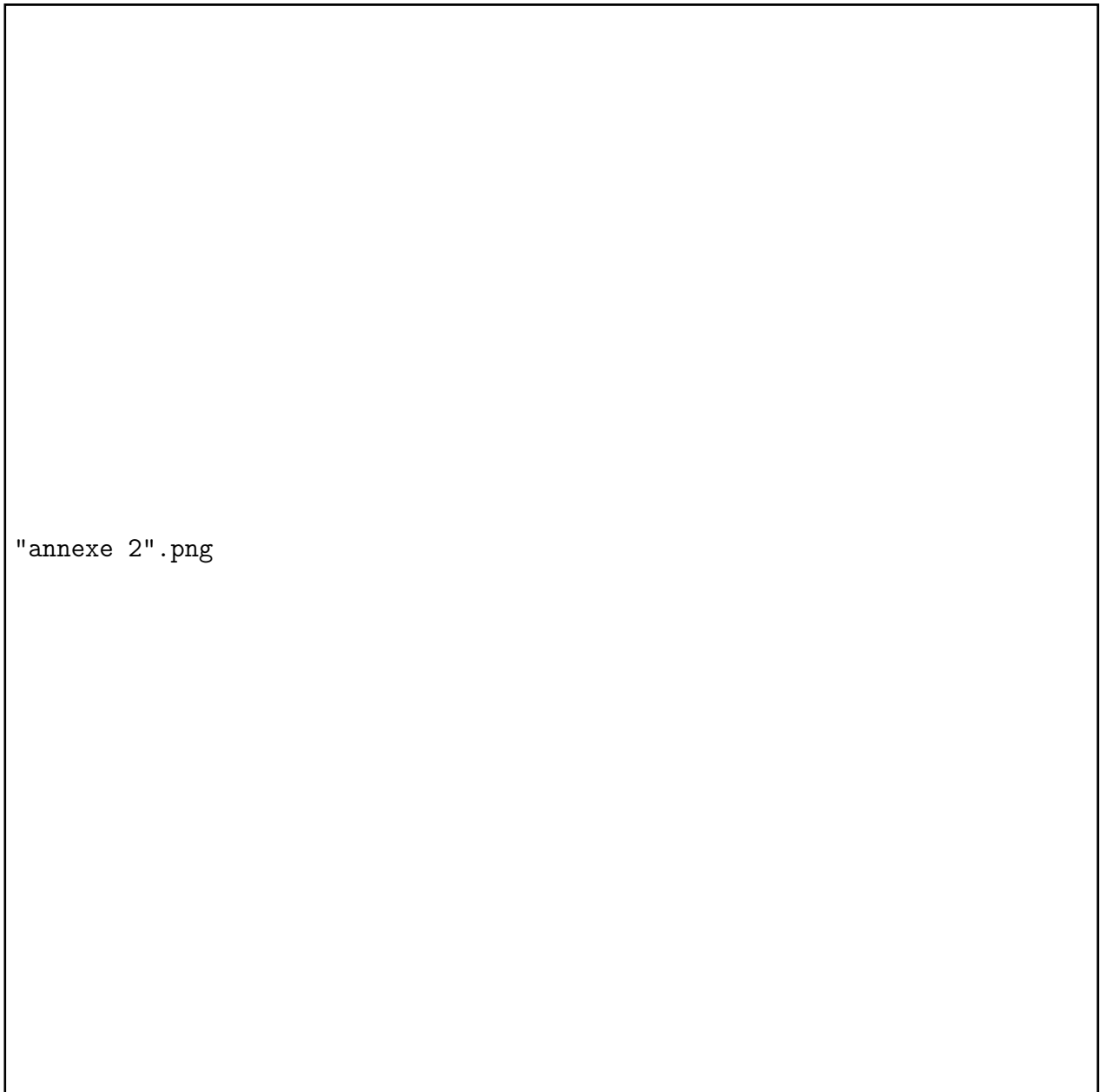


FIGURE 40 – Segmentation des lignes avec la méthode dichotomique

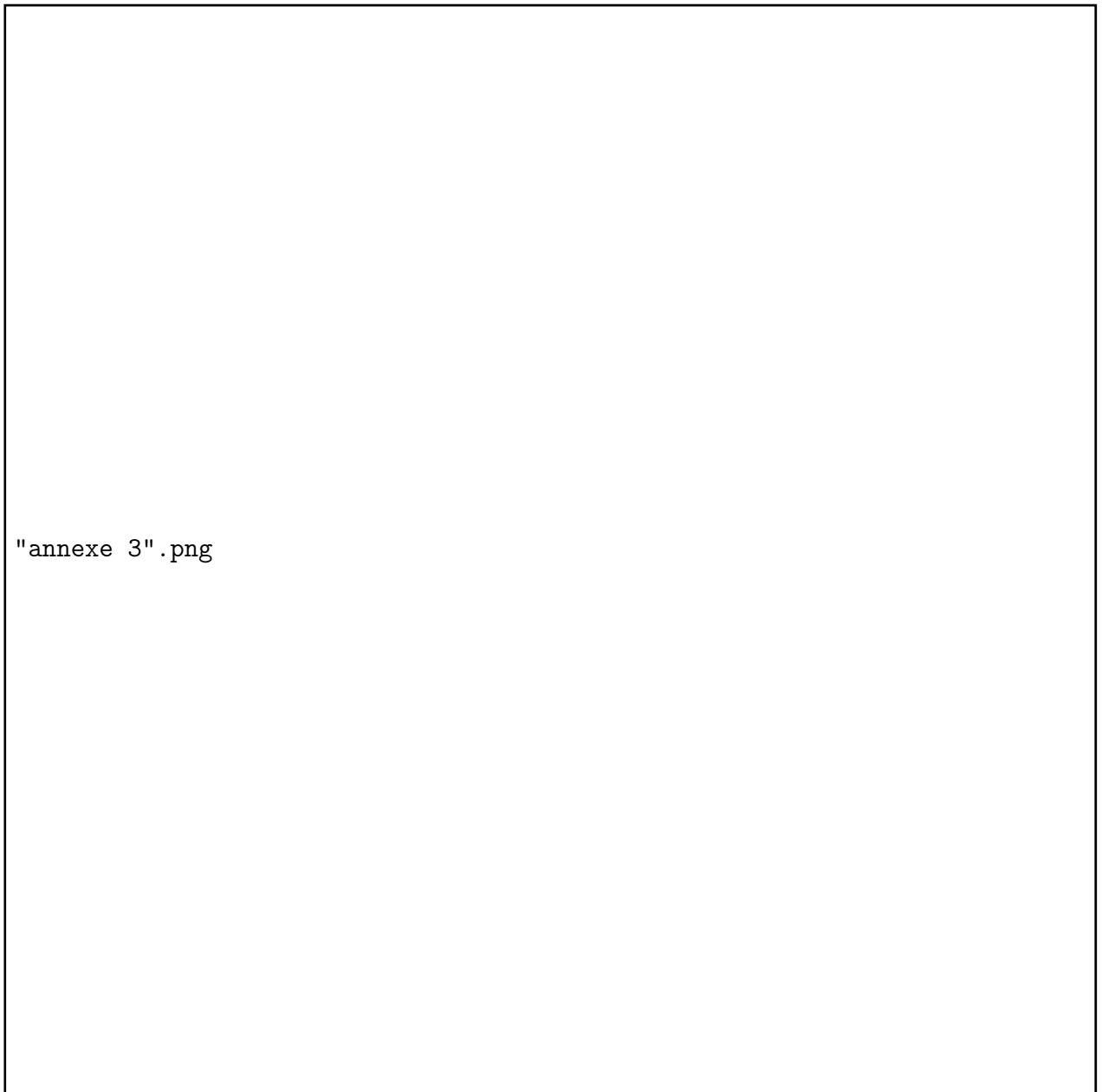


FIGURE 41 – Exemple de segmentation ratée lors de l'application de ma méthode

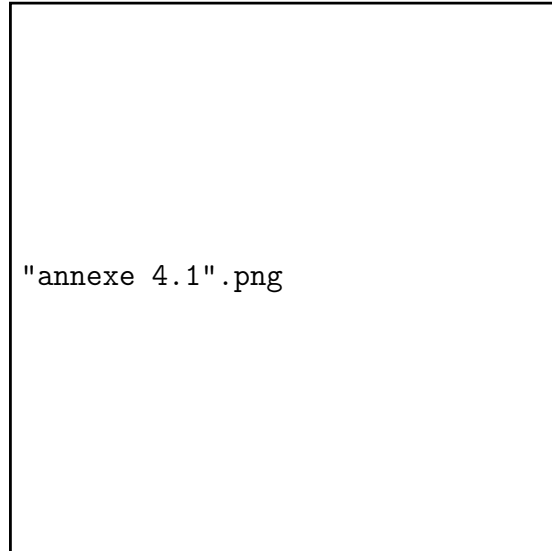


FIGURE 42 – Observation de segmentation ratée avec l'ancienne méthode grâce à la fonction de colorisation

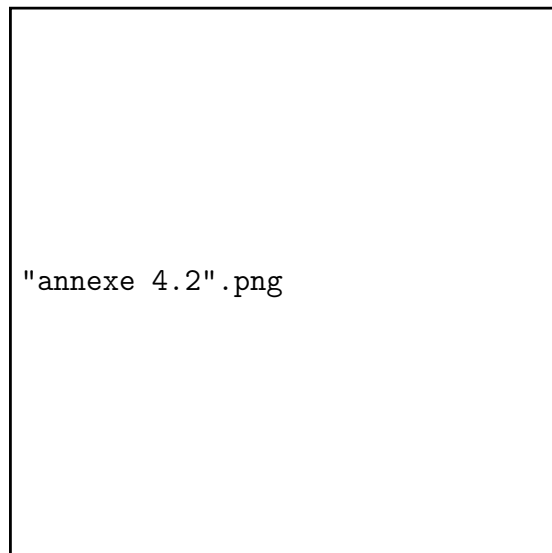


FIGURE 43 – Correction en enlevant les valeurs aberrantes lors du calcul des barycentres et des classes d'espaces