RAPPORT DE PROJET

Parebrise numérique et intelligent

Projet de recherche

Université de Haute-Alsace, année 2017/2018

Professeurs référents : Karim Hammoudi et Mahmoud Melkemi

UNIVERSITÉ HAUTE-ALSACE

Julien Arnold / Laurent Galassi

Sommaire

Introduction

Le projet qui nous a été proposé par M. Hammoudi et M. Melkemi porte sur les technologies que l'on pourrait apporter sur un parebrise numérique et intelligent. Le principe de ce parebrise est d'afficher une vidéo filmée par caméra en temps réel de l'avant de la voiture.

Plusieurs problématiques ont été soulevées concernant le traitement de l'image qui serait diffusée sur le parebrise afin d'en améliorer la qualité perçue par l'automobiliste. Les principales améliorations que nous avons évoquées sont :

- **Débrumage** : Consiste à supprimer le brouillard dans la mesure du possible
- **Alpha blending** : Affichage sur le parebrise d'informations concernant le véhicule, générales comme la température ambiante ou encore la navigation GPS
- **Traitement de la pluie** : Suppression des gouttes de pluie
- **Traitement de l'ensoleillement** : Réduction du niveau de luminosité si celle-ci est trop forte
- **Traitement de l'aveuglement** : On imagine qu'un rayon de soleil intense vienne perturber le conducteur, ce rayon serait traité afin d'être supprimé avant même qu'il n'ait pu attendre la vision de celui-ci

Les nouvelles technologies sont présentes au quotidien autour de nous, notamment dans le domaine automobile, où il ne cesse d'y avoir des avancées notables, comme les véhicules autonomes, la connectivité avancée ou encore au niveau de la sécurité. Les voitures sont devenues des condensés de technologie, alliant intelligence artificielle, caméras, aides à la conduite et encore bien d'autres fonctionnalités.

De nos jours la sécurité au volant pose encore problème et les conditions météorologiques sont souvent un facteur à risque, c'est pour cela que l'amélioration de la visibilité permettrait de réduire les chances d'accident ou même d'augmenter le confort de conduite.

Les idées proposées pourraient avoir un réel impact dans les véhicules du futur et jouer un rôle important sur la sécurité des automobilistes.

Nous avons utilisé différents outils afin de mener à bien ce projet :

- **Visual Studio C++** : environnement de développement simple d'utilisation, performant et qui est facilement configurable
- **OpenCV** : bibliothèque graphique de traitement d'images ou vidéos, nous l'avons choisi car c'est une technologie connue de nos professeurs qui nous l'ont recommandée pour effectuer les différents traitements
- **GitKraken** : outil de versioning graphique en vogue aujourd'hui
- **Discord** : logiciel de discussion pour échanger des informations à distance

Motivation

Nous avons tous deux été très intéressés par ce sujet car il est question d'un travail de recherche sur des nouvelles technologies dans le domaine automobile.

Le parebrise numérique intelligent est une technologie de pointe qui n'existe pas encore en tant que tel, c'est-à-dire que des recherches existent déjà sur l'amélioration de la visibilité mais rarement dans le domaine automobile. Nous avons été attirés par le côté concret de ce projet, par l'analyse des documents existants et le traitement des informations qui permettent de créer une approche qui correspond aux aspects que nous voulions traiter.

Nous avons particulièrement apprécié le débrumage, c'est pourquoi nous avons choisi de nous focaliser dessus afin de réaliser un travail poussé et proposer une solution fonctionnelle sur cet aspect dans le temps imparti. Pour réaliser cela des traitements d'image et de vidéo décrits dans la partie méthodologie ont été nécessaires.

On peut classifier les différents types de brouillard :

- Radiatif
- D'advection
- De précipitations
- D'évaporation
- De mer arctique
- Orographique
- D'inversion
- De vallée

L'algorithme que nous proposons a pour but de fonctionner sur les brouillards radiatifs et de précipitation car ils sont graduels (une image visible sur le bas et qui se floute de plus en plus en montant).

État de l’art du débrumage

	Dark Channel	Alpha Blending	Débrumage variationnel	Débrumage statistique	Contraste et luminosité
DEFADE				Oui	Oui
Amélioration et restauration de Yong Xu	Oui	Oui			
Projet débrumage de K. Sun et J. Tang	Oui				
Notre méthode					Oui

Dark Channel :

Le Dark Channel est calculé comme une érosion dans l'espace RGB. Il faut prendre un certain minimum de l'image calculé. Le nombre de pixels va changer la granularité de l'image obtenue.

Alpha Blending :

C’est une fusion de deux images qui vise à améliorer sa visibilité dans le cas de l’amélioration d’une image.

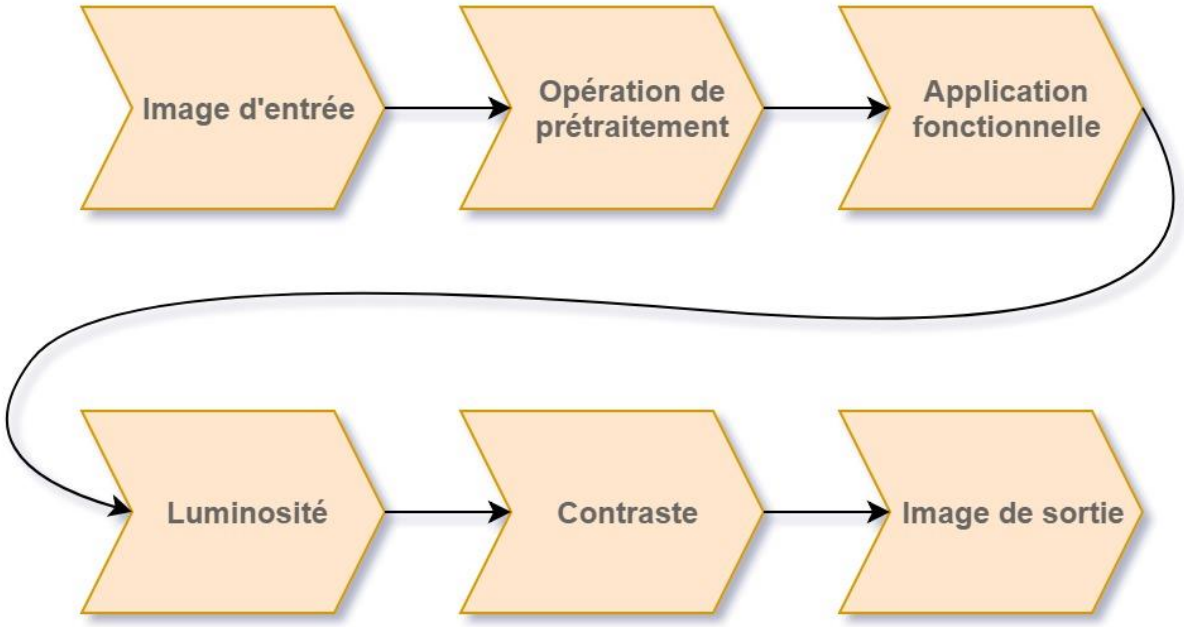
Débrumage variationnel : plusieurs modèles sont utilisés pour modifier la couleur du brouillard, rehausser le contraste et effectuer des traitements par approximation pour accélérer les temps de calculs.

Débrumage statistique : utilise des régularités observées sur des images avec des sans brouillard pour extraire une information visible et ainsi appliquer des traitements d’amélioration adapté.

Nous n’avons pu réaliser de tests sur les algorithmes aucuns d’eux n’étaient disponibles mais les méthodes étaient décrites, ce qui nous a permis de nous faire notre opinion sur les différences de rendu et la complexité des traitements.

Les différentes recherches que nous avons effectuées nous ont montré que modifier le contraste et la luminosité d’une image pourrait améliorer sa visibilité, c’est pour cela que nous avons décidé de nous pencher sur cette méthode.

Méthodologie de débrumage proposée



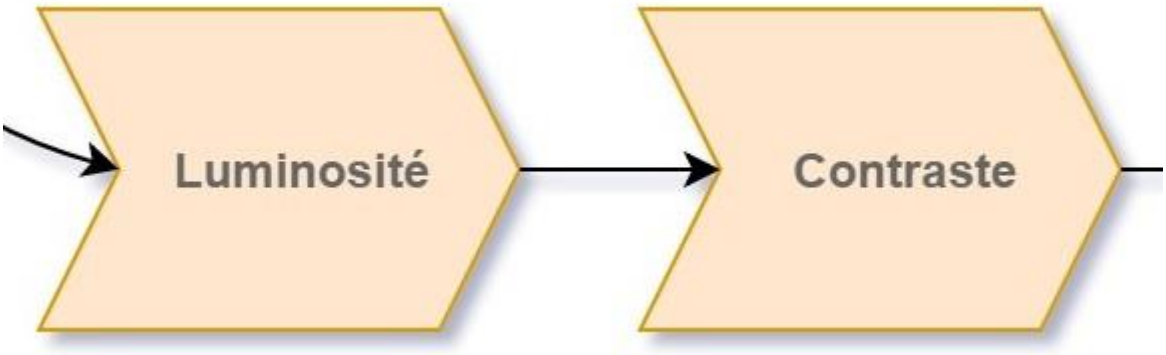
Dans un premier temps, pour pouvoir effectuer un débrumage, nous avons proposé une méthodologie. Nous allons voir en détail chaque partie de cette méthodologie.

Pour bien commencer, il nous fallait un nombre d'images avec des types de brumes différentes et des environnements différents. Nous voulions aussi avoir un point de référence avec un autre algorithme de débrumage, ici DEFADE, pour pouvoir comparer nos images débrumées avec les leurs.
Par exemple, des images de brume en ville et hors ville et des types de brumes différentes comme ci-dessous pour avoir plus de résultats et améliorer plus facilement les algorithmes (taille réelle en annexe) :

Dans notre méthodologie, il n'y a pas d'opération de prétraitement particulière à faire pour que les algorithmes marchent. En revanche, nous avons pensé au cas où un matériel pourrait être défectueux jusqu'à un certain degré. Nous avons donc simulé un bruit artificiel poivre et sel pour tester la robustesse de notre algorithme dans ce cas. Cette étape est donc optionnelle.

La plupart du travail repose ici. Comment savoir quel degré de notre algorithme doivent être appliqués et à quel endroit sur l'image ? Nous verrons cela plus en détail dans la partie Expérimentations mais nous avons choisi de

gérer ce paramètre par rapport à la hauteur d'un pixel sur l'image. Une bonne partie des brumes forme un dégradé ce qui nous a conduit à ce choix.



Nous avons décidé d'effectuer les opérations sur la luminosité avant celle du contraste car cela donnerait un meilleur résultat de visibilité par rapport à la lisibilité (inversement si on prend le contraste d'abord), plus particulièrement pour les brumes qui font des dégradés de couleurs.
Nous avons d'opter pour une méthode sur la luminosité et le contraste, car l'œil distingue mieux les différences de contraste élevé. Quant à la luminosité, elle permet de réduire l'effet visuel de la brume dans les hauteurs. Cependant, celle-ci est à double tranchant car on perd de l'information sur les couleurs plus foncées.

Pour la luminosité, nous effectuons une simple translation du vecteur de couleur, chaque composante se voit retirer le coefficient en luminosité :
Cr = Cr – CoefLum
Cv = Cv - CoefLum
Cb = Cb – CoefLum

Pour ce qui est du contraste, comme pour la luminosité, nous affectons chaque composante de la même manière, cette fois-ci basé sur le coefficient de contraste :
Cr = Cr + (CoefContr - 128) / 255 * (Cr - 127)
Cv = Cv + (CoefContr - 128) / 255 * (Cv - 127)
Cb = Cb + (CoefContr - 128) / 255 * (Cb - 127)

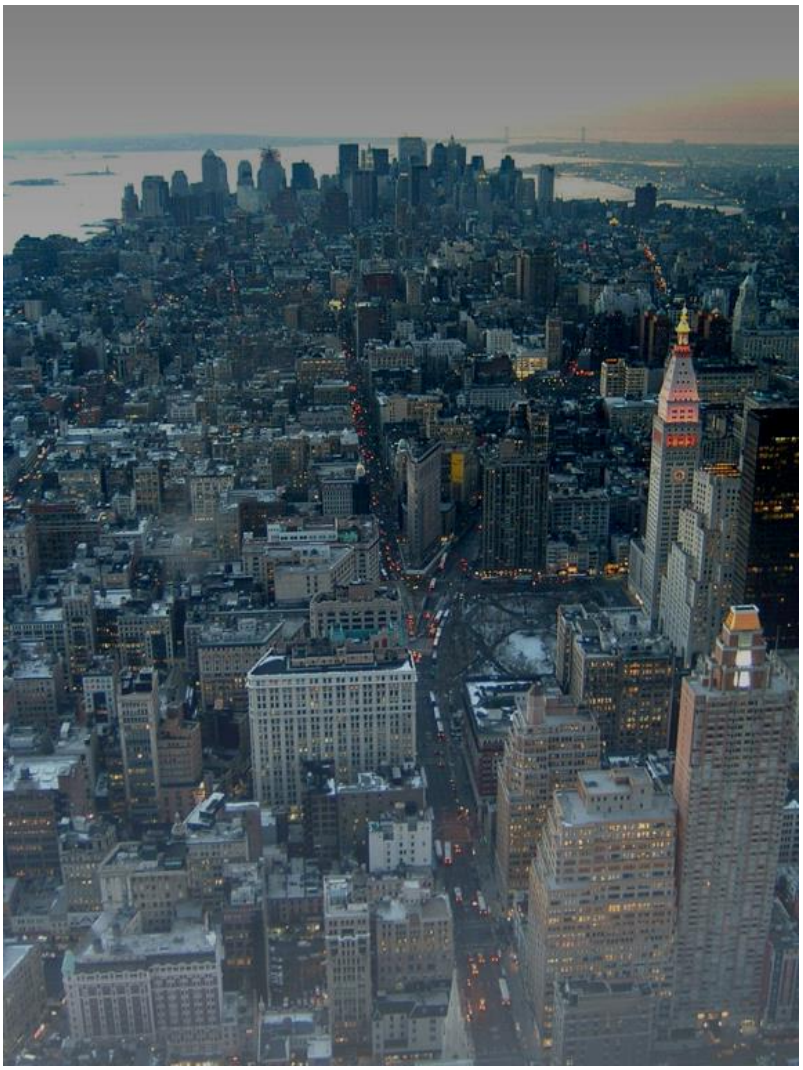
Au final, nous perdons un peu en lisibilité mais gagnons pas mal en visibilité. Par exemple, pour revenir à l'information que l'on perd, sur les deux images que l'on voit plus tôt, nous perdons l'information de la couleur sur le feuillage et un peu de lisibilité sur la ville (taille réelle en annexe) :

Expérimentations

Pour commencer, nous avons décidé de nous habituer à utiliser OpenCV. Pour cela, nous avons utilisé des tutoriels fournis par nos professeurs ainsi que trouvé sur internet. De plus, nous avions commencé par parler d' « alpha blending » et la librairie OpenCV gère ceci avec ses propres fonctions. Nous avons donc pu tester l' « alpha blending » sur une image et comprendre comment traiter les images grâce à OpenCV rapidement. (Voir images dans l'Annexe)

Nous nous sommes ensuite attaqués à la méthodologie pour le débrumage. Après avoir trouvé des images qui semblaient correspondre à notre sujet, nous avons commencé par nous occuper de l'application fonctionnelle. Comme notre but était de gérer les brumes qui ajoutent des dégradés de contrastes et de luminosité sur l'image, nous avons décidé de gérer le problème par la hauteur de l'image. Pour commencer, nous pensions mettre une fonction linéaire allant de 0 à 1. Ce coefficient est en réalité le pourcentage normé du contraste et de la luminosité que nous appliquerons après. Cela veut dire, que nous allons réduire la luminosité au maximum en haut de l'image ainsi que d'augmenter le contraste au maximum en haut de l'image.

Évidemment, nous ne pouvons pas réduire la luminosité totalement en haut de l'image, sinon elle deviendrait tout simplement noire. Nous avons donc décidé, comme solution temporaire, de diviser le facteur de luminosité par deux. Cependant, cela affectait aussi légèrement le bas de l'image et on perdait en visibilité sur le bas de l'image. Sur l'image ci-dessous, on peut remarquer que le bas de l'image est très blanc alors que le haut de l'image est sombre, ce qui permet de mieux voir au loin (taille réelle en annexe) :



Cependant, sur certaines images, comme celle ci-dessus, le dégradé posait déjà problème. Il était tout simplement trop graduel (taille réelle en annexe) :

Nous avons donc choisi de nous tourner vers des fonctions non linéaires comme l'exponentielle et le logarithme. L'intervalle de la fonction exponentielle

est de [1;1/e (~=0.37)] quant à celui de la fonction logarithmique, il était originalement de [1;ln(e - 3/4) (~=0.68)]. Le début de l'intervalle correspond au haut de l'image et la fin au bas de l'image.

La fonction exponentielle semblait donner de bien meilleurs résultats aux premiers abords, nous nous sommes donc concentrés sur celle-ci. Cependant, la fonction logarithmique donnait des images bien trop sombres, et celle de l'exponentielle était déjà assez sombre aussi. Nous avons alors estimé que nous devions nous intéresser d'abord à ce problème de luminosité pour pouvoir améliorer ensuite les fonctions.

Nous avons donc décidé de borner la réduction de luminosité à ⅓ de la réduction de luminosité maximale. Ainsi, la fonction logarithmique apparaissait bien meilleure sur certaines images et nous perdions que très peu d'informations sur la fonction exponentielle. Nous avons donc gardé ces paramètres de luminosité pour la suite.

Pour avoir une idée de la différence entre les différentes luminosités, voici des parties d'images qui montrent les différences (taille réelle en annexe) :

De gauche à droite, luminosité divisée par 2, puis par 2.5 puis par 3.

On remarque immédiatement que celle de gauche est tellement sombre que l'on ne distingue plus très facilement, voir même plus difficilement que dans l'image originale les différences de contrastes en dehors du haut de l'image. Dans la deuxième et la troisième, elles ont l'air très similaire mais on remarque que le

feuillage est plus lisible au premier plan. On voit aussi qu'il n'y a aucune perte dans le haute de l'image et que le bas de l'image est aussi plus lisible avec une luminosité plus élevée.

La fonction exponentielle marchant déjà correctement à ce stade, nous avons décidé de nous concentrer sur la fonction logarithmique pour essayer de changer ses paramètres et avoir des intervalles différents.

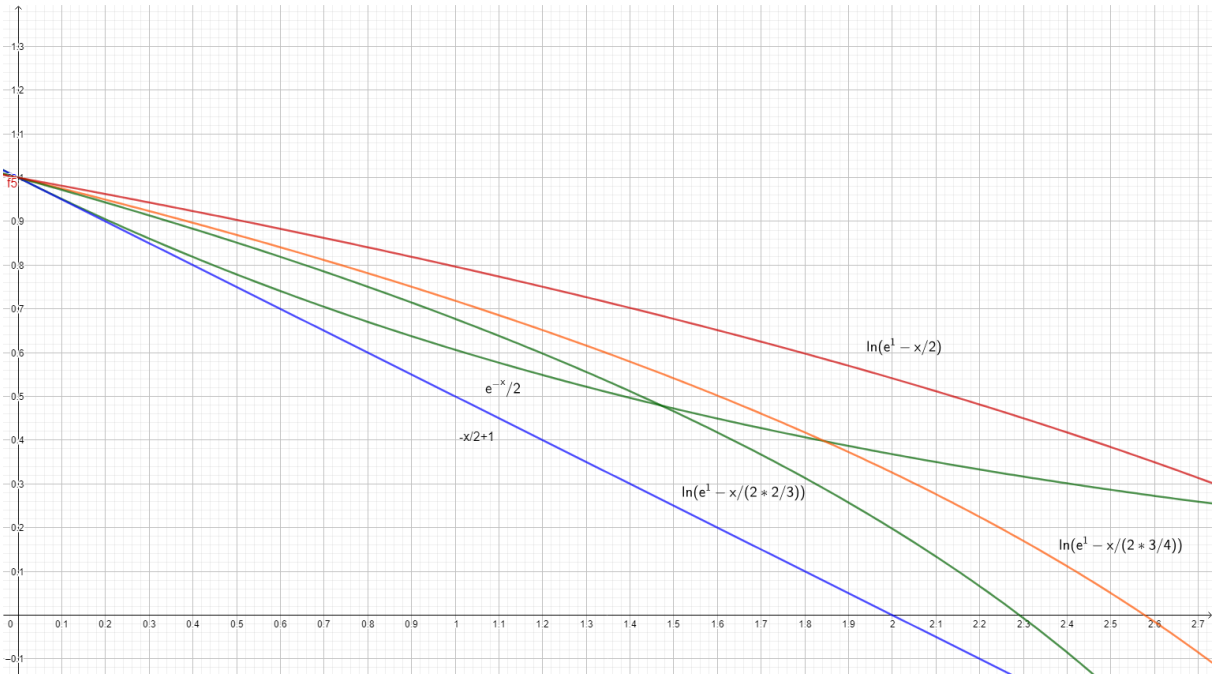
Les 3 essais avec la fonction logarithmique étaient :

- log(exp(1) - i / imgBrumee.rows)
- log(exp(1) - i / (imgBrumee.rows * 3.0 / 4.0))
- log(exp(1) - i / (imgBrumee.rows * 2.0 / 3.0))

Avec i la hauteur de l'image, et imgBrumee.rows le nombre de lignes de l'image.

Voici des exemples d'images, de gauche à droite dans le même ordre que les fonctions du dessus (taille réelle en annexe) :

Voici un schéma illustrant les différentes expérimentations de fonctions pour le degré de contraste et de luminosité :



Ces courbes correspondent à la fonction sur une image d'une hauteur de 2 pixels.

On peut voir que le logarithme que l'on finit par utiliser réduit le plus lentement et finit proche de la fonction exponentielle que nous utilisons et qui semblait bonne dès le départ. Si le degré de contraste et de luminosité est trop bas, la visibilité est mauvaise au bas de l'image, cependant elle reste bonne en haut de l'image. En fonction des images, la fonction exponentielle et la fonction logarithme vont donner de meilleurs résultats au milieu de l'image.

Enfin, pour ce qui est du prétraitement, nous avons créé un filtre qui va appliquer un bruit poivre et sel sur l'image de base. En essayant ceci, on teste la robustesse de notre algorithme sur des images de mauvaises qualités ou des caméras légèrement défectueuses.

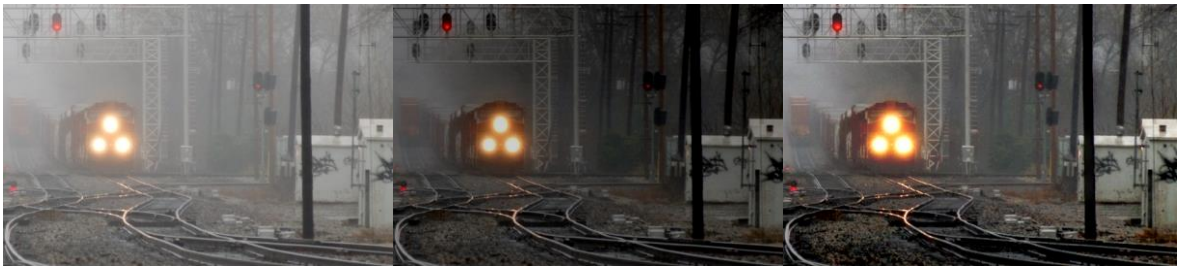
Voici quelques exemples de ce filtre :

Évaluations

Tout d'abord, nous devons faire la différence en lisibilité et visibilité. La lisibilité est la facilité de perception d'un objet, quel que soit sa distance. La visibilité est la distance à laquelle on peut percevoir le dernier objet lisible. Ces deux notions sont liées, mais dans notre cas, le but est de pouvoir voir au loin. La lisibilité sera donc quelque peu négligée mais nous n'oublierons pas de préciser ce que l'on perd.

Le premier type d'évaluation auquel nous avons procédé est l'évaluation de manières qualitatives. C'est-à-dire que nous regardons à l'œil nu si des détails sont plus visibles ou moins visibles. Cette étape a été très utile pour tester les différentes applications fonctionnelles et les modifier à vue d'œil.

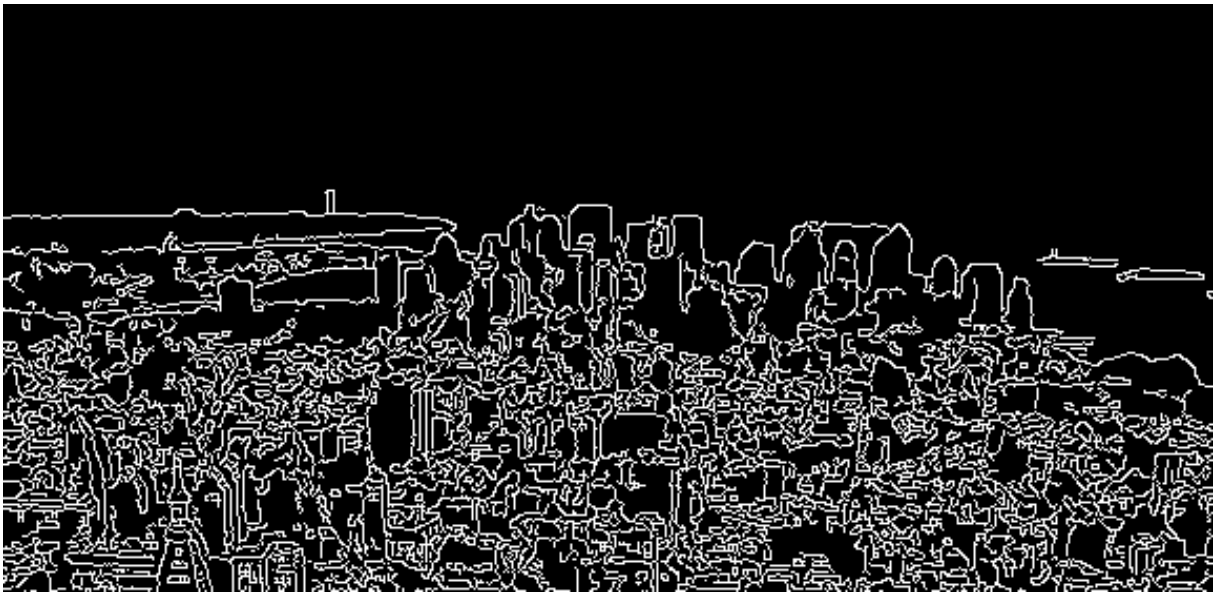
Par exemple, entre ces 3 images, on peut voir une bonne évolution, la première à gauche montre l'image originale, la deuxième montre l'image débrumée avec notre méthode via la fonction logarithmique et la troisième image montre l'image que nous avons pris pour référence pour notre débrumage, celle du logiciel DEFADE (taille réelle en annexe) :

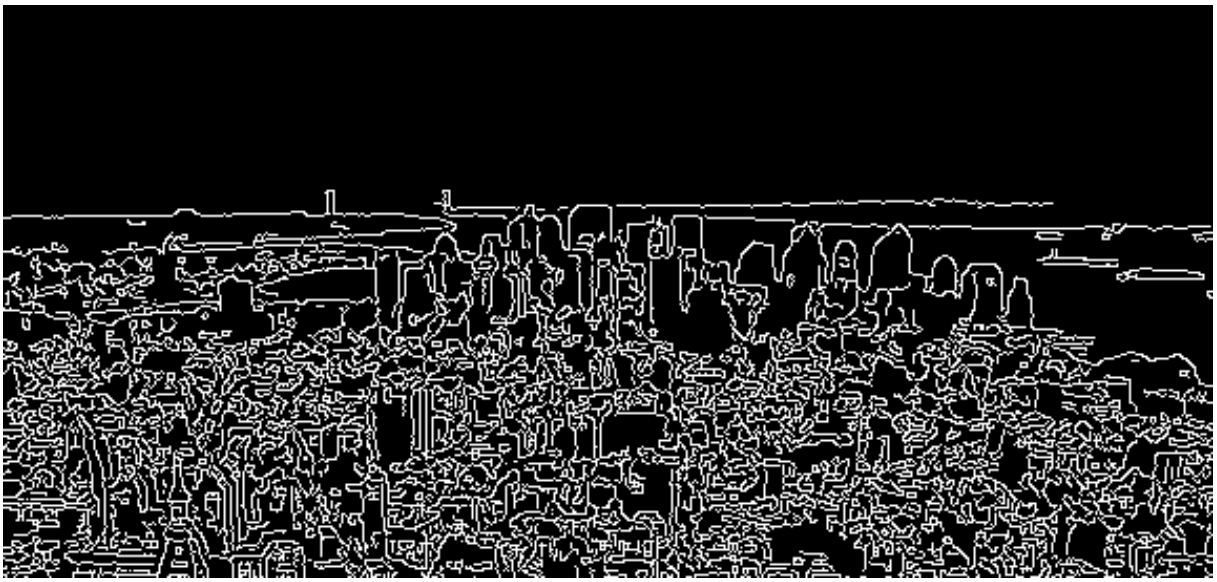


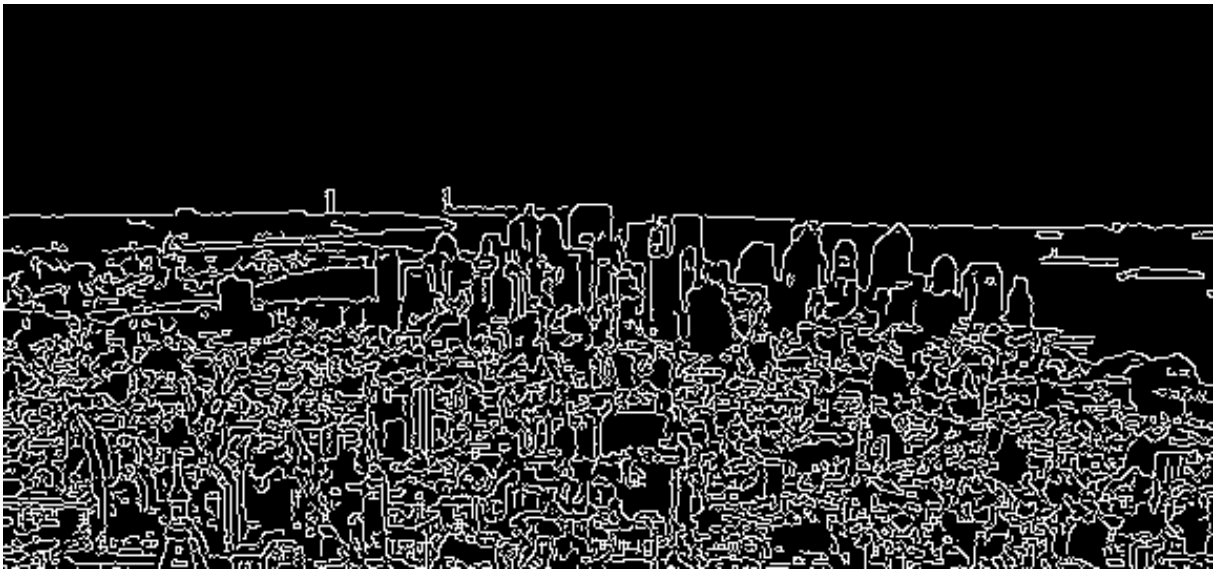
On peut voir (même avec la taille ici présente des images) que notre image est beaucoup plus visible au niveau des rails et du train que l'image originale. Cependant, on remarque aussi que tout le fond en haut à droite de l'image est difficilement discernable, on perd en lisibilité ici.

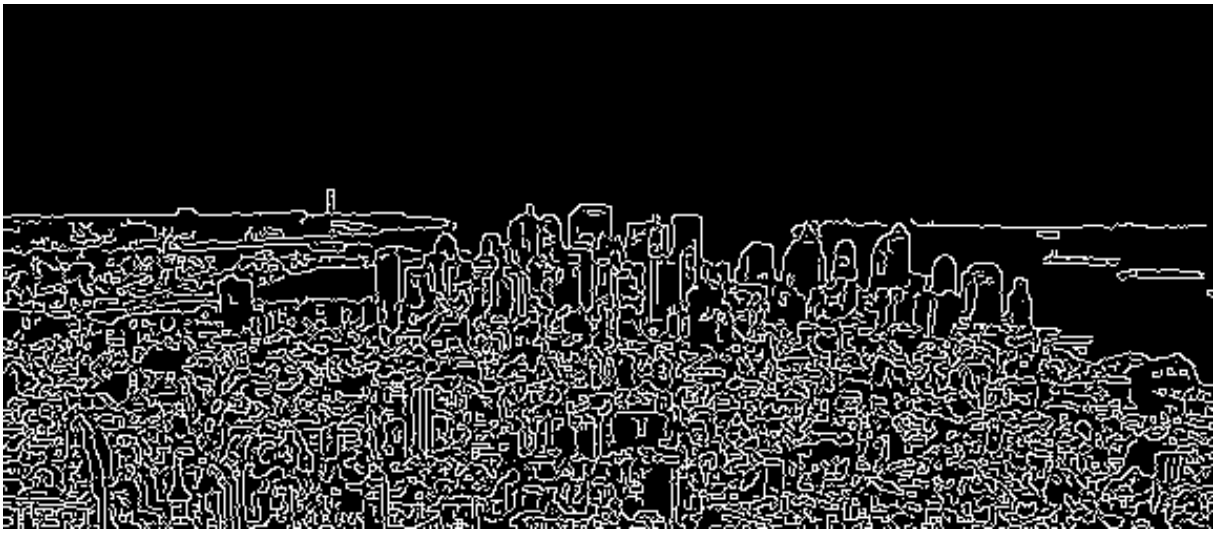
- Evaluation des fonctions avec openCV et les contours

Notre deuxième type d'évaluation des fonctions avec OpenCV est d'utiliser Canny, des fonctions d'OpenCV permettant de dessiner les contours visibles d'une image. Cela nous permet de voir des détails que nous n'avions pas vus avec la première méthode d'évaluation. Par exemple, sur ces parties d'image, on peut clairement voir les différences (taille réelle de l'image non découpée en annexe) :





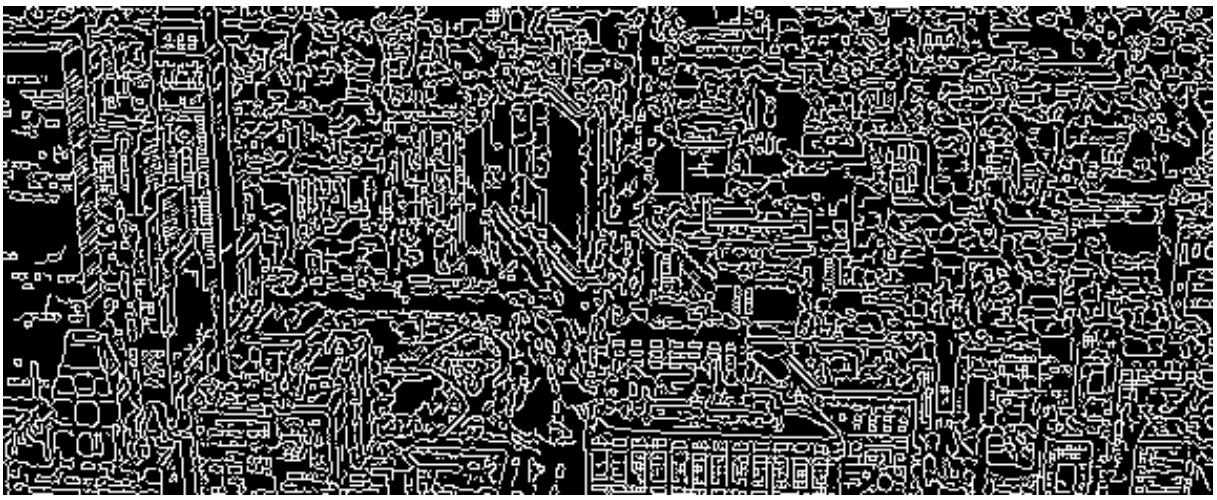


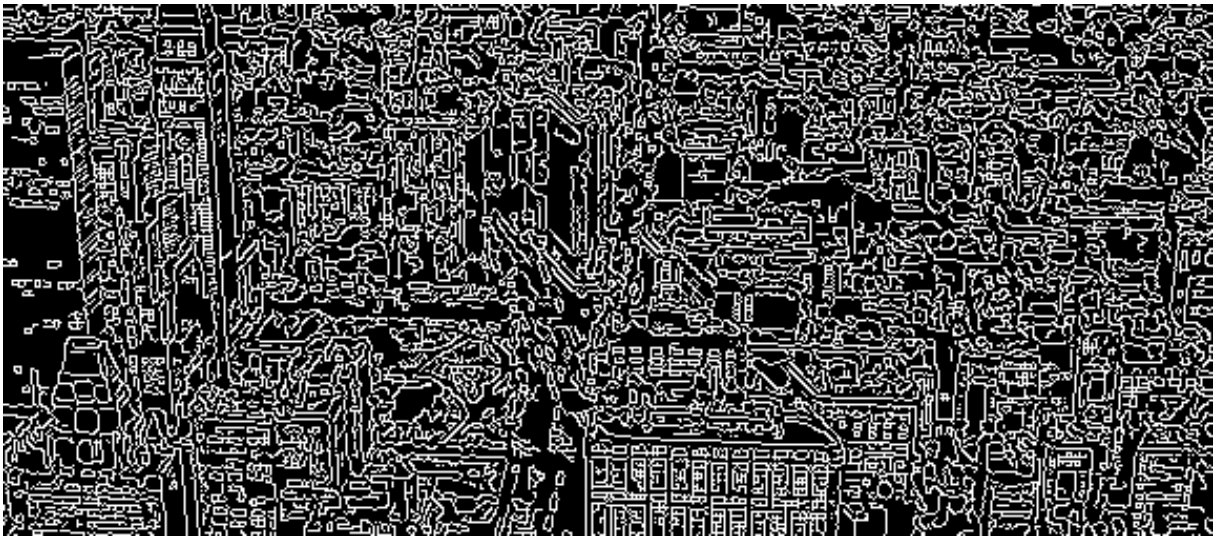


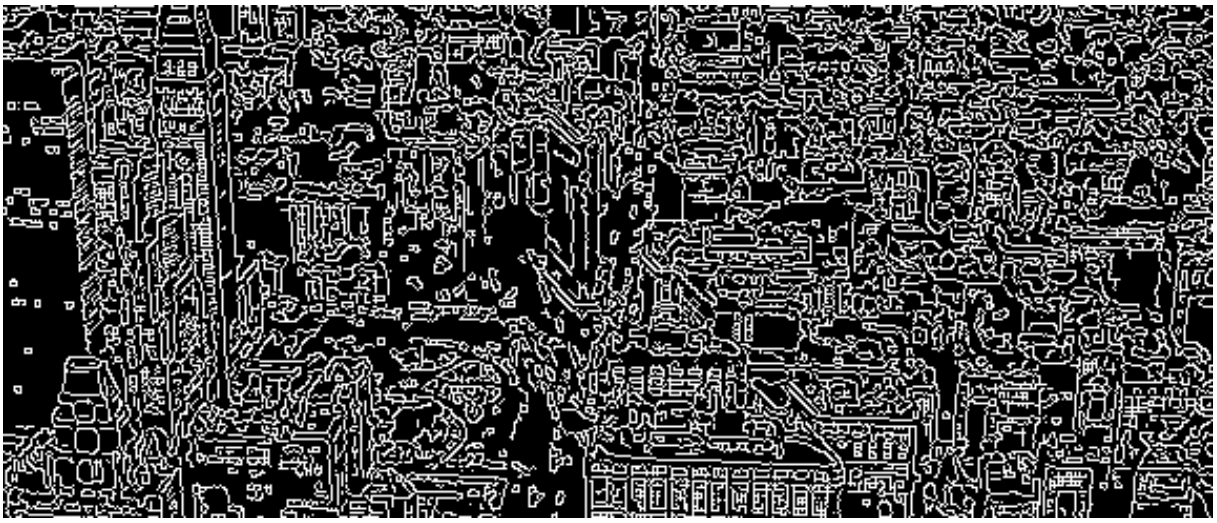
La première image est les contours de l'image originale, la deuxième, celle du débrumage de notre méthode avec l'exponentielle, la troisième, celle du débrumage de notre méthode avec le logarithme et enfin la dernière est celle du débrumage avec DEFADE.

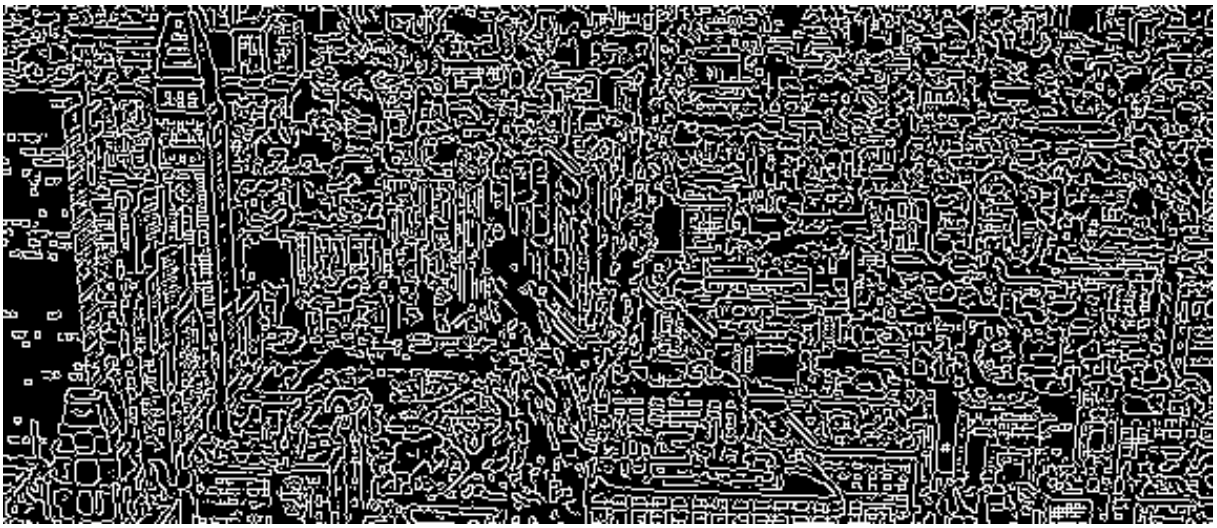
On peut clairement voir que nos deux méthodes (ici l'exponentielle l'emporte sur le logarithme) donnent de meilleurs résultats que l'image originale. Mais si on regarde l'image de DEFADE, nous avons aussi de meilleurs résultats vis-à-vis du haut de l'image, on peut apercevoir une berge qui n'est visible ni sur les contours de DEFADE ni sur les contours de l'image originale.

Cependant, comme dit plus tôt, on va perdre en lisibilité à certains endroits de l'image, par exemple, toujours dans le même ordre :









Ici, on peut voir que nos deux méthodes ont des pertes de contours, particulièrement le logarithme, alors que DEFADE possède des contours encore plus précis. Comme voir loin était notre but original, nos algorithmes sont, de part cette méthode, assez pertinent malgré la légère perte d'information aux premiers plans.

Notre troisième méthode pour évaluer les fonctions était d'essayer de créer une fonction pour évaluer les images de manières quantitatives. Nous recherchons les contours avec OpenCV et nous calculons le nombre de points blancs sur chaque image séparément, puis on normalise tous les nombres de points blancs de chaque image par le nombre de points blancs sur l'image originale.

Cela nous donne des résultats parfois bons, parfois mauvais car nous devrions ne pas faire cela sur toute l'image, mais seulement sur une partie de l'image qui nous intéresse en particulier, comme les endroits où il y a la brume et les endroits où l'on pourrait gagner en visibilité tout simplement.

Par exemple, pour les dernières images de contours, nous obtenons cela :

L'image des immeubles étant la sixième.

On peut voir que comme nous l'avons aperçu, l'exponentielle donne de meilleurs résultats que le logarithme pour nous, mais nous pouvons aussi voir que DEFADE donne de meilleurs résultats alors que le haut de l'image est bien moins net sur le débrumage par DEFADE.

Cependant, pour l'image 7, avec cette fonction, nous voyons que nous perdons en contours par rapport à l'image originale de par le ratio inférieur à 1. Même l'algorithme de DEFADE ne se classe pas très bien.

Conclusion et perspectives

Nous avons uniquement pu traiter le débrumage car le travail de recherche et d'expérimentations est très long et nous voulions proposer un résultat à la fin. Cependant, les autres problématiques que nous avons évoquées en introduction sont un sujet qui pourrait faire l'objet de recherches postérieures avec plus de temps disponible.

La technique de débrumage que nous avons appliquée est efficace et relativement simple mais malgré cela le traitement image par image en temps réel est trop lent et il faudra traiter le flux vidéo différemment. Nous avons également confronté certains problèmes comme la perte de visibilité due à l'assombrissement de l'image sur certaines zones que nous pourrions imaginer pallier en affectant mieux les zones brumées et non brumées, comme DEFADE le fait par exemple.

Nous pourrions générer ou dégénérer de la brume sur des images de la même manière que DEFADE afin d'évaluer statistiquement à quels endroits effectuer les traitements de façon optimale.

La méthode d'évaluation quantitative que nous avons mise en place est trop générique et il faudrait focaliser les calculs sur les zones moins nettes pour avoir une mesure plus précise et juste de chaque image.

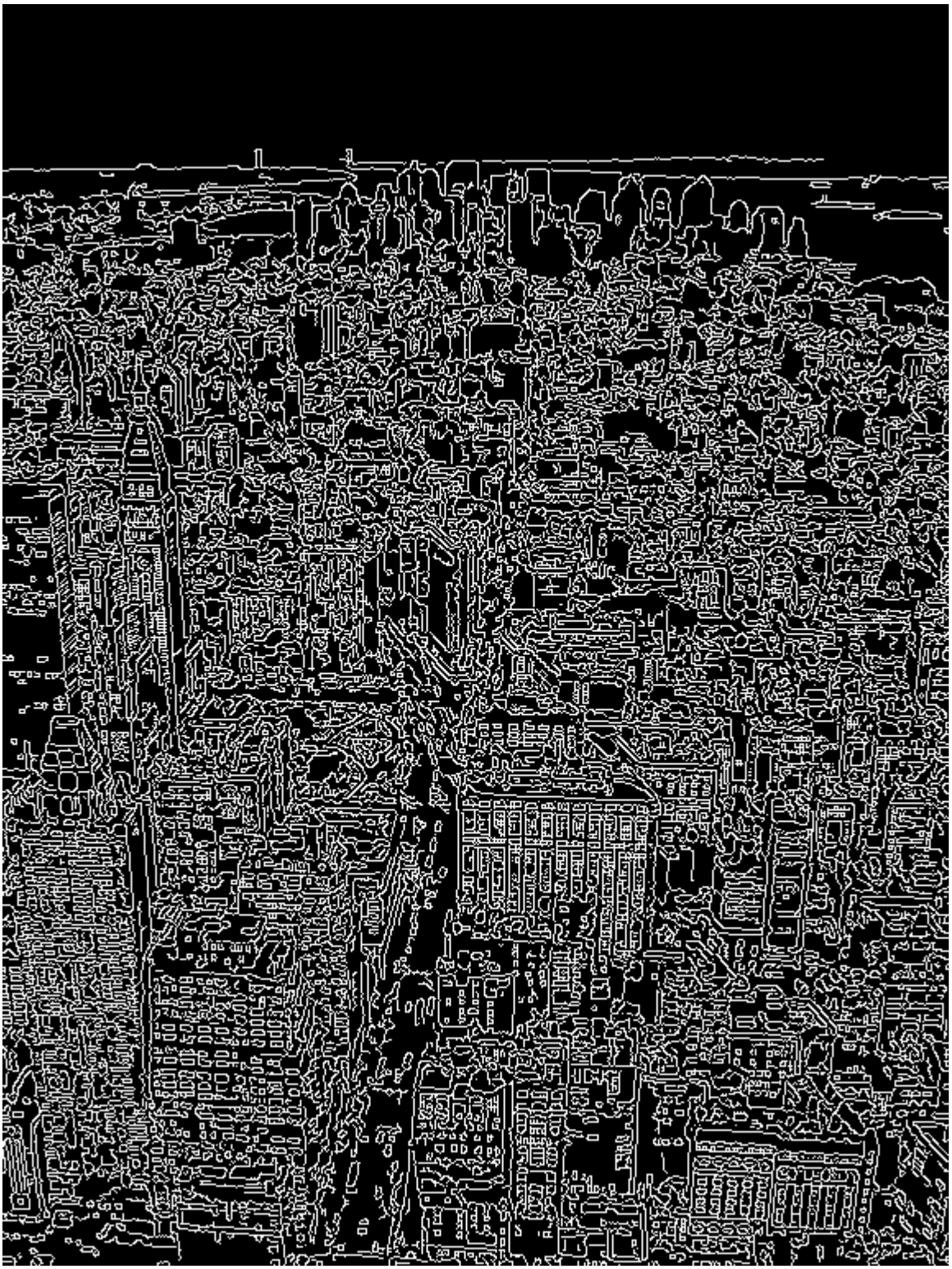
Pour finir, ce projet a été la conclusion d'une année riche en enseignements que nous avons pu mettre en pratique dans les différentes étapes de celui-ci et nous remercions nos professeurs référents pour leur aide, leur suivi et investissement qui ont permis de mener à bien nos tâches.

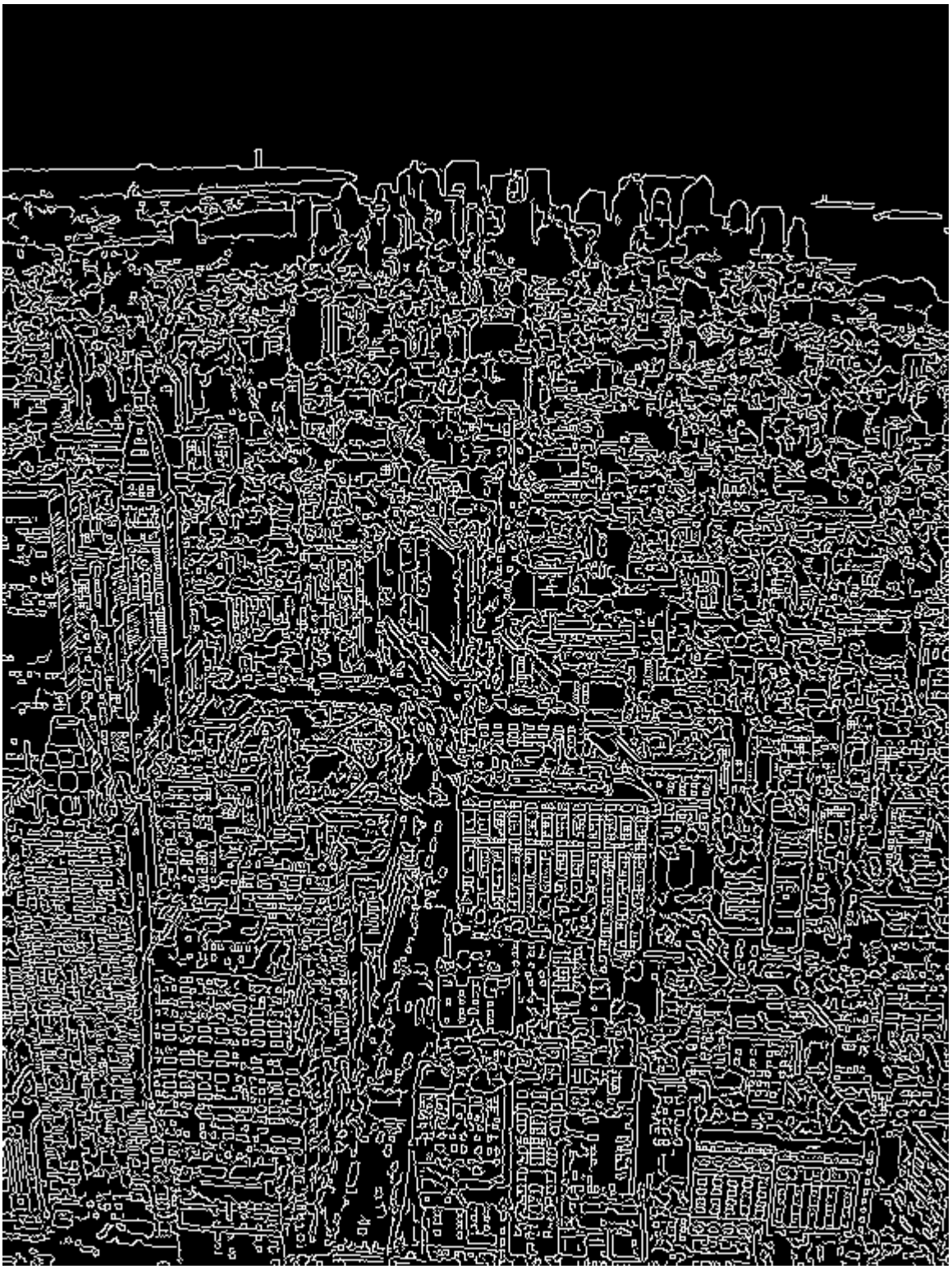
Annexes

Annexe liée à la méthodologie proposée :

Annexe des images liées à l’expérimentation :

Annexe des images liées à l’évaluation :





Bibliographie

Alpha blending :

https://www.learnopencv.com/alpha-blending-using-opencv-cpp-python/

http://www.labri.fr/perso/guenneba/pghp_2015/Cours_PGHP_2015_09-Transparence.pdf

https://www.ac-strasbourg.fr/fileadmin/pedagogie/isn/ISN/Conferences/Traiement_d_images_ISN2016_Hammoudi.pdf

Débrumage :

https://hal.archives-ouvertes.fr/hal-01168553/document

https://ieeexplore.ieee.org/document/7365412/

https://pdfs.semanticscholar.org/5898/6c4184081cb48d4008014784b44f5f386f03.pdf

http://vision.gel.ulaval.ca/~jflalonde/cours/4105/h17/tps/results/projet/111071392/index.html

https://tel.archives-ouvertes.fr/tel-00786898/document

http://live.ece.utexas.edu/research/fog/fade_defade.html

Vision œil contraste/luminosité :

http://ressources.univ-lemans.fr/AccesLibre/UM/Pedago/physique/02b/coursoptique/oeil.html

https://fr.wikipedia.org/wiki/Contraste

Source des images :

http://live.ece.utexas.edu/research/fog/fade_defade.html