Julien Arnold / Laurent Galassi

Bonjour les enfants

Université de Haute-Alsace, année 2017/2018  
Professeurs référents : Melkemi & Hammoudi

Rapport de projet

Parebrise numérique et connecté



Sommaire

[Introduction 2](#_Toc516150246)

[Motivation 3](#_Toc516150247)

[Etat de l’art 4](#_Toc516150248)

[Méthodologie proposée 5](#_Toc516150249)

[Expérimentations 6](#_Toc516150250)

[Evaluations 7](#_Toc516150251)

[Conclusion et perspectives 8](#_Toc516150252)

# Introduction

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

* Parler des différentes problématiques liées au traitement d’image ? (débrumage, aveuglement, etc…)
* Reprise de la description du sujet original
* Nouvelles technologies
* Domaine automobile
* Problématique actuelle, amélioration du quotidien
* Outils (openCV, C++, Visual Studio, etc…) et pourquoi ceux-ci

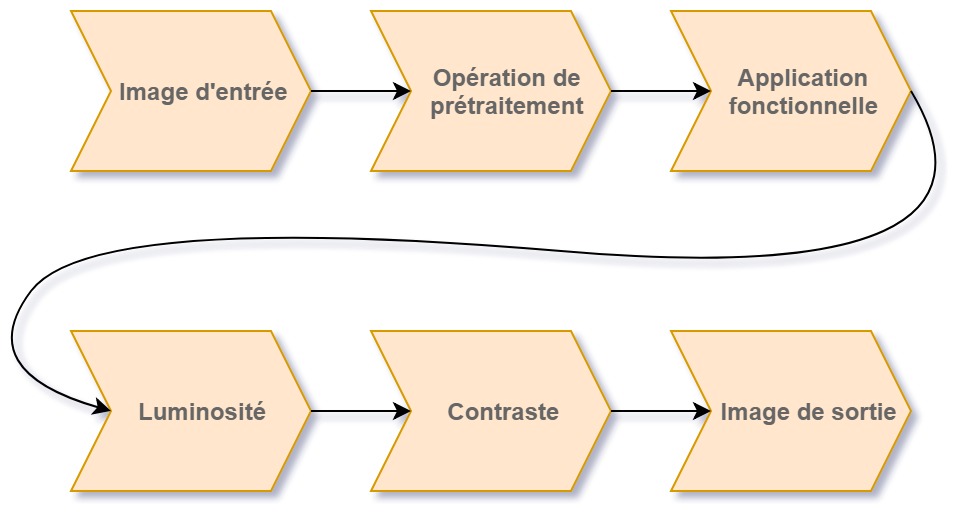
# Motivation

* Sujet de recherche (parebrise numérique augmenté)
* Différentes possibilités de travaux sur les vidéos
* Choix porté principalement sur le débrumage
* Types de brumes
* Traitement d’images / de vidéos

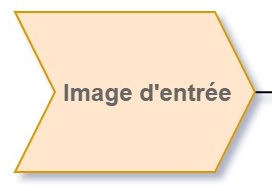
# Etat de l’art du débrumage

* Voir image tableau (méthodologie utilisée)
* DEFADE avec le reste

# 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.

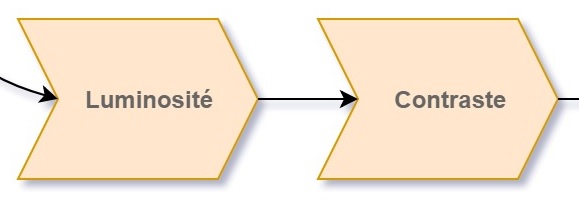
Par exemple, des images de brume en ville et hors ville et des types de brumes différentes comme ci-dessous pour avoir plus de résultats et améliorer plus facilement les algorithmes (taille réelle en annexe) :



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

* Fonctions de blending (expérimentation, prise en main openCV)
* Problème évident de luminosité, solutions temporaires
* Fonctions exponentielle et logarithme pour gérer pourcentage contraste/luminosité
* Exponentielle semble meilleure aux premiers abords
* Problème fonction logarithme sur la luminosité, changement de la solution temporaire vers une solution définitive qui semble correcte pour l’exponentielle
* Retour à d’anciens paramètres « simples » pour la fonction logarithme
* Fonctions de débrumage robustes au bruit artificiel poivre et sel (pourquoi ce bruit etc…)

# Evaluations

* Définition visibilité/lisibilité
* Evaluation des fonctions de manières qualitatives
* Evaluation des fonctions avec openCV et les contours
* Evaluation des fonctions de manières quantitatives (openCV + contour)

# Conclusion et perspectives

* Uniquement débrumage de traiter
* Possibilités d’améliorer le projet pour traiter les autres types de problèmes
* Un débrumage relativement efficace avec des algorithmes simples
* Le traitement image par image en temps réel est trop lent
* Focaliser fct netteté sur les zones moins visibles pour avoir une meilleure mesure
* Approche générer brume pour débrumer (DEFADE/FADE)
* Problèmes qu’on aurait pu gérer (introduction)
* Traitement vidéo (temps réel) problématique