



ÉCOLE NATIONALE
DES SCIENCES
GÉOGRAPHIQUES



DÉVELOPPEMENT D'UN OUTIL DE TRACKING D'OBJETS DANS DES VIDÉOS

RAPPORT DE PROGRAMMATION

COMMANDITAIRES : JEAN-FRANÇOIS VILLEFORCEIX ET NICOLAS HOLVOET
BUREAU D'ENQUÊTES ET D'ANALYSES

Hugo Tardy

Février 2021

Table des matières

1 Rappels de l'analyse du projet	2
1.1 Contexte	2
1.2 Fonctionnalités attendues	2
1.2.1 Tracking	2
1.2.2 Segmentation	4
2 Réalisation du projet	7
2.1 Architecture	7
2.2 Interface graphique	7
2.3 Tracking des petits objets	9
2.4 Masquage des gros objets	12
2.4.1 K-Moyennes	14
2.5 Données d'entrée et de sortie de l'application	16
2.6 librairies/modules utilisés	18
2.6.1 Python	18
2.6.2 OpenCV	18
2.6.3 Github	18
2.6.4 L ^A T _E X	18
3 Organisation du projet	19
3.1 Communication avec les commanditaires	19
3.2 Livrables	19
3.3 Respect du cahier des charges et perspectives	19
3.4 Respect du calendrier prévisionnel	19
4 Remerciements	21
5 Conclusion	21

1 Rappels de l'analyse du projet

Cette partie reprend de manière succincte ce qui a été présenté dans le rapport d'analyse du projet.

1.1 Contexte

Le Bureau d'Enquêtes et d'Analyse pour la sécurité de l'aviation civile est l'organisme français chargé des enquêtes de sécurité pour les accidents impliquant des aéronefs civils. Pour déterminer les circonstances précises de certains accidents le BEA utilise Micmac, un logiciel de photogrammétrie open source développé par l'IGN, pour reconstruire la trajectoire et certains paramètres de vol de l'avion à partir d'enregistrements vidéo embarqués ou au sol.

Les vidéos utilisées par le BEA dans le cadre de ses enquêtes présentent souvent des éléments inintéressants voire gênants qu'il convient d'exclure dans les calculs de MicMac en créant des masques sur les images. Ces objets sont mobiles et doivent aujourd'hui être détournés à la main ce qui est très chronophage.

Le but du projet était donc de trouver une solution permettant d'automatiser la création de masques dans des vidéos d'accidents. Ceci permettrait d'accélérer la phase de pré-traitement et donc l'enquête tout en libérant des ressources humaines pour d'autres tâches.

1.2 Fonctionnalités attendues

L'analyse du sujet avait mis en évidence la difficulté de segmenter précisément des objets dans le temps. L'outil devait aussi pouvoir servir si possible à tracker des objets. Sa réalisation s'était donc divisée en deux parties principales.

1.2.1 Tracking

La première partie était le tracking des petits objets. En effet dans le cas d'objets de petite taille, un masque grossier entraîne peu de pertes d'informations sur l'image. La forme du masque n'étant pas déterminante dans ce cas de figure, il suffit de connaître la position de l'objet, sa taille approximative et de suivre sa position.



FIGURE 1: Exemples d'objets gênant
Un masque rectangulaire autour de l'objet cache très peu d'informations au sol

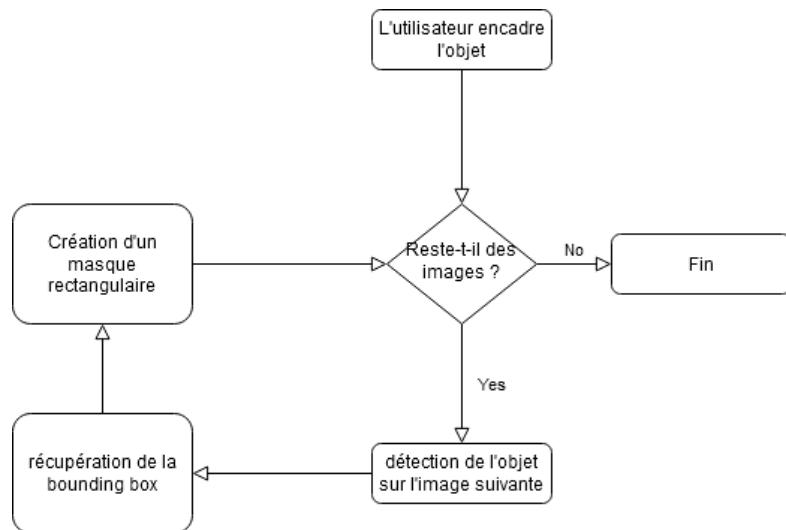


FIGURE 2: Schéma de traitement des petits objets

Pour suivre un objet, plusieurs algorithmes de tracking existent. La bibliothèque OpenCV en intègre 8 à ce jour. Ces algorithmes se différencient par des critères tels que la vitesse d'exécution, la précision, et la robustesse :

- Boosting Tracker
- MIL Tracker
- KCF Tracker
- CSRT Tracker
- MedianFlow Tracker
- TLD Tracker
- MOSSE Tracker
- GOTURN Tracker

1.2.2 Segmentation

Dans le cas d'objets plus gros, on ne peut pas se permettre de réaliser un masque rectangulaire sans perdre trop d'informations comme le montrent les figures 3 et 4.



FIGURE 3: En réalisant un masque rectangulaire on perd toute l'information située dans la zone hachurée



FIGURE 4: Un masque rectangulaire n'est pas non plus adapté dans le cas où on souhaite masquer les éléments du cockpit qui forment le pourtour de l'image

L'idée est donc d'utiliser une segmentation pour construire le masque. Pour déterminer les régions qui y appartiennent, on se base sur le masque de l'image précédente lui-même récursivement basé sur le masque précédent, le premier étant le masque défini par l'utilisateur (fig. 5).

En utilisant un algorithme de détection et de description des points clés du masque, on peut établir la position de l'objet dans l'image suivante. Une segmentation adaptée réalisée en parallèle sur l'image suivante nous permet de dire si chaque région appartient au masque ou non en utilisant les descripteurs.

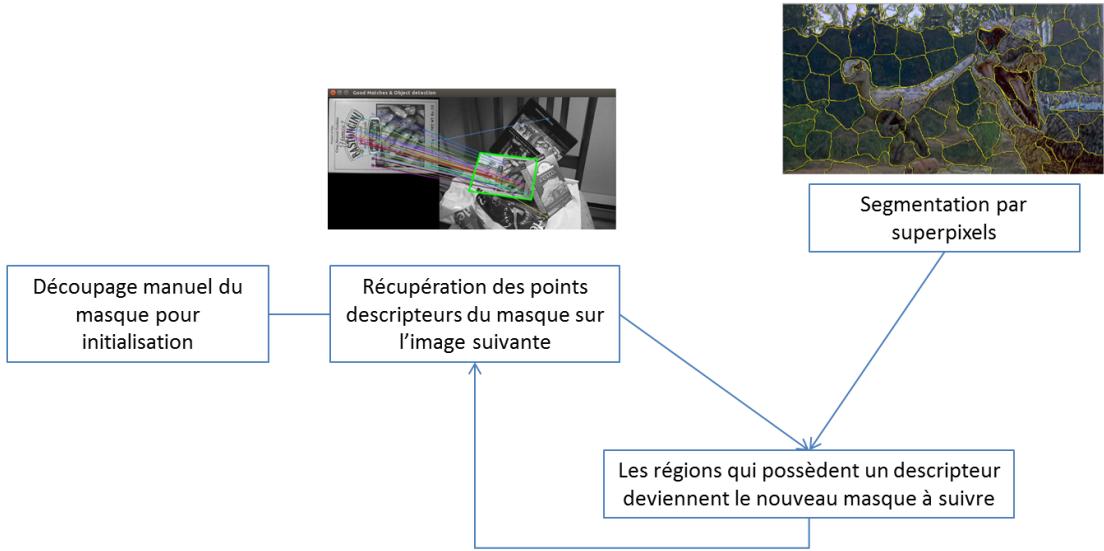


FIGURE 5: Schéma de traitement des gros objets

Le choix de la segmentation s'est porté sur la méthode des superpixels qui a l'avantage de sursegmenter. Ceci permet de minimiser les régions ambiguës qui comporteraient des éléments d'un objet à masquer et du reste de l'image non masquée. Ces objets gênants massifs seront probablement la principale difficulté du projet.

2 Réalisation du projet

J'aborderais dans cette partie l'implémentation qui a été faite des fonctionnalités et les changements qui ont pu arriver par rapport à ce qui a été prévu.

2.1 Architecture

L'application est construite autour de deux classes principales : la classe Vidéo et la classe Objet. La première contient les informations sur les caractéristiques de la video (nombre de frames, chemin, nom, taille des frames) et la liste des objets que l'utilisateur aura défini. Chaque objet est lié à une vidéo et possède la liste des masques à appliquer sur chaque frame.

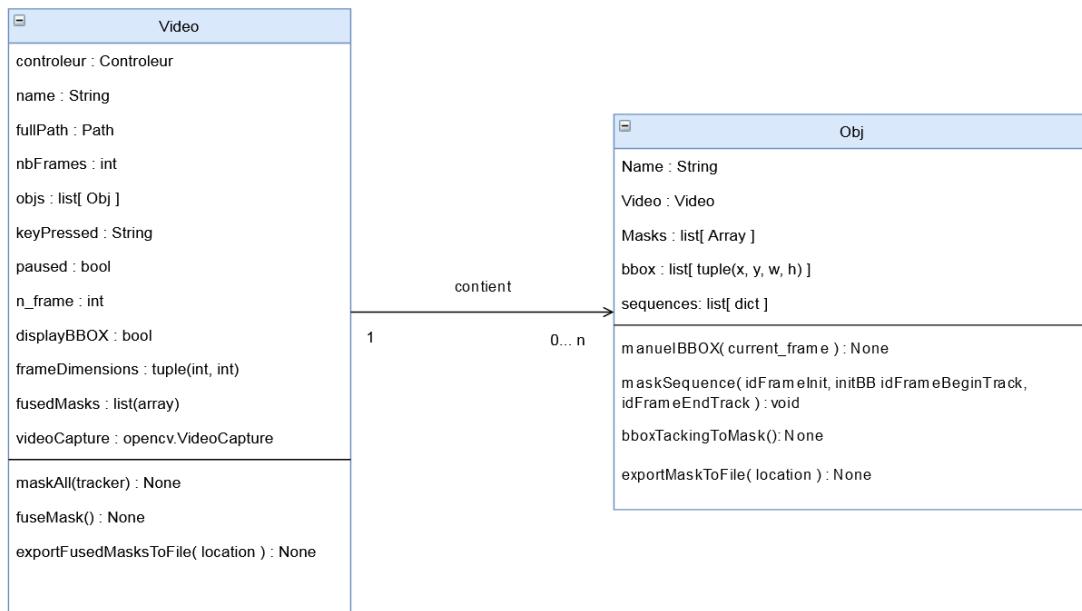


FIGURE 6: Une partie du diagramme de classe du projet

2.2 Interface graphique

Une des difficultés survenues assez tôt dans le développement était de rendre l'application simple à prendre en main. De par sa nature (traitement des vidéos) l'outil devait être visuel et l'utilisateur devait pouvoir sélectionner les séquences où démarrer le tracking et où l'arrêter. Un outil sous forme de ligne de commande aurait forcé l'utilisateur à connaître exactement les fonctions à utiliser, à ouvrir sa vidéo dans une fenêtre externe, trouver les frames exactes de début et de fin de segmentation puis visualiser avec difficulté les résultats.

J'ai donc décidé de mettre en place une interface graphique qui répond à ces problématiques :

Rapport d'analyse

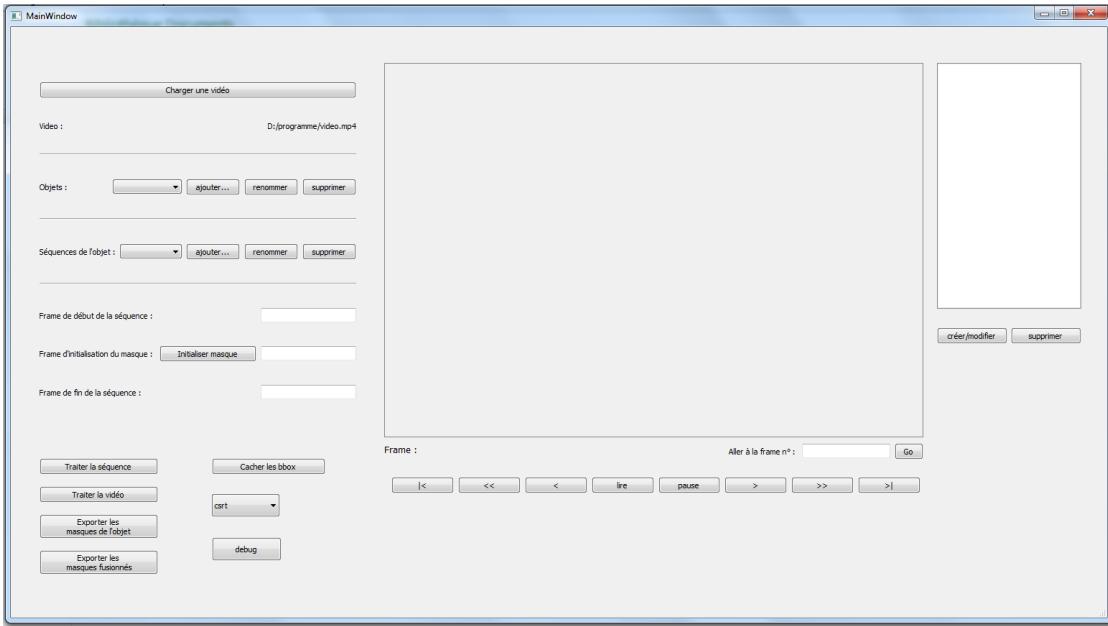


FIGURE 7: Page d'accueil de l'application

J'ai intégré l'interface dans l'application en utilisant le motif d'architecture **Modèle - Vue - Controleur** (voir fig. 8). Cette architecture sépare ce qui tient de l'affichage (la Vue) des données et traitement (Modèle) ici représenté par l'objet Vidéo. Les deux sont liés par le contrôleur chargé de recevoir les données et signaux de chacun et d'agir en conséquence.

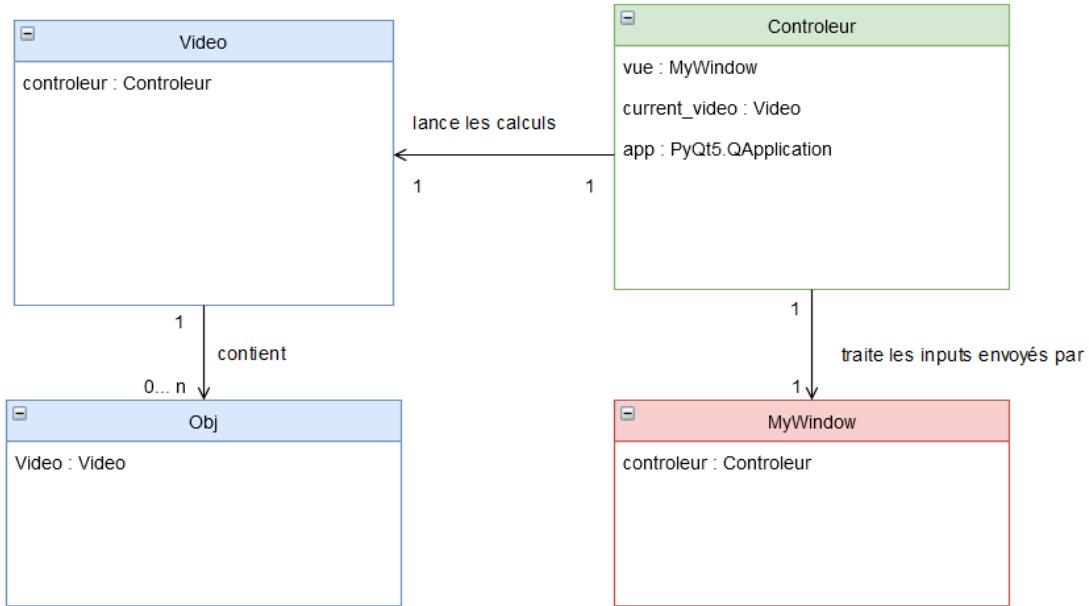


FIGURE 8: Diagramme de classe simplifié et architecture Modèle-Vue-Controleur de l'application

2.3 Tracking des petits objets

Le suivi des petits objets a été entièrement intégré à l'aide de la bibliothèque OpenCV qui propose plusieurs fonctions de tracking.

L'utilisateur fournit 3 informations à l'application :

- l'index de la frame où le tracking doit débuter
- l'index de la frame où le tracking doit s'arrêter
- l'index de la frame où l'utilisateur initialise une bounding box autour de l'objet ainsi que la bounding box en question

L'application ouvre alors une nouvelle fenêtre et montre en temps réel l'avancement du tracking avec notamment un progrès en pourcentage, le nombre d'images traitées par seconde et la frame actuelle (fig 9).



FIGURE 9: Résultat du tracking en temps réel

L'utilisateur peut alors visualiser les résultats sur la vidéo dans l'interface graphique. S'il n'est pas satisfait, il peut décider de recommencer tout ou partie du tracking.

L'algorithme le plus prometteur de suivi d'objet parmi ceux proposés dans OpenCV est CSRT en termes de robustesse et de précision du suivi. Il est en revanche parfois mis en échec lors des occlusions et dérive alors sur d'autres objets. De plus les vidéos avec une fréquence d'image réduite semblent compliquer la tâche de l'outil.

Les résultats sont cependant bons voire très bons de manière générale et en dehors des conditions extrêmes évoquées ci-dessus, le tracking répond bien à la problématique du BEA de masquer des objets ou de suivre des cibles spécifiques. La figure 10 suivante montre les résultats de 2 tracking et illustre la possibilité de gérer le multi masques.

Rapport d'analyse

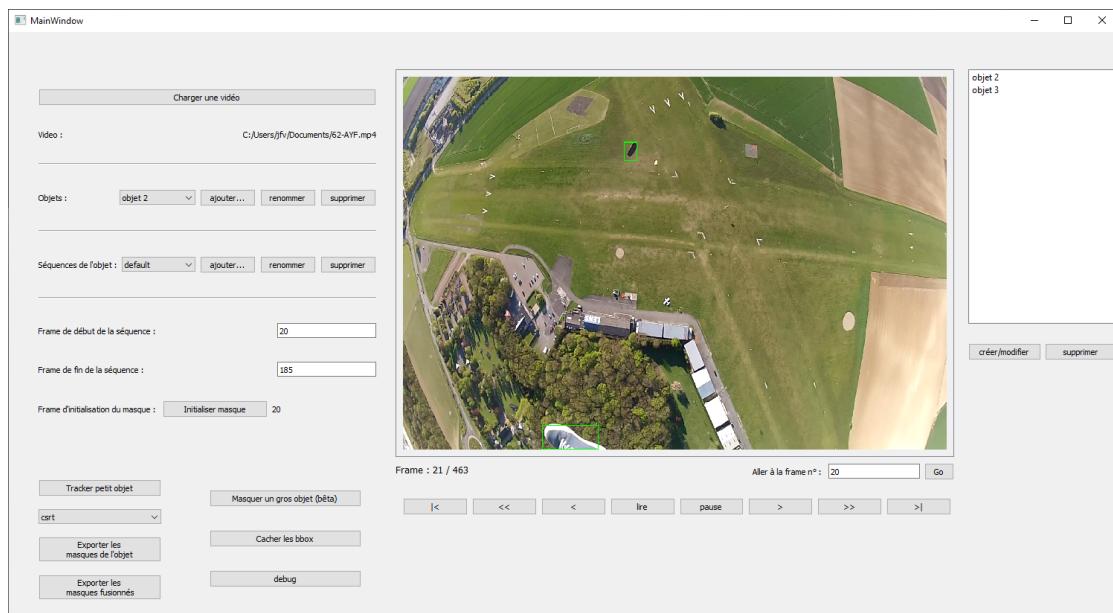


FIGURE 10: Résultat du tracking de 2 objets différents



FIGURE 11: Résultat du tracking d'une cible au sol

2.4 Masquage des gros objets

En cohérence avec ce qui avait été prévu dans l'analyse, masquer les gros objets s'est fait à l'aide d'une segmentation par superpixel associée à des points d'intérêt.

De la même manière que pour les petits objets, l'utilisateur choisit où le tracking doit commencer, terminer et doit dessiner un masque sur l'objet qui l'intéresse.



FIGURE 12: points d'intérêt

Cependant cette méthode s'est montré insuffisante, il n'est en effet pas possible de garantir que toutes les zones auront des points d'intérêt permettant de leur attribuer la classe "masque" ou "non masque".

Il a donc été nécessaire d'utiliser une méthode pour assigner une région indécise à l'une de ces deux catégories. La première idée a donc été d'utiliser une métrique de distance entre les régions. Pour commencer, il fallait donc déterminer des régions de référence appartenant au masque ou non.

Ceci s'est fait par l'utilisation d'une érosion et d'une dilatation pour détourer grossièrement le masque sur l'image suivante. Si on prend l'hypothèse que d'une frame à l'autre d'une vidéo l'objet ne bouge pas beaucoup, on peut déterminer deux zones où nous sommes sûrs que les pixels appartiennent ou non au masque sur l'image suivante. Sur la figure 15 la zone verte correspond à l'érosion, donc à la zone où nous sommes sûrs que les pixels appartiennent au masque. A l'inverse tous les pixels en dehors de la dilatation (rouge) n'y appartiennent pas (zone noire).



FIGURE 13: Superposition du masque utilisateur (blanc), de l'érosion (vert) et de la dilatation (rouge)

La définition de la métrique s'est d'abord basée sur une distance colorimétrique entre chaque région indécise et les zones masquées/non masquées.

$$d = \sqrt{(R_{reg} - R)^2 + (V_{reg} - V)^2 + (B_{reg} - B)^2}$$

- R_{reg} , V_{reg} , B_{reg} les moyennes colorimétriques de la région à classifier
- R , V , B les moyennes colorimétriques du masque ou du non masque selon la classe avec laquelle on calcule la distance

Les résultats se sont montrés inutilisables, les classifications étant erronées pour la plupart des régions.

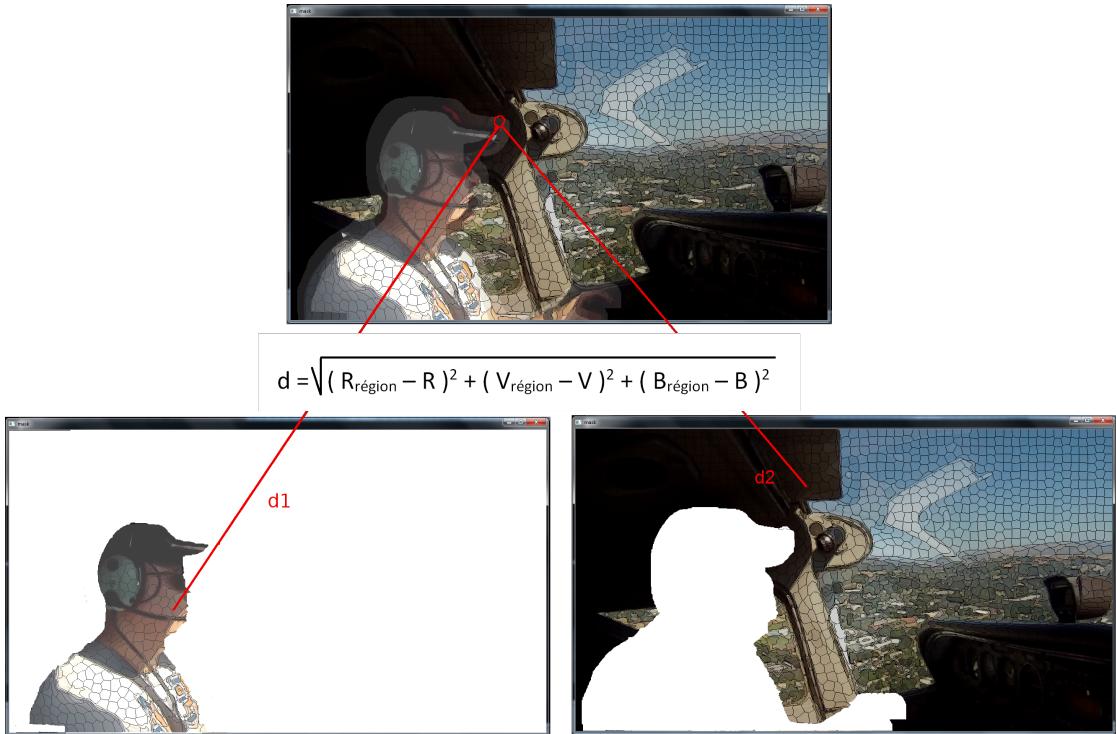


FIGURE 14: Calcul de distance colorimétrique entre une région indécise (en haut) et les deux classes "masque" (à gauche) et "non masque" (à droite)

2.4.1 K-Moyennes

Afin d'améliorer la classification des zones indécises et après des échanges avec Arnaud le Bris, intervenant à l'ENSG, j'ai mis en place une classification par K-moyennes :

1. classification par K-moyennes sur la partie masque
2. classification par K-moyennes sur la partie non-masque
3. Assigner chaque région indécise à la classe la plus proche

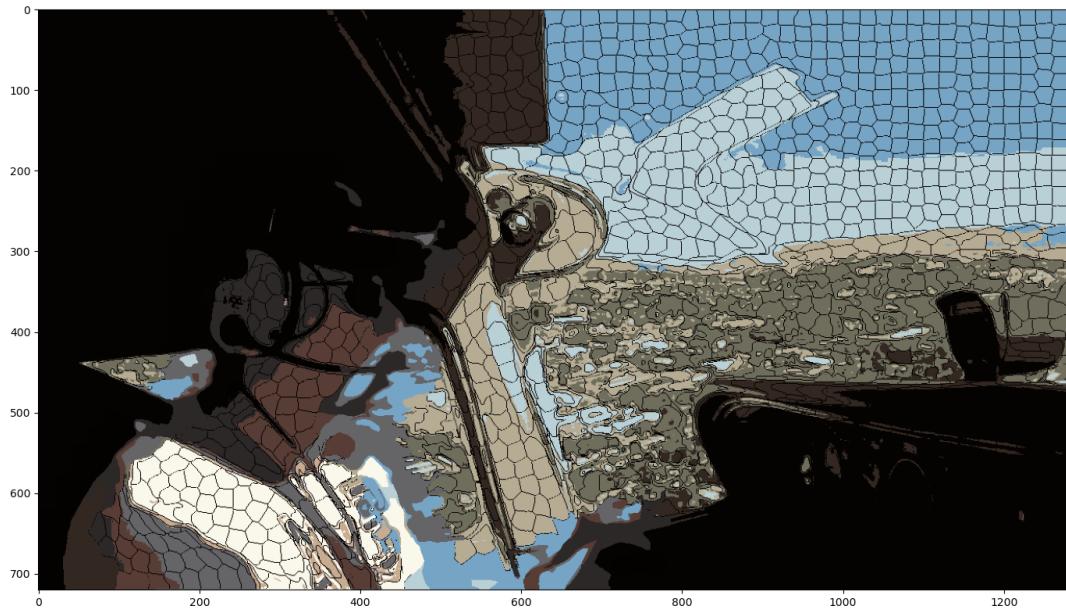


FIGURE 15: Résultats des K-moyennes après classifications
Chaque pixel a la couleur du centroïde de sa classe

Il existe encore plusieurs fausses associations cependant d'autres essais sur des données du BEA montrent des résultats encourageants (fig 16 et 17)



FIGURE 16: Résultat d'un masquage automatique sur la tête du pilote



FIGURE 17: Résultat d'un masquage automatique sur la tête du pilote

Les résultats sont donc à confirmer, la fonctionnalité n'étant pas encore implémenté entièrement. De plus il reste à tester l'impact de la taille des régions pour la segmentation et du nombre de classes utilisés pour l'algorithme des K-moyennes qui pourraient avoir un impact sur le résultat.

Une autre amélioration possible serait de ne pas considérer toute l'image pour l'algorithme. Le ciel par exemple sur la figure 15 induit plusieurs fausses classifications qui pourraient être améliorées en découplant l'image autour de l'objet d'intérêt (ici le pilote).

2.5 Données d'entrée et de sortie de l'application

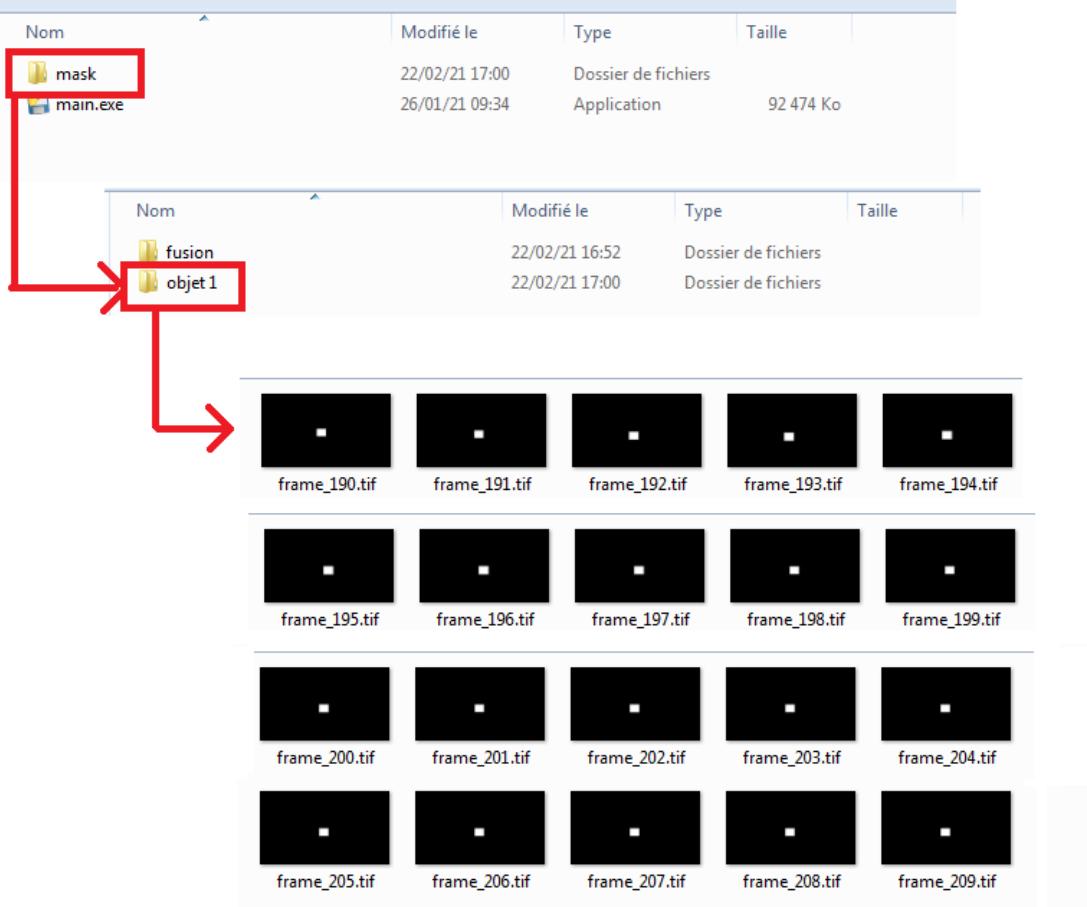
L'application permet en entrée tous les formats de vidéo acceptés par OpenCV.

Lorsque l'utilisateur est satisfait des masques créés, il peut les sauvegarder soit individuellement par le bouton "exporter les masques de l'objet", soit exporter la fusion des masques. L'application crée un dossier "mask" qui contient les masques de chaque frame au format .tiff et les fichiers .xml correspondant pour Micmac.

Rapport d'analyse

Nom	Modifié le	Type	Taille
mask	22/02/21 17:00	Dossier de fichiers	
main.exe	26/01/21 09:34	Application	92 474 Ko

Nom	Modifié le	Type	Taille
fusion	22/02/21 16:52	Dossier de fichiers	
objet1	22/02/21 17:00	Dossier de fichiers	



 frame_190.tif	 frame_191.tif	 frame_192.tif	 frame_193.tif	 frame_194.tif
 frame_195.tif	 frame_196.tif	 frame_197.tif	 frame_198.tif	 frame_199.tif
 frame_200.tif	 frame_201.tif	 frame_202.tif	 frame_203.tif	 frame_204.tif
 frame_205.tif	 frame_206.tif	 frame_207.tif	 frame_208.tif	 frame_209.tif

FIGURE 18: Arborescence des masques en sortie

2.6 librairies/modules utilisés

2.6.1 Python

Le projet a été réalisé avec le langage Python en version 3.6 pour assurer une compatibilité avec OpenCV.

2.6.2 OpenCV

OpenCV est une librairie python spécialisée dans la vision par ordinateur. Certains algorithmes tels que SURF utilisés pour la récupération de points descripteurs sont sous licence dans les versions au delà de la 3.4.2. La bibliothèque OpenCV a donc été utilisée en version 3.4.2.

2.6.3 Github

Le suivi du projet s'est fait sur un Github partagé aux commanditaires dans le but d'assurer un suivi du projet.

2.6.4 L^AT_EX

Les rapports et comptes rendus du projet sont rédigés avec L^AT_EX qui est un langage et un système de composition de documents. L'éditeur utilisé est Overleaf.

3 Organisation du projet

3.1 Communication avec les commanditaires

Les contacts avec les commanditaires se sont fait par mail en général et par téléphone ou visio conférence lors des points d'avancement qui se faisaient toutes les 5 séances environ. Une visite au Bureau d'Enquêtes et d'Analyse a également été réalisée dans le cadre du projet afin d'échanger sur son avancement et de découvrir en quoi celui-ci s'inscrit dans les travaux du BEA.

Le partage de l'application s'est fait par un accès libre des commanditaires au compte github hébergeant l'application et par l'utilisation d'exécutables envoyés aux commanditaires pour les phases de test et de retour.

3.2 Livrables

Les livrables du projet qui ont été remis aux commanditaires sont

- Une application fonctionnelle
- Une documentation utilisateur
- Une documentation développeur sous la forme d'un diagramme de classe et d'un code commenté pour faciliter la prise en main du code

3.3 Respect du cahier des charges et perspectives

L'application a bien rempli le premier objectif du projet qui était de pouvoir masquer les petits objets. De par son intégration, l'application permet aussi de répondre à un besoin de suivi d'objets pour lesquels les commanditaires se sont montrés intéressés tels que des cibles terrains sur des vidéos de drones. L'application pourrait éventuellement être mise en relation avec le travail réalisé par d'autres élèves pour le BEA tel que la lecture automatique de paramètres de vol sur des cadrans.

Le second objectif de masquage des objets de grande taille a été partiellement atteint. Le code pose les bases de cette fonctionnalité mais nécessitera du temps supplémentaire pour être intégré de manière fluide à l'application.

Les perspectives d'améliorations sont donc principalement sur le masquage des objets de grande taille. Une amélioration possible pour le tracking des petits objets concerne la résistance à l'occlusion lors desquelles CRST n'échoue jamais mais dérive sur d'autres objets. L'implémentation OpenCV ne permet pas de régler de paramètres sur le tracker dans l'implémentation Python, une option serait donc de recompiler OpenCV en C++ avec une fonction modifiée, ou de mesurer la similarité dans la bounding box pour détecter une occlusion et tenter de retrouver l'objet plus loin dans la vidéo par corrélation.

3.4 Respect du calendrier prévisionnel

Le planning prévisionnel a globalement été respecté. Certains points ont demandé un temps conséquent comme le développement de l'interface graphique mais son développement n'a pas empiété sur les séances assignées au projet. Le masquage des gros objets n'a pas été totalement finalisé au moment du rendu du rapport en raison de la difficulté à l'implémenter et à estimer le temps nécessaire pour obtenir un résultat acceptable.

Rapport d'analyse

Tâches	Séances																						
	1	2	3h	3h	3h	4h	3h	3h	3	4	5	6	7	8	9	1h30	10	11	12	13	14	1h30	6h
Implémentation données entrées choix utilisateurs : initialisation masque, frame...																							
Tracking petits objets																							
Visualisation pratique du résultat																							
Comprendre Qt Designer et imaginer comment intégrer l'interface graphique																							
Mettre en place l'interface graphique																							
Débugguer sur ubuntu																							
Créer un auto executable ubuntu																							
Fusion et création des masques compatibles avec Micmac																							
Correction bugs et améliorations																							
Menu pour modifier/supprimer masques																							
Point d'avancement et légères améliorations																							
Diagramme classe + documentation code/docstring																							
Visite BEA																							
Tracking gros objets																							
Mis en place des K-moyennes																							
Rédaction du rapport																							
Préparation soutenance																							

FIGURE 19: planning du projet

4 Remerciements

J'ai pris beaucoup de plaisir à réaliser ce projet qui m'a permis de me confronter à la réalisation d'une interface graphique avec PyQt et d'utiliser OpenCV et mes cours de traitements d'images. J'ai pu ainsi répondre à une problématique qui ne présentait pas de solution simple à partir d'un socle de connaissance qui m'étaient familières. Je souhaite remercier mes commanditaires pour leur accueil au Bureau d'Enquête et d'Analyse et leur intérêt visible pour mon travail et Jean-François Villeforceix pour son encadrement.

5 Conclusion

Dans l'ensemble, l'application a répondu aux attentes des commanditaires et devrait permettre de répondre au moins partiellement à leurs problématiques de suivi et de masquage d'objet. La soutenance de ce rapport aura lieu à l'Ecole Nationale des Sciences géographiques le mardi 2 mars entre 11h35 et 12h05.