# FAQ-4 du manuel d’extraction des trajectoires / Relier les objets entre eux (CombineBlobs.jar)

Logiciel : CombineBlobs.jar – version du 23 septembre 2013

Manuels à disposition : Combine Blobs Manuel

***remarque concernant les applications java***

Il est nécessaire d’avoir une machine virtuelle JAVA installée sur l’ordinateur

Si besoin la télécharger à partir du site java et choisir la configuration de l’OS (pour ordis de l’ENTPE – 64 bits : jre-8u91-windows-x64)

*Attention, il ne faut pas télécharger l’application pour les navigateurs mais celle de la MV.*

Deux possibilité pour faire tourner les applications Java

1/ exécuter le fichier .bat

Ce fichier doit contenir la ligne de commande qui lance l’application java

java -Xmx1500m -jar CombineBlobs.jar

si le nom du fichier .jar est modifié, changer son nom dans le fichier .bat

2/ utiliser l’invite de commande d’une fenêtre Ms-Dos (MicroSoft Disk Operating System)

Ouvrir la fenêtre Ms-Dos : drapeau Windows / rechercher programmes / taper cmd

Dans la fenêtre cmd.exe

i. Se placer dans la bonne partition : au départ le répertoire courant est C : \ >

pour accéder à un autre disque, taper simplement sa lettre de référence --- > F :

Nouvelle ligne F : \ >

ii. saisir le chemin d’accès au fichier .jar de CombineBlobs

rappel de quelques commandes Ms-Dos

cd .. pour redescendre dans l’arborescence ; cd nom\_dossier pour avancer ; dir pour lister le contenu du dossier ; cd\ pour revenir à la racine

penser à utiliser la touche tabulation pour ne pas saisir la totalité du nom du dossier

iii. une fois placé dans le bon répertoire, taper la commande java telle que contenue dans le fichier .bat

Calcul de l’échelle (mètres par pixel) sur chaque zone. Comme, sur chaque zone, les images ont été stabilisées par rapport à une seule image de référence, il suffit de calculer l’échelle de ces images de référence. Pour cela, j’ai pris deux points facilement repérables sur chaque image de référence, j’ai mesuré la distance d entre ces deux points en mètres sur GoogleEarth, et j’ai repéré leurs coordonnées (x,y) en pixels sur l’image de référence. Pour deux points P1 et P2 de coordonnées respectives (x1,y1) et (x2,y2), l’échelle s’exprime comme suit :

$$scale=\frac{d}{\sqrt{\left(x\_{2}-x\_{1}\right)^{2}+\left(y\_{2}-y\_{1}\right)^{2}}}$$

La détermination des points se faisant à la main, il peut y avoir une erreur d’environ 2 pixels sur chaque dimension entre les points P1, P2 et les points utilisés sur GoogleEarth pour la mesure de d. Les résultats seront donc exprimés avec leur incertitude, suivant l’expression suivante :

$$scale\_{\pm }=\frac{d}{\sqrt{\left(x\_{2}-x\_{1}\pm 2δx\right)^{2}+\left(y\_{2}-y\_{1}\pm 2δy\right)^{2}}}$$

Où $δx=2 pixels$ et $δy=2 pixels$.

|  |  |
| --- | --- |
| Zone 1 : d = 306.70 mPoint 1 : (x,y) = (885,629)Point 2 : (x,y) = (2335,496) $scale=0.2106 \pm 0.0005 m/pixel$ | Zone 2 bis : d = 803.84 mPoint 1 : (x,y) = (48,383)Point 2 : (x,y) = (2380,769) $scale=0.3401 \pm 0.0007 m/pixel$ |
| Zone 2 : d = 603.86 mPoint 1 : (x,y) = (397,489)Point 2 : (x,y) = (2300,555) $scale=0.3171 \pm 0.0007 m/pixel$ | Zone 3 :  d = 244.68 mPoint 1 : (x,y) = (733,553)Point 2 : (x,y) = (1733,639)$scale=0.244 \pm 0.001 m/pixel$  |

Pour utiliser CombineBlobs, il faut, comme pour ObjectFinder, avoir un fichier .bat que l’on lance directement et qui précise le jour de l’exécution et est associé à une copie d’écran qui précise les paramètres utilisés.

Contenu d’un fichier CombineBlobs.bat

java **-Xmx1600m** -jar **CombineBlobs.2012-02-03.jar** FirstFile=.\images\sCam\_1\_Zone\_1\_140911\_FM\_film2\_00001\_07-49-20.152.txt LastFile=.\images\sCam\_1\_Zone\_1\_140911\_FM\_film2\_36239\_08-15-02.449.txt

Quelques explications sur les lignes précédentes :

**-Xmx1600m :** c’est là qu’on précise l’espace mémoire à allouer à la machine virtuelle. Si cet espace n’est pas suffisant, un message d’erreur s’affiche : java heap space

1600m fonctionne pour un fichier image de 40.5 Go

**CombineBlobs.2012-02-03.jar :** fait référence au fichier .jar situé dans le même dossier et qui permet de lancer l’application. Faire correspondre le nom du fichier .jar présent dans le dossier avec le nom inscrit dans le .bat.

Les fichiers textes sont placés dans le dossier images, on spécifie alors le nom du premier et du dernier file.

**.\images\sCam\_1\_Zone\_1\_140911\_FM\_film2\_00001\_07-49-20.152.txt**

**.\images\sCam\_1\_Zone\_1\_140911\_FM\_film2\_36239\_08-15-02.449.txt**

A noter que si l’application java est lancée depuis une fenêtre Ms-Dos, les lignes First et Last sont renseignées directement depuis l’application via les boutons réservés à cet effet : Select First / Select Last

Une fois l’application lancée, régler les paramètres (faire une copie d’écran pour garder trace de ce réglage)

* Attention surtout au scale : ce paramètre dépend de la zone
* Décocher Frame Rate is fixed
* Laisser les autres paramètres en valeurs par défaut (0.5 / 2 / 3 / 6 / 6)
* Min chain length : 20
* Et Max speed km/h : 200
* Cocher link\_chains

Pour lancer l’application : cliquer sur Link Blobs

A la fin du processus, sauvegarder les trajectoires : écrire le nom du fichier avec son extension .txt

Pour info, dans le déroulement de l’application, les steps correspondent aux images successives.

# Annexe

(texte écrit par Peter Knoppers, TU Delft, NL)

## CombineBlobs - Object tracking software developed at TU Delft

CombineBlobs is software that interprets files created by [ObjectFinder](http://www.regiolab-delft.nl/?q=node/79) (cf FAQ-3). CombineBlobs expect to find (in one directory):

* Stabilized image files of the scene of interest
* Text files describing dark and light objects in the format created by ObjectFinder

The files should have the same names, except for the file types. Any common image file type is supported, the text files must have type "txt". CombineBlobs loads the text files and tries to match objects in one image to objects in the surrounding frames. As it does this with the *easy* objects, it learns the local motion field and this helps connecting the *not so easy* objects. Roughly the algorithm looks like this

1. Try to link objects from each image to the preceding image assuming zero speed.
2. Compute the strength of the movement field from the linked images at each object location (excluding the influence of the object itself)
3. Destroy the links and once again try to link the objects in each image to the object in the preceding image now assuming the computed speed of the movement field.
4. Same steps as before, but now in the forward direction
5. Resolve differences in forward linked chains and reverse linked chains.
6. Try to link chains over (increasingly longer) gaps. A successful link can result in the breakup of another chain which is then processed in the same way. Chains shorter than 10 are ignored.

The objects in the input file are defined by their octagonal bounding box, *weight* (number of filled pixels inside the bounding box), center of *gravity* and *average* color. Very small objects and objects that have large bounding box surface to weight ratio are less credible than larger objects and objects that are more solid.

Connecting objects is controlled by a function that determines *credibility* of the link. This credibility depends on

* 1. Credibility of the connecting object and the connected object
	2. Similarity of actual displacement of the center of gravity and the expected displacement.
	3. Similarity of the weights of the objects.
	4. Similarity of the colors of the objects.

Additionally there is a speed limit that determines how far an object can move.
Objects are linked to their most *credible* target. If the target is already linked to by another object and the newly proposed link has a higher credibility, that old link is severed. Another attempt will be made to link objects that got disconnected in this way.
The linking of chains is done like this:

* 1. Compute the median and standard deviation of the weight of the last (maximum 50) objects in the chain.
	2. Select the last (maximum 20) objects in the chain that have weight in the range *median - standard\_deviation .. median + standard\_deviation* and fit a least squares line through the centers of gravity of these objects. If less than two objects have weights in this range, processing this chain ends here.
	3. For each time step up to the maximum range, using the least squares line, extrapolate the position of the object and add a margin that corresponds to a maximum acceleration/deceleration/course change of 0.5 m/s2 and a fixed margin of 2.0m. These values can be changed by the user.
	4. If the extrapolated line plus the margin hits the center of gravity of an object within an additional margin depending on the weight of that object and that object is contained in a chain that is at least 10 links long, try to merge these chains.

A chain merging attempt consists of a number of checks:

* 1. If the leading chain ends before the continuing chain starts (there is no overlap) the last object of the leading chain is linked to the first object of the continuing chain (the connection is made).
	2. If the chains overlap and one chain completely extends over the time range of the other, no connection is made.
	3. If the end of the leading chain overlaps the beginning of the continuing chain, the optimal time-step is searched from where the objects of continuing chain must be merged with the objects in the leading chain. If this is **not** the first object in the continuing chain, the continuing chain would have to be broken. If the existing link in the continuing chain is less credible than the new link would be, the continuing chain is broken, otherwise, merging is attempted at the next object in the chains. If no suitable starting point is found, no connection is made.
	If a connection is made, the objects in the parallel section are merged. Merging extends the octagonal bounding box, adds the weights and assigns the color of the heaviest of the objects to the new object.

## Scale

There are two scale values that must be set by the user:

|  |  |
| --- | --- |
| Scale | m / pixel |
| Frame rate | Hz |

Both scales must be constant throughout the image series. To allow CombineBlobs to operate on images taken at a non-constant frame rate there is a possibility to encode the acquisition time of each frame in the file name.
The distance scale must be consistent throughout the area of the image where objects are to be tracked. This can be achieved by setting the correct output transform in [ImageTracker](http://www.regiolab-delft.nl/?q=node/76), provided that the area of interest lies in a plane.
The chain connecting algorithm (described below) is controlled by these parameters:

|  |  |
| --- | --- |
| Maximum catch acceleration | m / s2 |
| Maximum catch distance | m |
| Maximum catch gap | s |

# Use of the CombineBlobs program

The main window consists of two areas:

* 1. Buttons, controls and list of input files (left).
	2. Tab control (right)

The tab control shows either the currently selected input image (top) and the trajectories (bottom), or the content of the currently selected input file as a list of objects (with optional links).

### XXX describe object list format here

### Scene image

This area shows the image of the scene with the locations of the objects in green (more credible objects) or yellow (less credible objects). The objects are drawn by their octagonal bounding box plus a circle with area proportional to the weight of the object and centered on the center of gravity. When the blobs are linked, an arrow from the center of gravity to the next position is drawn. Additionally, the strength of the movement field is drawn in a 20x20 pixels grid.
At the left of the image is a range selector with a red and a green slider. With these sliders the y-range of interest can be set. Objects outside the selected range will not be shown in the trajectory graph below.

### Trajectory graph

This graph is shown with the time-scale running from top to bottom and the x-component of the location of the objects from left to right. This x-component is identical to the x-component in the image shown above the trajectory graph. Therefore, both images scroll horizontally using the same scroll bar. The wide gray line that runs from left to right corresponds to the time of the selected blob file. The thin gray lines are 10 time steps (frame times) away from the wide gray line. CombineBlobs keeps the wide gray line in the center of the image, unless the selected object file is too close the beginning or the end of the series.

The graph shows the locations of the objects over a range of time steps. The X-position of the objects is copied from the image above the trajectory graph. The y-position is translated into a color in the range from Red to Green. Only objects between the knobs of the range control are shown. The objects with y-coordinate close to the Red knob are shown in Red, those near the Green knob in Green, objects between those are shown in color on a scale from Red, via Yellow to Green.
After the objects have been linked (with a click on the *Link blobs* button) the links are shown as well. The centers of gravity are connected and the leftmost and rightmost x-coordinates are linked. If the link is credible, the connection of the centers of gravity is a fat line. Series of connected objects are called chains.

If the *Link chains* check box is checked, an attempt is made to link up those chains that have a minimum length (currently 10 time steps; not user-settable). This process can cover fairly long distances in time (currently 3 seconds; not user-settable). It will also fix cases where a vehicle moved under a fixed object, or over a road marking that has approximately the color of the vehicle. In such cases, a vehicle disappears at one point and shows up at another point. The disappearance and reappearance can happen simultaneously and when that happens, blobs are combined. Combined blobs are indicated by super-imposing a blocked line over the X-range of the object that was merged into another object (which was enlarged).

When some distance in time is covered by the chain linking algorithm, white lines indicate successful connection between the chains. When unsuccessful, a thick, dashed line connects the last object with credible size of a chain to the last object of the chain and then to the extrapolated location of that chain at the time of that last object. A Cyan funnel shape indicates the range that was (unsuccessfully) searched to connect interrupted chains.

## Selecting input files

The buttons *Select first file ...* and *Select last file ...* are used to select the text files that contain the objects that must be linked.
Click on the button, navigate to the file and select it. The selected files are listed under *Blob files*.
Click on a blob file to show the contents in the road image area or as a list of objects.
The bottom left area contains several buttons, check boxes, sliders and edit areas:

|  |  |
| --- | --- |
| Link blobs button | Click to start the blob linking process (meter / pixel and frames / second values must be set first) |
| Cancel | Cancels the link blobs operation |
| Minimum chain length slider | Suppress showing of objects that are not linked in a chain of at least the selected length |
| Maximum speed (km/h) slider | Maximum speed at which an object can move |
| meter / pixel edit | Enter the size scale of the images here |
| frames / second edit | Enter the frame rate here |

Currently, these controls are shown in a rather illogical order.

## Connecting the objects

Click on the *Link blobs* button to link the objects. This can take a while and progress information is displayed along the bottom of the window. The operation can be cancelled with the *Cancel* button.

## Saving the results

Click on the *Save trajectories* button to save the chains to a text file. The file has the following format.

*Marginal Note : one should specify the extension (.txt) of the file when entering the name.*

Comment lines start with a # (these lines are used to document the file format).
All other lines consist of 21 fields separated by a tab character: <td s< td=""> </td s<>

|  |  |  |
| --- | --- | --- |
| Field | Content | unit |
| 0 | Not used (blank) |  |
| 1 | Chain number (first chain has number 1) |   |
| 2 | Time since first selected object file | s |
| 3 | Left of bounding octagon (minimum(X) | m |
| 4 | Top of bounding octagon (minimum(Y)) | m |
| 5 | Right of bounding octagon (maximum(X)) | m |
| 6 | Bottom of bounding octagon (maximum(Y)) | m |
| 7 | Top-left of bounding octagon (minimum (X+Y)) | m |
| 8 | Top-right of bounding octagon (maximum(X-Y) | m |
| 9 | Bottom-left of bounding octagon (minimum(X-Y) | m |
| 10 | Bottom-right of bounding octagon (maximum(X+Y) | m |
| 11 | Center of gravity X-coordinate | m |
| 12 | Center of gravity Y-coordinate | m |
| 13 | Mass of object (number of pixels) |   |
| 14 | Color of the object (RRGGBB, hexadecimal) |   |
| 15 | Predecessor time steps difference (integer) |   |
| 16 | Predecessor rank number (integer) |   |
| 17 | Predecessor link credibility (positive floating point value; higher values indicate higher credibility) |   |
| 18 | Successor time steps difference (integer) |   |
| 19 | Successor rank number (integer) |   |
| 20 | Successor link credibility (positive floating point value; higher values indicate higher credibility) |   |

Floating point values are output using the radix symbol of the locale setting of the computer. Each chain will occupy a series of lines; each line describes the location of the object at one particular time step.
The predecessor and successor fields are empty when no such link was made.