SEMINAR *SEAL Team — Exploring Robotics — Remote Sensing*

Cyclone monitoring

Direct interpretation of satellite images

Laurent Beaudoin & Loïca Avanthey

Épita

In this first **concrete application**, we seek to follow the **evolution of a cyclone** by determining :

a) its trajectory over time

b) its instantaneous speed (norm and direction)

1 Image dataset

As we have seen in class, **meteorology** is one of the most important **field of application** for satellite imagery. Geostationary and polar satellites cover the Earth 24 hours a day. The former offer a **global view** at **high frequency**, while the latter provide more **precise information**, but on areas with **smaller swaths** and with a **longer revisit time**.

For our application, we will use a time series from the **SEVERI sensor** carried by the geostationary satellite **Météosat 8** (MSG1). The images (examples in figure 1 page 1) are those of **channel 9**, that is to say the $10.9\mu m$ **infrared band**.

By the way, why take the images in an **infrared band** and not in the **visible band** to track a cyclone ?

They are in **full resolution** but **truncated** (1373×1253) compared to the global image (3712×3712). The **upper left corner** of the truncated image is **offset** by 407 lines and 132 columns from the upper left corner of the complete image. This information will be used in the section 3 page 3.

The set of images available is composed of **21 images** selected at a regular time interval (10h) over a period of 10 days (the SEVERI sensor takes an image every quarter of an hour, the total set is therefore composed of 430 images).

To find out **more about** this satellite, its sensors and its missions, we can refer to [ESA18].

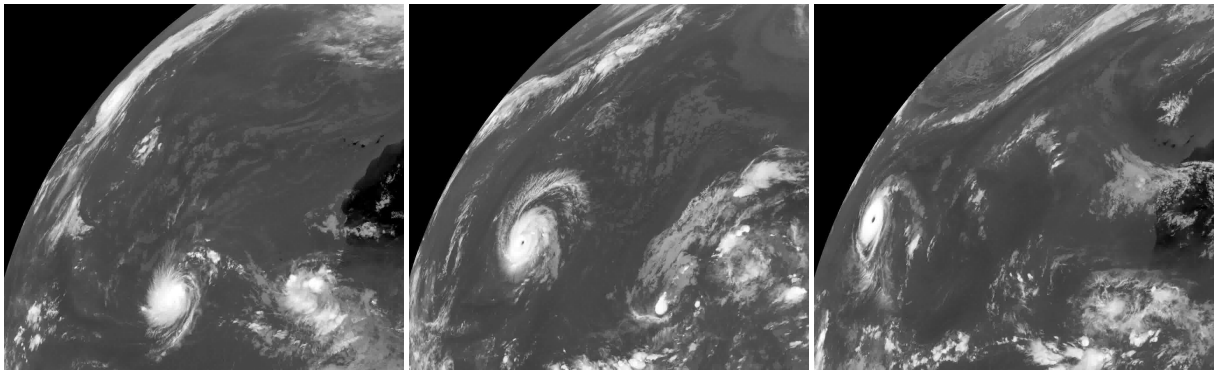

Figure 1 – *Some images of the cyclone extracted from the data set taken by the geostationary satellite Meteosat 8 (MSG-1) in the* $10.9\mu m$ *infrared band (channel 9, truncated images).*

2 Objective of the work

The **objective** of the work to be done for this seminar is to study the **evolution of a cyclone** by following its **trajectory in time** and by calculating its **instantaneous speed** (value and direction). For this, we will base ourselves on the **position of the center of the eye** of the cyclone. We will collect these successive pixel coordinates **manually**. But to calculate the **real speeds**, we need the **real distance** between these coordinates.

Can we calculate the **distance** in kilometers **directly** from the pixel coordinates as we did in the previous exercises ? Why ?

As we have seen in class, and in particular for limbo to limbo satellites as is the case for this MSG 1 satellite (the swaths go from one horizon to another), the **resolution** on the ground **varies greatly** between the **nadir** (ie the vertical point of the satellite) and the visible **edges** of the Earth. Each pixel does not represent the **same surface** on the ground, so we cannot calculate a distance in kilometers **directly** from a distance in pixels between two points of the image.

To get around the problem of the **complex variation** of the **resolutions** according to the viewing angle, and therefore of the problem of the **calculation of the real distances**, we will have to convert the pixel coordinates into geographic coordinates (longitude, latitude) using the satellite acquisition specifications.

The **general workflow** of the program we are trying to develop is as follows :

— Manually select the **pixels** corresponding to the **center of the eye of the cyclone** on each image to obtain their **images coordinates** (rows, columns)
— Convert the **images coordinates** (rows, columns) into **geographics coordinates** (latitude, longitude) (see section 3 page 3).
— Calculate the **angular differences** between **two geographical positions** (see section 6 page 4).
— Calculate the **actual distance** traveled between two geographic positions (see section 6 page 4).
— Calculate the **travel speed** and its **orientation** between two geographic positions (see section 6 page 5).

The **final results** can be **reported** in the **table** 6 page 6. The **details** of these different steps are described in the **following sections** and to help you in this work, we provide you with a **code frame** to complete [1]. It will **load** the **images**, to retrieve the **image coordinates** of a clicked pixel and **display** the **trajectory**. The calculations you will have implemented will be applied as you go and you will be able to **export** the **image** of the **trajectory** and the **results table** in a CSV file.

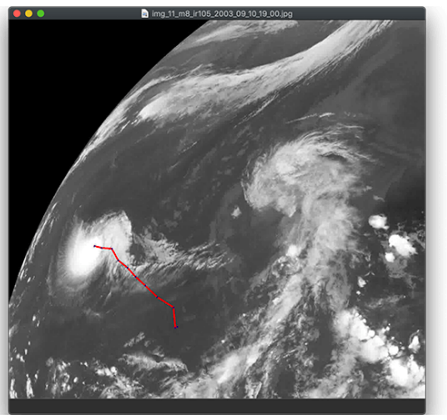

Figure 2 – *Screenshot of the program in action*

1. Thanks to Thomas Michelot (GISTRE 2021) for the development of the interface.

3 Conversion of image coordinates into geographic coordinates

The **reference document** to carry out this operation is the technical document [EUM13] and more precisely pages 24 to 32. You must **program the functions** which allows you to pass from **image coordinates** (column, line) to **geographic coordinates** (latitude, longitude) by coding the equations of this document.

To help you go faster, here are some explanations :

You **cannot go directly** from entire row / column coordinates (c, l) of the pixel to the geographic coordinates latitude / longitude (lat, lon). You have to go through an **intermediate coordinate system**, that we will called (x, y) thereafter.

EXERCICE 1 (*Image to Intermediate coordinates*)

Start by writing two **functions** to calculate (x, y) knowing (c, l) :

- `double Convert::intermediateXFromPixelC(int c);`
- `double Convert::intermediateYFromPixelL(int l);`

To do this, **reverse** the **system of equations** in 4.4.4 page 32 of the **reference document**.

Remember to apply the **offsets** to the image coordinates (see section 1 page 1) as we work on **cropped images**. Set these offsets as constants

Take these values for the **constants** (put them in `src/maths/Convert.hh`) :

- `CFAC = 781648343`
- `LFAC = 781648343`
- `COFF = 1857`
- `LOFF = 1857`

EXERCICE 2 (*Intermediate to Geographics coordinates*)

Write the eight **functions** which makes it possible to pass from (x, y) to (lat, lon). For this, we will use the **equations** on page 29 of the **reference document**. Let $s_d = \sqrt{s_a}$ and start by programming :

- `double Convert::getSA(double x, double y);`
- `double Convert::getSN(double sd, double x, double y);`
- `double Convert::getS1(double sn, double x, double y);`
- `double Convert::getS2(double sn, double x, double y);`
- `double Convert::getS3(double sn, double y);`
- `double Convert::getSXY(double sn, double x, double y);`

Finally calculate (lat, lon). If the value of s_a is negative, it means that you are beyond the limb and therefore you will return $(-999.999, -999.999)$ as an outlier. Take 0.0 as the value for the constant `SUB_LON` and 1.006739501 is the value of the constant `SQUARED_ELLIPSOID_FLATNESS`.

- `double Convert::latitudeFromIntermediate(double x, double y);`
- `double Convert::longitudeFromIntermediate(double x, double y);`

Remember to **convert** longitude and latitude from radians to **degrees** !

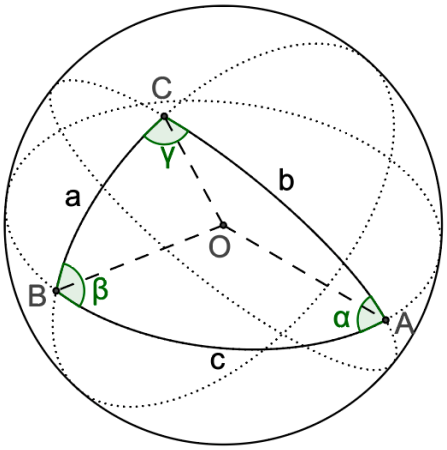

Figure 3 – *Relationship between the angles in spherical trigonometry :* `a` *is the angle* $\widehat{COB}$, `b` *is the angle* $\widehat{COA}$ *and* `c` *is the angle* $\widehat{AOB}$.

4 Calculation of displacement speeds of the eye of the cyclone

To calculate the **distance between two points** on the **surface** of a **sphere** whose geographic coordinates are known, we will use **spherical trigonometry**. Using the figure 3 page 4, we remind you of the cosine formula :

$$\cos c = \cos a \cos b + \sin a \sin b \cos \gamma \tag{1}$$

In our case, we have **two points** A (lat_A, lon_A) and B (lat_B, lon_B) in geographic coordinates (in degrees) corresponding to **two positions** of the eye of the cyclone. We want to calculate the **distance** between these two points. For that, simply calculate the **angle** c and **multiply** it by the **Earth's radius**.

To **simplify** the calculations, we are going to place the point C at the **North Pole**. Thus the angle a becomes the **complementary of the latitude** of the point B (ie $a = 90 - lat_B$), the angle b becomes the **complementary of the latitude** of the point A (ie $b = 90 - lat_A$) and the angle γ becomes the **the difference in longitudes** (ie $\gamma = lon_B - lon_A$). With these modifications, the equation 1 becomes :

$$\begin{aligned} \cos c &= \cos(90 - lat_B)\cos(90 - lat_A) + \sin(90 - lat_B)\sin(90 - lat_A)\cos(|lon_B - lon_A|) \\ &= \sin(lat_B)\sin(lat_A) + \cos(lat_B)\cos(lat_A)\cos(|lon_B - lon_A|) \end{aligned} \tag{2}$$

$$\Leftrightarrow c = \arccos\left(\sin(lat_B)\sin(lat_A) + \cos(lat_B)\cos(lat_A)\cos(|lon_B - lon_A|)\right) \tag{3}$$

And so if we assimilate the Earth to a **perfect sphere** with a **radius** of $6,367.3764km$ (`EARTH_RADIUS` constant), we get the **distance** between point A and B with the following equation :

$$d_{AB} = Radius_{Earth} \times c \tag{4}$$

EXERCICE 3 (*Angle and Distance*)

Code these two equations 3 and 4 in the functions :

- `double Convert::getAngularAtoB(double lat_A, double lon_A,double lat_B, double lon_B);`
- `double Convert::getDistance(double angle_rad);`

Remember to **convert** longitude and latitude from degrees to **radians** ! We want to obtain the **angle c** in **radians** to calculate the distance afterwards.

Finally, we recall that to estimate the value of a **speed** at a **point** P_i, we calculate the **distance** ΔL_i between the points P_{i+1} and P_{i-1} which correspond respectively to points B and A of the formulas seen in the previous section. And then we **divide** it by ΔT_i, the **time elapsed** between the shooting of P_{i-1} and P_{i+1}.

EXERCICE 4 (*Speed*)

Knowing that there are 10 hours between each image, thus ΔT_i is worth 20 hours. Code this function to calculate the **speed of the eye of the cyclone** for each point P_i where this is possible :

- `double Convert::getSpeed(double lat_A, double lon_A,double lat_B, double lon_B);`

As we are working on P_{i+1} et P_{i-1}, the speed value (and therefore the orientation value thereafter) **doesn't make sense** for the **first** and **last** images, so it is normal for the program to displays N/A for these.

5 Calculation of the orientations of the displacement speeds

Finally, to calculate the **orientation**, that is to say the **angle** between the **speed vector** and the **north direction**, we will use another **well-known spherical trigonometry** formula which is the **sine formula** :

$$\frac{\sin(a)}{\sin(\alpha)} = \frac{\sin(b)}{\sin(\beta)} = \frac{\sin(c)}{\sin(\gamma)} \tag{5}$$

In our case, we are therefore looking for the **angle** α knowing the angles a, c and γ. We therefore calculate :

$$\sin(\alpha) = \frac{\sin(a)\sin(\gamma)}{\sin(c)} \tag{6}$$

With the same simplifications used as previously in the equation 2, we have :

$$\begin{aligned} \alpha &= \arcsin\left(\frac{\sin(90-lat_B)\sin(|lon_B-lon_A|)}{\sin(c)}\right) \\ &= \arcsin\left(\frac{\cos(lat_B)\sin(|lon_B-lon_A|)}{\sin(c)}\right) \end{aligned} \tag{7}$$

EXERCICE 5 (*Orientation*)

Code the equation 7 in the following function :

- `double Convert::getOrientation(double lat_A, double lon_A, double lat_B, double lon_B);`

6 Across the whole dataset

Run the program on the entire set of images to calculate the entire path and retrieve all velocities and orientations. You can export the result table.

Image ID	**Row**	**Column**	**Latitude**	**Longitude**	**Speed** (km/h)	**Orientation** $(^o)$
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						

Table of cyclone monitoring results.

7 To go further

QUESTION 1

 What is the **name** of this cyclone ?

QUESTION 2

 Get the **official trajectory** and **compare** with your results.

Bonus !

Do not hesitate to use this first (small) **database** to **test prediction algorithms** like Kalman's for example (you can come and ask us for 430 images of the total sample).

Références

[ESA18] ESA. Meteosat Seconde Generation (MSG). https://directory.eoportal.org/web/eoportal/satellite-missions/m/meteosat-second-generation, 2018. [site-web, ESA Portal]. 1

[EUM13] EUMETSAT. Lrit/hrit global specification. Technical report, Coordination Group for Meteorological Satellites, 2013. 3