3 Challenges in Source Detection

Some of these notes are directly from the Sextractor 2.5 manual. They are marked with a ** in the beginning of a paragraph.

First: Background Estimation.

** First, on most astronomical images, the background is not constant over the frame, and its determination can be ambiguous in crowded regions.

Second: Filtering

** Second, the software has to operate on noisy data, and some filtering adapted to the characteristics of the image has to be applied prior to detection, to reduce the contamination by noise peaks.

Third : Detection

** Third, many sources that overlap on the image are unlikely to be detected separately with a single detection threshold, and require a de-blending procedure, which is actually multi-thresholding in SExtractor.

We focus on the detection part since we are interested in knowing which pixels are considered as a part of the source and what is the extent of a source as detected by Sextractor

Detection

** Thresholding is applied to the background-subtracted, filtered image to isolate connected groups of pixels. Each group defines the approximate position and shape of a basic SExtractor detection that will be processed further in the pipeline. Groups are made of pixels whose values exceed the local threshold and which touch each other at their sides or angles (“8-connectivity”).

** DETECT THRESH sets the threshold value. If one single value is given, it is interpreted as a threshold in units of the background’s standard deviation. For example: DETECT_THRESH 1.5 will set the detection threshold at 1.5σ above the local background.

** DETECT MINAREA sets the minimum number of pixels a group should have to trigger a detection. In most cases it is therefore recommended to keep DETECT MINAREA at a small value, typically 1 to 5 pixels, and let DETECT THRESH and the filter define SExtractor’s sensitivity.

Deblending

** Each time an object extraction is completed, the connected set of pixels passes through a sort of filter that tries to split it into eventual overlapping components. This case appears more frequently when the field is crowded or when the detection threshold is set very low. The deblending method adopted in SExtractor, is based on multi-thresholding, and works on any kind of object; but it is unable to deblend components that are so close that no saddle is present in their profile. However, as no assumption has to be made on the shape of the objects, it is perfectly suited for galaxies as well as for high galactic latitude stellar fields.

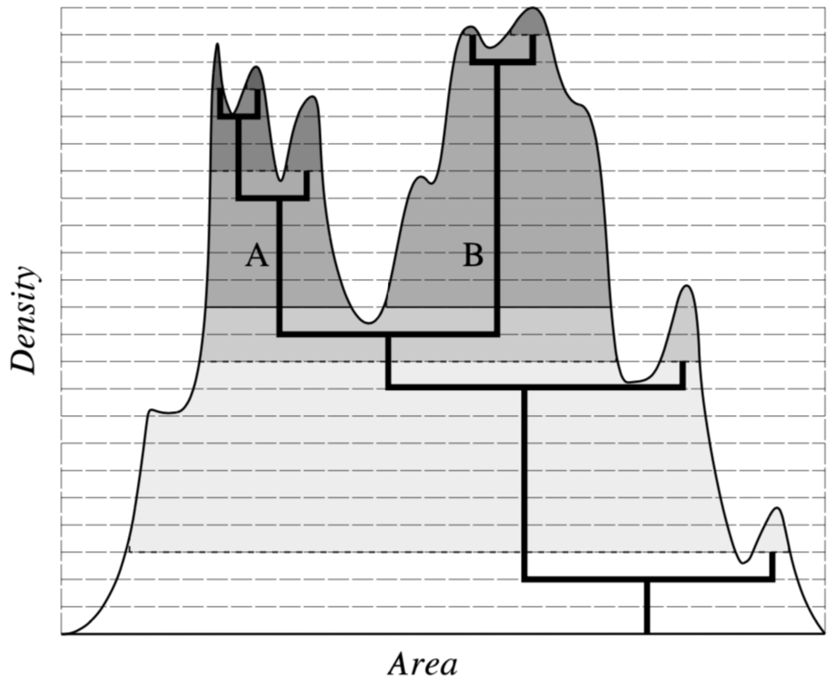
** each extracted set of connected pixels is re-thresholded at N levels linearly or exponentially spaced between its primary extraction threshold and its peak value. This gives us a sort of 2-dimensional “model” of the light distribution within the object(s), which is stored in the form of a tree structure (fig. 3). Then the algorithm goes downwards, from the tips of branches to the trunk, and decides at each junction whether it shall extract two (or more) objects or continue its way down. To meet the conditions described earlier, the following simple decision criteria are adopted: at any junction threshold ti, any branch will be considered as a separate component if:

(1) the integrated pixel intensity (above ti) of the branch is greater than a certain fraction δc of the total intensity of the composite object.

(2) condition (1) is verified for at least one more branch at the same level i.

In [41]:
```python
from IPython.display import Image
Image('documentation/deblend.png',width=500,height=500)
```

Out[41]:



Measurements

** Once sources have been detected and deblended, they enter the measurement phase. There are in SExtractor two categories of measurements. Measurements from the first category are made on the isophotal object profiles. Only pixels above the detection threshold are considered.

** Measurements from the second category have access to all pixels of the image. These measure- ments are generally more sophisticated and are done at a later stage of the processing (after CLEANing and MASKing).

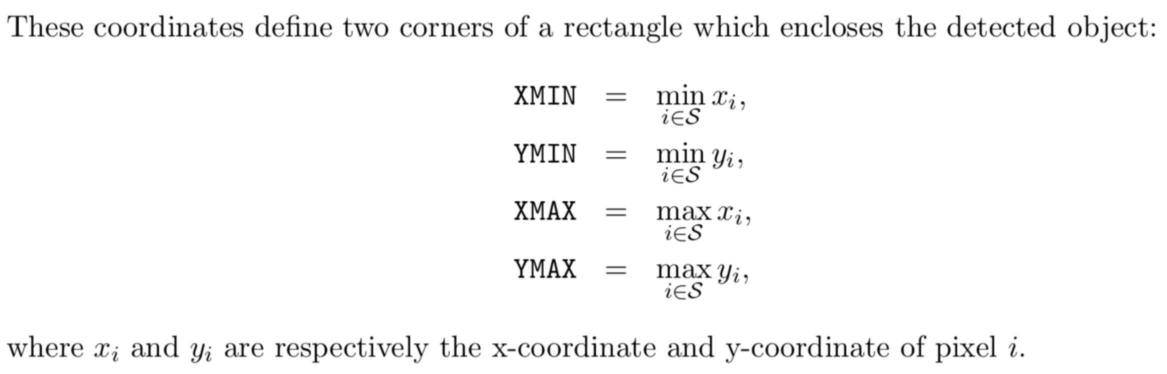
Measurement of coordinates of detected object

** The following parameters are derived from the spatial distribution S of pixels detected above the extraction threshold. The pixel values Ii are taken from the (filtered) detection image.

Limits XMIN, XMAX, YMIN, YMAX

```
In [42]: from IPython.display import Image
         Image('documentation/xmin.png',width=900,height=900)
```

Out[42]:



These coordinates define two corners of a rectangle which encloses the detected object:

$$\begin{aligned} \texttt{XMIN} &= \min_{i\in\mathcal{S}} x_i, \\ \texttt{YMIN} &= \min_{i\in\mathcal{S}} y_i, \\ \texttt{XMAX} &= \max_{i\in\mathcal{S}} x_i, \\ \texttt{YMAX} &= \max_{i\in\mathcal{S}} y_i, \end{aligned}$$

where x_i and y_i are respectively the x-coordinate and y-coordinate of pixel i.

Barycenter X, Y

** Barycenter coordinates generally define the position of the “center” of a source, although this definition can be inadequate or inaccurate if its spatial profile shows a strong skewness or very large wings. X and Y are simply computed as the first order moments of the profile:

In [43]:
```python
from IPython.display import Image
Image('documentation/xy.png',width=500,height=500)
```

Out[43]:

$$\mathrm{X} = \overline{x} = \frac{\sum\limits_{i\in\mathcal{S}} I_i x_i}{\sum\limits_{i\in\mathcal{S}} I_i},$$

$$\mathrm{Y} = \overline{y} = \frac{\sum\limits_{i\in\mathcal{S}} I_i y_i}{\sum\limits_{i\in\mathcal{S}} I_i}.$$

So we now know that the center is simply a first order moment of the pixel coordinates of the pixels of the detected object (which are above a certain threshold).

** Actually, xi and yi are summed relative to XMIN and YMIN in order to reduce roundoff errors in the summing.

Position of Peak XPEAK, YPEAK

** It is sometimes useful to have the position XPEAK,YPEAK of the pixel with maximum intensity in a detected object, for instance when working with likelihood maps, or when searching for artifacts. For better robustness, PEAK coordinates are computed on filtered profiles if available. On symetrical profiles, PEAK positions and barycenters coincide within a fraction of pixel (XPEAK and YPEAK coordinates are quantized by steps of 1 pixel, thus XPEAK IMAGE and YPEAK IMAGE are integers). This is no longer true for skewed profiles, therefore a simple comparison between PEAK and barycenter coordinates can be used to identify asymetrical objects on well-sampled images.

2nd order moments X2, Y2, XY

** (Centered) second-order moments are convenient for measuring the spatial spread of a source profile. In SExtractor they are computed with:

In [44]:
```
from IPython.display import Image
Image('documentation/x2y2.png',width=500,height=500)
```

Out[44]:

$$\mathtt{X2} \quad = \overline{x^2} = \quad \frac{\sum\limits_{i\in\mathcal{S}} I_i x_i^2}{\sum\limits_{i\in\mathcal{S}} I_i} - \overline{x}^2,$$

$$\mathtt{Y2} \quad = \overline{y^2} = \quad \frac{\sum\limits_{i\in\mathcal{S}} I_i y_i^2}{\sum\limits_{i\in\mathcal{S}} I_i} - \overline{y}^2,$$

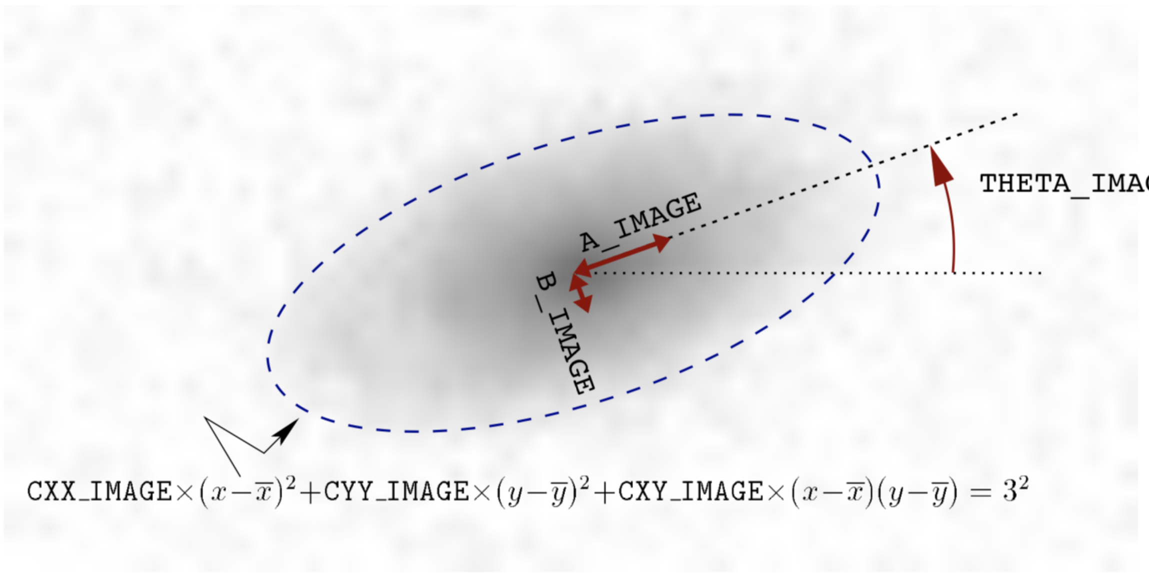
$$\mathtt{XY} \quad = \overline{xy} = \quad \frac{\sum\limits_{i\in\mathcal{S}} I_i x_i y_i}{\sum\limits_{i\in\mathcal{S}} I_i} - \overline{x}\,\overline{y},$$

** These expressions are more subject to roundoff errors than if the 1st-order moments were sub- tracted before summing, but allow both 1st and 2nd order moments to be computed in one pass. Roundoff errors are however kept to a negligible value by measuring all positions relative here again to XMIN and YMIN.

Ellipse around the detected object

In [45]:
```
from IPython.display import Image
Image('documentation/ellipse.png',width=800,height=800)
```

Out[45]:


THETA_IMAGE
A_IMAGE
B_IMAGE
CXX_IMAGE$\times(x-\overline{x})^2+CYY_IMAGE\times(y-\overline{y})^2+CXY_IMAGE\times(x-\overline{x})(y-\overline{y}) = 3^2$

In [ ]:

A, B, THETA

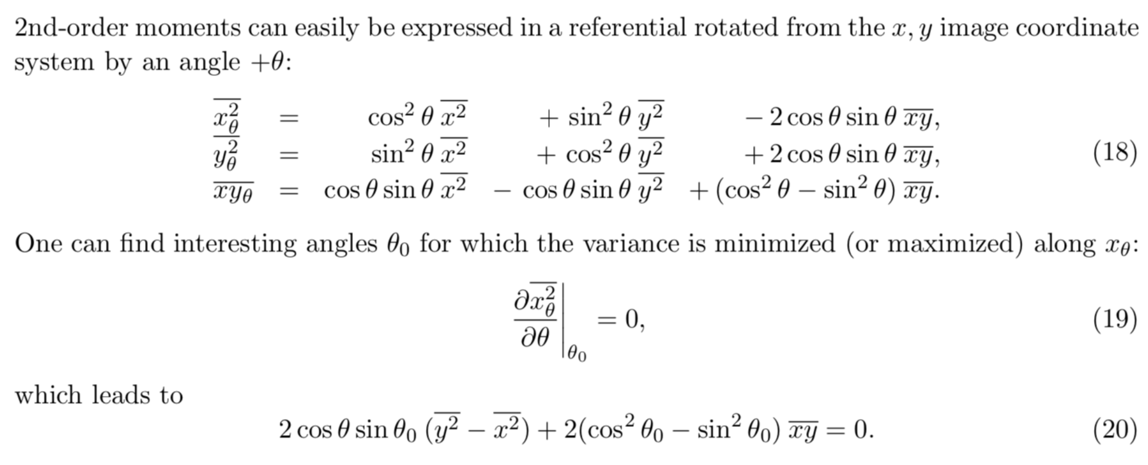
** These parameters are intended to describe the detected object as an elliptical shape. A and B are its semi-major and semi-minor axis lengths, respectively. More precisely, they represent the maximum and minimum spatial rms of the object profile along any direction. THETA is the position-angle between the A axis and the NAXIS1 image axis. It is counted counter-clockwise.

The important part to see is that the angle theta is found out for which the variance is minimized or maximized.

Insert formulae for A,B and THETA

In [46]:
```
from IPython.display import Image
Image('documentation/xtheta.png',width=900,height=900)
```

Out[46]:



2nd-order moments can easily be expressed in a referential rotated from the x, y image coordinate system by an angle $+\theta$:

$$\begin{aligned}
\overline{x_\theta^2} &= \cos^2\theta\,\overline{x^2} + \sin^2\theta\,\overline{y^2} - 2\cos\theta\sin\theta\,\overline{xy},\\
\overline{y_\theta^2} &= \sin^2\theta\,\overline{x^2} + \cos^2\theta\,\overline{y^2} + 2\cos\theta\sin\theta\,\overline{xy},\\
\overline{xy_\theta} &= \cos\theta\sin\theta\,\overline{x^2} - \cos\theta\sin\theta\,\overline{y^2} + (\cos^2\theta - \sin^2\theta)\,\overline{xy}.
\end{aligned} \tag{18}$$

One can find interesting angles θ_0 for which the variance is minimized (or maximized) along x_θ:

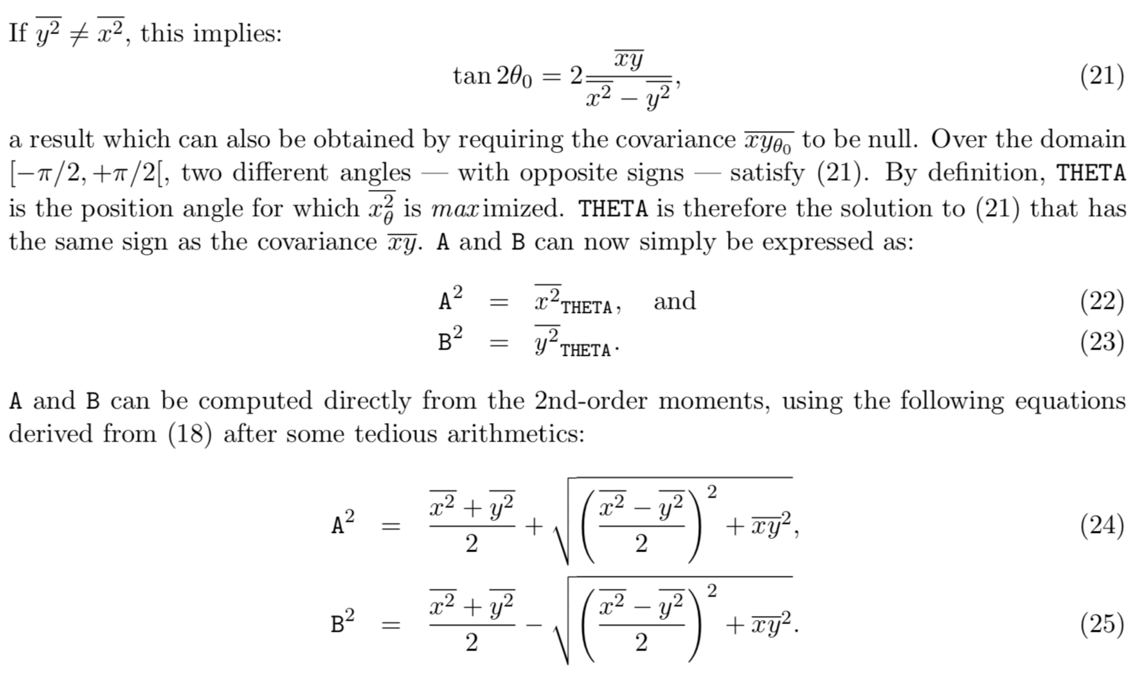
$$\left.\frac{\partial\overline{x_\theta^2}}{\partial\theta}\right|_{\theta_0} = 0, \tag{19}$$

which leads to

$$2\cos\theta\sin\theta_0\,(\overline{y^2} - \overline{x^2}) + 2(\cos^2\theta_0 - \sin^2\theta_0)\,\overline{xy} = 0. \tag{20}$$

In [47]:
```
from IPython.display import Image
Image('documentation/ab.png',width=900,height=900)
```

Out[47]:



If $\overline{y^2} \neq \overline{x^2}$, this implies:

$$\tan 2\theta_0 = 2\frac{\overline{xy}}{\overline{x^2} - \overline{y^2}}, \tag{21}$$

a result which can also be obtained by requiring the covariance $\overline{xy}_{\theta_0}$ to be null. Over the domain $[-\pi/2, +\pi/2[$, two different angles — with opposite signs — satisfy (21). By definition, `THETA` is the position angle for which $\overline{x^2_\theta}$ is *max*imized. `THETA` is therefore the solution to (21) that has the same sign as the covariance $\overline{xy}$. `A` and `B` can now simply be expressed as:

$$\begin{aligned} \mathtt{A}^2 &= \overline{x^2}_{\mathtt{THETA}}, \quad \text{and} \qquad (22)\\ \mathtt{B}^2 &= \overline{y^2}_{\mathtt{THETA}}. \qquad (23) \end{aligned}$$

`A` and `B` can be computed directly from the 2nd-order moments, using the following equations derived from (18) after some tedious arithmetics:

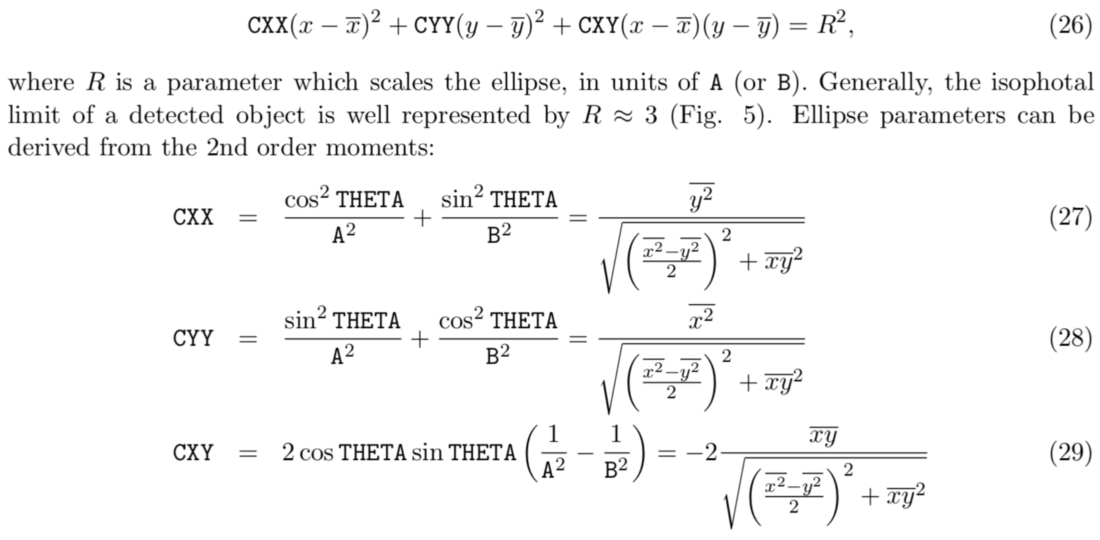
$$\begin{aligned} \mathtt{A}^2 &= \frac{\overline{x^2} + \overline{y^2}}{2} + \sqrt{\left(\frac{\overline{x^2} - \overline{y^2}}{2}\right)^2 + \overline{xy}^2}, \qquad (24)\\ \mathtt{B}^2 &= \frac{\overline{x^2} + \overline{y^2}}{2} - \sqrt{\left(\frac{\overline{x^2} - \overline{y^2}}{2}\right)^2 + \overline{xy}^2}. \qquad (25) \end{aligned}$$

CXX, CYY, CXY

** A, B and THETA are not very convenient to use when, for instance, one wants to know if a particular SExtractor detection extends over some position. For this kind of application, three other ellipse parameters are provided; CXX, CYY and CXY. They do nothing more than describing the same ellipse, but in a different way: the elliptical shape associated to a detection is now parameterized as:

In [48]:
```
from IPython.display import Image
Image('documentation/cxcy.png',width=900,height=900)
```

Out[48]:



$$\mathtt{CXX}(x - \overline{x})^2 + \mathtt{CYY}(y - \overline{y})^2 + \mathtt{CXY}(x - \overline{x})(y - \overline{y}) = R^2, \tag{26}$$

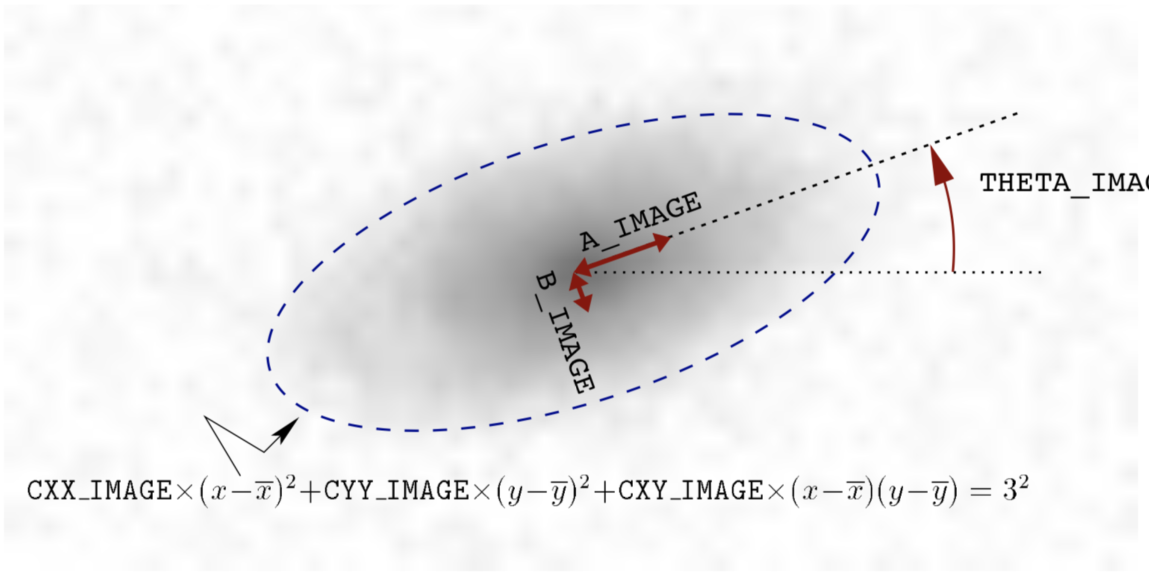
where R is a parameter which scales the ellipse, in units of `A` (or `B`). Generally, the isophotal limit of a detected object is well represented by $R \approx 3$ (Fig. 5). Ellipse parameters can be derived from the 2nd order moments:

$$\begin{aligned} \mathtt{CXX} &= \frac{\cos^2 \mathtt{THETA}}{\mathtt{A}^2} + \frac{\sin^2 \mathtt{THETA}}{\mathtt{B}^2} = \frac{\overline{y^2}}{\sqrt{\left(\frac{\overline{x^2} - \overline{y^2}}{2}\right)^2 + \overline{xy}^2}} \qquad (27)\\ \mathtt{CYY} &= \frac{\sin^2 \mathtt{THETA}}{\mathtt{A}^2} + \frac{\cos^2 \mathtt{THETA}}{\mathtt{B}^2} = \frac{\overline{x^2}}{\sqrt{\left(\frac{\overline{x^2} - \overline{y^2}}{2}\right)^2 + \overline{xy}^2}} \qquad (28)\\ \mathtt{CXY} &= 2 \cos \mathtt{THETA} \sin \mathtt{THETA} \left(\frac{1}{\mathtt{A}^2} - \frac{1}{\mathtt{B}^2}\right) = -2\frac{\overline{xy}}{\sqrt{\left(\frac{\overline{x^2} - \overline{y^2}}{2}\right)^2 + \overline{xy}^2}} \qquad (29) \end{aligned}$$

So now we can actually find out the extent of an object and see whether our object of interest in within the area of the ellipse or not.

In [49]:
```
from IPython.display import Image
Image('documentation/ellipse.png',width=900,height=900)
```

Out[49]:


THETA_IMAGE
A_IMAGE
B_IMAGE
CXX_IMAGE×$(x-\overline{x})^2$+CYY_IMAGE×$(y-\overline{y})^2$+CXY_IMAGE×$(x-\overline{x})(y-\overline{y}) = 3^2$

In [ ]: