PHOTOMETRY: CALIBRATION OF "VISIBILITY" , DATA FIT AND RELATED CALCULATIONS ON T80

ZAKARIA ABDELRAHMAN[1]
ZERGUINE SOHAIB[2]

20/02/2021

ABSTRACT

During our stay on the **O**bservatoire **H**aute de **P**rovence, we took a lesson about the photometry & Astrophysical interferometry. We didn't proceed in data taking because of the upgrades that were performed during the stay from a side, and the bad sky from the otherside.
We used the famous software IDL to do the Fourrier Transform corresponding to our measures and performed cantour integral in order to extract visibility. Next, we calibrated the data that we had extracted, with the help of some Python coding, we fitted our results to the theoretically predicted model, and finally we calculated the angular diameter of Mars.

Guided by **Dr. Hervé LE COROLLER**

[1,2] *Department of Physics, Aix–Marseille University, Marseille, France*

CONTENTS

1 INTRODUCTION

Our practical work on photometry had been passed on OHP at 80cm telescope, which was built in 1932 in Paris and had been installed in its final location since 1945. Here after we presented its main characteristics:

Telescope Specifications		CCD Specifications	
Features	**Property**	**Features**	**Property**
Mount Type	English(Yoke)	Imaging	3072 x 2048
Configuration	Cassegrain	Dark Current	0.5 e-/pixel/s
focal distance	$f = 12$m	Pixel Size	9 µm
Aperture	$f/15$	Cooling	Air(Delta"tech") down to -60 C
field of view	2.8 arc-min/cm	Optics type	Adaptive/(auto Filters)
Guiding	Manual/Auto	Circuitry	Antiblooming

T80 & its digital detector CCD **SBIG stxl-6303** specifications

2 PHOTOMETRY

2.1 Rayleigh Criteria

For any image-forming optical system, the **Rayleight Criterion** measures the ability of this device to distinguish between close objects. The parameter that characterizes this ability is called the **Angular resolution**, given by the following formula:

$$\theta = 1.22\frac{\lambda}{D}$$

So that: {λ: is the wavelength of the incoming light, D: is the diameter of the aperture}.

2.2 Resolution limits

Here after we will present the pioneering telescopes and their limits:

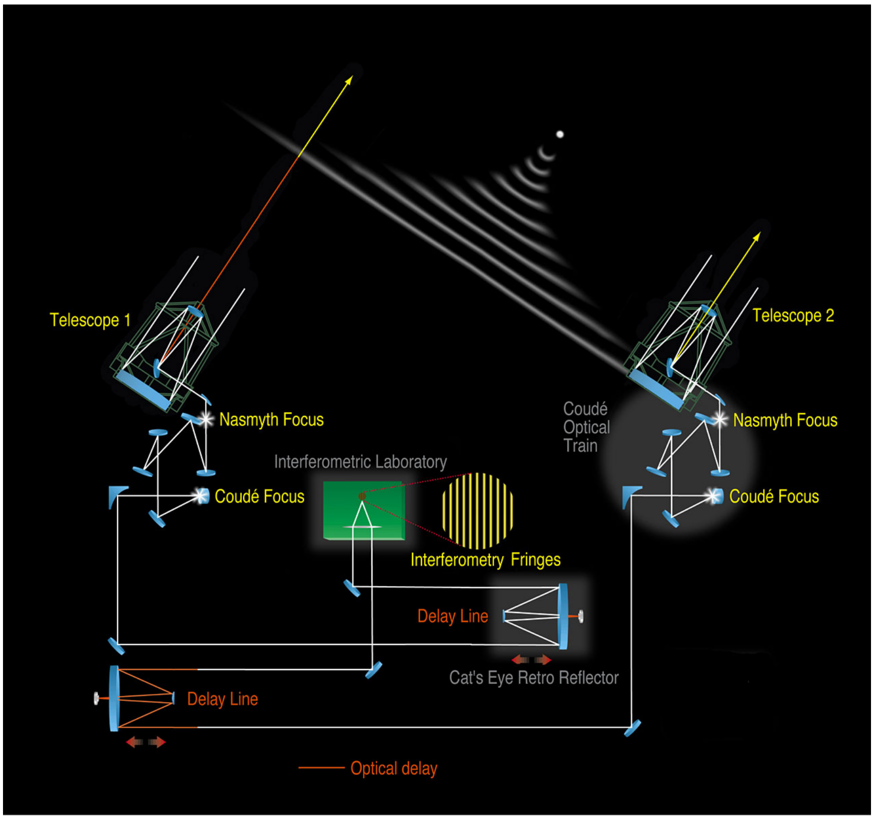
Telescope :	**Built in (year, country)**	**Mirror size**	**Angular resolution**
Keck:	KeckI (1990), KeckII (1996) in Hawaii, US	10m	KeckI(0.04 arcsec), KeckII(0.4 arcsec) & (5 milliarcsec) both combined (Baseline = 85m).
VLT:	Antu(1998), Kueyen(1999), Melipal & Yepun(2000) in Atacama, Chili	8.2m	50 milliarcsec
E-ELT:	Future (2025) in Atacama, Chili	39m	5 milliarcsec

2.3 Resolved Object

The Stars on Andromeda Galaxy are very difficult objects to be resolved on telescopes, since it is located at 2,537 million light years from earth. The resolving ability of Keck which is the most powerful telescope now, could not resolve these stars, we need a mirror of the order of 10^5 km, which is impossible with classical telescopes. And it's very Challenging.

2.4 Interferometery

Even if the working on mirrors in a size of thousands of km is impossible, an alternative solution is possible, it is briefly interferometry, if we make the received light at same distance from two or more telescopes, we will find the usual Young aperture, with completely lightning and dark fringes, if and only if the light is coming from only one source, if light the dark fringes start to be lightned, then the object is said resolved, because we receive it's light from more of one direction.



Blueprint showing interferometry process from **ESO.org**

3 EXPERIMENTAL SET-UP AT T80

During our stay on **OHP**, unfortunately, we didn't get the opportunity to manipulate **T80**, because of a technical issue besides the bad weather. We took instead the data taken by some people the last year.

3.1 schema of the experiment

During the data gathering, disks of different appertures and baselines were used, in our data, we only had found data that belongs to one aperture size(4mm), and three distinct baselines which are respectively (8,12,20)cm.

3.2 Atmospheric turbulence

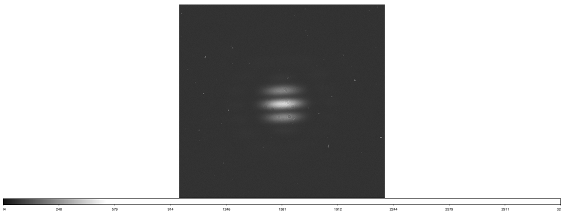
In our case, we are dealing with Mars planet, it is not too far, then with a small aperture of (4mm), the noise effect from turbulance of the atmosphere is minimal.

3.3 solar system

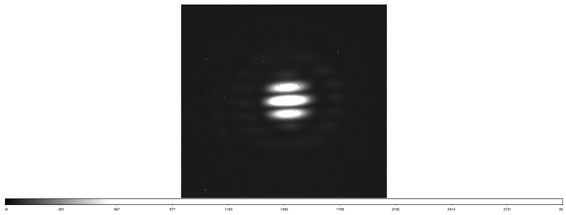
In instance, all of solar planets could been resolved in our telescope, the most promising are Mars & Jupiter, because they are both a good combo of (distance, size) objects.

3.4 fixed baseline

In a fixed baseline of 8mm, we can observe above these two images of two differnt objects: Mars planet which is locating at an average distance of 227944000km, and Sirius which is considered at a distance of 8611 light years($8,15227480810^{16}$km:



Franges in Mars 8mm baseline



Franges in Sirius 8mm baseline

We can observe a huge contrast in Sirius, opposite of when we look at Mars franges, graphical settings are the same for them both

3.5 Angular size of the fringes

The angular size of fringes:

$$\Delta r = 1.22 \times \left(\frac{\lambda \times f}{\text{Aperture}} \right) = 0,004392 \text{rad} = 15,09'$$

The Image we got is a composed of central fringes and an exterior enveloppe, the number of fringes is sensitive to the ratio of Baseline/Aperture, we can give an approximative calculation of the number of fringes, as it satistfies the **Schuster** Criterion by:

$$N_{(fringes)} = \text{Round}\left(2.44 \times \frac{\text{Baseline}}{\text{Aperture}} \right)$$

The envelope size: $2 \times N_{(fringes)} \times angularsizeoffringes$ The more the number of fringes increases the more we loose contrast, each pairs of fringes take theoretically $\sim 9\%$ from the overall Intensity when performing Modulation Transforming, which are the source of decrease of contrast.

4 FINDING THE ANGULAR DIAMETER OF MARS

4.1 the formula of the fringes visibility

For each object, we assign the quantity called Visibility, denoted V, defined as:

$$V = \frac{I_{max} - I_{min}}{I_{max} + I_{min}}$$

4.2 Computing visibility in Sirius

The visibility should keep invariant in Sirius looking to its far distance, in case if it is not, then we will calibrate our measures of Mars on Sirius as a reference. The Calculations are attached in the Python Code.

4.3 Computing visibility in Mars

All the computations of visibility in Mars are given in the Python code.

4.4 Finding B_0 so that $V(B_0) = 0$

The curve_ fit and the trial to find a guess point for B_0 are described well in the Python code, and the angular diameter as well.

4.5 Main steps for processing fringes

The process of fringes on order to extract out the visibility measurements:

1. We substract the Dark-noise from the picture, the dark-image should be taken with the same exposure time as the pictures (180s) in our experiment.

2. We verify the quality of background reduction manually, and we try to regularize it with adding or substructing little scalar values.
3. We Crop the pictures to a square format in order to prepare them for **IDL**, in our case (850×850) pixels.
4. Organizing them in ordered directories as described in **IDL**.
5. Processing them with **IDL**.

5 PROCESSING OF THE FRINGES

5.1 Van-Citter Zernike theorem

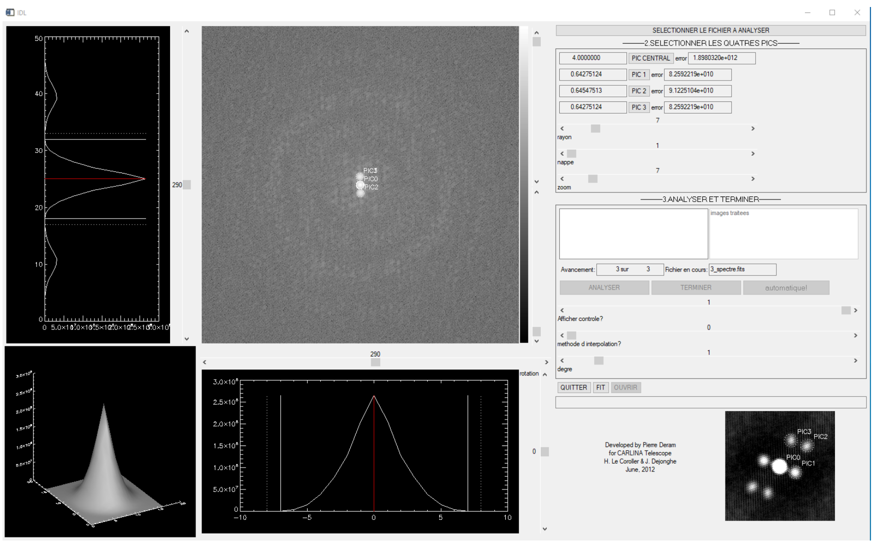
For a uniform disk, the visibility is calculated through Fourrier Transform of the Intensity in Baseline space. Then the result of this transform is always a curve of three peaks such that the value of central peak equals the sum of the two pics on the edges. The cantour integral formula to calculate these 3D peaks is a result of Van-Citter Zernike theorem, it is given as in the following:

$$V(u,v) = \frac{\int\int I(\alpha,\beta)e^{\frac{-2i\pi(\alpha u+\beta v)}{\lambda}}d\alpha d\beta}{\int\int I(\alpha,\beta)d\alpha d\beta}$$

5.2 IDL processing

With the Use of **IDL**, explicitly of widget1 & widget3 routines, we could do the following:

1. **widget1:** Calculates the Fourrier transform for the cropped-images and returns back new images in Intensity space. As a result we get the corresponding peaks plotted in images.
2. **widget3:** Gives us the ability to chose the Integration Cantour coordinates and dimensions around the peaks. Then integrates them, and gives the Modulous square of visibility.



Screen shot of our working on Sirius in **widget3** routine

5.3 Fitting the visibility

The fit of calibrated data and every calculations related to it are provided in the Python code, after the processing in **IDL** we extracted the data from the text-file to the Code.

5.4 Credibility of Results

In the Literature The Angular diameter of Mars is defined between $[3.5'', 25.1'']$, In our experiment, we found it 11.18″ as the data was taken when Mars was sufficiently close, the result is Satisfying.
I should notice that I've tried to access the astronomical ephemerides, but They are not free to use.

6 APPENDIX (PYTHON3 CODE)

6.1 Libraries we need

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy as sc
from scipy import special as sp
from scipy import optimize as op
from scipy import interpolate as ip
from IPython.display import Latex
plt.rcParams['figure.figsize'] = (16,8)
```

6.2 Creating data lists

[2]:
```
Baseline=np.array([8,12,20])
N=len(Baseline)
Mars_squared_visibility=[
    [[0.53702909,0.53986554,0.53986554],[0.49502340,0.49744273,0.49744273],[0.49501966,0.
 ↪49742943,0.49742943]]
    ,[[0.21522365,0.21527814,0.21527814],[0.20380409,0.20631468,0.20631468],[0.
 ↪21200288,0.21179992,0.21179992]]
    ,[[0.014119630,0.014119630,0.014470544],[0.012059569,0.012512654,0.012512654],[0.
 ↪0097094168,0.0097094168,0.0096995073]]]
Sirius_squared_visibility=[[[0.70916327,0.71228920,0.71228920],[0.66915086,0.67190125,0.
 ↪67190125],[0.64275124,0.64547513,0.64275124]]
    ,[[0.47154292,0.47300350,0.47154292],[0.51743026,0.51911452,0.51911452],[0.
 ↪47199861,0.47358245,0.47358245]]
    ,[[0.17976266,0.18152939,0.18152939],[0.11521714,0.11235728,0.11235728],[0.
 ↪14922414,0.14992092,0.14992092]]]
```

[3]:
```
def ROOTED(SQUARED_VISIBILITY):
    #SQUARED_VISIBILITY = np.array()
    ROOTED_VISIBILITY=[]
    for i in SQUARED_VISIBILITY:
        ROOTED_VISIBILITY.append(abs(np.sqrt(i)))
    return ROOTED_VISIBILITY
```

[4]:
```
Mars_visibility= ROOTED(Mars_squared_visibility)
Sirius_visibility= ROOTED(Sirius_squared_visibility)
```

[5]:
```
Mars_visibility
```

[5]:
```
[array([[0.73282269, 0.73475543, 0.73475543],
        [0.70357899, 0.7052962 , 0.7052962 ],
        [0.70357634, 0.70528677, 0.70528677]]),
 array([[0.46392203, 0.46398075, 0.46398075],
        [0.45144666, 0.45421876, 0.45421876],
```

```
        [0.4604377 , 0.46021725, 0.46021725]]),
 array([[0.11882605, 0.11882605, 0.12029357],
        [0.10981607, 0.11185997, 0.11185997],
        [0.09853637, 0.09853637, 0.09848608]])]
```

```
[6]: Sirius_visibility
```

```
[6]: [array([[0.84211832, 0.84397227, 0.84397227],
        [0.81801642, 0.81969583, 0.81969583],
        [0.80171768, 0.80341467, 0.80171768]]),
 array([[0.68668983, 0.6877525 , 0.68668983],
        [0.71932625, 0.72049602, 0.72049602],
        [0.68702155, 0.68817327, 0.68817327]]),
 array([[0.42398427, 0.42606266, 0.42606266],
        [0.3394365 , 0.33519737, 0.33519737],
        [0.38629541, 0.38719623, 0.38719623]])]
```

6.3 Defining Functions

```
[7]: #Mean and standard deviations:
def std_mean_visibility(Lin):
    Lout = []
    Lavg = []
    for i in Lin:
        Lavg.append(np.mean(i))
        Lout.append(np.std(i))
    expression = print('\n The average visibility:' ,Lavg ,'\n','The standard deviation␣
 ↪for each visibility:', Lout)
    return Lavg, Lout, expression
```

```
[8]: #Calibrated standard deviation:
def CALIB_STD(STD_OBJECT,STD_REFERENCE,V_OBJECT,V_REFERENCE):
    CALIBRATED_STANDARD_DEVIATION = []
    for i in range(len(STD_OBJECT)):
        CALIBRATED_STANDARD_DEVIATION.append((V_OBJECT[i]/V_REFERENCE[i])*np.
 ↪sqrt(((STD_OBJECT[i]/V_OBJECT[i])**2)+((STD_REFERENCE[i]/V_REFERENCE[i])**2)))
    return CALIBRATED_STANDARD_DEVIATION

#CHISQUARE for the plot:
def CHISQUARE(N_Baseline, Calibrated_Visibility, Fitted_Visibility, Standard_Deviation):
    CHI=[]
    c1=0
    c2=0
    for i in Calibrated_Visibility:
        c1+=1
        c2=0
        c3=0
        for j in Fitted_Visibility:
            c2+=1
            c3=0
            for k in Standard_Deviation:
                c3+=1
                if ((c1==c2) and (c1==c3)):
                    CHI.append((1/(N_Baseline-1))*(((i-j)**2)/k))
    CHISUM = np.sum(CHI)
    return CHISUM
```

6.4 Results

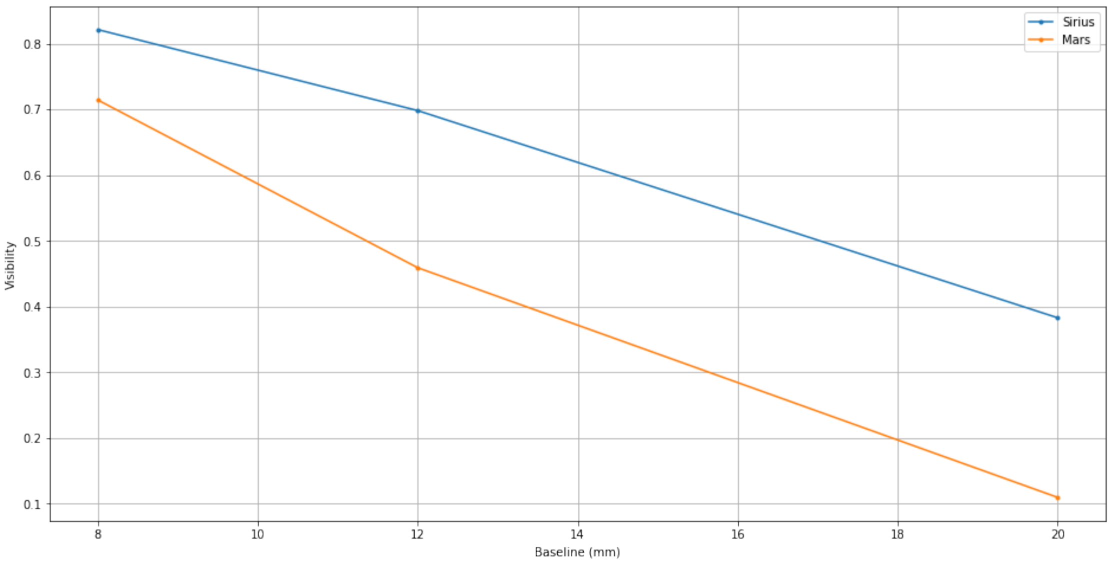
[9]:
```
Mars_Mean,Mars_Standard_deviation,l= std_mean_visibility(Mars_visibility)
Sirius_Mean,Sirius_Standard_deviation,l= std_mean_visibility(Sirius_visibility)
```

```
 The average visibility: [0.7145172018133503, 0.45918221422791944,
0.10967116826893893]
 The standard deviation for each visibility: [0.013880701292633612,
0.004488840863810076, 0.008583806166489417]

 The average visibility: [0.8215912195579255, 0.6983131719098874,
0.3829587449727887]
 The standard deviation for each visibility: [0.016876846427008176,
0.01542211771871696, 0.03636621881759011]
```

6.4.1 Plot of data

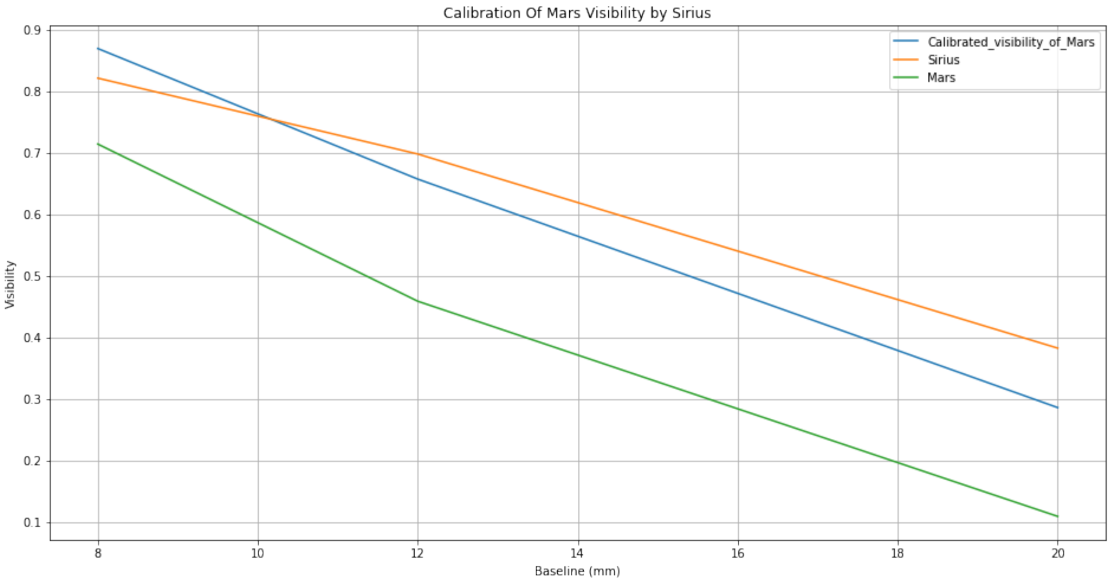
[10]:
```
Sirius_plot= plt.plot(Baseline,Sirius_Mean,'.-',label='Sirius')
Mars_plot= plt.plot(Baseline,Mars_Mean,'.-',label='Mars')
plt.xlabel('Baseline (mm)')
plt.ylabel('Visibility')
plt.legend()
plt.grid()
```



[11]:
```
Calibrated_Visibility=[]
cs=0
cm=0
for i in Sirius_Mean:
    cs+=1
    cm=0
    for j in Mars_Mean:
        cm+=1
        if cs==cm:
            Calibrated_Visibility.append(j/i)
print(Calibrated_Visibility)
```

```
[0.869674827096876, 0.6575591478133737, 0.28637854523137124]
```

[12]:
```
Calibrated_plot= plt.
 ↪plot(Baseline,Calibrated_Visibility,label='Calibrated_visibility_of_Mars')
Sirius_plot= plt.plot(Baseline,Sirius_Mean,label='Sirius')
Mars_plot= plt.plot(Baseline,Mars_Mean,label='Mars')
plt.legend()
plt.grid()
plt.xlabel('Baseline (mm)')
plt.ylabel('Visibility')
plt.title('Calibration Of Mars Visibility by Sirius')
plt.draw()
plt.savefig('123.pdf')
```



6.4.2 *Fitting function*

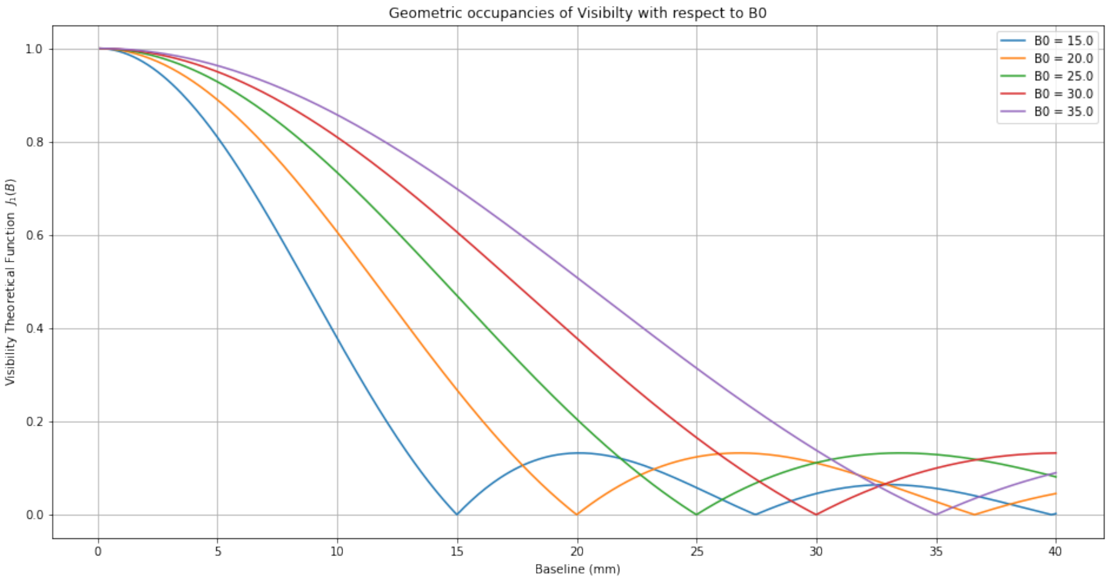
[13]:
```
#B0 parameter
def fit_model_function(B,B0):
    fit_result1= (2*sp.jv(1,B*np.pi*1.22/B0) )/(1.22*np.pi*B/B0)
    FFResult = [abs(number) for number in fit_result1]
    return FFResult
```

[14]:
```
BNew = np.linspace(0,40,401)
```

6.4.3 *Speculation of the Guess prameter for the optimization*

[15]:
```
B0_test = np.linspace(15,35,5)
for i in B0_test:
    plt.plot(BNew,fit_model_function(BNew,i),label=f"B0 = {i}");
plt.legend()
plt.title('Geometric occupancies of Visibilty with respect to B0')
plt.xlabel('Baseline (mm)')
plt.ylabel(r'Visibility Theoretical Function  $J_{1}(B)$')
plt.grid()
plt.draw()
plt.savefig('Guessing_the_guess.pdf')
```

```
<ipython-input-13-397a439bfcda>:3: RuntimeWarning: invalid value encountered in
true_divide
  fit_result1= (2*sp.jv(1,B*np.pi*1.22/B0) )/(1.22*np.pi*B/B0)
<ipython-input-13-397a439bfcda>:3: RuntimeWarning: invalid value encountered in
true_divide
  fit_result1= (2*sp.jv(1,B*np.pi*1.22/B0) )/(1.22*np.pi*B/B0)
<ipython-input-13-397a439bfcda>:3: RuntimeWarning: invalid value encountered in
true_divide
  fit_result1= (2*sp.jv(1,B*np.pi*1.22/B0) )/(1.22*np.pi*B/B0)
<ipython-input-13-397a439bfcda>:3: RuntimeWarning: invalid value encountered in
true_divide
  fit_result1= (2*sp.jv(1,B*np.pi*1.22/B0) )/(1.22*np.pi*B/B0)
<ipython-input-13-397a439bfcda>:3: RuntimeWarning: invalid value encountered in
true_divide
  fit_result1= (2*sp.jv(1,B*np.pi*1.22/B0) )/(1.22*np.pi*B/B0)
```



7 CALCULATIONS

[16]:
```
B0, Y0 = op.curve_fit(fit_model_function, Baseline, Calibrated_Visibility, p0=25)
B0, Y0
```

[16]:
```
(array([27.02728182]), array([[0.27422126]]))
```

[17]:
```
FY= fit_model_function(np.array(Baseline),B0)
FY
```

[17]:
```
[0.8475189467747707, 0.6791529359784151, 0.2803456530475009]
```

[18]:
```
FY1= fit_model_function(BNew,B0)
```

```
<ipython-input-13-397a439bfcda>:3: RuntimeWarning: invalid value encountered in
true_divide
  fit_result1= (2*sp.jv(1,B*np.pi*1.22/B0) )/(1.22*np.pi*B/B0)
```

[19]:
```
C0=.07073068*np.ones(120)
```

[20]:
```
CALIBRATED_MARS_STANDARD_DEVIATION =␣
 ↪CALIB_STD(Mars_Standard_deviation,Sirius_Standard_deviation,Mars_Mean,Sirius_Mean)
CHISQUARED_DATA= CHISQUARE(N,Calibrated_Visibility,FY,CALIBRATED_MARS_STANDARD_DEVIATION)
```

[21]:
```
CHISQUARED_DATA
```

[21]: 0.025179109555519508

[22]:
```
Calibrated_Visibility,FY
```

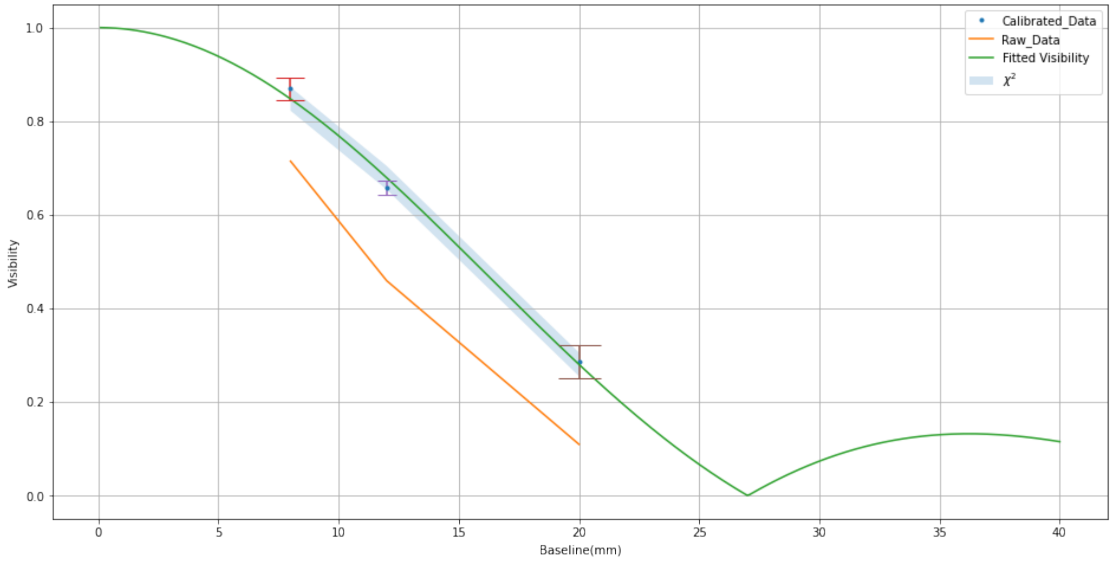
[22]:
```
([0.869674827096876, 0.6575591478133737, 0.28637854523137124],
 [0.8475189467747707, 0.6791529359784151, 0.2803456530475009])
```

[23]:
```
CALIBRATED_MARS_STANDARD_DEVIATION
```

[23]: [0.02458821396706298, 0.01588116240937992, 0.03524155100242703]

8 PLOTTING THE RESULT OF OUR FIT

[24]:
```
plt.plot(Baseline,Calibrated_Visibility,'.',label='Calibrated_Data')
plt.plot(Baseline,Mars_Mean,label='Raw_Data')
plt.plot(BNew,FY1,label='Fitted Visibility')
plt.fill_between(Baseline, FY-CHISQUARED_DATA*np.ones(3), FY +CHISQUARED_DATA*np.
 ↪ones(3), alpha=0.2, label=r'$\chi^{2}$')
plt.errorbar(Baseline[0],Calibrated_Visibility[0],␣
 ↪yerr=CALIBRATED_MARS_STANDARD_DEVIATION[0], capsize=␣
 ↪500*CALIBRATED_MARS_STANDARD_DEVIATION[0])
plt.errorbar(Baseline[1],Calibrated_Visibility[1],␣
 ↪yerr=CALIBRATED_MARS_STANDARD_DEVIATION[1], capsize=␣
 ↪500*CALIBRATED_MARS_STANDARD_DEVIATION[1])
plt.errorbar(Baseline[2],Calibrated_Visibility[2],␣
 ↪yerr=CALIBRATED_MARS_STANDARD_DEVIATION[2], capsize=␣
 ↪500*CALIBRATED_MARS_STANDARD_DEVIATION[2])
plt.legend()
plt.grid()
plt.xlabel('Baseline(mm)')
plt.ylabel('Visibility')
plt.draw()
plt.savefig('fitted_plot.pdf')
```



8.1 Calculation of Thetha

[25]:
```
llambda = 1.2*10e-6 # micro m
index_min=np.nanargmin(FY1)
B_min = BNew[index_min]
Theta =1.22*llambda/(B_min*10e-3) # mm
Theta
```

[25]: 5.422222222222222e-05

8.1.1 Theta in arcseconds

[26]:
```
Theta_arcsec = Theta*(3600*180)/np.pi
Theta_arcsec
```

[26]: 11.18413616095367

9 BIBLIOGRAPHY

- **Web pages :**
  1. http://www.obs-hp.fr/guide/t80.shtml
  2. http://www.obs-hp.fr/guide/80.html
  3. http://www.obs-hp.fr/www/guide/camera-80.html
  4. https://diffractionlimited.com/product/stxl-6303/
  5. https://quantumimaging.com/knowledge-base/
  6. https://cdn.eso.org/images/screen/eso0020b.jpg