IVC

Interferometric Visibility Computations

Version 0.9.2 (12th July 2006)

G. Li Causi, INAF - OAR

An IDL software tool for
interferometry simulations and model fitting

User Manual

Dr. Gianluca Li Causi
INAF – Osservatorio Astronomico di Roma
licausi@mporzio.astro.it

Table of Context

Abbreviations used in this manual

IVC	Interferometric Visibility Computations, i.e. this software
IDL	Research Inc. Interactive Data Language
IDL-VM	Research Inc. IDL Virtual Machine, i.e. the free tool to run IDL programs outside IDL
IDL-DE	IDL Development Environment, i.e. the graphical framework of IDL
GUI	Graphical User Interface
FoV	Field of View
SED	Spectral Energy Distribution
ETC	Exposure Time Calculator
FFT	Fast Fourier Transform

Version-related items

--- IMPORTANT INFORMATION ---

Current version V0.9.2 of IVC is ***not yet a distribution version*** , but a test-level one which comes with ***limited functionalities*** and ***no warranty for scientific applications*** .

All what described in this User Manual should work properly in current version, with the exception of the following items:

- ***Subroutine -mode (sec. 2.2, a) is not yet tested (foreseen for Version 1.0);***
- ***the possibility to call IVC from other languages (sec. 2.2, b) has not yet been implemented;***
- ***speed is not optimized in current version (foreseen for Version 1.0);***
- ***the current version is not given in source code (foreseen for Version 1.0).***

Lots of minor improvements have been already planned and will be implemented while new versions will be released.

Any user's suggestion or bug notice will be greatly appreciated in this test phase of the software.

--- IMPORTANT INFORMATION ---

Versions history (inverse chronology)

Version 0.9.2:

- Released on 12th July2006 at web site http://www.mporzio.astro.it/~licausi/IVC/
- Bug fixed: GUI now works fine also under Unix/Linux;
- Added: `TOY MODELS TOOL` and updated this manual;
- Bug fixed: `GUI` now enters into a 1024x768 screen;
- Bug fixed: multi-baseline chi-squared computation now working;
- Bug fixed: `Mark at Baseline` slider is now editable;
- Added: `baseline number` tag in observations ASCII files and updated this manual;
- Improved: error checks on inputs and sampling;
- Improved: code relative to sampling has been revised;
- Bug fixed: error in plots titles: baseline 1.0m;
- Improved: some menu titles changed for better understanding;

Version 0.9.1:

- Released on 23th May 2006 at web site http://www.mporzio.astro.it/~licausi/IVC/
- Added: `ALL MODELS AND OBSERVATIONS` tab and updated this manual;
- Added: `CHI SQUARED VS MODEL` tab and updated this manual;
- Added: MIDI and AMBER instrument parameters example files plus Appendix 2 documentation in this manual;
- Bug fixed: maximum value of `Mark at Baseline` slider;
- Bug fixed: error in mark positioning with `Mark at Baseline` slider;
- Bug fixed: `Baseline` drop list now display current baseline number;
- Bug fixed: error in cut of FoV for .fits input, now the cut is always centered
- Bug fixed: errors when using a single baseline

Version 0.9:

- Released on 8th May 2006 at web site http://www.mporzio.astro.it/~licausi/IVC/

1. Introduction

IVC, which stands for “Interferometric Visibility Computations”, is a software tool for computing the interferometric visibilities of a light source model and for matching these simulations with real observation data.

It has been designed to be user-friendly and to serve as a generic tool for the user who wants to perform interferometry simulations of his own models on a given instrument, or needs to test his models against interferometric observations and non-interferometric ones simultaneously.

Within IVC the user will provide his own grid of models as 1-D profiles or 2-D images for a set of wavelengths, thus describing their Spectral Energy Distributions (SEDs).

A specific Toy Model Tool is also available to create a model grid out of simple geometries.

Regarding the instruments, only a restricted set of instrument descriptions will be available in the software distribution (i.e. MIDI and AMBER at VLTI), leaving the user free to insert custom description of the instrument, modes and observing conditions to simulate.

IVC can be used both as a Graphical User Interface (GUI), or as a Subroutine to be called from within the user’s radiative transfer code, as clarified in this manual.

2. What is it for ?

Within the framework of the growing interest of the italian astronomical community in the usage of interferometric facilities (expecially the VLTI) and facing the lack of available software tools to compute interferometric observables by interfacing with user's modelling softwares, we have developed IVC at the Rome Astronomical Observatory.

IVC finds its place aside the most commonly used softwares for the computation of interferometric data, namely the *VisCalc* and *CalVin* web pages from ESO[1] and the *ASPRO* software from JMMC[2].

These programs are effective in providing various results, but they have three major drawbacks: *i) VisCalc* does not allow the user to insert the Spectral Energy Distribution (SED) of his own model; *ii)* them both cannot receive a grid of user's models as input, so that to perform model fitting among hundreds of models becomes unpractical; *iii)* they cannot be calles as sub-routines within the user's modelling code in order to perform simultaneous fitting on different data, e.g. a double fit on interferometric visibilities and spectral observations together.

IVC has been designed specifically to overcome these difficulties, but not to duplicate the work of the ESO and JMMC tools. So that IVC is expected to input user's models with SEDs and perfrom computations and fitting on interferometric data, as well as to be called from within the user's code for multiple fitting, but it is NOT supposed to compute instrument-specific baselines coverages, to calculate target's observability, to select calibrators, to make ETC simulations, etc.

[1] European Southern Observatory, Garching b. Munchen, Germany
[2] Jean-Marie Mariotti Center, Observatoire de Grenoble, France

3. Installation and Run

IVC is written in IDL (Interactive Data Language), but it comes also pre-compiled to run outside the IDL framework thanks to the free IDL Virtual Machine (IDL-VM, available for free download at the IDL homepage http://www.rsinc.com/idlvm/).

Hereafter the installation and usage of the program is described for both GUI-mode and Subroutine-mode, launched from within an IDL framework or from outside.

3.1 GUI-mode

Either ***IDL*** or ***IDL-VM version 6.2 or later*** must be installed on your system to properly run IVC in GUI-mode.

You can run IVC from inside the IDL framework as explained in a) or use the IDL-VM stand-alone version as explained in point b) as follows:

a) Within IDL :

Do the following steps:

- unzip the **`IVC.zip`** file in a directory of your choice;
- launch IDL;
- restore the compiled routines by writing at the IDL prompt:

```
RESTORE, 'start.sav', /ROUTINES
```

then launch it simply writing the word "**`start`**" and pressing enter.

The program will automatically recognize its source directory, so that no manual editing of the IDL path or working directory are needed.

b) Outside IDL :

If you don't have an IDL licence, you can use the stand-alone version of IVC by just installing the IDL-VM on your computer.

Do the following steps:

- download and install the IDL Virtual Machine from http://www.rsinc.com/idlvm/ ;
- unzip the **`IVC.zip`** file in a directory of your choice;
- launch the program by just double click on the **`start.sav`** file.

3.2 Subroutine-mode

a) IDL users :

One of the most relevant properties of IVC is the possibility to be called from within any user's IDL code, just like an IDL procedure.
Thanks to this capability, the user can include constraints to interferometric observations to his own source modelling code, as for example if he had a program which fits a model to spectroscopic data and wanted to add a simultaneous fitting to interferometric ones.
More, the Subroutine-mode reveals its utility when huge sets of models are involved and computation speed is required: the final result will be returned instead of wasting time by visualizing any output in a GUI.
See section 6.1 for the subroutine call syntax and parameters.

b) Non IDL users :

The Subroutine-mode can be called also by codes written in other languages: TBD

4. Program description

4.1 The GUI

In Figure 1 a screenshot of the GUI is presented, where we can recognize the following scheme:

a) **Command Menu**: it collects the commands to load model data and to set instrument's parameters, as well to launch a computation or to exit the program.

b) **Graphical Tabs**: the various graphical outputs are organized into a number of tabs that the user can select by clicking on the respective labels: both input models and output visibilities are displayed here.

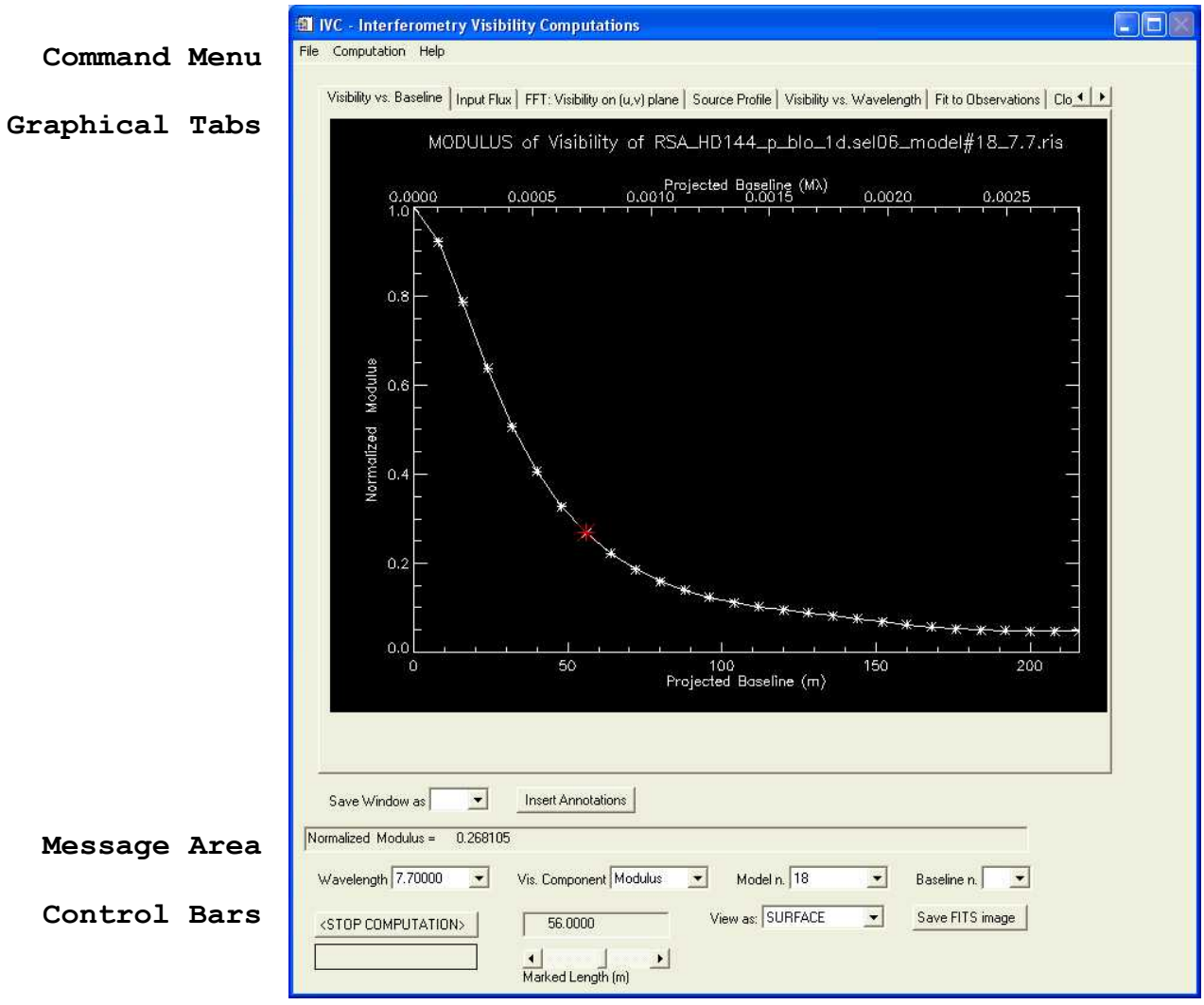

Figure 2: A screenshot of the GUI of the IDL Interferometry Visibility Calculator, with decription labels (see text).

c) **Control Bars**: here a series of drop lists and buttons allow the user to browse the various wavelengths and models in the input and output graphs, or to analyse how the visibilities change as a function of baseline or spectral channel. The graphics will update in real time at any user's intervention.

A further buttons bar is provided for user's convenience to add annotations and to save the plots for user's reporting.
At the right side of the Control Bar, a progress bar will show in real time the progress percentage of the active tasks.

d) **Message Text Area**: finally, a textual message line is used by the program to write temporary results and process information.

4.2 Input of the Data

To start working, IVC needs one or more source models to be provided as input. It accepts two kind of input data (Figure 3):

i) **2-D images**, which represents the source intensity distribution on the sky, ***I(x,y)*** for any wavelength;
ii) **1-D intensity profiles**, ***I(r)*** for any wavelength, if the source has a circular symmetry.

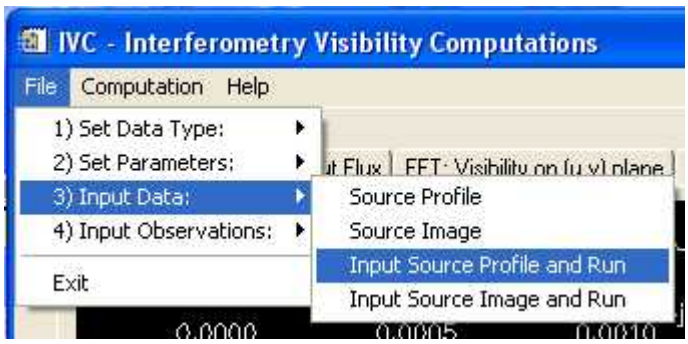

Figure 3: The FILE menu offers both Image and Profile input.

A very minimal set of rules, described hereafter, must be satisfied by the data format in both cases, so that it is very easy for the user to submit a model for computation.

4.2.1 Intensity Profiles

The 1D intensity profiles must be ASCII files with the format shown in Tab. 1: a short header must be present where the tags **`lambda=`** and **`model=`** must be present (and set to the current wavelength and model identification, respectively), while any comment should preceded by a **#** sign; the **`lambda`** keyword must be expressed in ***micron*** .

header:

```
# Lambda in micron:
lambda = 10.5

# Model ID:
model = 01

# Any other comment
# .......
# .......
# .......
```

data:

```
0.316888 6.583E+10  other_data    other_data .....
0.362316 6.583E+10  other_data    other_data .....
0.414392 6.583E+10  other_data    other_data .....
0.473947 6.583E+10  other_data    other_data .....
0.542089 6.583E+10  other_data    other_data .....
0.620203 6.583E+10  other_data    other_data .....
........ .........  ..........    ................
```

Tab. 1: Format of the input ASCII file containing the Intensity Profile.

Past the header, there are the user's model data ordered by columns: the only requirements here is that two of them must contain the **radius** ***r*** and the **intensity profile** ***I(r)*** and that the radius ***r*** is given in units of ***arcsec*** (previous conversion from different units are up to the user).
These two can be any couple of columns, irrespectively of their locations, which will be specified once and for all by the user at the first reading of the data.

For 1-D intensity profiles, both regularly or irregularly spaced ***r*** coordinate values are accepted.
The only requirement is that the header length, in number of rows, must be the same for at least all the files representing the same kind of data, e.g. all the files of a grid of models to use for a fit to observations. If not, just make all the header's rows, but the tags, begin with the **#** sign.

At the first access to a new kind of ASCII file, the user will be asked to define a template for that format, by means of a dialog panel, or he can load an existing template suitable for the selected file type.

A SED can be given by giving one ASCII file for each wavelength channel, thanks to the multiple files selection capability of the input panel.

You can practise with some example model images given under the **`/Sample_Models_1D_Profiles`** directory in the IVC distribution.

4.2.2 Images

In the case of 2D images, they must be provided in the form of FITS files, where the following keywords must be present in the header (if not present the user will be asked for editing them on each file):

`CDELT1:` X Pixel Scale (in **`arcsec/pixel`**)
(Y Scale is assumed equal to X Scale)
(pixel scale is assumed uniform across the image)

LAMBDA: Effective Wavelength (in **microns**)
MODEL: Model ID

The pixel scale must be given in ***arcsec per pixel*** ; previous conversions from different units are up to the user.
A SED can be given by giving one FITS file for each wavelength channel, thanks to the multiple files selection capability of the input panel.

You can practise with some example model files given under the **/Sample_Models_2D_Images** directory in the IVC distribution.

4.3 Instrument Description

Different interferometric instruments have different properties and observational modes, e.g. different shapes and sizes of the Field of View (FoV), different possible baselines and various wavelength channels. All these parameters can be edited via the Instrument Parameters Editor (Figure 4).
In particular, the user can specify whether the focal plane aperture is ***circular***, giving its radius in arcsec, or is a ***slit***, giving its width, length and position angle.
More, a ***Gaussian FoV*** can be defined, i.e. having a Gaussian illumination instead of a flat one, and so on.

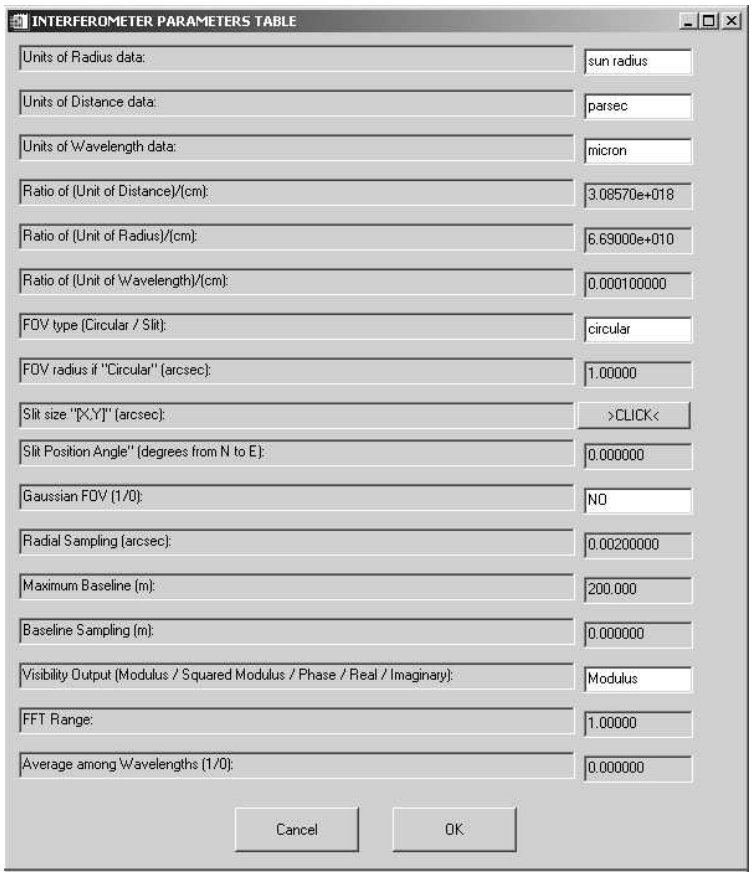

Figure 4: The Instrument Parameters Editor panel.

The values of Instrument Parameters are stored in an ASCII file with conventional **`.cfg`** extension (showed and explained in Appendix 2) and can also be changed by manually editing this file.
Once changed, the user can save the current Instrument Parameters in a .cfg file by the command menu **`Computation > Save Instrument Parameters`**.

Two example Instrument Parameters files are given with the IVC distribution: a VLTI-MIDI with UT1-UT2 instrument description (file: **`MIDI_UT1-UT2.cfg`**) and a VLTI-AMBER example used with UT1-UT2-UT4 (file: **`AMBER_UT1-UT2-UT4.cfg`**).

4.4. Interferometric Computations

Once the input files are selected, the computation starts by first classifying them according to the header tags.
An outer cycle, spanning over the models, and two inner ones, spanning over the available wavelengths defined for each model and over the defined baselines directions, will drive the following operations:

1. if the input is a 1-D profile ***I(r)***, defined on whatever ***r***-sampling, build a flux-conserving uniformly-pixeled 2-D revolution surface ***I(x,y)***, else, if it is an image, use the sampling defined in the FITS file;

2. cut the 2-D intensity distribution with the defined, circular or slit, FoV;

3. if a Gaussian FoV has been defined, multiply the 2-D distribution by a normalized Gaussian with a sigma equal to the FoV radius.

4. compute the complex visibility in the ***(u,v)*** plane by means of 2-D Fast Fourier Transform (FFT) of the intensity distribution;

5. extract the 1-D Visibility vs. Baseline profile ***V(B)*** along the projected baseline direction.

Finally the various outputs are stored and displayed in the GUI, and the steps are repeated for each wavelength within each selected model.

Each time the **`Computation > Run Computation`** command is selected the whole interferometric computation starts again with the last selected parameters.

5 The meaning of FoV in interferometry

It's worth nothing to clarify at this point what the phrase "Field of View" actually means when referred to interferometric instruments.
There are two different meanings in the literature for it, which the various authors refer to, sometimes rising confusion to the not used reader: the so called ***interferometric FoV*** , and the so called ***detector FoV*** .

- The ***interferometric FoV*** is defined by the following formula:

$$FoV_{INTERF} = \frac{OPD_{RANGE}}{B} \approx \Delta_{COER}$$

where:
B is the baseline length,
Δ_{COER} is the coherence length of the wave packet in the interferogram,
OPD_{RANGE} is the length of optical path difference scanned to form the interferogram.

This quantity represents an angle on the sky which depends on the length of the scanned optical path difference and which actually corresponds to some $\lambda_0/\Delta\lambda$ times the baseline resolution λ_0/B (λ_0 is the central wavelength and $\Delta\lambda$ is the band width of a single spectral channel of the instrument) and it is typically of ***some tenths of milliarcsecs*** .

- The ***detector FoV*** is defined by a circular or slit diaphragm in the optics (or by a fiber core) placed at the telescope focal plane to delimit the observation area, which size is of the order of the Airy disk's radius of a single telescope: λ_0/D (where ***D*** is the telescope diameter) and it is typically of the order of ***one arcsec*** .

The difference among them is that, while the ***detector FoV*** selects all the flux which effectively makes the interference, the ***interferometric FoV*** just tells what portion of the former is actually scanned to form the registered interferogram.
This explains why in the data reduction of some interferometric observations (e.g. with MIDI@VLTI) a so called ***visibility correction*** is usually applied to compensate the measured visibilities for the so called ***incoherent flux*** , i.e. the total flux within the area between the small ***interferometric FoV*** and the wide ***detector FoV*** , flux that is lost outside the scanned part of the OPD:

$$V^2_{corrected} = V^2\left(\frac{1}{1+10^{\Delta m/2.5}}\right)^2$$

where:
V is the measured visibility amplitude,
$V_{corrected}$ is the corrected visibility amplitude,
Δm is the magnitude difference between the flux in the FoV_{INTERF} to $FoV_{DETECTOR}$ annulus and the flux inside the FoV_{INTERF} area.

Interferometric simulations like IVC instead, which are based on synthetic models of the source intensity distribution, computes the visibility of the entire ***detector FoV*** as all the flux passing it will interfere within the instrument.

This means that the simulated visibilities computed by IVC can be put in comparison only with observations ***already corrected for the lost incoherent flux*** .

6. GUI usage

Hereafter we describe how to practically use the GUI menus and buttons to perform an interferometric computation or an observation fitting.

6.1 Basic model simulations

Once the program has been launched in GUI-mode, just follow the numbered steps in the menu **`File`** in order to start a new computation:

1) **`Set Data Type`**: Skip this step if your input is 2-D images, else:

- if this is the ***first time*** you access your input, select **`New Data Type`** and follow the instructions to make a new template for your model ASCII files;

- if not, select **`Load Data Type`** and choose the right template file for your model ASCII files.

2) **`Set Parameters`**: - select **`Load Instrument Parameters`** and choose the **`.cfg`** instrument parameters file of your interest.
If your want to define a ***new instrument*** , select the file **`IVC_Default_Instrument.cfg`**.

The ***Instrument Parameters Editor*** (Figure 4) will pop up: make your modifications, if any, and confirm with the **`OK`** button;
Then select **`Save Instrument Parameters`** and write the **`.cfg`** filename of your choice.

3) **`Input Data`**: - if your models are given as 1D Intensity Profiles ***I(r)***, choose **`Input Source Profile and Run`**, then select one or more ASCII files from your models' folder.

- if your models are given as 2D Images ***I(x,y)***, choose **`Input Source Image and Run`**, then select one or more FITS images from your models' folder.

Upon selection the program will start the computation as described in section 3.4.

A number of plots will be drawn, for each model and wavelength, in the respective **`Graphic Tab`**, as extensively described in the following section.

6.2 Output Plots

`Tab "Visibility vs. Baseline":`

1-D Normali zed Visibility as function of ***Projected Baseline*** (expressed both in ***meters*** and in ***Mλ*** units). The **`Modulus, Squared Modulus, Real Part, Imaginary Part`** or **`Phase`** component of the Visibility is drawn on y-axis, depending on the **`MODE`** is selected in the **`Control Bar`**.

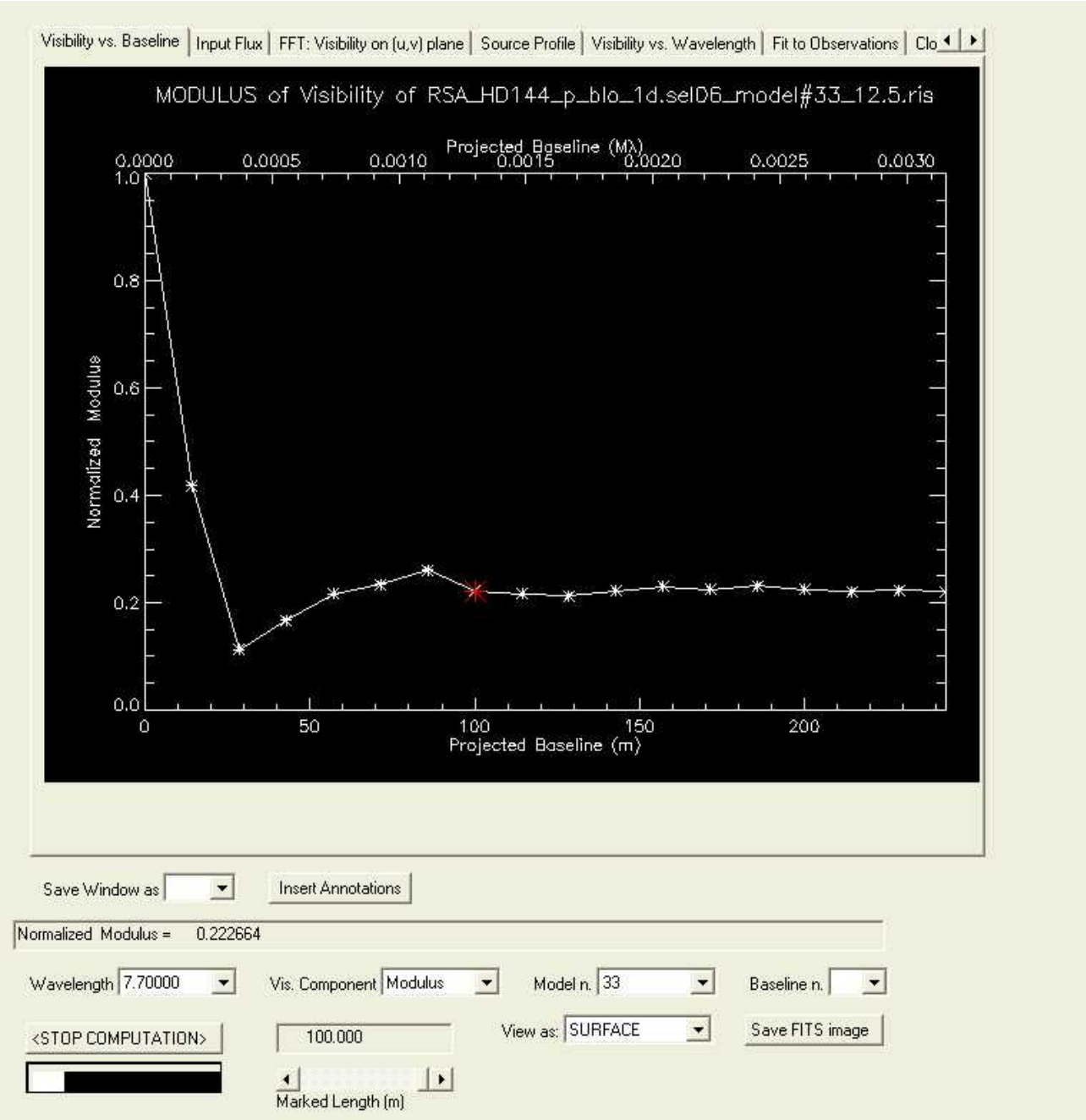

Figure 5: Visibility vs. Baseline plot.

Tab "Input Flux":

2-D Intensity Distribution within the defined FoV (the input 2D image, or revolution surface computed with uniform sampling from the input 1D profile).
The button **View as Image / View as Surface** is available to switch between **IMAGE** and **SURFACE** visualizations (the projected baseline direction is over-imposed on 2D views). The **Generate FITS** button will writes a FITS file with the 2D intensity distribution.

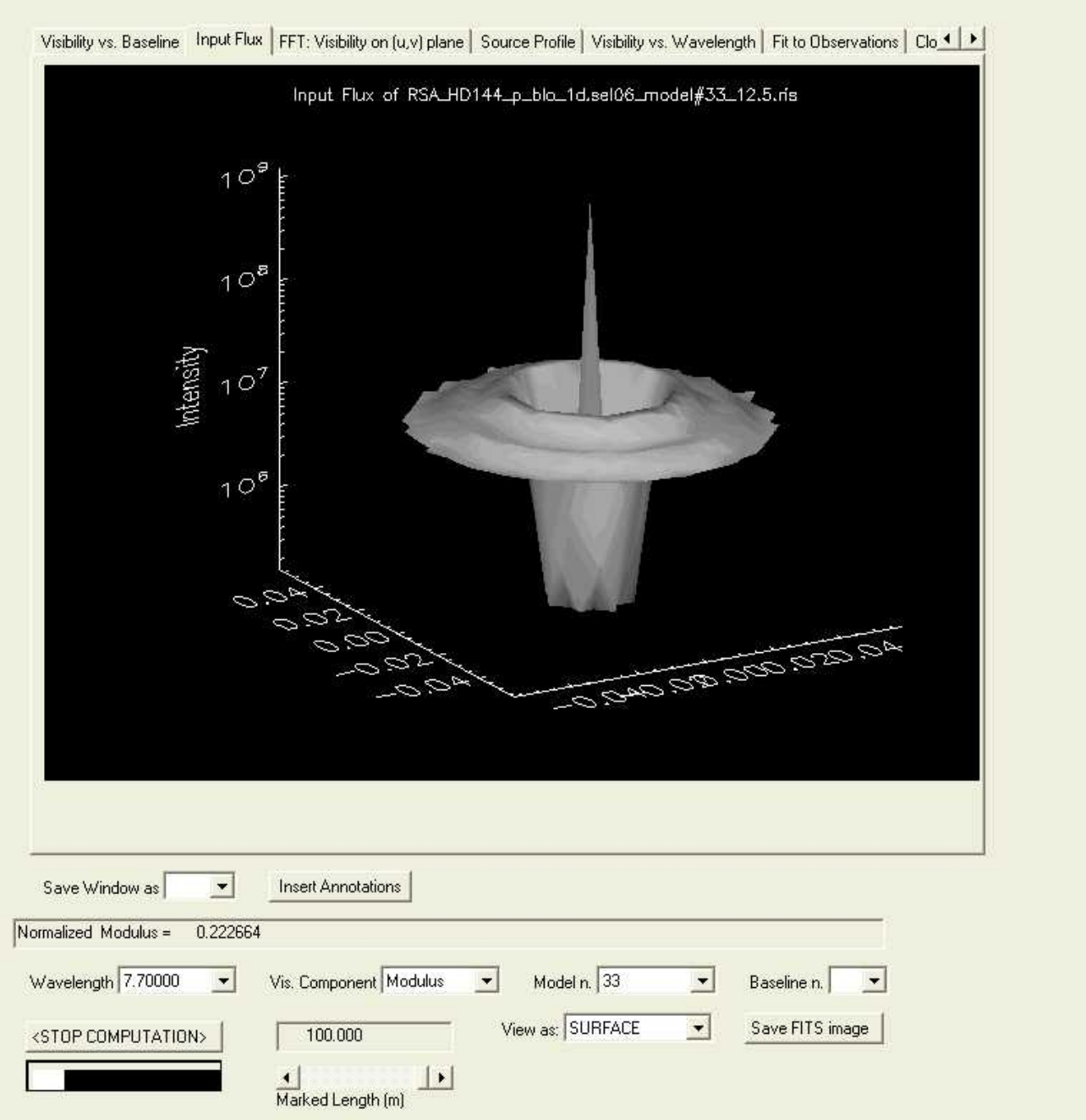

Figure 6: Input Flux displayed as SURFACE (see "View as" drop list value).

Tab "FFT: Visibility on (u,v) plane":

2-D Visibility on the (u,v) plane: the **Modulus, Squared Modulus, Real Part, Imaginary Part** or **Phase** component of the Visibility is drawn on z-axis, depending on the **MODE** selected in the **Control Bar**.
The button **View as Image / View as Surface** is available to switch between **IMAGE** and **SURFACE** visualizations (the projected baseline direction is over-imposed on 2D views). The **Generate FITS** button will writes a FITS file with the 2D Visibility distribution.

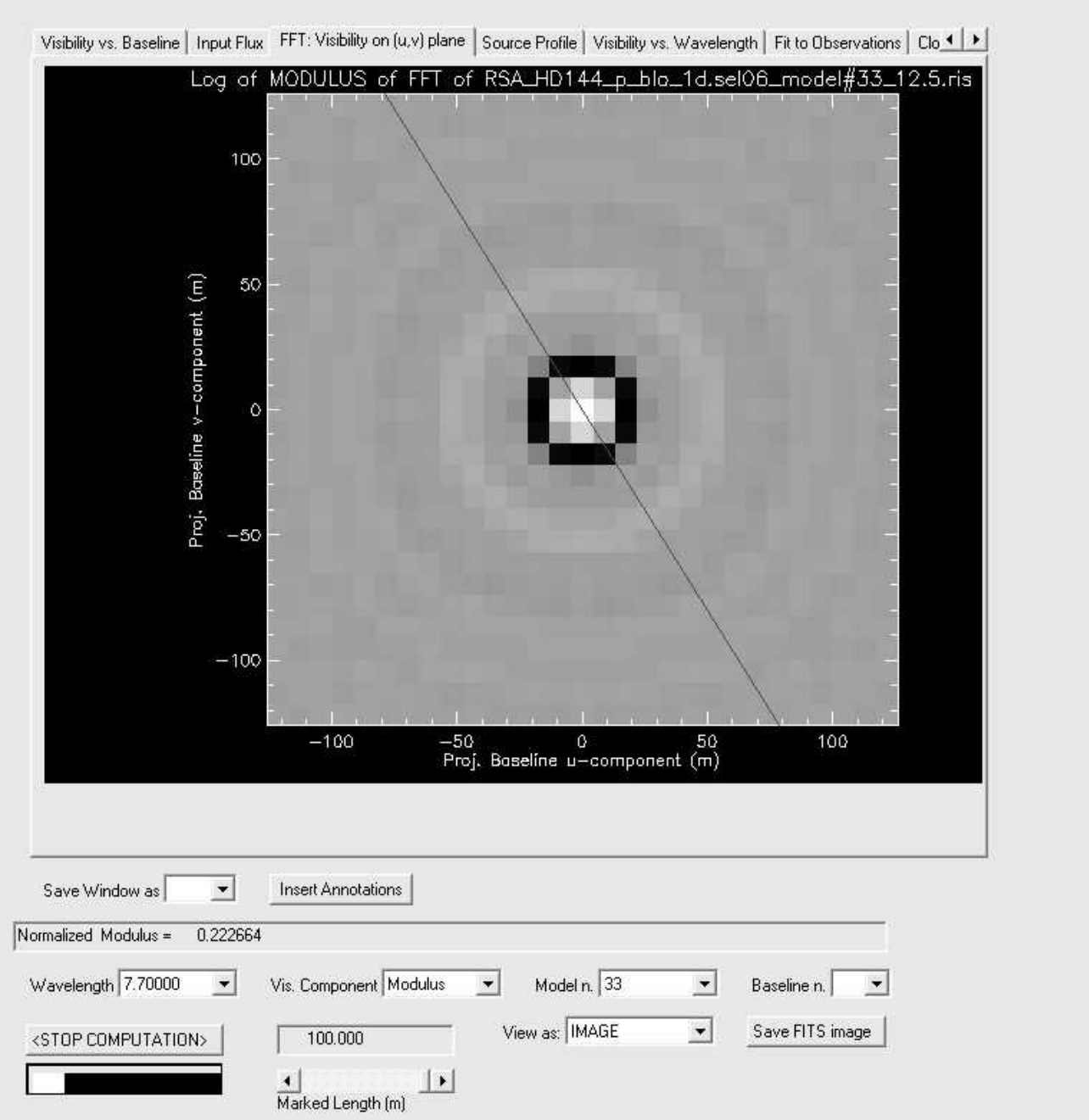

Figure 7: Visibility MODULUS displayed as IMAGE
(see "Vis. Component" and "View as" drop lists values).

Tab "Source Profile":

1-D Intensity Profile and ***1-D Projected Intensity Profile*** , both computed along the projected baseline direction.
The ***Intensity Profile*** is just a central section of the ***2-D Intensity Distribution*** along the projected baseline direction, while the ***Projected Intensity Prof ile*** is the projection of all the flux within the FoV along the same direction. The latter is the effective profile which affect the Visibility results.

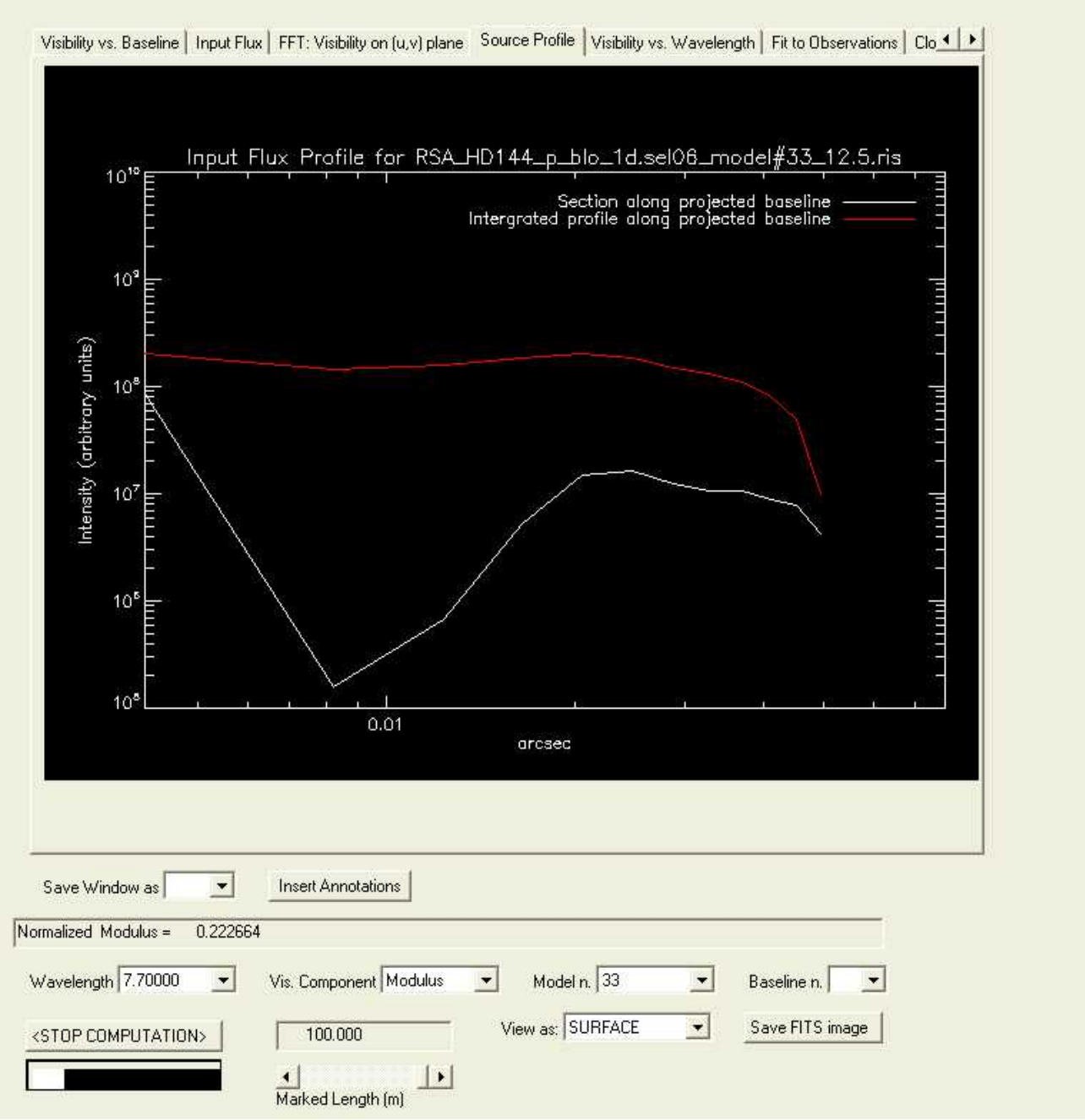

Figure 8: Source Profile of Model n. 33 at 7.7 micron along baseline direction (white = section profile, red = projection profile).

Tab "Visibility vs. Wavelength":

If a non-zero **Baseline Length** has been marked in the **Control Bar** and if more than one wavelength is available for the current model, this tab will show the **MODE**-selected visibility component computed for the **Marked Baseline** value as a function of the wavelengths.
Any change to the baseline's **Marked Length** updates the plot in real-time, thus the Spectral Visibility behaviour can be quickly examined for various baseline lengths.

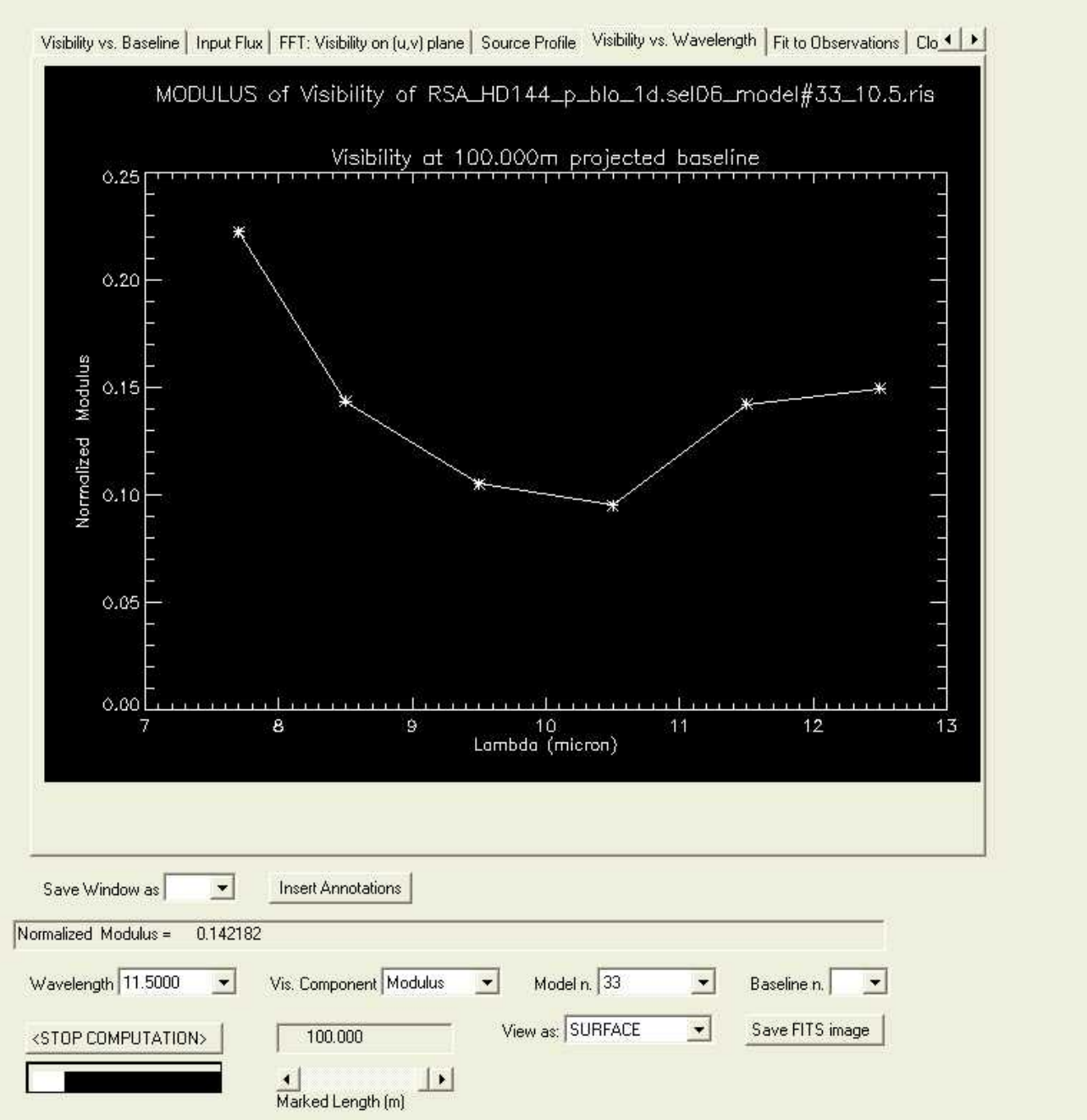

Figure 9: Visibility Modulus at 100m projected baseline plotted vs. wavelength.

Tab "Fit to Observations":

If some Observation Data have been selected in **File > Input Observations** and the selected Visibility Component is the same as in these data, a comparison between observations and current model is displayed here and the corresponding weighted χ^2 value is printed in the **Message Text** line.

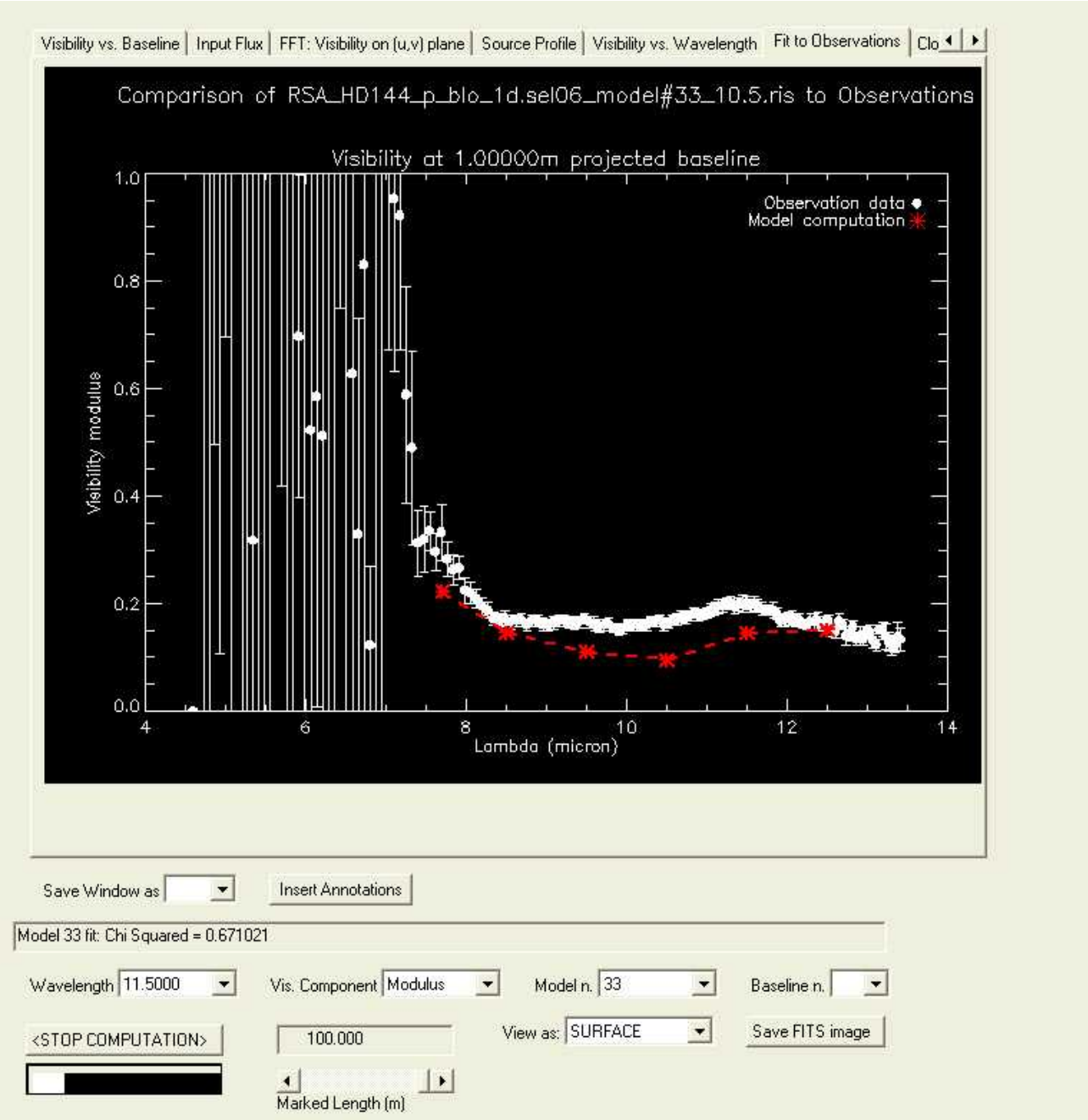

Figure 10: Visibility Modulus at 100m projected baseline vs. wavelength compared to Observation data (notice the corresponding χ^2 value on the Massage Text line).

Tab "Closure Phase":

If three baselines are specified in the **Intrument Parameters** configuration, the **Closure Phase** of the model is computed for each wavelength and displayed in this tab. A ***null*** or $\pm\pi$ closure phase value represents point-symmetric model geometries.

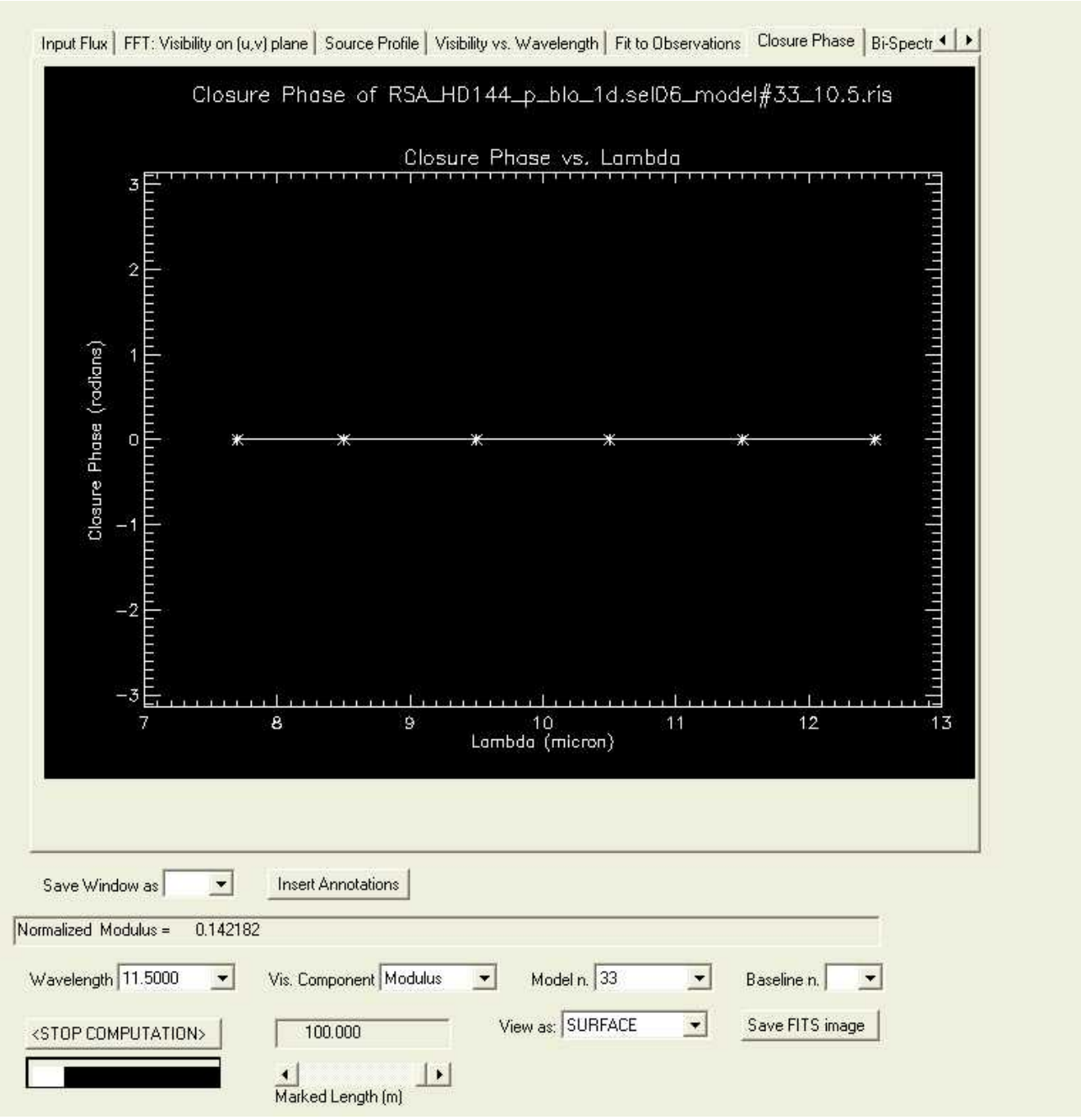

Figure 11: Closure Phase for the three baselines plotted vs. wavelength: here the closure phase is always zero, confirming that Model n. 33 is indeed point-symmetric.

Tab "BiSpectrum":

The same values of the **Closure Phase** tab is also plotted here in the complex plane representaion, the so called **Bi-Spectrum**, where each closure phase point is represented by a vector.

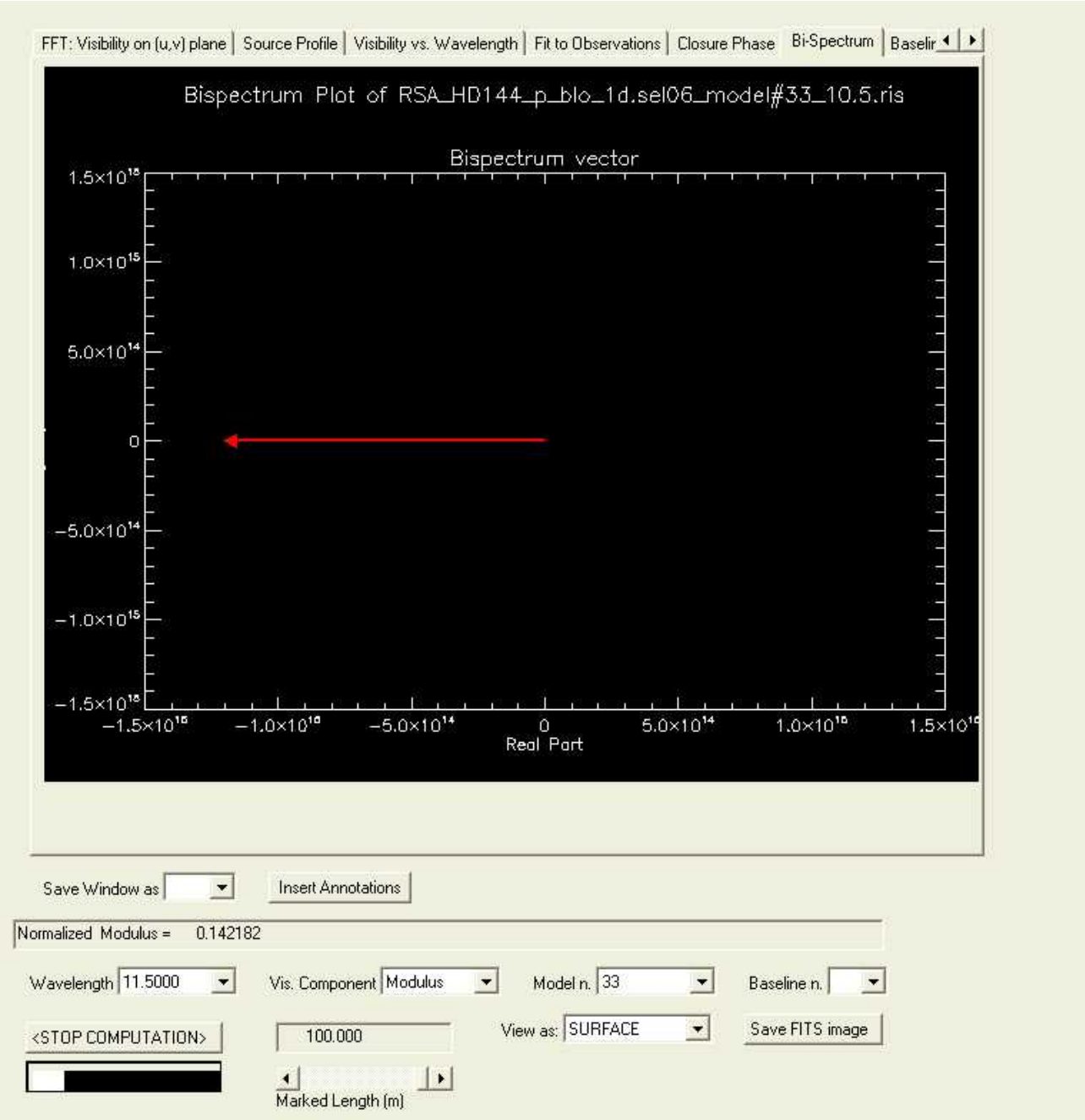

Figure 12: Bi-Spectrum vector representing the same data as the Closure Phase plot.

Tab "Baselines Geometry":

This tab shows an isometric display of the geometry of the three baselines as they appears when projected on the target's tangent plane in the sky.
It is very useful to over-impose plots like this on an image of the target when planning interferometric observations, in order to check any relation among the projected baselines and the target's features.

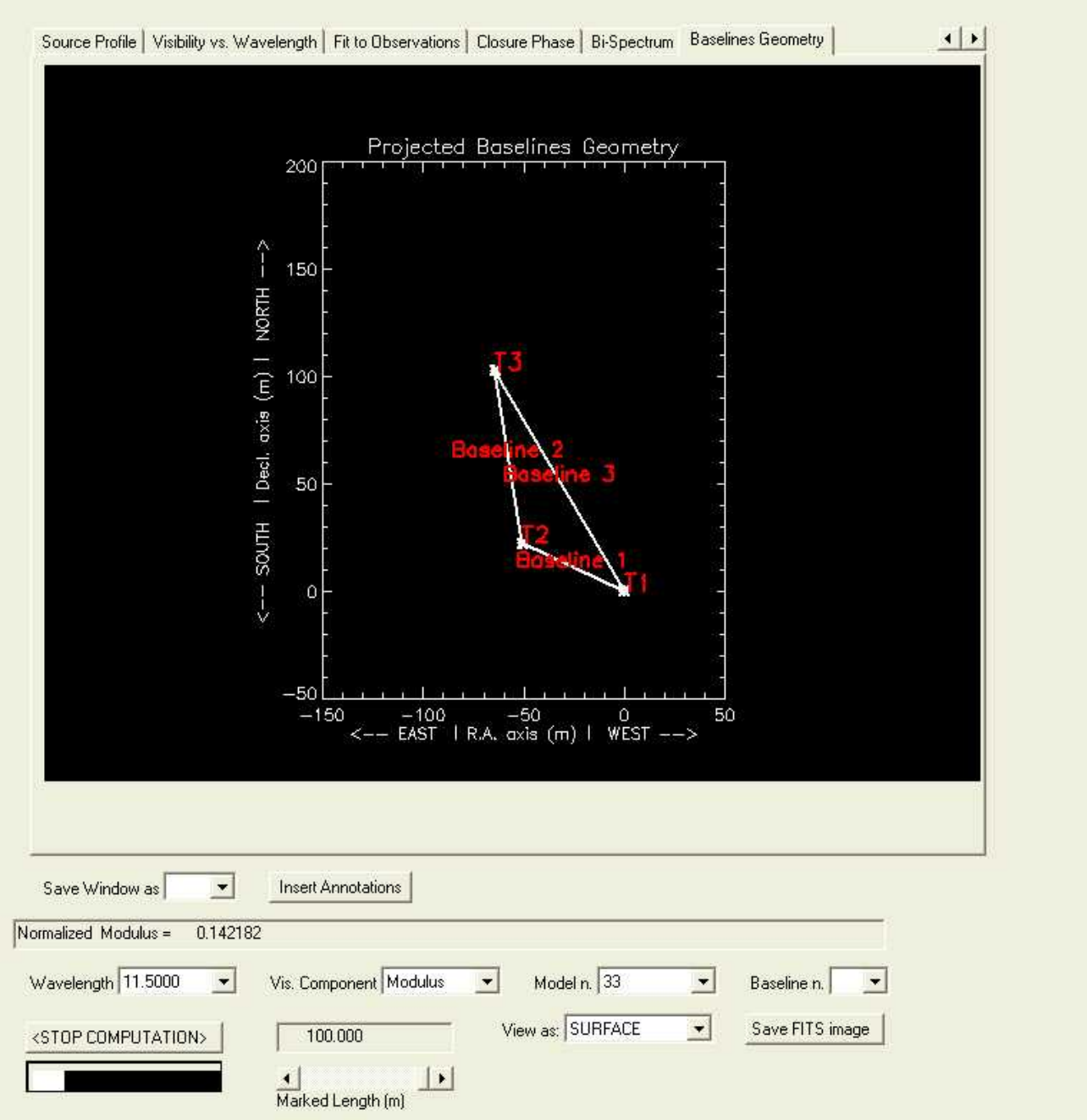

Figure 13: Isometric display showing the geometry of the three baselines projected on the target's tangent plane in the sky: it is very useful to over-impose this plot on an image of the target in order to check the baselines directions respect to any target's feature.

Tab "All Models and Observations":

This is the same as the **Fit to Observations** tab, but all the models are over-imposed on the plot of the Observation data.

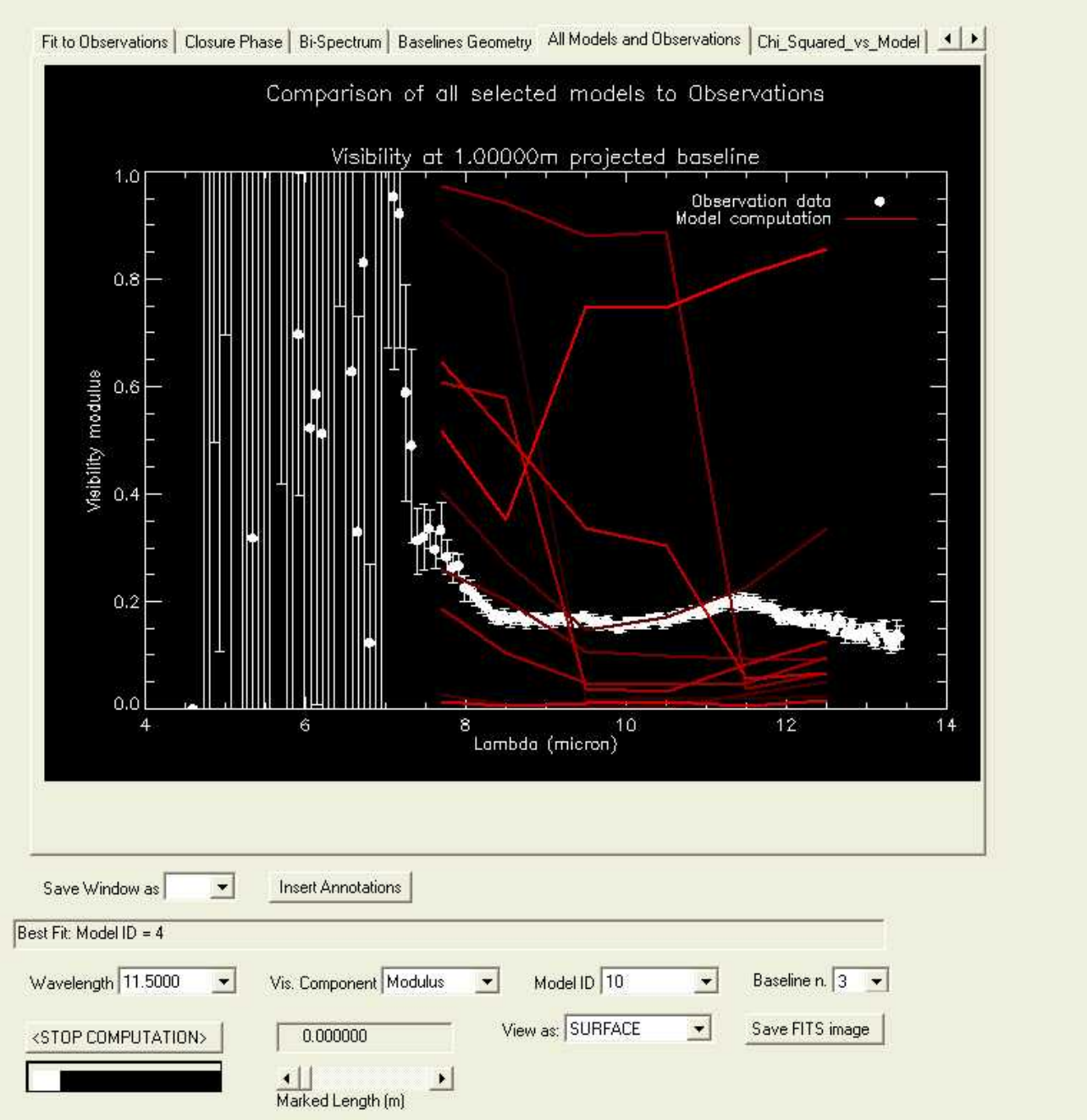

Figure 14: Panel showing the visibility of all the models over-imposed to the observations.

Tab "Chi Squared vs. Model":

On this graph the fit result of all the selected models to observation data is shown and the best fit model indicated in the `Text Message` area.

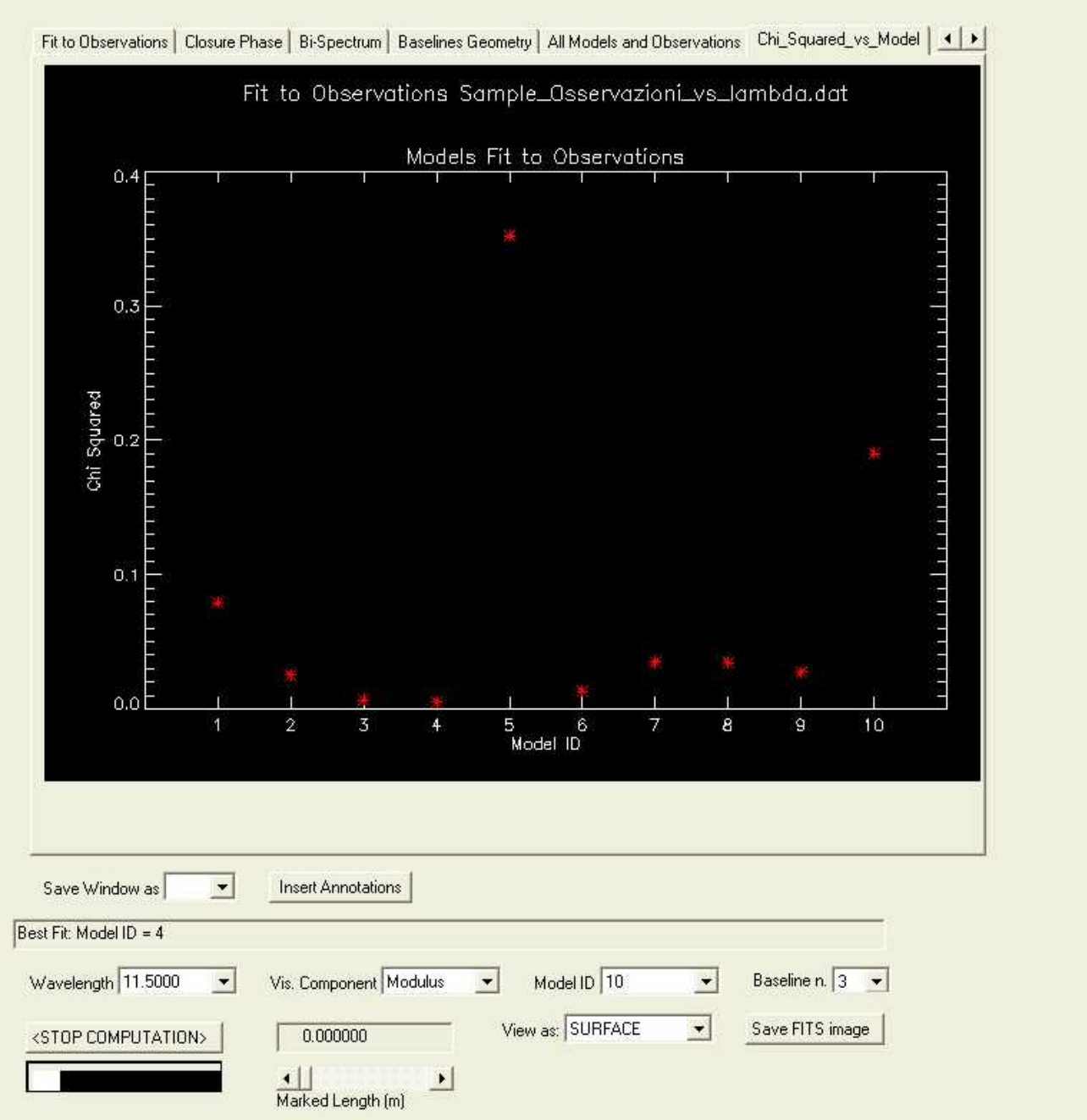

Figure 15: Behaviour of χ^2 model fit to observations vs. Model ID

6.3 User interaction

The user can browse the plots and interact with them via the GUI buttons and drop lists widgets.

Different wavelengths can be visualized in real-time by selecting them via the **Wavelength** drop list while different components of the Complex Visibility can be selected via the **Mode** drop list. Similarly, any model among those selected in input can be displayed by choosing it via the **Model** drop list (changing model requires more time because the computation will be run again for it).

The **Mark at Baseline** slider widget can be used to mark the Visibility plot at a certain baseline with a red asterisk and to update the **Visibility vs. Wavelength** plot (the corresponding visibility value is printed in the **Message Text** area).

In the lower control bar of the GUI, some buttons are available to the user to **Save** a JPEG or TIFF image of the currently exposed window or to insert annotations for presentation scopes.

A running progress-bar showing the status of on-going computations is drawn on the bottom left area of the GUI. Long computation runs can be stopped at any moment by simply pressing the **STOP COMPUTATION** button.

When a computation is terminated the user is free to select for example the **Edit Instruments Parameters** again from the **Computation** menu and change e.g. the **FoV type** or **Slit Position Angle** or whatever, and re-run the whole computation on the same models by simply choosing **Run Computation** from the same menu. The same can do e.g. with the **Target Parameters** and so on, the selected input files will always be taken until the user changes it selecting **Input Data** from the **File** menu.

6.4 Fitting a set of models to Observation data

When working with a multiple parameterized model of an astronomical source, the user's modelling code does normally produce a wide grid of results in the parameters space, which means a huge number of profiles or image files; in this case the user could be interested in comparing its grid of models with some available interferometric observations, in order to find the best χ^2 fit of them.

Observations fitting can be easily done in IVC by selecting a **Visibility vs. Wavelength** or **Visibility vs. Baseline** observation ASCII file; the program will compute the fit and add to the outputs a plot of the χ^2 vs. model number, with the indication of the best fit model ID.

To do this you must first of all select your observation data from **Input Observations** in the **File** menu as follows:

1) - if this is the ***first time*** you access your observation data, select **New Observation Data Type** and follow the instructions to make a new template for it;

   - if not, select **Load Observation Data Type** and choose the right template file for your observations ASCII file.

2) - if your observation data are visibilities as function of ***wavelengths*** , select **`Visibility vs. Wavelength`** and choose the file;

- if they are visibilities as function of ***baselines*** , select **`Visibility vs. Baseline`** and choose the file.

An Observation File must be written in a suitable ASCII format, similar to the one adopted to describe the intensity profile (Tab. 1), in which the **`baseline`**, **`baseline_number`**, **`mode`** and **`observation`** tags must be present and set, respectively, to the ***baseline length*** used for the observation, the ***baseline number*** of the instrument baselines the data refer to, the ***component*** of the complex visibility reported in the file ("Modulus", "Squared Modulus", "Phase", "Real Part", "Imaginary Part") and the ***observation ID*** to distinguish among various observation files.

header:

```
# Baseline in meters:
baseline = 100.0

# Baseline number (starting from 0):
baseline_number = 0

# Visibility Mode:
mode = "modulus"

# Observation ID:
observation = 01

# Results for MIDI observation of HD144432_02:
# Wavelength, Vis, ErrVis
#
# Calibrator2
#
```

data:

```
      13.4152     0.133064    0.0207250
      13.3799     0.137796    0.0265207
      13.3438     0.121556    0.0115012
      13.3071     0.122783    0.0184080
      13.2696     0.129415    0.0176089
      13.2315     0.138606    0.0244342
      13.1928     0.152022    0.0139849
       .....
```

Tab. 3: Format of the Observation ASCII file containing visibility vs. wavelength data.

The data must contain at least two columns with the ***wavelength*** (in ***micron***) and the ***normalized visibility*** measurements of the component described in the **`mode`** tag.
A third column with the ***measurement error*** will be used if present.

In the case of observation data of ***visibilities vs. baseline*** for a fixed wavelength, in place of the **`baseline`** tag is required a **`wavelength`** tag (value in ***micron***) and in place of the ***wavelengths*** column there will be a column with ***baselines*** values, expressed in ***meters***.

When the computation is launched with **`Run Computation`** and an Observation File has been selected, the **`Fit to Observations`**, **`All Models and Observations`** and **`Chi Squared vs. Model`** plots are added as described in Sec. 5.2.

You can practise with an example Observation File given under the **/Sample_Observation_Vis_Modulus_vs_wavelength** directory in the IVC distribution.

6.5 The Toy Models tool

If the user has not his own models grid, or if a qualitative simulation is required, IVC offers a powerful tool for producing model grids wich adopt combinations of simple geometric entities, e.g. a uniform disk star or binary, a gaussian envelope, a limb-darkened disk, or any combination of some of them.
Up to 10 components can be joined together allowing to simulate more complex models and any parameter of each component can be varied in a range a with a step defined by the user, in order to generate a parametric grid of FITS images. These images can then be imported into IVC who will compute the interferometric observables (visibilities, closure phases, etc.) of any of them and, if required, fit them to a set of observations.

The tool is available from the `File` menu under `Input Models > 2-D Toy Models`.
The tool will ask for the number of components of the model you want to create, then, for each component, it will ask the geometry (selectable from a checkbutton list) and the value of the parameters for that geometry.
Tab.4 hereafter lists all the available geometries and their parameters; by combining these simple shapes a wide range of sources can be modelled, e.g. pointlike or resolved binary stars, proto-planetary disks and envelopes, binary systems with complex morphologies for each star, etc.

GEOMETRY	PARAMETERS
`Uniform Disc or Ellipse`	`x_center, y_center, major_axis, minor_axis, position_angle`
`Gaussian Disc or Ellipse`	`x_center, y_center, major_axis_sigma, minor_axis_sigma, position_angle`

Tab. 4: Available toy models geometries and their parameters.

While defining each component, the components combination are displayed, as shown in Figure 16, where the red parts are the previous components and the white part is the last defined one.
The peak intentisites of each component is finally asked and the final combination displayed and saved as a FITS file.

At this stage you can choose if only use this model or create a parametric grid by varying some specific parameters among the ones pertaining to the various components.

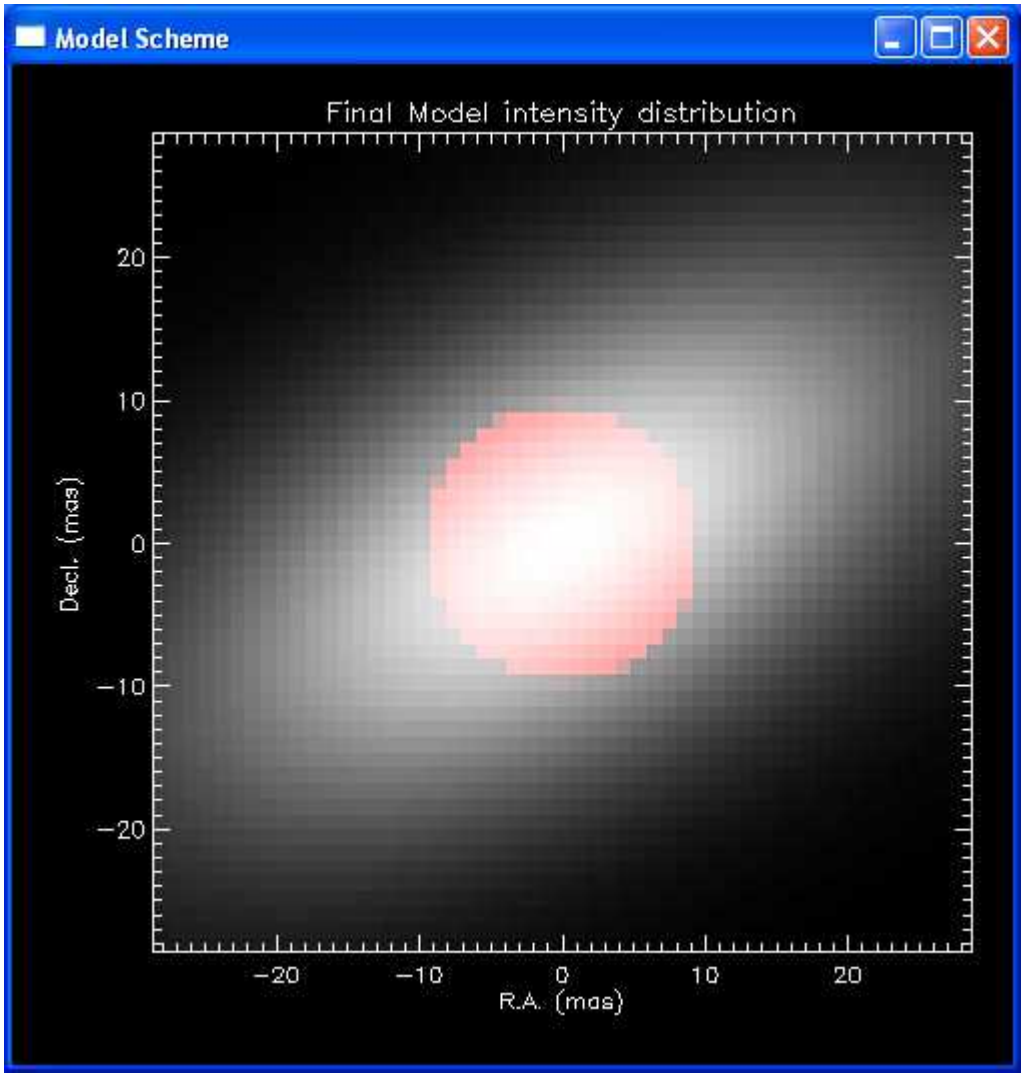

Figure 16: Toy Models tool in action: sum of a uniform disc on face and an inclined gaussian envelope.

If you choose to build a grid a checkbuttons panel is presented for each component, in which you can select the parameters to be varied (don't select anything if you don't want to vary the current component).

Now you are required to enter the range (low_limit and high_limit) and the step values for each parameter that you've choosen to vary.

Finally the Toy Model program generates a number of images with the correct sampling and FoV for the instrument selected in IVC and puts their FITS files under the `/toy_models` directory.

When comparing these models to an observation, as explained in sec. 6.4, IVC will tell which of them best fits the data, so that a finer grid can be created if needed by scanning the parameters' space in the neigborhood of the best fit model with a second turn of the Toy Models tool.

7 Subroutine usage for Multiple fitting on a Grid of models

In sec. 4.4 we have described how to use IVC to perform model fit to interferometric visibility observations.

In that approach the user comes with an already computed grid of parametric models which could be themselves best fit solutions to other astronomical observations, e.g. spectral SEDs or wide band photometries.

If this is the case, and if the grid of models has been computed by a user's code, a more suitable fitting scheme would be to perform a χ^2 fit to the observations already used ***and*** the interferometric ones ***simultaneously*** .

In other words, we want to perform the minimization on the sum of the squared deviations $\chi^2_{TOT} = \chi^2_{OTHER} + \chi^2_{INTERF}$ instead of separately on each of them.

To do this the user can use IVC in the Subroutine-mode by calling it from within his code, outside or inside his loop which spans over the models grid.

This way IVC will be used just as a function ***value=IVC(arguments)*** , which returns the value of χ^2_{INTERF} for each model of the grid, that the user's code will use as it needs.

The Subroutine usage of IVC as a function is straightforward if the user's code is also written in IDL, and just a little more complex if it is in another language, as explained in the following subsection.

7.1 Calling IVC from within IDL

Almost all the parameters accessible from the GUI can be specified in the functional call to IVC, thus making it possible to run it on simple input and complex data as well.

In order to use IVC from IDL just plave the **IVC.sav** file within your current directory and restore it by writing the following command at IDL prompt, or better at one of the first line of your own IDL code:

```
RESTORE, 'IVC.sav', /ROUTINES
```

this will make available the ***IVC*** function, whose schematic syntax is the following

```
value = IVC(arguments)                    ...TBC
```

The detailed explanation of the syntax and usage of this function is given in Appendix 1 hereafter.

7.2 Calling IVC from other languages

This function is not yet implemented in current version.

8. Acknowledgements

The IVC software has been written with the support of the Infrared and Technology Working Group of the INAF - Rome Astronomical Observatory.
It's a pleasure to thank my colleagues D. Lorenzetti, B. Nisini, S. Antoniucci and T. Giannini for their suggestions and helpful discussions.

Thanks a lot to all the people who have tested the previous versions of this software and reported their useful suggestions and bugs advices: S. Antoniucci, B. Demory, S. Sacuto, K. Shabun.

9. Disclaimer

IVC makes use of various IDL procedures and functions taken from the following libraries:

IDL Astronomy User's Library , by Wayne Landsman, Science Systems and Applications (SSAI), NASA, http://idlastro.gsfc.nasa.gov/homepage.html;

Coyote Program Library , by David Fanning, Fanning Software Consulting, http://www.dfanning.com;

Markwardt IDL Library , by Craig B. Markwardt, http://cow.physics.wisc.edu/~craigm/idl/idl.html

OAR IDL Library , by Gianluca Li Causi, INAF – Rome Astronomical Observatory, http://www.mporzio.astro.it/IDL_Library/

All these routines are used and distributed within the IVC package in respect of their copyrights and licences.

10. References

- Li Causi G., “An IDL tool for Interferometric Visibility Computations”, OAR Technical Report, IR/05/02, May 2005
- Li Causi G., “A couple of tools for sparse uv-coverage interferometry: IVC and RaT”, SHARP kick-off meeting, Physics Dep. Univ. Padova, Italy, April 2006
- Li Causi G., “IVC: Interferometric Visibility Computations. An IDL software tool for interferometry simulations and model fitting”, Poster at Euro Summer School “Observation and Data reduction with the Very Large Telescope Interferometer”, 4-16 June 2006, Goutelas (France)

Appendix 1: IVC function details for Subroutine-mode

TBD

Appendix 2: Instrument Parameters description files

Here the two .cfg instrument parameters examples files which comes with the IVC distribution are reported and explained: the first is for the MIDI single baseline instrument and the second is for the AMBER three-baselines instrument.

Following this examples, the user can describe other interferometric instruments.

`` ################################### # IVC Instrument Parmeters file # ################################### ```	
`` # FOV Description (circular / slit): FOV_type ="slit" ```	FoV: this parameter describe the type of focal plane mask, i.e. the "detector FoV" (see sec. 4). For MIDI it is a slit of 0.5x2 arcsec.
`` # FOV Radius (arcsec), only for Circular FOV: FOV_radius =0. ```	
`` # FOV Slit (arcsec), only for Slit FOV: FOV_slit =[0.5,2.0] ```	FoV_Slit: two-element vector with the slit FoV dimensions, in arcsec.
`` # FOV Slit Position Angle (degrees, N to E, on sky): slit_angle =30. ```	Slit_Angle: the P.A. of the major axis of the slit FoV on the sky plane: here an example of 30 degrees.
`` # Gaussian FOV (yes/no): gaussian_fov =yes ```	Gaussian_FoV: the MIDI FoV is not uniform but has a Gaussian illumination.
`` # Baselines Lenght (meters, on Earth) # (set to zero the tird element to get a closed triangle): # UT1-UT2 baselines_length =56.56854249 ```	Baselines_Lenght: the length, in meters, of the baselines on the Earth. Here UT1-UT2 has been taken.
`` # Baseline Position Angle (degrees, N to E, on Earth) # (set to zero the tird element to get a closed triangle): baselines_angle =63.984 ```	Baselines_Angle: the P.A. of the baselines on the Earth. Here UT1-UT2 has been taken.
`` # Maximum Projected Baseline for computations (meters, on sky): max_baseline =200.000 ```	Max_Baseline: IVC will compute the behaviour of visibility in a range of projected baselines: this parameter sets the maximum projected length for wich we want to know the visibility.
`` # Geographical Latitude (degrees): GeoLat = -24.62794830 ```	GeoLat: geographical latitude of the instrument: here VLTI location has been taken.

Tab. 2: The `MIDI_UT1-UT2.cfg` instrument parameters file

`` ################################### # IVC Instrument Parmeters file # ################################### ```	
`` # FOV Description (circular / slit): FOV_type ="circular" ```	FoV: this parameter describe the type of focal plane mask, i.e. the "detector FoV" (see sec. 4). For AMBER it is a fiber core of 56mas.
`` # FOV Radius (arcsec), only for Circular FOV: FOV_radius =0.056 ```	FoV_Radius: circular FoV radius, in arcsec. For AMBER it is 56mas.
`` # FOV Slit (arcsec), only for Slit FOV: FOV_slit =[0,0] ```	
`` # FOV Slit Position Angle (degrees, N to E, on sky): slit_angle =0 ```	
`` # Gaussian FOV (yes/no): gaussian_fov =no ```	
`` # Baselines Lenght (meters, on Earth) # (set to zero the tird element to get a closed triangle): # UT1-UT2, UT2-UT4 baselines_length =[56.56854249, 89.44271910, 0] ```	Baselines_Lenght: a vector with the length, in meters, of the baselines on the Earth. Here UT1-UT2, UT2-UT4 and UT1-UT4 has been taken. The third element has been set to zero in order to let IVC compute it to form a closed triangle.
`` # Baseline Position Angle (degrees, N to E, on Earth) # (set to zero the tird element to get a closed triangle): baselines_angle =[63.984, 8.67915349, 0.] ```	Baseline_Angle: a vector with the P.A. of the baselines on the Earth. Here UT1-UT2, UT2-UT4 and UT1-UT4 has been taken. The third element has been set to zero in order to let IVC compute it to form a closed triangle.
`` # Maximum Projected Baseline for computations (meters, on sky): max_baseline =200.000 ```	Max_Baseline: IVC will compute the behaviour of visibility in a range of projected baselines: this parameter sets the maximum projected length for wich we want to know the visibility.
`` # Geographical Latitude (degrees): GeoLat = -24.62794830 ```	GeoLat: geographical latitude of the instrument: here VLTI location has been taken.

Tab. 3: The AMBER_`UT1-UT2-UT4.cfg` instrument parameters file