

1 Documenting the code

vx1 style of documentation is based on doxygen with slight modification. The documentation comment block has the format

```
///  
// Do not leave any blank lines without the ///  
// will think you have ended the documentation comment, and  
// started an ordinary code comment.  
//  
// If you want to start a new paragraph, put it a line with just ///
```

Apart from this, you can use full doxygen notation. Useful commands are

- `\brief` brief description
- `\param a` comment for parameter *a*.
- `\return` comment for returned value

If you want to do some special stuff such as formulas, have a look at the doxygen documentation <http://www.stack.nl/~dimitri/doxygen/>

1.1 Example

```
///  
// (This is an example of how to code document your )  
// ( file. Comments with brackets, like this one, )  
// ( should be read but not copied into your file. )  
  
#ifndef rretl_feature_h_  
#define rretl_feature_h_  
  
// This is rpil/rretl/rretl_feature.h  
  
// (The above line indicates the file name )  
// ( the \file command below tells Doxygen which )  
// ( file the comments correspond to, so should )  
// ( be left blank in general to minimise the )  
// ( risks of mistakes )  
// (Don't forget to leave an empty line after this one.)  
  
///  
// \file  
// \brief format for storing, reading and writing features  
// \author Charlene Tsai  
// \author Someone else  
// \date 5 May 2001
```

```

// \bug not yet
//
// more description of the file can be given here

#include <vcl_iosfwd.h>
#include <vcl_stlfwd.h>
#include <rretl/rretl_fwd.h>

//: Here you put the description for the class.
//
// Here is the format for storing, reading and writing features
// Every field is required. All must appear on a single, separated line of
// input. The input ends with a blank line.
//
// \verbatim
// image = image_id <string>
// location = x y <double double>
// size = n <int>
// signature =
// x1 y1 <double double>
// ...
// xn yn <double double>
// \endverbatim
class rretl_feature: public vbl_ref_count {
public:
    //: \brief default constructor
    rretl_feature();
    //: \brief destructor
    ~rretl_feature(){}

    //: \brief sets the location to (x,y)
    //
    // more detailed description of the function
    // \param x x_position
    // \param y y_position
    void set_location(double x, double y);

    //: \brief returns location
    //
    // \return returns the location
    vnl_vector_fixed<double,2> location();

    //: \brief checks if location set
    bool is_location_set() const;

private:

```

```
    vnl_vector_fixed<double,2>          location_;
    bool is_location_set_;
};

#endif
```

2 Testing the documentation

Doxygen uses a configuration file to determine all of its settings. A sample configuration **Doxyfile.test** can be copied from `/projects/vision/web/doc`. The `INPUT` tag is the only tag that you are required to edit. You have to specify source files or directories with absolute paths or relative paths to where you start the doxygen.

To generate the documentation, run

```
/projects/vision/local/bin/doxygen Doxyfile.test
```

assuming **Doxyfile.test** is in the current directory. Doxygen will create a html directory under the directory specified by `OUTPUT_DIRECTORY`. The default output directory is the directory in which doxygen is started. The directory to which the output is written can be changed using the `OUTPUT_DIRECTORY` and `HTML_OUTPUT` tags of the configuration file. If the output directory does not exist, doxygen will try to create it for you. Make sure you test build the documentation after using any of the advanced stuff, such as formulas.

The generated HTML documentation can be viewed by pointing a HTML browser to the `index.html` file in the directory specified by `HTML_OUTPUT` tag.