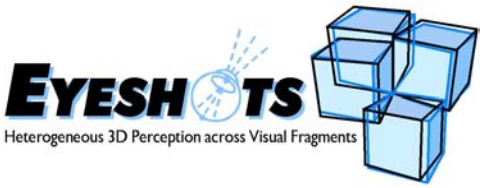


Project no.: FP7-ICT-217077
Project full title: Heterogeneous 3-D Perception across Visual Fragments
Project Acronym: EYESHOTS
Deliverable no: D1.4a
Title of the deliverable: Bioinspired Stereovision Robot System. Robot Prototype. (Intermediate version Month 24)

Date of Delivery:	02 March 2010	
Organization name of lead contractor for this deliverable:	UG	
Author(s):	G. Cannata, A. Trabucco	
Participant(s):	UG	
Workpackage contributing to the deliverable:	WP1	
Nature:	Prototype/Other	
Version:	1.2	
Total number of pages:	55	
Responsible person:	Giorgio Cannata	
Revised by:	S.P. Sabatini	
Start date of project:	1 March 2008	**Duration:** 36 months

Project Co-funded by the European Commission within the Seventh Framework Programme		
Dissemination Level		
PU	Public	
PP	Restricted to other program participants (including the Commission Services)	**X**
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Abstract:
In order to meet the objectives of workpackage WP1 it has been decided to develop a software simulator which could be used as a tool for the analysis of ocular motion and its interplay with vision both within the scope of WP1 and also within other workpackages and in view of the integration activities planned as final demonstration of the project's results. This document describes a software package designed as a simulation tool for the analysis of bio-inspired dynamic ocular models suitable for the study of bio-inspired ocular motion control strategies, as a support tool for the design of a bio-inspired robot eye (expected as deliverable D1.4b), and, if required, to perform comparative analysis with the common pan-tilt platforms commonly used in robot vision. The package has been designed to be an analysis tool bearing in mind various operational requirements. The first is that it could be used with a reasonably limited background knowledge of the Simulink programming environment. To this aim a particular care has been given to limit the number of blocks required to set-up the simulation environments and to provide an extensive use of graphical user interfaces (GUIs) to support the parameterization of the models. The second important requirement has been that of making possible the integration with the virtual reality simulator developed as part of WP1 for the synthesis of ground truth stereo images datasets.

This document is in final form and no updated versions of it will be issued. However, as this report is the User's Manual of the release 1.1 of a software package, some sections could be subject to modification or upgrade. Additional release notes related to further development of the software described in the present report could be produced during the forthcoming project activities.

Contents

List of Figures

1 Executive Summary

This document is the *User's Manual* of a custom made Simulink Toolbox software package which constitutes the Deliverable D1.4a entitled *Bioinspired Stereovision Robot System. Robot Prototype. (Intermediate Version)* of the EU Project EYESHOTS. Deliverable D1.4a is part of the workpackage WP1: *Eye Movements for Exploration of the 3D Space*. In particular D1.4a is the first *outcome* of the activities of the worktasks **Task 1.3** *Control of voluntary eye movements in 3D*, and **Task 1.4** *Bioinspired Stereovision Robot System*.

The workpackage WP1 is devoted to the study of ocular mechanics and oculomotor control, for both single eye and conjugate movements. The target is to investigate how mechanics of the eye plant affects the strategies implemented by the brain to drive typical biological motions ocular motions (including saccades and smooth pursuit). A second goal is the study of the geometric and kinematic effects of ocular motions on image flow, for supporting the estimation of 3D information from ocular motions. Finally, from the engineering point of view the major expected achievement is to develop a bio-inspired stereoscopic robot system capable to emulate the ocular motions to be used during the planned experimental tests.

In order to meet these objectives it has been decided to develop a software simulator which could be used as a tool for the analysis of ocular motion and its interplay with vision both within the scope of WP1 and also within other workpackages and in view of the integration activities planned as final demonstration of the project's results.

This document describes a software package designed as a simulation tool for the analysis of bio-inspired dynamic ocular models suitable for the study of bio-inspired ocular motion control strategies, as a support tool for the design of a bio-inspired robot eye (expected as deliverable D1.4b), and, if required, to perform comparative analysis with the common pan-tilt platforms commonly used in robot vision.

The package has been designed to be an analysis tool bearing in mind various operational requirements. The first is that it could be used with a reasonably limited background knowledge of the Simulink programming environment. To this aim a particular care has been given to limit the number of blocks required to set-up the simulation environments and to provide an extensive use of graphical user interfaces (GUIs) to support the parameterization of the models. The second important requirement has been that of making possible the integration with the virtual reality simulator developed as part of WP1 for the synthesis of ground truth stereo images datasets [15].

2 Introduction

This document describes a software package designed as:

- a simulation tool for the analysis of bio-inspired dynamic ocular models suitable for the study of bio-inspired ocular motion control strategies,
- a support tool for the design of a bio-inspired robot eye, and, if required, for a comparative analysis with the common pan-tilt platforms commonly used in robot vision.

The system allows to model fully bio-inspired or robotic head eye platforms.

The simulator has been fully implemented using MATLAB/Simulink in order to guarantee flexibility and portability (and furthermore has been designed as a Simulink toolbox). The package has been designed to be an analysis tool bearing in mind various operational requirements. The first is that it could be used with a reasonably limited background knowledge of the Simulink programming environment. To this aim, a particular care has been taken to limit the number of blocks required to set-up the simulation environments and to provide an extensive use of graphical user interfaces (GUIs) to support the parameterization of the models. The second important requirement has been that of making possible the integration with the virtual reality simulator developed as part of WP1 for the synthesis of ground truth stereo images [15].

In order to make the usage of the package as simple as possible an extensive on line help documentation has been included in order to provide the required background on the bio-inspired eye modelling and on the robot eye kinematics.

The structure of this document is the following. Section 3 describes the set-up procedures required to install the software package[1]. Section 4 describes the geometric and kinematic conventions adopted during the implementation of the models (first part), and provides the description of the implementation of the various custom block including reference to the scientific literature (second part).

[1]The software makes use of several files generated at configuration and run time. These files are required to manage the data which parameterize each model instance, and they should be never directly modified, moved or deleted by the user.

3 System description

This documentation addresses the procedure to model the human oculomotor plant for the binocular vision and the kinematic of a pan-tilt system. In particular, the goal is to provide guidelines for the implementation of the various functions responsible for human eye movements and binocular vision.
In the first part we describe the steps and the software requirements necessary to configure the system, then we analyze how to implement the various functions of the oculomotor plant and the kinematics of a pan-tilt system.

3.1 Software requirements

To correctly run the system it is necessary to install the following programs and tools:

- WindowsXp or Windows Vista Operating System;
- MATLAB R2008b;
- Simulink 7.2;
- SimMechanics 3.0.

3.2 Directories and Files

The system is organized in a set of directories and files. Fig. 1 shows the directories structure of the system. The base folder is called $EyeShotsLib$ and it contains the following folders and files:

- setup.m: MATLAB file that is central to configure the EyeShotLib package (see the following section for more details);
- data folder: directory that contains ($*.mat$) files with the saved parameters of the Simulink models;
- doc folder: contains web files ($*.html$) for the Matlab help browser and the $User'sGuide.pdf$;
- examples folder: contains some examples about the library block such as a pan-tilt system and the model of the human oculomotor plant;
- lib folder: contains Simulink Library files ($*.mdl$) that implement the functions of a pan-tilt system and of the human eye system;

- src folder: contains all the Matlab files ($*.m$, $*.mat$ and $*.fig$) necessary for the library blocks.

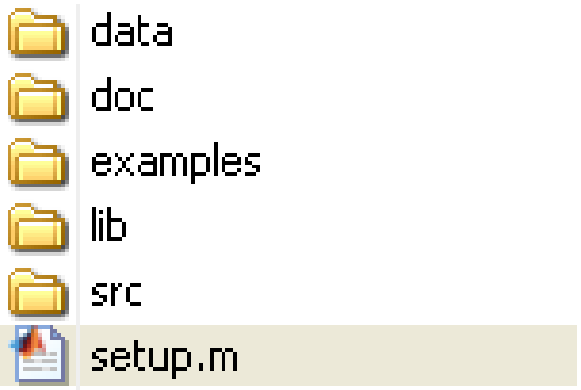


Figure 1: Directories structure of the EyeShotsLib package.

3.3 Installation and Startup

To install the EyeShotsLib package you need to execute the following steps:

1. save the package on a local disk (e.g. $C:\backslash Work$);
2. open Matlab;
3. in the *command window* type:
   ```
   cd C:\Work\EyeShotsLib
   ```
4. in the *command window* type:
   ```
   setup(path)
   ```
   where path is the string of the path where you have saved the package; for example type:
   ```
   setup('C:\Work')
   ```
   and then press enter.
5. finally, close and restart Matlab.

3.3.1 The *setup.m* file

This is the main file. Do not delete the $setup.m$ file. It is located in the EyeShotsLib folder and it comprises three sections.
The first is called $addpath$ and adds to the MATLAB path all the folders of the package $EyeShotsLib$. In this way MATLAB and Simulink know where the files of the system are located.

```
function setup(path)
    % addpath section
    us=userpath;
    s=size(us);
    us=us(1:s(2)-1);
    p=path;
    addpath([p '/EyeShotsLib/lib']);
    addpath([p '/EyeShotsLib/doc']);
    addpath([p '/EyeShotsLib/data']);
    addpath([p '/EyeShotsLib/src']);
    addpath([p '/EyeShotsLib/src/parameters']);
    addpath([p '/EyeShotsLib/examples']);
    savepath
    ....
```

The second section, $Creation\ of\ the\ startup.m\ file\ section$, creates the $startup.m$ file. See the following for more information.

```
    ....
    %Creation of the startup.m file section
    fid = fopen([us, '\startup.m'],'w');
    fprintf(fid,'p=''%s'';\n',path);
    fprintf(fid,'fid = fopen(
                [p ''/EyeShotsLib/src/get_path.m''],''w'');\n');
    fprintf(fid,'fprintf(fid,''function out = get_path()\\n'')
                ;\n');
    fprintf(fid,'fprintf(fid,''out='''''%%s''''';'',p);\n');
    fprintf(fid,'fclose(fid);');
    fclose(fid);
    ....
```

If you change the $user\ path$ or the location of the package you must re-run the $setup.m$ file.

3.3.2 The *startup.m* file

[2] When the MATLAB program starts, automatically executes the master M-file $matlabrc.m$ and, if it exists, $startup.m$. The file matlabrc.m invokes the file startup.m if it exists on the search path that MATLAB uses. You can create a

[2]Part of this section is extracted from the help guide of MATLAB.

startup.m file in your own startup directory for MATLAB. In our case it is saved on the user path (the location of the current directory when MATLAB is open), and creates the file $get\_path.m$.
Code:

```
    p='C:\Work';
fid = fopen([p '/EyeShotsLib/src/get_path.m'],'w');
fprintf(fid,'function out = get_path()\n');
fprintf(fid,'out=''%s'';',p);
fclose(fid);
```

Every time MATLAB is opened,- the $startup.m$ file is called and the $get\_path.m$ is created.

3.3.3 The *get_path.m* file

The get_path file, created by the startup.m, implements a function that returns the path where the package is located. It is used by the functions that save and update the parameters of the model. This file is saved in the src folder.
Code:

```
function out = get_path()
out='C:\Work';
```

3.3.4 The *slblocks.m* file

With this file the library, that contains all the blocks of the oculomotor plant, is added to the Simulink Library Browser. This file is located in the lib folder. The following code is an example of the slblocks file:

```
function blkStruct = slblocks
%SLBLOCKS Defines a block library.
% Library's name. The name appears in the Library Browser's
% contents pane.
blkStruct.Name = ['My' sprintf('\n') 'Library'];
% The function that will be called when the user double-clicks
%on the library's name. ;
blkStruct.OpenFcn = 'mylib';
% The argument to be set as the Mask Display for the subsystem.
% You may comment this line out if no specific mask is desired.
% Example: blkStruct.MaskDisplay =
'plot([0:2*pi],sin([0:2*pi]));';
% No display for now.
% blkStruct.MaskDisplay = '';
% End of blocks
```

4 Library blocks description

In this section we analyze each block of the library EyeLib. In particular the guidelines to create or modify a library block are supplied. In this library the blocks of the oculomotor plant and of the pan-tilt system are present.
The oculomotor plant developed is composed of the head, the two eyeballs and the muscles that drive each eye to reach a particular position. The pan-tilt system is composed of the head, the two pan-tilt cameras and the block that computes the joint velocities of the pan-tilt camera system. The eyeball and the head are modelled using SimMechanics, that is a block diagram modelling environment for the engineering design and simulation of rigid body machines. Conversely the computational block of the oculomotor plant, the eye's muscle and the pan-tilt system are modelled in Simulink, that is a software that models, simulates, and analyzes dynamic systems.

4.1 Geometrical parameters

The system, from a geometric point of view, is composed of a fixed reference frame called world $< w >$ with the origin of the axes in $(0\ 0\ 0)$. A second frame $< h >$, called head, is positioned and oriented in the space with respect to the reference frame $< w >$. On the head two other frames $< l >$ (left camera: eye or pantilt) and $< r >$ (right camera: eye or pantilt) are defined, that describe the position and the orientation of these two frames with respect to the head reference frame.
Now all the geometrical parameters of the system and the transformation matrices between the frames are described (Fig. 2):

- ${}^{w}p_{h/w}$: position vector of the frame $< h >$ with respect to the reference frame $< w >$.
- ${}^{h}p_{l/h}$: position vector of the frame $< l >$ with respect to the reference frame $< h >$.
- ${}^{h}p_{r/h}$: position vector of the frame $< r >$ with respect to the reference frame $< h >$.
- ${}^{w}_{h}R$: rotation matrix of the frame $< h >$ with respect to the reference frame $< w >$.
- ${}^{h}_{l}R$: rotation matrix of the frame $< l >$ with respect to the reference frame $< h >$.

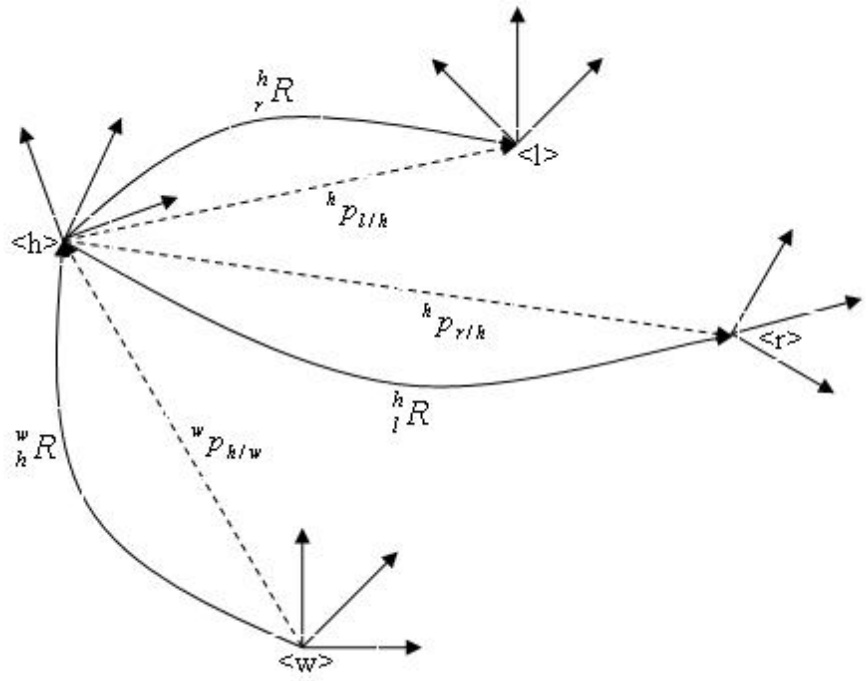


Figure 2: Reference frames and geometrical parameters.

- $^{h}_{r}R$: rotation matrix of the frame $< r >$ with respect to the reference frame $< h >$.
- $^{w}_{h}T$: transformation matrix of the frame $< h >$ with respect to the reference frame $< w >$.
- $^{h}_{l}T$: transformation matrix of the frame $< l >$ with respect to the reference frame $< h >$.
- $^{h}_{r}T$: transformation matrix of the frame $< r >$ with respect to the reference frame $< h >$.

Where:

$$^{w}_{h}T = \begin{bmatrix} ^{w}_{h}R & ^{w}p_{h/w} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{1}$$

$$^{h}_{l}T = \begin{bmatrix} ^{h}_{l}R & ^{h}p_{l/h} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2}$$

$$^{h}_{r}T = \begin{bmatrix} ^{h}_{r}R & ^{h}p_{r/h} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{3}$$

4.2 SimMechanics Toolbox

[3] SimMechanics software is a block diagram modelling environment for the engineering design and simulation of rigid body machines and their motions, using the standard Newtonian dynamics of forces and torques.
With SimMechanics software, you can model and simulate mechanical systems by a suite of tools to specify bodies and their mass properties, their possible motions, kinematic constraints, and coordinate systems, and to initiate and measure body motions. You can represent a mechanical system by a connected block diagram, like other Simulink models, and then incorporate hierarchical subsystems. The

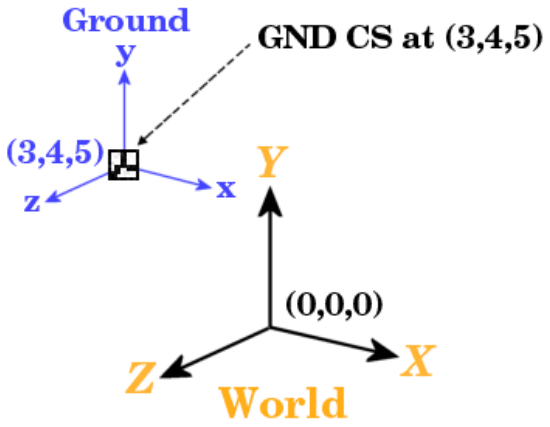


Figure 3: SimMechanics reference frames.

SimMechanics master coordinate system and reference frame is called World. All grounds are at rest in World. The connector port of each Ground block defines a grounded coordinate system called GND. The GND coordinate system's axes are parallel to World.
The SimMechanics block library provides the following blocks specifically for modelling machines:

- Machine Environment blocks set the mechanical environment for a machine. Exactly one Ground block in each machine must be connected to a Machine Environment block.
- Body blocks represent a machine's components and the machine's immobile surroundings (ground).
- Joint blocks represent the degrees of freedom of one body relative to another body or to a point on ground.

[3] Part of this section is extracted from the help guide of MATLAB.

- Constraint and Driver blocks restrict motions of or impose motions on bodies.
- Actuator blocks specify forces, motions, variable masses and inertias, or initial conditions applied to bodies, joints, and drivers.
- Sensor blocks measure the forces on and motions of bodies, joints, and drivers.
- Force element blocks model interbody forces.

Simscape mechanical elements model one-dimensional motion and, with certain restrictions, can be interfaced with SimMechanics machines.

4.3 Creating block Libraries

In this section we illustrate the procedure to create the library EyeShots Blockset, and its sublibrary, and how to add it to the Simulink Library Browser. All the files of the library are in the 'lib' folder.
The root of the library developed for the project is called EyeShotsBlockset. To create this library proceed as follows:

- Select **Library** from the **New** submenu of the **File** menu. Simulink creates a model (*.mdl) file for storing the new library and displays the file in a new model editor window.
- Create an empty subsystem with the name of the sublibrary EyeLib.
- Save the library's model file under EyeShotsBlockset.
- Create another library with the blocks of the oculomotor plant and of the pan-tilt system and save it under EyeLib.
- Open the root library and select the subsystem EyeLib.
- In the Matlab command window type:

  ```
  set_param(gcb,'OpenFcn','EyeLib')
  ```

  in this way, when you double click on the EyeLib subsystem you open the EyeLib library.
  To execute this command in the command window, the EyeLib subsystem must be selected.
- save and close the library.

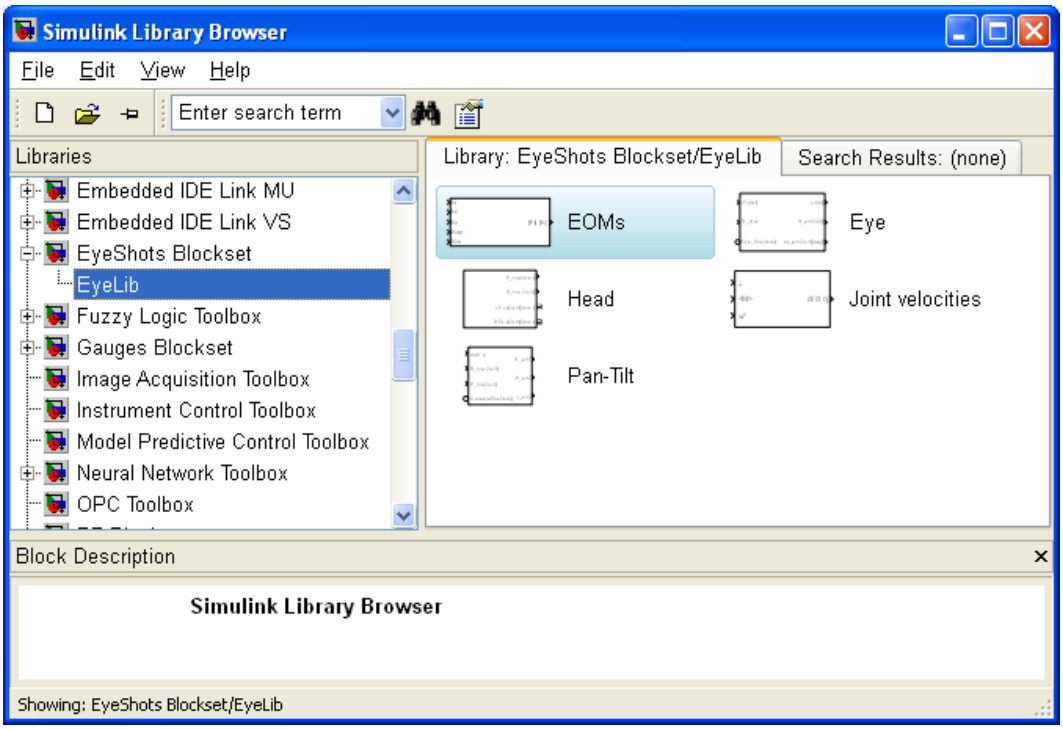


Figure 4: Library EyeLib in the Simulink Library Browser.

When you open a library, this is automatically locked and you cannot modify its contents. To unlock the library, select **Unlock** Library from the **Edit** menu. Closing the library window locks the library. To add the library to the Simulink Library Browser it is necessary to create the slblocks.m file and save it in the same directory of the libraries.

4.4 Creating Matlab GUI (Graphical User Interfaces)

[4] To view or modify the parameters of the oculomotor plant and of the pan-tilt system a graphical interface for each block of the library it is developed.
A graphical user interface (GUI) is a graphical display in one or more windows containing controls, called components, that enable a user to perform interactive tasks.
The GUI components can be menus, toolbars, push buttons, radio buttons, list boxes, and sliders just to name a few. GUIs created in MATLAB software can group related components together, read and write data files, and display data as tables or as plots.

[4]Part of this section is extracted from the help guide of MATLAB.

Most GUIs wait for their user to manipulate a control, and then respond to each action in turn. Each control, and the GUI itself, has one or more user-written routines (executable MATLAB code) known as callbacks, named for the fact that they ”call back” to MATLAB to ask it to do things. The execution of each callback is triggered by a particular user action such as pressing a screen button, clicking a mouse button, selecting a menu item, typing a string or a numeric value, or passing the cursor over a component. The GUI then responds to these events. You, as the creator of the GUI, provide callbacks which define what the components do to handle events.
To cretae a GUI:

- Start GUIDE from the MATLAB **File** menu by selecting **New GUI**.
- Select **Create New GUI**.
- Select a template in the left pane. A preview displays in the right pane.
- Optionally, name your GUI now by selecting ”Save new figure as” and typing the name in the field to the right. GUIDE saves the GUI before opening it in the Layout Editor. If you choose not to name the GUI at this point, GUIDE prompts you to save it and give it a name the first time you run the GUI.
- Click **OK** to open the GUI template in the Layout Editor.
- The component palette at the left side of the Layout Editor contains the components that you can add to your GUI. You can display it with or without names.

By default, the first time you save or run a GUI, GUIDE stores the GUI in two files:

- A FIG-file, with extension .fig, that contains a complete description of the GUI layout and the GUI components, such as push buttons, axes, panels, menus, and so on. The FIG-file is a binary file and you cannot modify it except by changing the layout in GUIDE.
- An M-file, with extension .m, that initially contains initialization code and templates for some callbacks that are needed to control GUI behavior. You must add the callbacks you write for your GUI components to this file. As the callbacks are functions, the GUI M-file can never be a MATLAB script. When you save your GUI the first time, GUIDE automatically opens the M-file in your default editor.

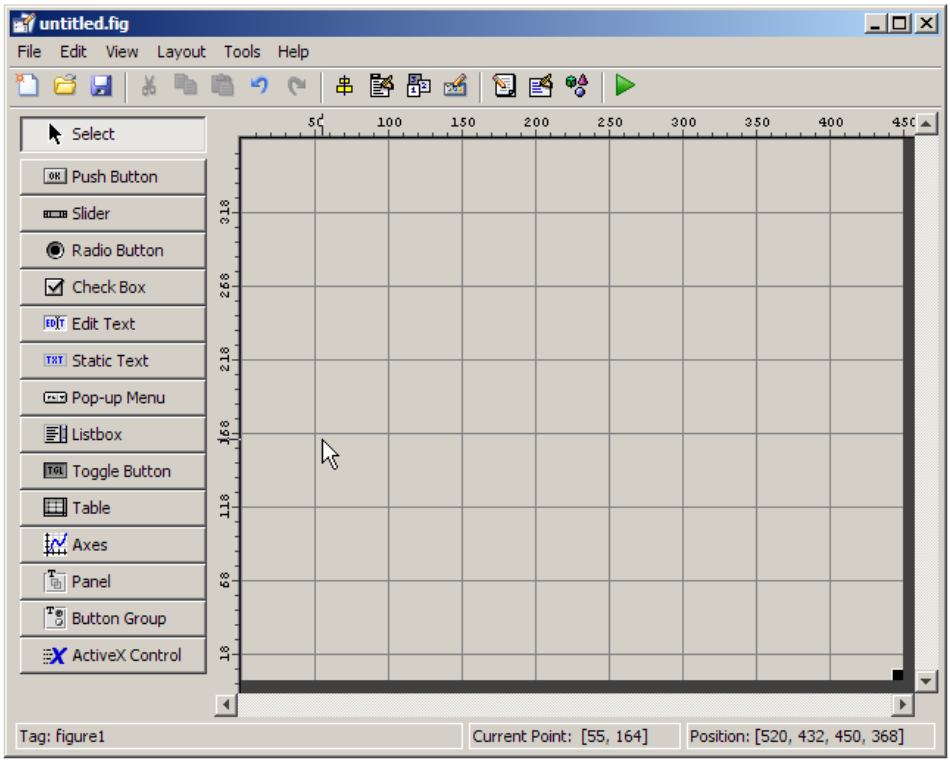


Figure 5: GUI template in the Layout Editor.

The FIG-file and the M-file must have the same name, usually reside in the same directory. They correspond to the tasks of laying out and programming the GUI. When you lay out the GUI in the Layout Editor, your work is stored in the FIG-file. When you program the GUI, your work is stored in the corresponding M-file. See the MATLAB help for more informations about the GUIs.

4.5 Source code of the library blocks

The source code of the blocks is in the ”src” folder and it contains:

- ”parameters” folder: with the source code of each block and another folder (models parameters) with the saved parameters of the Simulink models created by the user;
- get_model.m: this file returns the full path of the block in the model. For example if the name of the model is Eye_model and the block is called $Head$, this function returns the string Eye_model_Head. The source code is:

```
function model = get_model()
    string = gcb;
    lenght = size(string);
    for i=1:lenght(2)
        if (string(i)=='/')
            string(i)='_';
        end
    end
    model = string;
```

For each block the following files have been created (here are shown only the files for the eye block):

- eye_parameters.m contains the code of the callbacks relative to the GUI components.
- eye_parameters.fig contains the components, of the GUI, necessary to configure the parameters of the block.
- eye_param.mat contains the value of the parameters used to initialize the GUI the first time that is opened.
- eye_default_param.mat conatins the default parameters of the block.

4.6 Library blocks

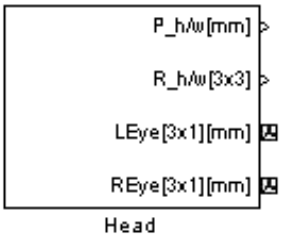
The library Eyelib contains the blocks that describe the oculomotor plant, such as:

- Head: models the human head,
- Eye: model the human eye,
- EOMs: model the muscles of the eye,

and a pan-tilt system:

- Head,
- Pan-tilt: models a pan-tilt camera with two degrees of freedom,
- Joint velocities: computes the joint velocities for the pan-tilt camera.

4.7 Head block



The head block, of the library EyeLib, models the human head. Here we assume that the head is fixed with respect to the reference frame world. The head is modeled like a rigid body regardless the mass, the dimensions and the inertia of the body.
The parameters used in the head model are:

- H: head position with respect to the world reference frame, vector [x y z] in mm.
- O: head orientation vector with respect to the world reference frame, Euler angles in deg.
- L: left camera position with respect to the head reference frame, vector [x y z] in mm.
- R: right camera position with respect to the head reference frame, vector [x y z] in mm.
- LR: left camera orientation with respect to the head reference frame, rotation matrix [3x3].
- RR: right camera orientation with respect to the head reference frame, rotation matrix [3x3].
- LO: left camera orientation vector with respect to the head reference frame.
- RO: right camera orientation vector with respect to the head reference frame.

LO and RO are used as input parameters in the GUI, instead LR and RR are used as input parameters in the model of the head. Thus there is a conversion from LO, RO to the respective rotation matrices.

- $eyeball$: defines the representation of the orientation (LO or RO) for the connected blocks. If it is equal to 1 the connected block is the Eye block and the orientation is expressed with the axis angle representation.
- $pantilt$: defines the representation of the orientation (LO or RO) for the connected blocks. If it is equal to 1 the connected block is the Pan-tilt block and the orientation is expressed with the euler angles representation.

The outputs used of the head block are:

- $LEye$: position vector of the left camera with respect to the head ([x y z] in mm).
- $REye$: position vector of the right camera with respect to the head ([x y z] in mm).
- P_h/w: position vector of the head with respect to the world reference frame ([x y z] in mm).
- R_h/w: orientation matrix of the head with respect to the head reference frame.

$LEye$ and $REye$ are SimeScape signals. $LEye$ must be connect to the Eye_Pos input port of the Eye block for the left eye, in the same way the $REye$ output port must be connect to the Eye_Pos input port of the Eye block for the right eye.

4.7.1 Model description

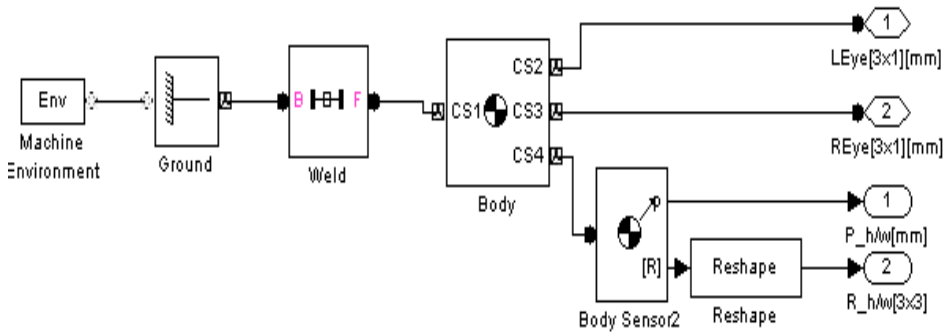


Figure 6: Simulink/SimMechanics model of the head

In each system modelled in SimMechanics a $ground\ body$ must be present, i.e., a body without mechanical properties. We use this block to define the head position (parameter H) with respect to the reference frame world $< w >$. Connected to the ground there is a weld joint (joint without degrees of freedom) because the head is considered fixed with respect to the world reference frame. The body that implements the head is connected to the joint..
On the body four frames are defined:

- CG: the center of gravity or center of mass of an extended body, it's connected to the weld joint.
- $CS2$: this frame defines the position (L) and the orientation (LR) for the left camera with respect to the frame $< h >$.
- $CS3$: this frame defines the position (R) and the orientation (RR) for the right camera with respect to the frame $< h >$.
- $CS4$: this frame is fixed with CG and it is used to measure the orientation and the position of the head with respect to the $< w >$ frame.

Note that $CS2$ and $CS3$ are two outputs of the system (respectively $LEye$ and $REye$) and are SimMEchanics signals. With $CS4$ the other two outputs of this block (P_h/w and R_h/w) are created, which are Simulink signals. This block has no inputs.

4.7.2 Dialog box

The dialog box for the Head block is a MATLAB GUI (Fig. 7). It comprises four sections. The first one is a 'panel' component that contains a 'static text' component with a description of the block (inputs, outputs and parameters). The second is a panel with two 'check box' components. The third one is a 'panel' component that contains: on the left side the 'static text' components with the name of the parameters and on the right the 'edit text' components, used to view or modify the parameters of the block. With these components you can set the parameters for the head block. In the bottom side of the GUI there are seven 'button' components (Ok, Cancel, Apply, Default parameters, Open, Save as and Help).
Now we analyze the callbacks for each component and for the GUI figure itself.
The function $head\_parameters\_OpeningFcn$ is called when you open the GUI and it executes the following code:

```
name = get_param(gcb,'Name');
set(gcf,'Name',name);
```

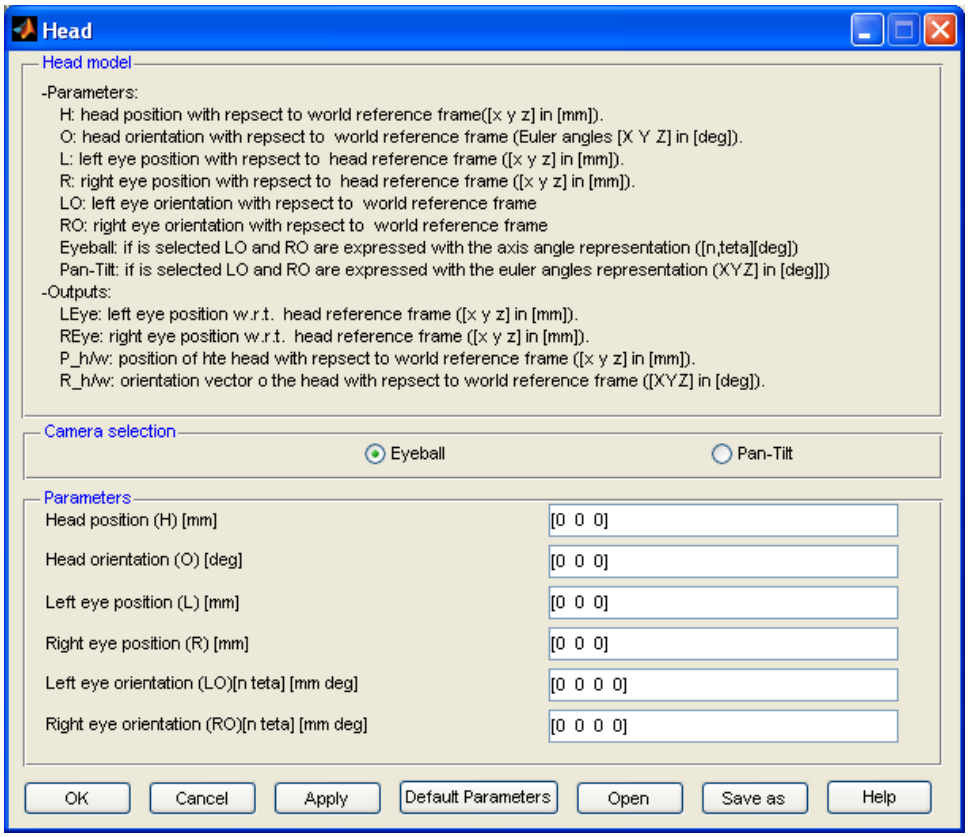


Figure 7: Head block interface.

The two previous lines of code set the name of the GUI with the current name of the block (for example if the block name is $Head1$ the name that appear when you open the GUI is $Head1$)

```
if (strcmp(bdroot,'EyeLib')==1)
    set(handles.H,'Enable','off');
    set(handles.O,'Enable','off');
    set(handles.LO,'Enable','off');
    set(handles.RO,'Enable','off');
    set(handles.L,'Enable','off');
    set(handles.R,'Enable','off');
```

This code is a control that disable all the 'edit text' component if the block is in the library model, or in the Simulink library browser.

```
if(exist([get_model(), '_head_param.mat'])==0)
        lo=load('head_param.mat');
    update_editbox(handles,lo);
    H=lo.H;
    O=lo.O;
    L=lo.L;
```

```
R=lo.R;
LR=lo.LR;
RR=lo.RR;
eyeball = lo.eyeball;
pantilt = lo.pantilt;
save([get_path() '/EyeShotsLib/src/parameters/models_parameters/'
    [get_model(), '_head_param.mat']],'H','O','L','R','LR','RR',
    'eyeball','pantilt');
else
    lo=load([get_model(), '_head_param.mat']);
    update_editbox(handles,lo);
end
if (lo.eyeball == 1)
    % Radio button is selected, take appropriate action
    set(handles.text12, 'string', 'Left eye orientation
            (LO)[n teta] [mm deg]')
    set(handles.text11, 'string', 'Right eye orientation
            (RO)[n teta] [mm deg]')
    set(handles.eyeball, 'Value',1);
else
    % Radio button is not selected, take appropriate action
    set(handles.text12, 'string', 'Left pan-tilt orientation
            (LO)[XYZ] [deg]')
    set(handles.text11, 'string', 'Right pan-tilt orientation
            (RO)[XYZ] [deg]')
    set(handles.pantilt, 'Value',1);
end
```

With this part of code we load the parameters (by the function $update\_editbox$) of the block in the 'edit box' component of the GUI and there is a control of which 'check box' is selected. When this block is used in a Simulink model for the first time the values saved in a file called $head\_param.mat$ are loaded, otherwise, the parameters saved in the '*.mat' file named with the full path name of the block. For example if the name of the model and of the block are, respectively, 'Model' and 'Head1' the name of this file is $Model\_Head1\_head\_param.mat$.
The function $update\_editbox()$ executes the following code:

```
H=lo.H;
O=lo.O;
L=lo.L;
R=lo.R;
LR=lo.LR;
RR=lo.RR;
eyeball = lo.eyeball;
pantilt = lo.pantilt;
if (eyeball==1)
% From rotation matrix to axis angle rapresentation for the left eye
    if (LR==eye(3))
        LO=[0, 0, 0, 0];
```

```
    else
        ql=dcm2quat(LR);
        LR=LR-LR';
        LO(1)= LR(6);
        LO(2)= LR(7);
        LO(3)= LR(2);
        LO=LO/norm(LO);
        LO(4)=rad2deg(2*acos(ql(1)));
    end

% From rotation matrix to axis angle rapresentation for the right eye
    if (RR==eye(3))
        RO=[0, 0, 0, 0];
    else
        qr=dcm2quat(RR);
        RR=RR-RR';
        RO(1)= RR(6);
        RO(2)= RR(7);
        RO(3)= RR(2);
        RO=RO/norm(RO);
        RO(4)=rad2deg(2*acos(qr(1)));
    end
end
if (pantilt==1)
    if (LR==eye(3))
        LO=[0, 0, 0];
    else
        [LO(1) LO(2) LO(3)]=dcm2angle(LR,'XYZ');
        LO(1)=rad2deg(LO(1));
        LO(2)=rad2deg(LO(2));
        LO(3)=rad2deg(LO(3));
    end

    if (RR==eye(3))
        RO=[0, 0, 0];
    else
        [RO(1) RO(2) RO(3)]=dcm2angle(RR,'XYZ');
        RO(1)=rad2deg(RO(1));
        RO(2)=rad2deg(RO(2));
        RO(3)=rad2deg(RO(3));
    end
end
set(handles.H,'String',['[' num2str(H') ']']);
set(handles.O,'String',['[' num2str(O') ']']);
set(handles.L,'String',['[' num2str(L') ']']);
set(handles.R,'String',['[' num2str(R') ']']);
set(handles.LO,'String',['[' num2str(LO) ']']);
set(handles.RO,'String',['[' num2str(RO) ']']);
set(handles.eyeball,'Value',eyeball);
```

```
set(handles.pantilt,'Value',pantilt);
```

The method $set(...)$ is used to load the values of the parameters in the 'edit box' components of the GUI.
The head body block orients the frames of the two cameras by the rotation matrix, but the GUI requires as the orientation parameter the angle axis representation (if the Eyeball check box is selected) or the Euler angles representation (if the Pan-tilt check box is selected). Thus, it is necessary to convert from rotation matrix ([3x3]) to axis angle ($[\mathbf{n}\ \Theta]$) representation by using the following procedure:

- from rotation matrix to quaternion representation

$$q = dcm2quat(R);$$

- rotation angle in deg:

$$\Theta = 2acos(q(0))$$

  where $q(0)$ is the first element of the quaternion.

- rotation vector :

$$S = R - R^T \tag{4}$$

$$\mathbf{n} = [S(6)\ S(7)\ S(2)]^T \tag{5}$$

$$\mathbf{n} = \frac{\mathbf{n}}{|\mathbf{n}|} \tag{6}$$

or to Euler angles representation ([XYZ] [deg]):

- from direct cosine matrix to Euler angles in radians

$$[XYZ] = dcm2angle(R,'XYZ');$$

- from radians to degrees

$$X = rad2deg(X);$$

$$Y = rad2deg(Y);$$

$$Z = rad2deg(Z);$$

where:

R is the rotation matrix

$\mathbf{n} = [n_x n_y n_z]$ is the axis of rotation

X,Y,Z are the Euler angles

$rad2deeg$ is a MATLAB function that converts from radians to degrees

$dcm2angle$ is a MATLAB function that computes the Euler angles from the rotation matrix.

The function $eyeball\_Callback$ (the $pantilt\_Callback$ is similar) is called when the user changes the state of the check box and it executes the following code:

```
function eyeball_Callback(hObject, eventdata, handles)
if (get(hObject,'Value') == get(hObject,'Min') &&
   (get(handles.pantilt,'Value') == get(handles.pantilt,'Min')))
        set(handles.ok,'Enable','off');
    set(handles.apply,'Enable','off');
    set(handles.cancel,'Enable','off');
end
    LO=[0, 0, 0, 0];
    RO=[0, 0, 0, 0];
    set(handles.LO,'String',['[' num2str(LO) ']']);
    set(handles.RO,'String',['[' num2str(RO) ']']);
    set(handles.text12, 'string', 'Left eye orientation
        (LO)[n teta] [mm deg]')
    set(handles.text11, 'string', 'Right eye orientation
        (RO)[n teta] [mm deg]')
if (get(hObject,'Value')==get(hObject,'Max'))
    set(handles.ok,'Enable','on');
    set(handles.apply,'Enable','on');
    set(handles.cancel,'Enable','on');
end
```

if all the check box are unchecked, the buttons () are disabled. Then the orientation vector for the two cameras are configured with the correct representation and if the check box is checked the buttons are enabled.
The Ok callback is called when you click on the Ok button. This function:

```
function ok_Callback(hObject, eventdata, handles)
if (strcmp(bdroot,'EyeLib')==1)
    close (gcf);
else
    get_params(handles)
    close(gcf);
end
```

- close the GUI if the current model is in the Simulink library browser, else calls the get_param(handles) that gets the parameters from the 'edit box' components and save the pare
- close the GUI.

The get_param(handles), called by the Ok callback, executes the following code:

```
function get_params(handles)
H=str2num(get(handles.H,'String'))';
O=str2num(get(handles.O,'String'))';
L=str2num(get(handles.L,'String'))';
R=str2num(get(handles.R,'String'))';
LO=str2num(get(handles.LO,'String'))';
RO=str2num(get(handles.RO,'String'))';
eyeball = get(handles.eyeball,'Value');
pantilt = get(handles.pantilt,'Value');
if (get(handles.eyeball,'Value') == get(handles.eyeball,'Max'))
    SL=[0 -LO(3) LO(2); LO(3) 0 -LO(1); -LO(2) LO(1) 0];
    LR = eye(3)+SL*sin(deg2rad(LO(4)))+SL^2*(1-cos(deg2rad(LO(4))));
    SR=[0 -RO(3) RO(2); RO(3) 0 -RO(1); -RO(2) RO(1) 0];
    RR = eye(3)+SR*sin(deg2rad(RO(4)))+SR^2*(1-cos(deg2rad(RO(4))));
end
if (get(handles.pantilt,'Value') == get(handles.pantilt,'Max'))
    LO(1)=deg2rad(LO(1));
    LO(2)=deg2rad(LO(2));
    LO(3)=deg2rad(LO(3));
    LR=angle2dcm(LO(1),LO(2),LO(3),'XYZ');
    RO(1)=deg2rad(RO(1));
    RO(2)=deg2rad(RO(2));
    RO(3)=deg2rad(RO(3));
    RR=angle2dcm(RO(1),RO(2),RO(3),'XYZ');
end
save([get_path() '/EyeShotsLib/src/parameters/models_parameters/'
        [get_model(), '_head_param.mat']],'H','O','L','R','LR','RR',
        'pantilt','eyeball','-append');
```

the $get(...)$ method gets the value (string) of the parameter relative to an 'edit box' component and save ($save(...)$ method) the parameters in the '.mat' file associated to the block. The $str2num(...)$ is a MATLAB function that converts the variable from string to a vector of double. Then if the eyeball check box is selected the angular axis representation is converted in the rotation matrix representation by using the Rodrigues formula:

$$R = I + S\sin(\Theta) + S^2(1 - \cos(\Theta)) \tag{7}$$

else if the Pan-tilt checkbox is selected:

$$X = deg2rad(X)$$

$$Y = deg2rad(Y);$$

$$Z = deg2rad(Z);$$

$$R = angle2dcm('XYZ');$$

where:

I is the identity matrix [3x3]

R is the rotation matrix [3x3]

X,Y,Z are the Euler angles

$deg2rad$ is a MATLAB function that converts from degrees to radians

$angle2dcm$ is a MATLAB function that computes the rotation matrix from the Euler angles

and

$$S = \begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{bmatrix} \tag{8}$$

where

$$\mathbf{n} = [n_x n_y n_z]$$

is the axis of rotation.
The Cancel callback load, in the 'edit text' components, the last parameters saved, to delete the last parameters insert by the user and close the GUI. The code is:

```
if (strcmp(bdroot,'EyeLib')==1)
    close(gcf);
else
    l=load([get_model(), '_head_param.mat']);
    update_editbox(handles,1);
    if (l.eyeball == 1)
        % Radio button is selected, take appropriate action
        set(handles.text12, 'string', 'Left eye orientation
                (LO)[n teta] [mm deg]')
        set(handles.text11, 'string', 'Right eye orientation
                (RO)[n teta] [mm deg]')
        set(handles.eyeball, 'Value',1);
    else
        % Radio button is not selected, take appropriate action
        set(handles.text12, 'string', 'Left pan-tilt orientation
                (LO)[XYZ] [deg]')
        set(handles.text11, 'string', 'Right pan-tilt orientation
                (RO)[XYZ] [deg]')
        set(handles.pantilt, 'Value',1);
    end
%     eyeball=l.eyeball;
%     pantilt=l.pantilt;
    save([get_path() '/EyeShotsLib/src/parameters/models_parameters/'
        [get_model(), '_head_param.mat']],'-append');
    close(gcf);
end
```

The Apply callback has the same behavior of the Ok callback, but it doesn't close the GUI, the code is:

```
function apply_Callback(hObject, eventdata, handles)
if (strcmp(bdroot,'EyeLib')==1)

else
    get_params(handles)
end
```

The $def\_param\_Callaback$ load in the GUI (when the user click on the Default parameters button) the default parameters of the block, saved in the '.mat' file, called $head\_default\_param.mat$.

```
function def_param_Callback(hObject, eventdata, handles)
if (strcmp(bdroot,'EyeLib')==1)

else
    l=load('head_default_param.mat');
    update_editbox(handles,l);
end
```

The $Open\_Callback$ is called when you click on the Open button. It gets a '.mat' file from the 'data' folder and loads, in the GUI, the parameters of the block saved in the file.

```
if (strcmp(bdroot,'EyeLib')==1)

else
    [File,Path]=uigetfile([get_path() '/EyeShotsLib/data/*.mat'],
        'Select the M-file');
    if (File)
        lo=load(File);
        update_editbox(handles,lo)
    end
end
```

The $uigetfile(...)$ is a MATLAB function that is used to load a file from a folder.
The $save\_as\_Callback$ is called when you click on the Save as button and it gets all the parameters of the block from the 'edit text' components and they are saved in a .mat file in the folder 'data'.

```
function saveas_Callback(hObject, eventdata, handles)
if (strcmp(bdroot,'EyeLib')==1)
else
H=str2num(get(handles.H,'String'))';
O=str2num(get(handles.O,'String'))';
L=str2num(get(handles.L,'String'))';
R=str2num(get(handles.R,'String'))';
```

```
LO=str2num(get(handles.LO,'String'))';
RO=str2num(get(handles.RO,'String'))';
if (get(handles.eyeball,'Value') == get(handles.eyeball,'Max'))
    SL=[0 -LO(3) LO(2); LO(3) 0 -LO(1); -LO(2) LO(1) 0];
    LR = eye(3)+SL*sin(deg2rad(LO(4)))+SL^2*(1-cos(deg2rad(LO(4))));
    SR=[0 -RO(3) RO(2); RO(3) 0 -RO(1); -RO(2) RO(1) 0];
    RR = eye(3)+SR*sin(deg2rad(RO(4)))+SR^2*(1-cos(deg2rad(RO(4))));
end
if (get(handles.pantilt,'Value') == get(handles.pantilt,'Max'))
    LO(1)=deg2rad(LO(1));
    LO(2)=deg2rad(LO(2));
    LO(3)=deg2rad(LO(3));
    LR=angle2dcm(LO(1),LO(2),LO(3),'XYZ');
    RO(1)=deg2rad(RO(1));
    RO(2)=deg2rad(RO(2));
    RO(3)=deg2rad(RO(3));
    RR=angle2dcm(RO(1),RO(2),RO(3),'XYZ');
end
[File,Path]=uiputfile([get_path() '/EyeShotsLib/data/*.mat'],
        'Save as file');
save([Path '/' File],'H','O','L','R','LR','RR');
end
```

The $uiputfile(...)$ is a MATLAB function used to save a file in a particular folder.
The Help callback open the $head\_help.html$ file in the MATLAB Help Browser that contains a description of the Head block:

```
function help_Callback(hObject, eventdata, handles)
web([get_path() '/EyeShotsLib/doc/head_help.html'],'-helpbrowser');
```

Now the GUI and the model of the head are separated. To connect the GUI and the model:

- open a Simulink file ('.mdl') and insert here the model of the head,
- create a subsystem of this model,
- mask this subsystem (To create the mask for this subsystem, select the Subsystem block and choose **Mask Subsystem** from the **Edit** menu),
- open the Mask Editor by selecting the subsystem and click on **Edit Mask** from the **Edit** menu (Fig. 8),
- in the Parameters panel insert the variables with the same name of the GUI,
- for each variable in the dialog callback type:

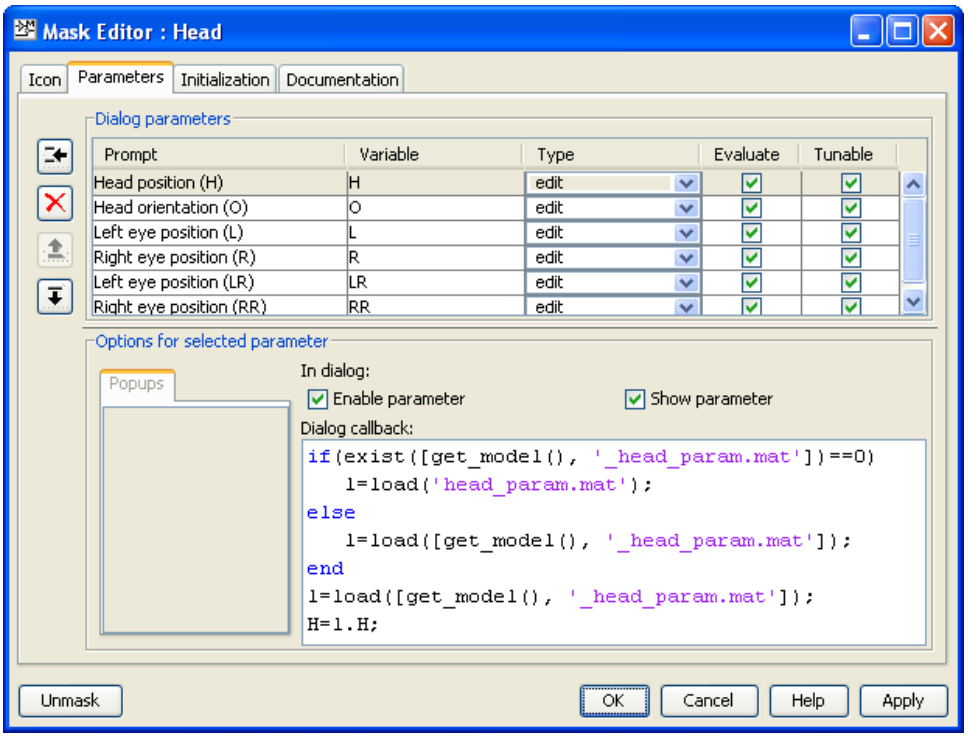


Figure 8: Mask editor for the head block.

```
if(exist([get_model(), '_head_param.mat'])==0)
    l=load('head_param.mat');
else
    l=load([get_model(), '_head_param.mat']);
end
l=load([get_model(), '_head_param.mat']);
var=l.var;
```

where var is the name of the parameters,

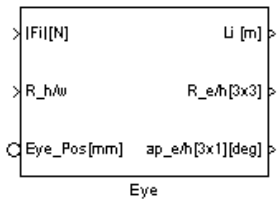
- close the mask editor and double click on the masked subsystem to open the default Dialog box. For each variable, set the value to H, O, L, R, LR, RR, respectively,
- select the head model and in the command window type:

  ```
  set_param(gcb,'OpenFcn','head_parameters')
  ```

- double click on the block to open the head GUI for the block.

In this way, the parameters that you insert in the GUI are 'writen' by Simulink in the blocks of the model.

4.8 Eye block



This block models the human eye. The eye in humans has an almost spherical shape and is actuated by six extra-ocular muscles (EOMs) [1] [14]. Each EOM has an insertion point on the sclera, and is connected with the bottom of the orbit at the other end. The four rectii extra-ocular muscles play a significant role for the implementation of saccadic motion wich obey to the so called Listing's law [3] [4] [5]. It has been found that the path of the recti muscles within the orbit is costrained by soft connective tissue, named soft-pulleys.
We model the soft pulley as fixed pointwise pulleys [6] [7] [8] [9] [10], and we consider only the four rectii muscles.
The eyeball is assumed to be modeled as a homogeneous sphere of radius R, having three degrees of freedom about its center (fig. 9). The eyeball has a moment of inertia (Jp) a mass (mass) and it is connected to viscosity element (Bp) and elastic element (Kp) [11] [12]. The parameters used in the eye model are:

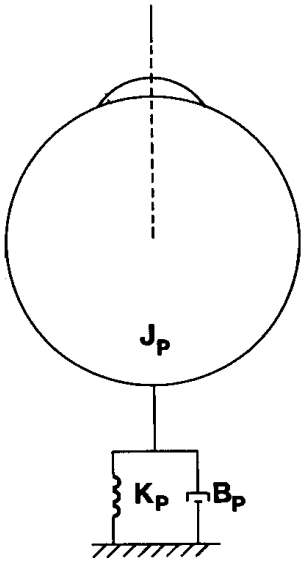


Figure 9: Mechanical properties of the eyeball.

- $C1$: insertion point of the left muscle on the eyeball ([x,y,z] in [mm]);
- $C2$ insertion point of the right muscle on the eyeball ([x,y,z] in [mm]);
- $C3$: insertion point of the upper muscle on the eyeball ([x,y,z] in [mm]);
- $C4$: insertion point of the lower muscle on the eyeball ([x,y,z] in [mm]);
- $P1$: position of the pointwise pulley for the left muscle ([x,y,z] in [mm]);
- $P2$: position of the pointwise pulley for the right muscle ([x,y,z] in [mm]);
- $P3$: position of the pointwise pulley for the upper muscle ([x,y,z] in [mm]);
- $P4$: position of the pointwise pulley for the lower muscle ([x,y,z] in [mm]);
- r: radius of the eyeball [mm];
- $mass$: mass of the eyeball [g];
- Jp: moment of inertia of the eyeball [Kg*m^2];
- Bp: viscous element of the orbit [N*s/m];
- Kp: elastic element of the orbit [N/m];

The inputs of the model are:

- $|Fi|$ [N]: signal with the four rectus muscle forces. These forces are scalar and F1,F2,F3,F4 are the forces of the left, right, upper and lower muscle, respectively. This input port is connected to the outport ($|Fi|$ [N]) of the muscle model.
- Eye_Pos [mm]: position vector of the eye with respect to the head reference frame. It is possible to connect this input port with the output port LEye of the head block, for the left eye; or Reye for the right eye. This is a SimMechanics signal.
- R_h/w: orientation matrix of the head with respect to the world reference frame.

The outputs of the model are:

- $|Li|$ [mm]: signal with the lengthening of the four rectus muscles. L1, L2, L3, L4 correspond to the lengthening of the left, right, upper and lower muscle, respectively. This outport is connected to the input port $|Li|$[mm] of the EOMs block.

- R_e/h[3x3]: rotation matrix of the eyeball with respect to the head reference frame.
- ap_e/h[3x1][deg]: angular position of the eyeball with respect to the head reference frame.

4.8.1 Model description

The Eye block, of the library Eyelib, models the human eye. The colored blocks are SimMechanics blocks, while the not-colored are Simulink blocks. The green block is the body that models the eyeball. On this body are defined the mechanical properties of the eye such as mass and moment of inertia, and a set of frames (Fig. 10), that are:

- CG that it is the center of gravity of the body, all the frames of the body are oriented and positioned respect to the CG reference frame. The CG frame is connected to the joint (grey block). This frame is oriented and positioned like the frame that is directly connected through the joint.
- $CS3$ and $CS4$ are two frames fixed with the CG refernce frame. Two actuator are connected to these two frames (magenta blocks), which are the elastic and viscosity passive forces of the orbit.
- $CS5$, $CS6$, $CS7$, $CS8$ are four frames that define the insertion points (Ci) of the four rectus muscles on the globe. To these frames are connected the actuators (blue blocks) that actuate the body with the muscle forces.
- $CS2$ is the frame connected to the sensor (light blue block) that measures the motion of the body. In our case it measures the position ([mm]), the angular velocity ([deg/s]) and the rotation matrix of the eyeball with respect to the world reference frame.
- $CS9$, $CS10$, $CS11$, $CS12$ are the frames positioned in the insertion point of the muscles on the globe and they are connected to the sensors that measure the position of the insertion points during the motion of the eyeball with respect to the world reference frame. These frames are useful to compute the muscle lenghts.

Note that the actuators and the sensors actuate and measure only with respect to the world reference frame. Now, we analyze the Simulink blocks (Muscle forces, Muscle length , Angular position and From world frame to head frame).

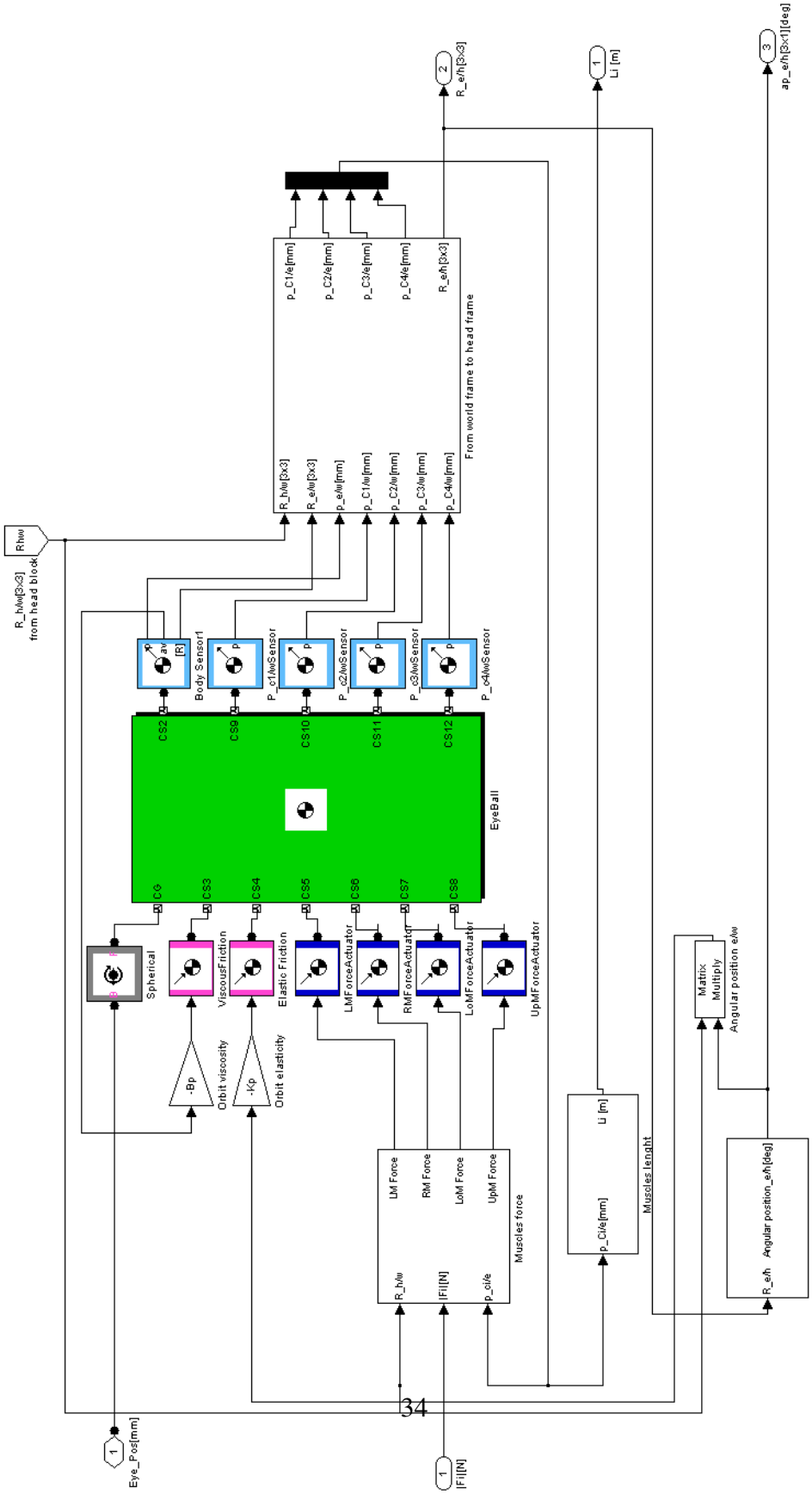


Figure 10: Simulink/SimMechanics model of the eye.

Muscle forces block

This Simulink block (Fig. 11 and Fig. 12) computes the four muscle forces that act on the globe and are connected to the actuators (blue block). The inputs of the block are:

- R_h/w rotation matrix of the head with respect to the world reference frame;
- $|Fi|[N]$ signal with the module of the four rectus muscle forces, that is the output of the EOMs model;
- p_Ci/e position of the four insertion point during the eye movement positioned with respect to the eye reference frame and oriented with respect to the head reference frame. This is the output of the 'From world frame to head frame'

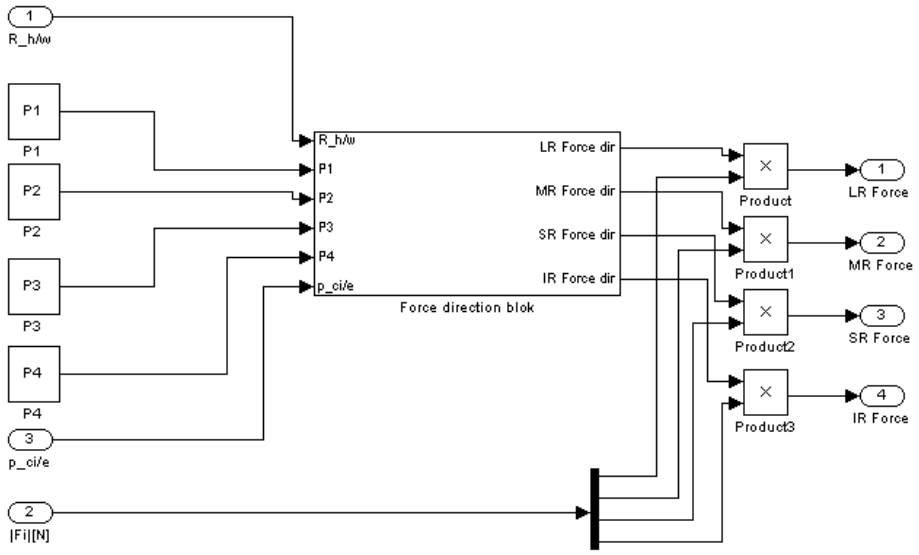


Figure 11: Simulink model of the Muscle forces block

The outputs of the this block are the vectors of the four rectus forces that are the inputs for the actuators (blue blocks):

- $LMForce$: left muscle force [N]
- $RMForce$: right muscle force [N]
- $LoMForce$: lower muscle force [N]

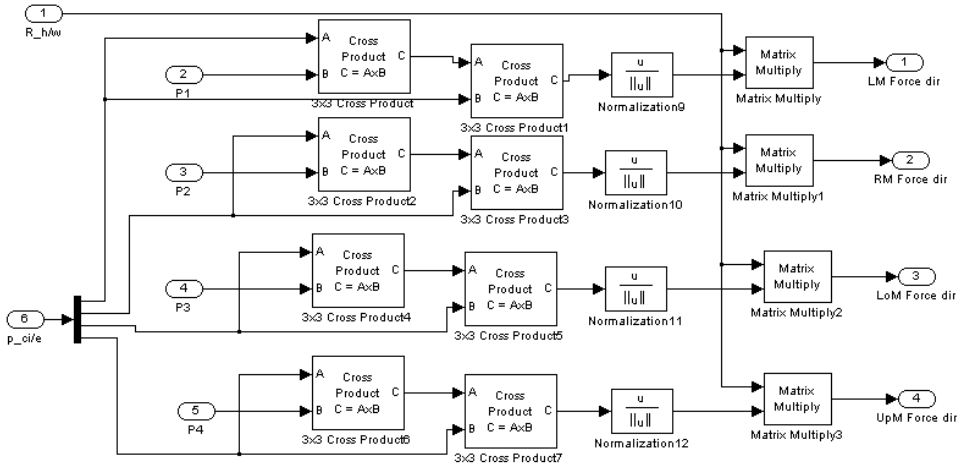


Figure 12: Simulink model of the force direction block

- $UpMForce$: upper muscle force [N]

The Simulink block model that calculates the force direction [14] for the four rectus muscles, is shown in Fig. 12. Mathematically we have:

$$\mathbf{n_i} = \frac{\mathbf{p_i} \times \mathbf{r_i}}{\mathbf{p_i} \times \mathbf{r_i}} \tag{9}$$

$$\mathbf{f_i} = \frac{\mathbf{n_i} \times \mathbf{r_i}}{\mathbf{n_i} \times \mathbf{r_i}} \tag{10}$$

where:

$i = 1 \ldots 4$

$\mathbf{p_i}$ is the position vector of the soft pulley fixed with respect to the head reference frame

$\mathbf{r_i}$ is the position vector of the insertion point (in the model p_Ci/e)

$\mathbf{f_i}$ is the force direction vector normalized.

The $\mathbf{f_i}$ vectors are premultiplied for the rotation matrix R_h/w, thus these vectors are expressed with respect to the world reference frame. Then the module of the four rectus forces are multiplied with the four force direction vectors and these are the four forces that act on the globe.

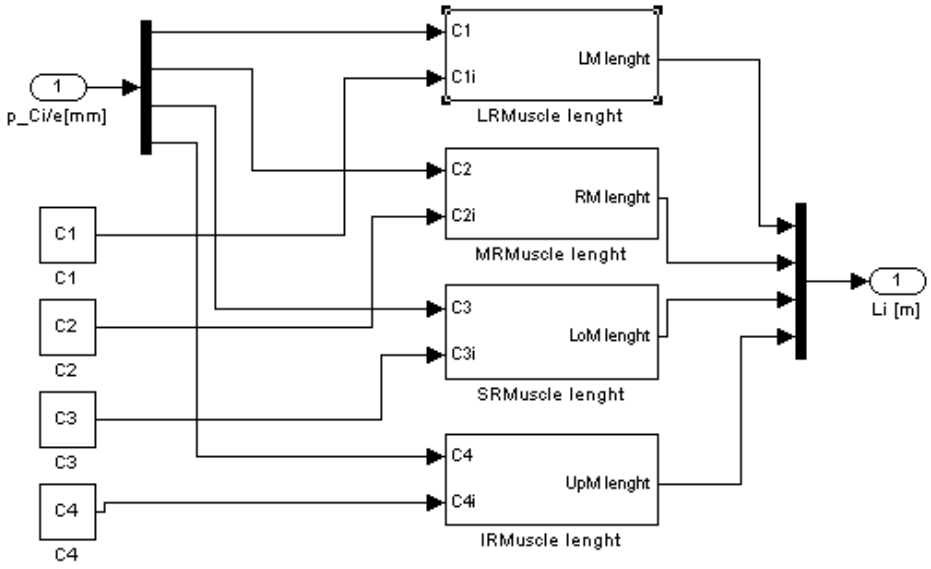


Figure 13: Simulink model of the Muscle length block

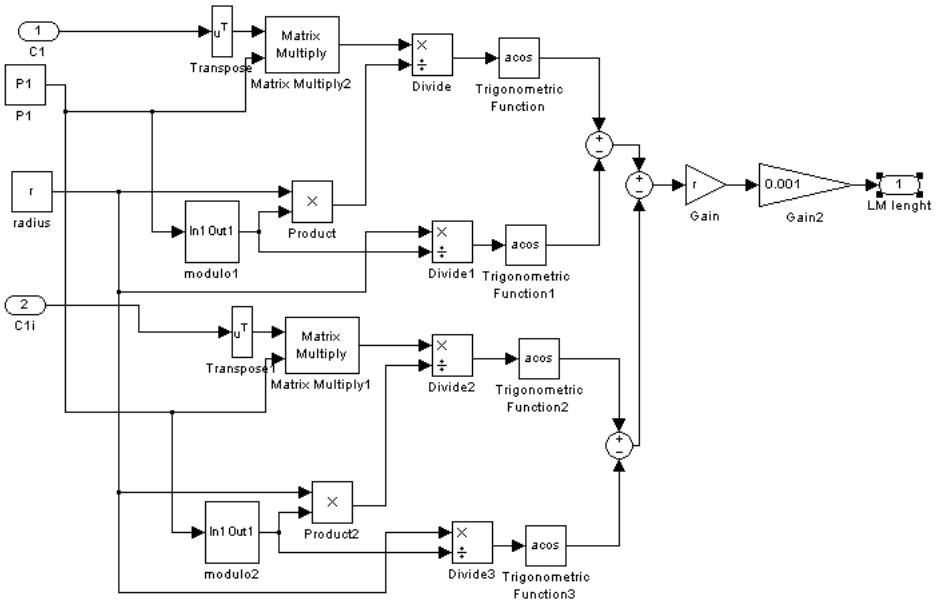


Figure 14: Simulink model of the Muscle length block for the left muscle

Muscle length block

This Simulink block (Fig. 13 and Fig. 14) computes the muscle lengths on geometrical basis. The input of this model are the four position vectors of the insertion

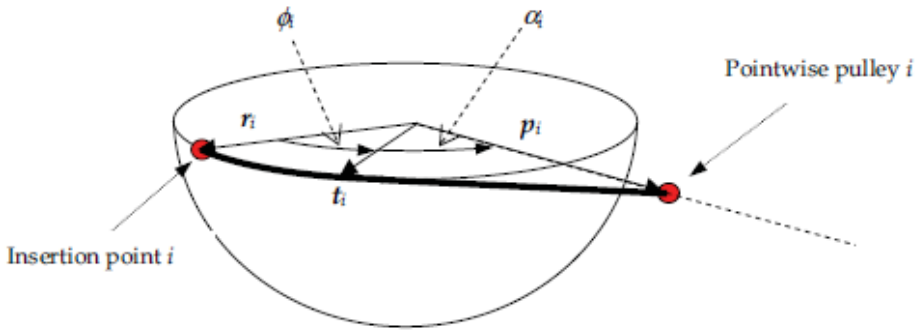


Figure 15: The plane of muscle i for a generic eye orientation

points (p_Ci/e) and the outputs are the four muscle lenghts. An algebraic mapping [14] relating the eye orientation to the displacement of the muscles can be computed by using the formula:

$$x_i = r(\phi_i - \phi_{0i}) \tag{11}$$

where x_i is the amount of the displacement of the free end of the muscle, while ϕ_i and ϕ_{0i} are the angles, formed by vectors $\mathbf{r_i}$ and $\mathbf{t_i}$ at a generic eye orientation, and at the primary position, respectively. In order to compute x_i the angle ϕ_i can be determined, as follows. According to figure 15, the angle α_i must be costant for any eye orientation and can be expressed by:

$$\alpha_i = cos^{-1}\left(\frac{r}{d_i}\right) \tag{12}$$

If the eye orientation is known with respect to frame $< h >$, then $\mathbf{r_i}$ is known, hence:

$$rd_i cos\left(\alpha_i + \phi_i\right) = \mathbf{r_i p_i} \tag{13}$$

and finally, we obtain:

$$\phi_i = cos^{-1}\left(\frac{\mathbf{r_i p_i}}{rd_i}\right) - cos^{-1}\left(\frac{r}{d_i}\right) \tag{14}$$

Angular position block

This Simulink block (Fig. 16) calculates the angular position vector (ap_e/houtput of the block) of the eyeball from the rotation matrix (R_e/h input of the model). Out of this block the angular position vector is premultiplied for the rotation matrix R_h/w, thus this vector is expressed with respect to the world reference frame.

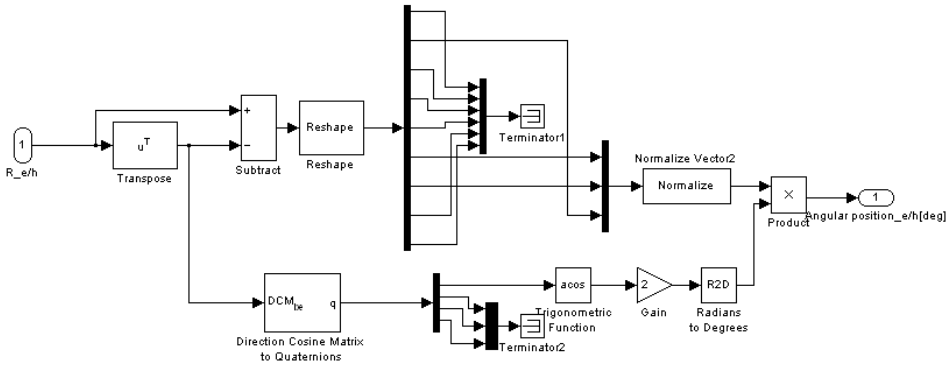


Figure 16: Simulink model of the Angular position block

From world frame to head frame block

This Simulink block (Fig. 17) executes a change of the reference system. The signals of the position of the four insertion points are positioned and oriented with respect to the world reference frame. We want these segnals expressed with respect to the head reference frame. Also the rotation matrix of the eyeball is expressed with respect to the $< w >$ frame. The input of this block are:

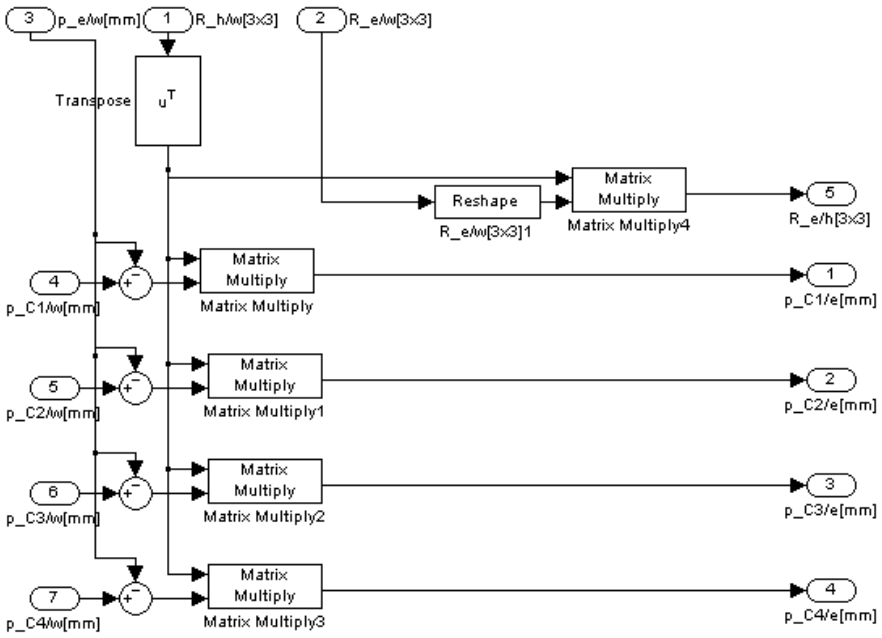


Figure 17: Simulink block of the conversion from world to head reference frame.

- p_e/w position vector of the eye traslated with respect to the $< w >$ frame
- R_h/w rotation matrix of the head with respect to the $< w >$ frame
- R_e/w rotation matrix of the eye with respect to the $< w >$ frame
- p_ci/w position vector of the insertion point i traslated with respect to the $< w >$ frame.

The output of this block are the rotation matrix of the eye (R_e/h) expressed with respect to the head reference frame and the position vector of the insertion points (p_ci/e) traslated with respect to the $< e >$ reference frame and oriented with respect to the $< h >$ frame. Mathematically we have:

$$R\_e/h = R\_h/w^T R\_e/w \tag{15}$$

$$\mathbf{p}\_ci/e = R\_h/w^T \left(\mathbf{p}\_ci/w - \mathbf{p}\_e/w\right) \tag{16}$$

4.8.2 Dialog box

The Dialog box (Fig. 18) of the Eye block is similar to the dialog box of the Head block. Here only the parameters change, but the structure of the GUI and the procedure to cretate and to connect it to the block remain the same.

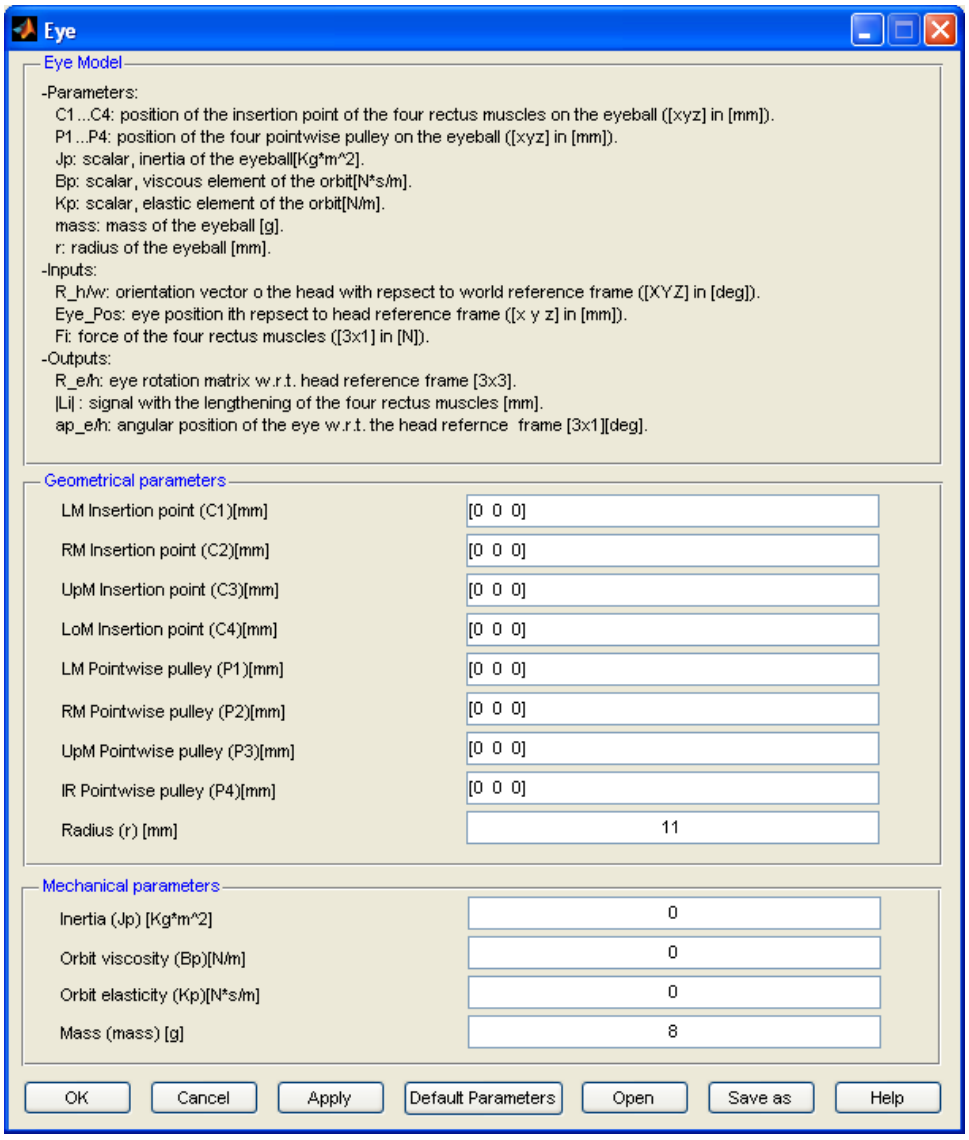


Figure 18: Eye block interface.

4.9 EOMs block

The EOMs block, of the library EyeLib, models the four rectus extraocular muscles. The left, right muscles and the upper, lower muscles form two agonist-antagonist muscle pairs.
Fig. 19 illustrates the mechanical components of the agonist and the antagonist muscles. The agonist muscle is modelled as a parallel combination of an active state tension generator F_{AG}, viscosity element B_{AG}, and elastic element K_{LT}, connected to a series elastic element K_{SE}. Similarly the antagonist muscle is modeled as a parallel combination of an active state tension generator F_{ANT}, viscosity element B_{ANT}, and elastic element K_{LT}, connected to a series elastic element K_{SE} [11] [12].
Each of the elements defined in the model of the muscles is ideal and linear. The

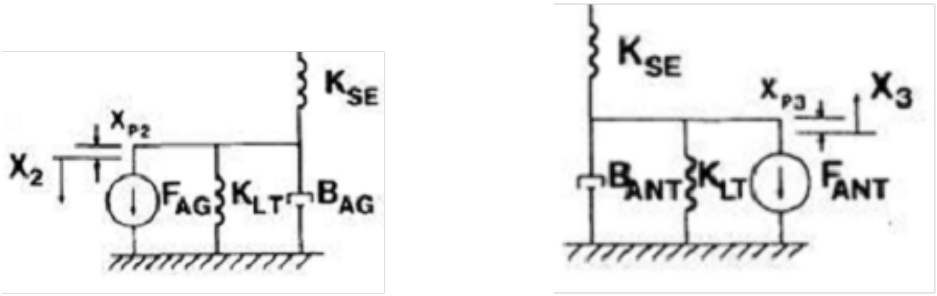


Figure 19: (a) Agonist muscle. (b) Antagonist muscle. [11]

parameters used in the muscle model are:

- B_{AG} Agonist viscous element of the agonist muscle [N*s/m]
- B_{ANT} Antagonist viscous element of the antagonist muscle [N*s/m]
- K_{LT} Elastic element (parallel) of the muscle [N/m]
- K_{SE} Elastic element (serie) of the muscle [N/m]
- Tac/Tde: activation and deactivation time costant of the force generators [s]

- $t1$ control signal switch time [s]

The inputs of the model are:

- $|Li|$ [mm]: signal with the lengthening of the four rectus muscles. L1, L2, L3, L4 that correspond to the lengthening of the left, right, upper and lower muscle, respectively. This input port is connected to the output port$|Li|$ [mm] of the eye block.
- $|Ni|$: neurological control signal for the four rectus muscles. N1, N2, N3, N4 that correspond to the neurological signals of the left, right, upper and lower muscle, respectively.

The outputs of the model are:

- $|Fi|$ [N]: signal with the four rectus muscle forces. These forces are scalar and F1, F2, F3, F4 are the forces, respective, of the left, right, upper and lower muscle. This output port is connected to the input port ($|Fi|$ [N]) of the eye model.

4.9.1 Model description

Each of the four rectus muscles it has been modeled in Simulink (Fig. 20 and Fig. 21). Now we show the equations that describe the agonist and the antagonist muscles pair [11] [12].
For the agonist muscle we have:

$$K_{SE}(x_2 - x_1) = F_{AG} - K_{LT}x_2 - B_{AG}\dot{x}_2 \tag{17}$$

$$\dot{F}_{AG} = (N_{AG} - F_{AG})/T_{ag} \tag{18}$$

$$T_{ag} = T_{ac}(u(t) - u(t - t_1)) + T_{de}u(t - t_1) \tag{19}$$

and for the antagonist muscle:

$$K_{SE}(x_1 - x_3) = F_{ANT} + K_{LT}x_3 + B_{ANT}\dot{x}_3 \tag{20}$$

$$\dot{F}_{ANT} = (N_{ANT} - F_{ANT})/T_{ant} \tag{21}$$

$$T_{ant} = T_{de}(u(t) - u(t - t_1)) + T_{ac}u(t - t_1) \tag{22}$$

where:

x_1 displacement from equilibrium for the stiffness element (K_{SE}) in each muscle

x_2 and x_3 displacement from equilibrium for the parallelel combination of elements in each muscle

T_{ag} and T_{ant} agonist and antagonist time costant

T_{ac} and T_{de} activation and deactivation time costant.

Therefore, according to the direction of the eye movements, each rectus muscle can be agonist or antagonist. The correct choice of the agonist and antagonist viscosity is based on the sign of the force produced by the parallel combination. If this force is greater than zero the muscle is antagonist else it is agonist.

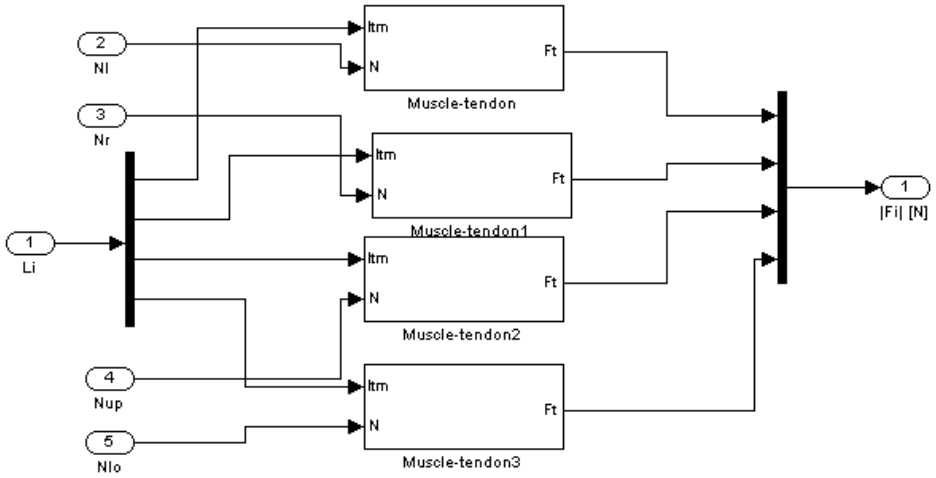


Figure 20: EOMs Simulink model.

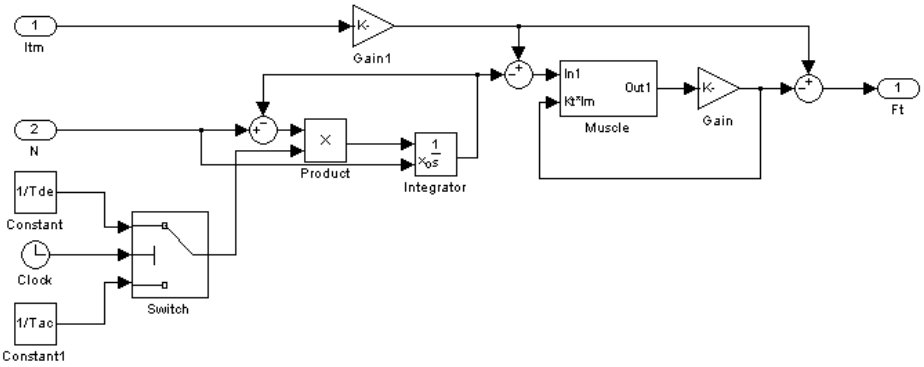


Figure 21: Simulink model of the left rectus muscle.

4.9.2 Dialog box

The Dialog box (MATLAB GUI) of the EOMs block is similar to the dialog box of the Head and Eye block. Here only the parameters change, but the structure of the GUI and the procedure to cretate and to connect it to the block are the same. In Fig. 22 the GUI of the block is shown.

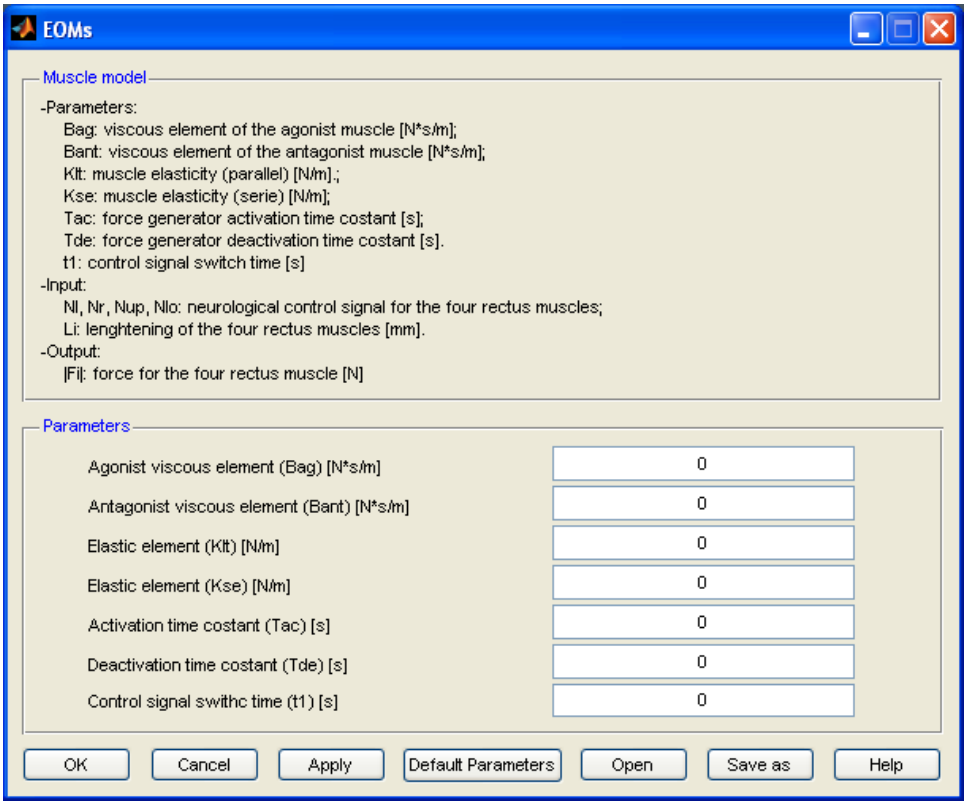
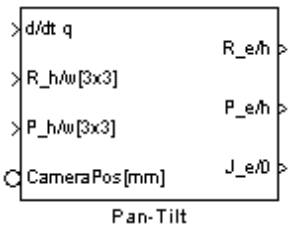


Figure 22: EOMs block interface.

4.10 Pan-tilt block



The pan-tilt block, models a pan-tilt system, that can be represented as a kinematic chain with two degrees of freedom. This system is composed of a revolute joint with one rotational degree of freedom about the x axis (tilt joint), a second revolute joint with one rotational degree of freedom about the y axis (pan joint) and the end-effector (the axis orientation is shown in Fig 23). For each joint and

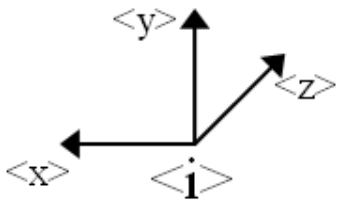


Figure 23: Axes orientation for a generic frame i.

for the end effector, a frame that identifies the position and the orientation of the joints and of the end effector in the space is defined (Fig. 24). The frame $< 0 >$ identifies the position and the orientation of the tilt joint with respect to the head referrnce frame $< h >$. The frame $< 1 >$ identifies the position and the orientation of the pan joint with respect to the tilt joint frame $< 0 >$ and the frame $< e >$ (that can be the left or the rignt camera) identifies position and orientation of the end-effector with respect to the pan joint reference frame $< 1 >$. Through the transformation matrices the position and the orientation of the end-effector can be expressed with respect to the head reference frame $< h >$ (see Fig. 25). The geometrical parameters of the pan-tilt system are:

- ${}^0p_{1/0}$: position vector of the frame $< 1 >$ with respect to the reference frame $< 0 >$.
- ${}^1p_{e/1}$: position vector of the frame $< e >$ with respect to the reference frame $< 1 >$.

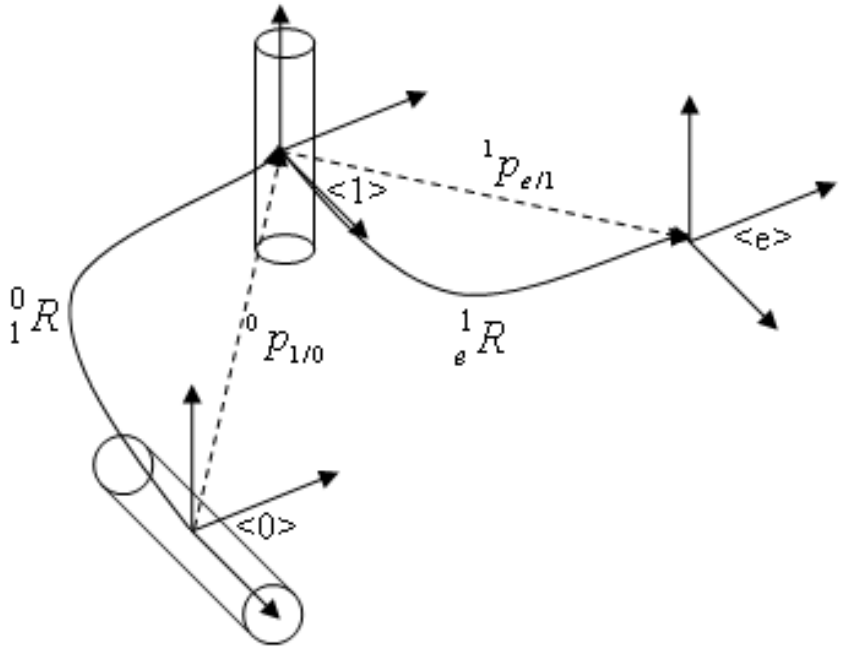


Figure 24: Geometry of the pan-tilt system.

- $^{0}_{1}R$: rotation matrix of the frame $< 1 >$ with respect to the reference frame $< 0 >$.
- $^{1}_{e}R$: rotation matrix of the frame $< e >$ with respect to the reference frame $< 1 >$.
- $^{0}_{1}T$: transformation matrix of the frame $< 1 >$ with respect to the reference frame $< 0 >$.
- $^{1}_{e}T$: transformation matrix of the frame $< e >$ with respect to the reference frame $< 1 >$.

Where:

$$^{0}_{1}T = \begin{bmatrix} ^{0}_{1}R & ^{0}p_{1/0} \\ \mathbf{0}^{T} & 1 \end{bmatrix} \tag{23}$$

$$^{1}_{e}T = \begin{bmatrix} ^{1}_{e}R & ^{1}p_{e/1} \\ \mathbf{0}^{T} & 1 \end{bmatrix} \tag{24}$$

The transformation matrices between the head and the world frame are the same ones of those defined in the section 4.1. In the case of an ideal pan-tilt system the

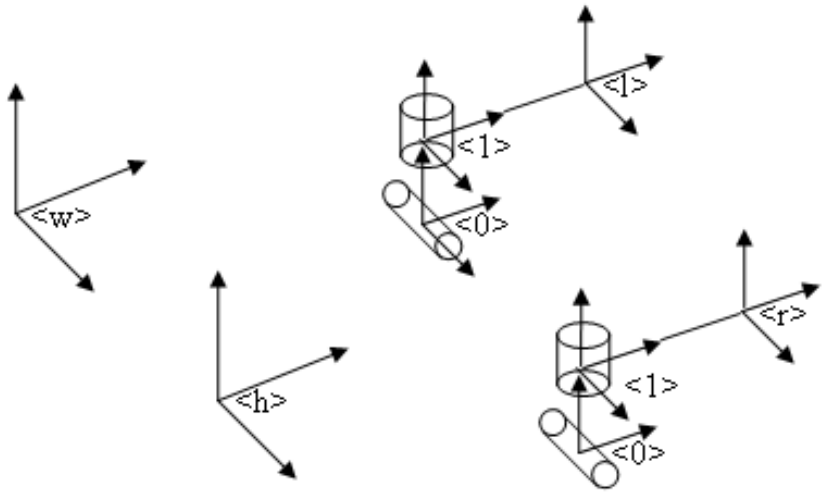


Figure 25: Geometry of the complete pan-tilt system.

frames of the joints and of the end-effector coincide and there isn't translational movement of the end-effector, for a given rotation, with respect to the tilt joint reference frame $< 0 >$. Conversely in the case of a real pan-tilt system the end-effector has a translational movement with respect to the tilt-joint frame. The parameters of the pan-tilt block are:

- $P\_1/0$: position vector of the pan joint frame $< 1 >$ with respect to the tilt joint reference frame $< 0 >$ ([x y z] in mm),
- $P\_e/1$: position vector of the end-effector frame $< e >$ with respect to the pan joint reference frame $< 1 >$ ([x y z] in mm).

The inputs of the model are:

- $d/dt\ q$: vector with the velocities for the two joints [rad/s],
- R_h/w: rotation matrix of the head with respect to the world reference frame,
- P_h/w: position vector of the head with respect to the world reference frame ([x y z] in mm),
- $CameraPos$: position vector and of the tilt joint frame $< 0 >$, and initial orientation of the end effector with respect to the head reference frame $< h >$ [mm].

The outputs of the model are:

- R_e/h: rotation matrix of the camera $< e >$ with respect to the head reference frame $< h >$,
- P_e/h: position vector of the camera $< e >$ with respect to the head reference frame $< h >$,
- $J\_e/0$: Jacobian of the pan-tilt system.

4.10.1 Model description

The pan-tilt block, of the library EyeLib, models the kinematic of a pan-tilt camera. From the joint velocities are computed the joint positions and from these the rotaion matrix of the end-effector with respect to the head reference frame.
The initial condition for the system are taken from the input signal $CameraPos$. Then the Jacobian matrix [13] of the system is computed, which describes the relation between the angular velocity of the end effector and the joint velocities:

$$^0\mathbf{w} =^0 J_{e/0}\dot{\mathbf{q}} \tag{25}$$

where J is composed of:

$$J = \begin{bmatrix} J_A \\ J_L \end{bmatrix} \tag{26}$$

where: J_A is the angular Jacobian and J_L is the linear Jacobian. For a revolute joint we have that:

$$J_A = \mathbf{k}_i \tag{27}$$

with $\mathbf{k}_i$ the axis of rotation of the joint i, and:

$$J_L = \mathbf{k}_i \wedge \mathbf{r}_{e/i} \tag{28}$$

with $\mathbf{r}_{e/i}$ the position of the end-effector with respect to the joint i.
Therefore for our system we have that:

$$J = \begin{bmatrix} J_A \\ J_L \end{bmatrix} = \begin{bmatrix} \mathbf{k}_0 & \mathbf{k}_1 \\ \mathbf{k}_0 \wedge \mathbf{r}_{e/0} & \mathbf{k}_1 \wedge \mathbf{r}_{e/1} \end{bmatrix} \tag{29}$$

Finally adding the position of the pan joint expressed with respect to the head reference frame and the position of the end-effector expressed with respect to the pan joint reference frame we have the position of the end effector expressed with respect to the head reference frame:

$$\mathbf{p}_{e/h} = \mathbf{p}_{e/0} + \mathbf{p}_{0/h} \tag{30}$$

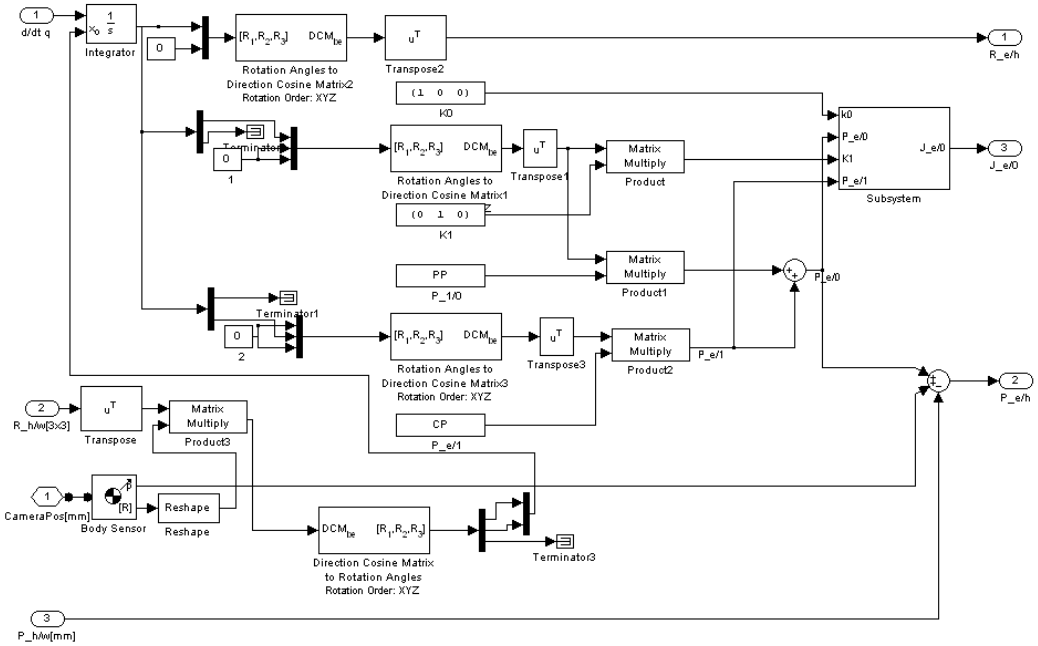


Figure 26: Simulink model of the pan-tilt block.

4.10.2 Dialog box

The Dialog box (Fig. 27) of the Pan-tilt block is similar to the dialog box of the Head, Muscle and Eye block. Here only the parameters change, but the structure of the GUI and the procedure to cretate and to connect it to the block are the same. In this GUI it isn't present the 'Default parameters' button, because it is a block with only two parameters.

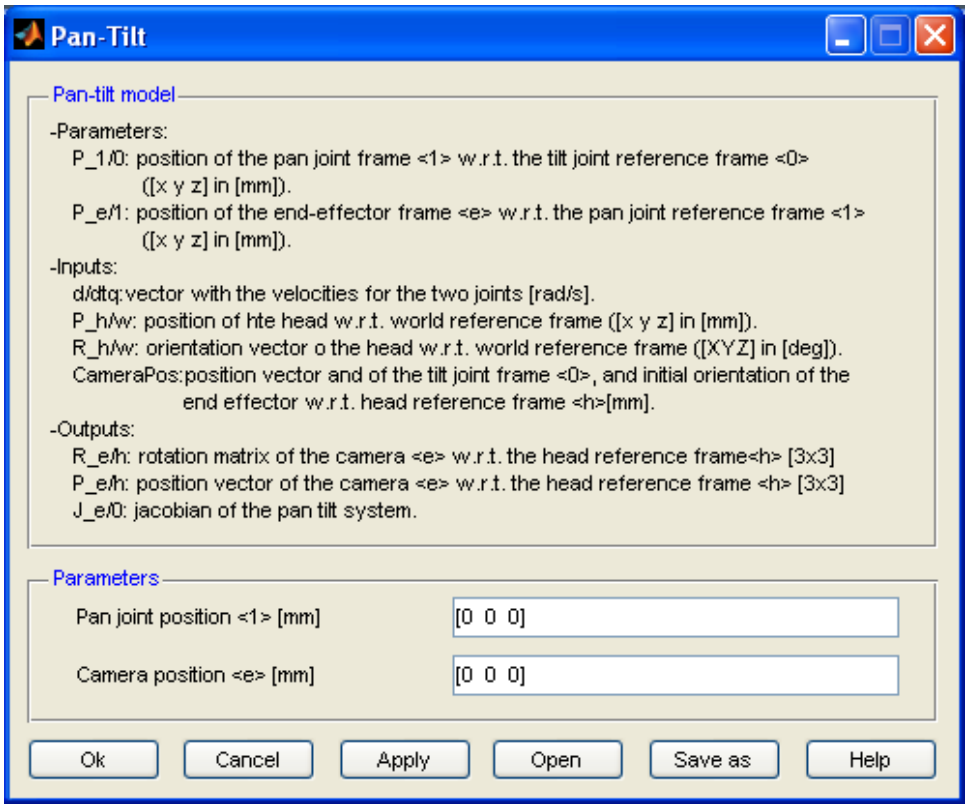
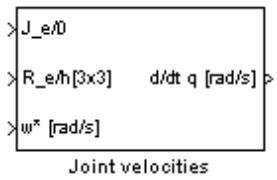


Figure 27: Pan-tilt block interface.

4.11 Joint velocities block



The Joint velocities block (Fig. 28), is used to compute the SVD (Singular Value Decomposition) of the Jacobian matrix of the pan-tilt kinematic chain. In this case, the system is $redundant$, so the Jacobian matrix has more columns than rows and infinite solution exist for the following equation [13]:

$$\dot{\mathbf{q}} = J^{-1}\mathbf{w} \tag{31}$$

We consider only the angular part of the Jacobian matrix because, for simplicity, we assume that the pan-tilt system is ideal. The inputs of the model are:

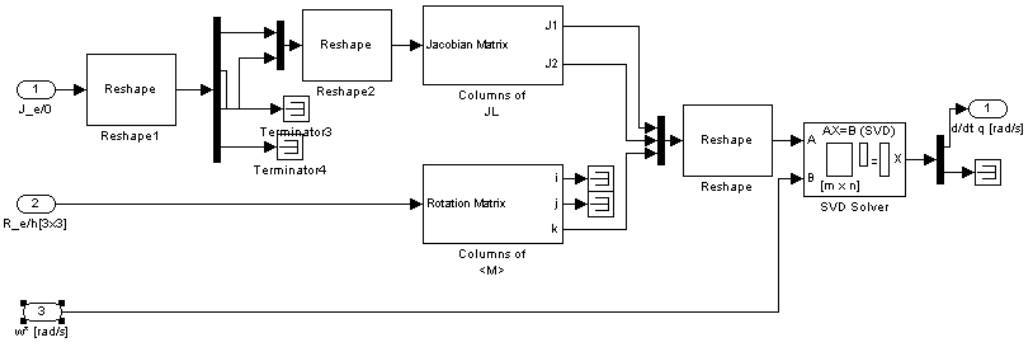


Figure 28: Simulink model of the Joint velocities block.

- R_e/h: rotation matrix of the camera $< e >$ with respect to the head reference frame $< h >$ [3x3],
- w^*: the angular velocity of the end-effector [rad/s],
- $J\_e/0$: Jacobian matrix of the pan tilt system.

The outputs of the model are:

- $d/dt\ q$: vector with the velocities for the two joints [rad/s],
- R_e/h: rotation matrix of the camera $< e >$ with respect to the head reference frame $< h >$ [3x3],
- P_e/h: position vector of the camera $< e >$ with respect to the head reference frame $< h >$ [mm],
- $J\_e/0$: jacobian matrix of the pan tilt system.

4.11.1 Dialog box

The Dialog box (Fig. 29) contains only the panel with a description of the inputs and the outputs of the block, because this block hasn't parameters. There are only two button, that in both cases, close the GUI.

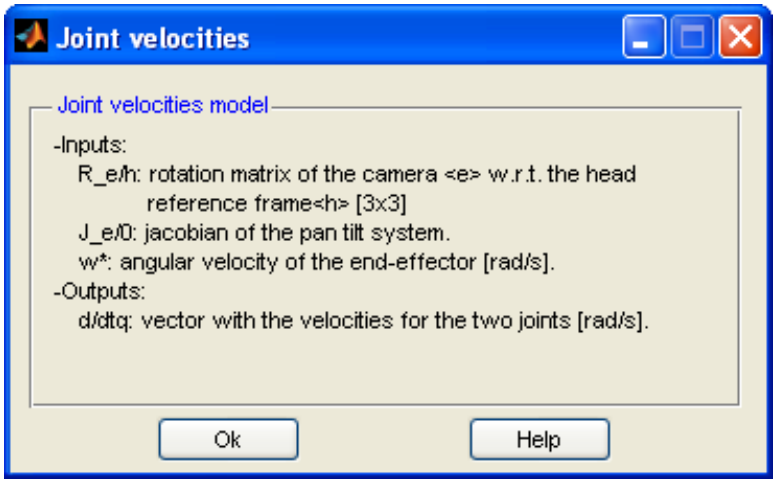


Figure 29: Joint velocities block interface.

References

[1] G. Cannata and M. Maggiali, "Models of the Design of Bioinspired Robot Eyes", IEEE Transaction on Robotics, 2008, vol.24 (no.1).

[2] A. R. Koene, C.J. Erkelens, "Properties of 3D rotations and their relation to eye movement control", Biol. Cybern., vol. 90, pp. 410-417, Jul. 2004.

[3] D. Tweed, T. Vilis, "Geometric relations of eye position and velocity vectors during saccades", Vision. Res., vol. 30, n. 1, pp. 111-127, 1990.

[4] A. D. Polpitiya and B. K. Ghosh, "Modeling the Dynamics of Oculomotor System in Three Dimensions", Proceedings of the Conference on Decision and Control, pp. 6418-6422, Maui,Dec. 2003.

[5] L. Koornneef, "The first results of a new anatomical method of approach to the human orbit following a clinical enquiry", Acta Morphol Neerl Scand, vol. 12, n. 4, pp. 259-282, 1974.

[6] J. M. Miller, "Functional anatomy of normal human rectus muscles", Vision Res., vol. 29, pp. 223-240, 1989.

[7] J. L. Demer, J. M. Miller,V. Poukens,H. V. Vinters and B.J. Glasgow, "Evidence for fibromuscular pulleys of the recti extraocular muscles", Investigative Ophthalmology and Visual Science, vol. 36, pp. 1125-1136, 1995.

[8] R. A. Clark, J.M. Miller, J. L. Demer, "Three-dimensional Location of Human Rectus Pulleys by Path Inflection in Secondary Gaze Positions", Investigative Ophthalmology and Visual Science, vol. 41, pp. 3787-3797, 2000.

[9] J. L. Demer, S. Y. Ho, V. Pokens, "Evidence for Active Control of Rectus Extraocular Muscle Pulleys", Invest. Ophtalmol. Visual Sci., vol. 41, pp. 1280-1290, 2000.

[10] A. T. Bahill, J. R. Latimer and B. T. Troost, "Linear homeomorphic saccadic eye movement", IEEE Trans. Biomed. Eng., vol BME-27, no. 11, pp 631-639, 1980.

[11] J. D. Enderle and J. W. Wolfe, "Time-Optimal Control of Saccadic Eye Movements", IEEE Trans. Biomed. Eng., vol BME-34, no. 1, pp 43-55, 1987.

[12] J. D. Enderle, J. W. Wolfe and J. T. Yates, "The linear homeomorphic saccadic eye movement model- A modification", IEEE Trans. Biomed. Eng., vol BME-31, no. 11, pp 717-720, 1984.

[13] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, "Robotics. Modelling, Planning and Control", Springer-Verlag London Limited 2009.

[14] G. Cannata, M. Maggiali, "Design of a Humanoid Robot Eye", Humanoid Robots, New Developments", I-Tech, pp. 582, 2007.

[15] Chessa, M., Solari, F., Sabatini, S.P., "A Virtual Reality Simulator for Active Stereo Vision System", International Conference on Computer Vision Theory and Applications 2009,VISAPP '09, Lisbon 5-8 February 2009.