HARPIA

Training vision tool

Harpia User Manual

version 0.1

This user manual belongs to the Harpia official documentation

S2i - Industrial Intelligent Systems

S2i

Powered by
Industrial Intelligent Systems

Harpia User Manual

Version 0.1 *Published at 31 July 2007*

Clóvis Scotti <scotti@das.ufsc.br>

Guilherme Rutzen <rutzen@das.ufsc.br>

Luís Carlos Dill Junges <lcdjunges@yahoo.com.br>

Sumário

1 Introduction

This document will present the basic concepts about the application *Harpia*. Its main objective is to be a reference document for those users who do not know anything about image processing techniques and also how to use the mentioned software.

As some brief and important considerations, it its valuable to inform that this project was created by a well-known brazilian research group and based upon its sponsors, the major documentation or related informations are in portuguese. Other really essential fact to bring up is its software license GPL, wich means that it is free to use for non-commercial purposes.

At this development stage, this tool is not so robust for complex or even safe operations in questions related to use image processing as a solution for a problem inside a company or manufacturing process. But, as a *Open-Source* project, it will be soon.

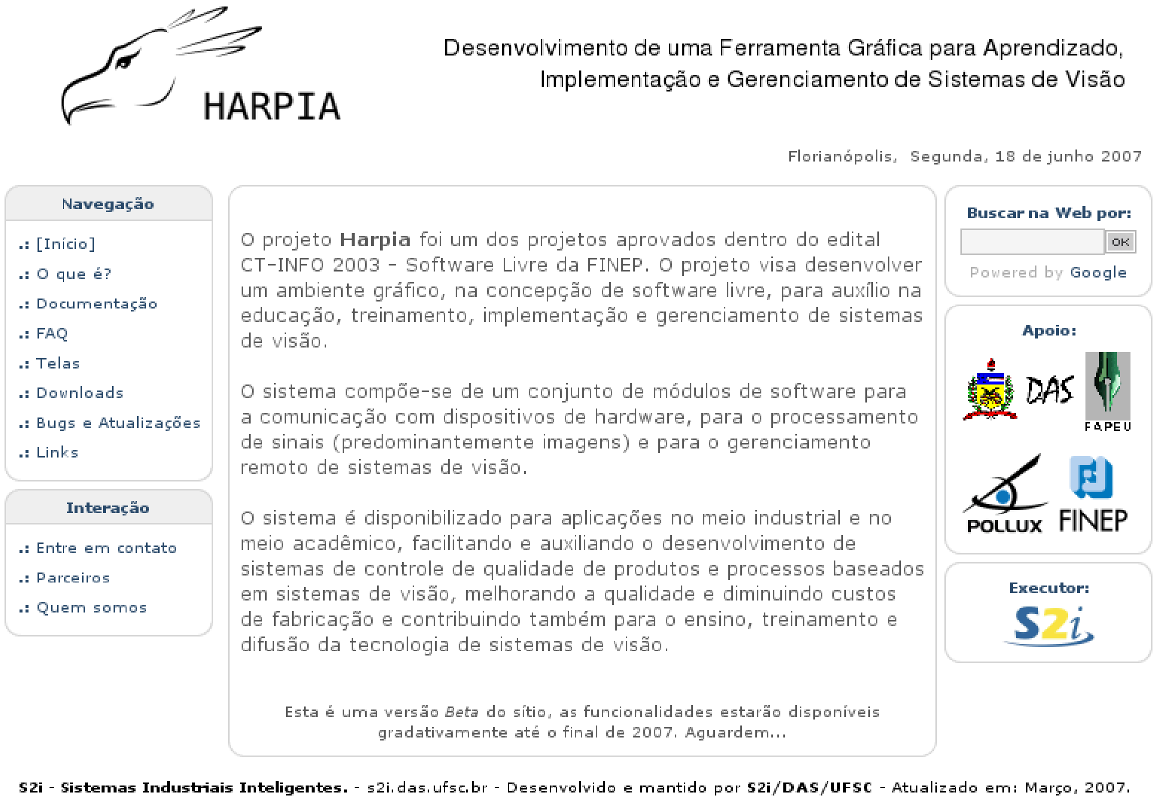
At first glance, the organization of this documentt makes some general descriptions as where you can find resources to improve your image processing models, reports bugs, find source-code documentation and also how to join our group as a programmer. After this, some points about installation procedure on machines with Microsoft Windows, GNU/Linux and others operational systems are showed. At the end, the manner of how to make image processing chains, the areas and available tools inside the graphic interface are commented.

2 Harpia overview

This section gives some internet points where more information can be found. A more complete list of interesting links are available at the documentation page.

2.1 Official website

In the official website `http://s2i.das.ufsc.br/harpia` you can find more information about the project, older released versions, tutorials about image processing as other relevant informations related to this area.



Desenvolvimento de uma Ferramenta Gráfica para Aprendizado, Implementação e Gerenciamento de Sistemas de Visão

Florianópolis, Segunda, 18 de junho 2007

O projeto **Harpia** foi um dos projetos aprovados dentro do edital CT-INFO 2003 - Software Livre da FINEP. O projeto visa desenvolver um ambiente gráfico, na concepção de software livre, para auxílio na educação, treinamento, implementação e gerenciamento de sistemas de visão.

O sistema compõe-se de um conjunto de módulos de software para a comunicação com dispositivos de hardware, para o processamento de sinais (predominantemente imagens) e para o gerenciamento remoto de sistemas de visão.

O sistema é disponibilizado para aplicações no meio industrial e no meio acadêmico, facilitando e auxiliando o desenvolvimento de sistemas de controle de qualidade de produtos e processos baseados em sistemas de visão, melhorando a qualidade e diminuindo custos de fabricação e contribuindo também para o ensino, treinamento e difusão da tecnologia de sistemas de visão.

Esta é uma versão *Beta* do sítio, as funcionalidades estarão disponíveis gradativamente até o final de 2007. Aguardem...

S2i - Sistemas Industriais Inteligentes. - s2i.das.ufsc.br - Desenvolvido e mantido por **S2i/DAS/UFSC** - Atualizado em: Março, 2007.

Figura 2.1: Website screenshot.

2.2 Documentation

As a *Open-Source* project, its entire source-code is available as its documentation at the official website under the `http://s2i.das.ufsc.br/harpia/doc.html` link. At this time some available documents are a tutorial about image processing concepts, project documentation(source-code), backend documentation and some reports, articles and presentations about the project.

2.3 Bug tracker

As this application is in its first versions, certainly some bugs will appear. In order to correct them for the next version, a bug tracker called mantis is running under `http://s2i.das.ufsc.br/mantis`. So, new bugs posts can be commented there taking aware that some informations as how the bug happened and how to reproduce it is really valuable to kill it.

3 Installation

One of the main requirements of the *Harpia* application is its capacity to run in several machines and through the architecture client-server execute image processing chain requests. This mean that the *Harpia* client can run on one machine and the server in another one. The section below presents the steps to install the necessary files to run just the client, just the server or both at the same machine.

3.1 Installation requirements

To install and use *Harpia* application, the following requirements must be reached:

- At least 100MB(Windows), 28MB(GNU/Linux) of free disk space
- 256 MB RAM
- Frame-grabber (only for those who what acquire images from cameras)

3.2 Microsoft Windows

The installation under Microsoft Windows platform use the automatic installer provided by nsis[1]. By default, it just necessary confirm all the step to install the minimal set of files, including those for the server, in order to run it. However, it is possible to install more packages as the documentation or even libraries.

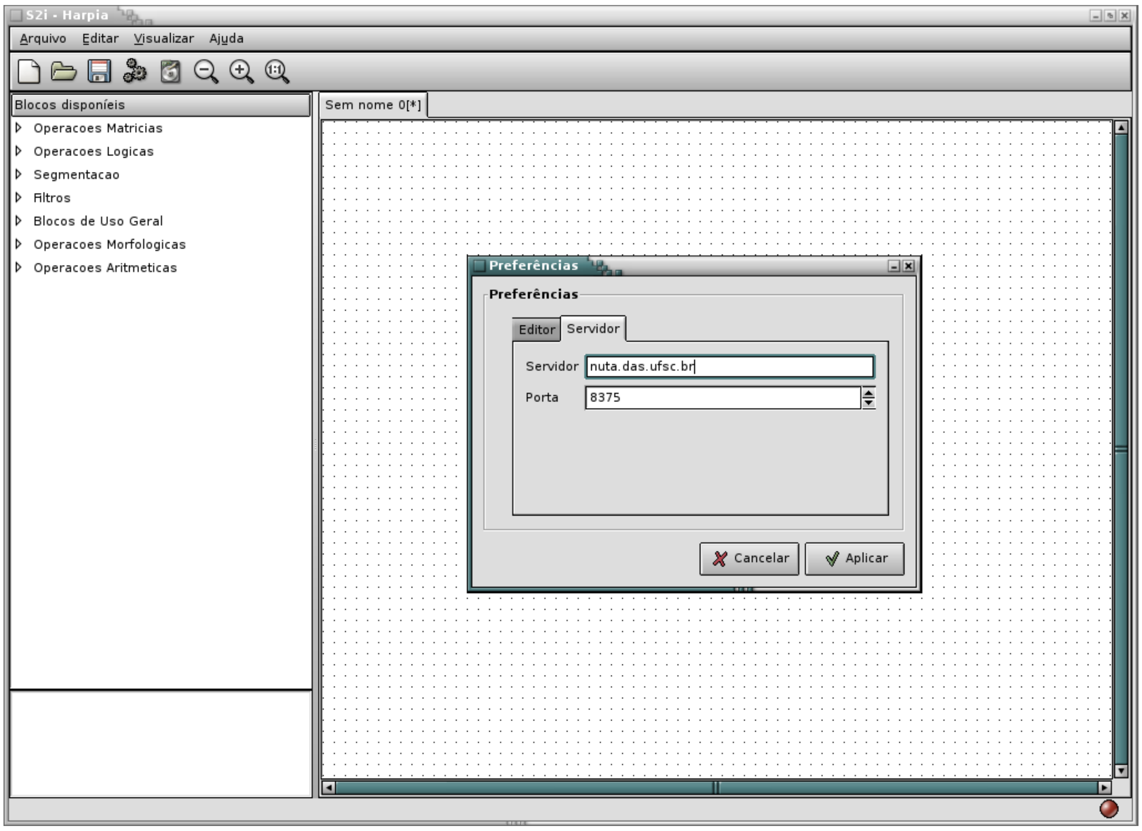
[1] http://nsis.sourceforge.net

3.3 GNU/Linux

The installation under GNU/Linux is provide as a desktop icon. So, it is only needed to drag and drop it at the desktop area and choose the installation path. By default, the path will be a folder inside the home directory. However, it is also possible to install *Harpia* at the system as a root user.

3.4 Harpia Server

As commented previously, the server can run in another machine. Our group left a server running at `nuta.das.ufsc.br` at port 8053 for tests. By default, the installation procedure installs the server and configure it properly to run at the same machine as a localhost service. Others server can be used; to do this, go to **editar** → **preferências** → **servidor** and type the address and port number of the desired server as the figure ?? show.



Preferências

Servidor nuta.das.ufsc.br
Porta 8375

Figura 3.1: Set-up a different server.

4 Harpia components

This section show the Graphic User Interface - GUI of the *Harpia* project. First it gives some description about the implemented blocks and how to construct image processing chains with those blocks.

4.1 Available blocks

For this initial version, only some really important blocks were implemented. For future versions, it is planned that an automatic tool will created an insert user specific blocks, in this case the user will have the option to create complex or even huge image processing chains and aggregate it over one simple block.

4.1.1 Block options

Each block has a common set of operations which are divided into properties, appearance and help about the block functionality.

Properties - This section is a block specific area where it is possible to change the defaults values of the mathematical operation provided by the block in question. As a universal option, the *disable box checker*[1] is available and if marked, the *Harpia* server will not execute this block. This mean that the input provided to the block pass right away through it. If the block has more than one input, like those listed at arithmetical operations, the choosen input that pass to the output is always the first one.

Appearance - This are provide color pallets to change the background and border color. By default, all blocks that belongs to the same set of operations have the same color.

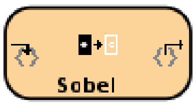
[1]This option is not working right now

Help - It gives a basic description and a more advanced help about how to set the parameters properly in order to achieve the desired result.

4.1.2 Filters

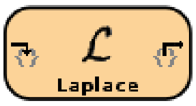
The blocks from this category implements some well-known filters that use common algorithms of image analysis. The blocks are Sobel, Laplace, Smooth, Canny.

Sobel



Filtering operation wich uses one sobel mask to enhance corners of the image. The sobel filtering looks forward to enhance the image regions where it is possible to find strong changes of luminosity. To do this, it makes the calculus of the first, second and third derivative, showing the objects corners. This derivative is based on one combination of smooth gaussin and image differentiation filters, which turns the sobel result pretty good even for images with heavy noises.

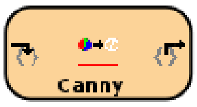
Laplace



The function cvLaplace calculates Laplacian of the source image by summing secon dx- and y- derivatives calculated using Sobel operator:

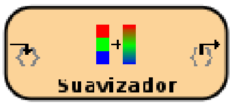
$$dst(x,y) = \frac{dsrc^2}{dx^2} + \frac{dsrc^2}{dy^2}$$

Canny



This block returns the edges on the input image using the Canny algorithm. The smallest of threshold1 and threshold2 is used for edge linking, the largest - to find initial segments of strong edges.

Smooth



Performs smoothing of the image in one of the following ways:

CV_BLUR (simple blur) Summation over a pixel param1xparam2 neighborhood with subsequent scaling by $\frac{1}{param1xparam2}$

CV_GAUSSIAN (gaussian blur) Convolving image with param1xparam2 Gaussian kernel.

CV_MEDIAN (median blur) Finding median of param1xparam1 neighborhood (i.e. the neighborhood is square).

4.1.3 Morphological operations

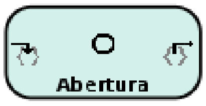
Erode

The block erodes the source image using the specified structuring element that determines the shape of a pixel neighborhood over which the minimum is taken:
dst=erode(src,element):
dst(x,y)=min((x',y') in element))src(x+x',y+y')
The function supports the in-place mode. Erosion can be applied several (iterations) times. In case of color image each channel is processed independently.

Dilate

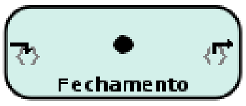
The block dilates the source image using the specified structuring element that determines the shape of a pixel neighborhood over which the maximum is taken:
dst=dilate(src,element):
dst(x,y)=max((x',y') in element))src(x+x',y+y')
The function supports the in-place mode. Dilation can be applied several (iterations) times. In case of color image each channel is processed independently.

Opening



Opening is the morphological transformation equivalent to dilate a image after eroding it.

Closing



Closing is the morphological transformation equivalent to erode a image after dilating it.

4.1.4 Segmentation operations

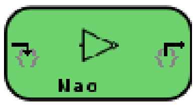
Threshold

Limiriarizacao

Creates the binarization of the image based on fixed value of luminosity (threshold value).

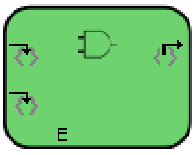
4.1.5 Logical operations

Not



Calculates the logical operation NOT of the light intensity values in all image elements, resulting the negative of the image.

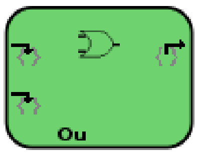
And



Calculates the logical operation AND with the light intensity values of all elements in both images, pixel per pixel. For this block, there are two possibilities:

- Executes the operation between two images element per element.
- Executes the operation between a constant value and each image element.

Or



Calculates the logical operation OR with the light intensity values of all elements of the two images, pixel per pixel. For this block, there are two posibilities:

- Executes the logical operation 'OR' between the two images pixel per pixel.
- Executes the operation between a constant value and each image pixel.

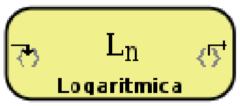
Xor

Calculates the logical operation XOR with the light intensity values of all elements of two images, pixel per pixel. For this block, there are two possibilities:

- Execution of the operation between two images, pixel per pixel.
- Execution of the operation between a constant value and each image pixel.

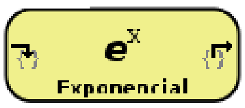
4.1.6 Matricial operations

Logarithmic



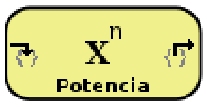
The Logarith consist in calculate the natural logarithm (ln) of the light intensity value of every pixel. ResultImage = ln(OriginalImage) Commonly used to enhance images with high light intensity.

Exponential



The exponent consists in powering the constant (e) to the light intensity value of each image pixel. ResultImage = $e^{OriginalImage}$ Commonly used to enhance images with low light intensity.

Power



The POW consist in raising each pixel to a fixed power value. ResultImage = $(OriginalImage)^{exponent}$ Commonly used to enhance images with low light intensity.

Power Value : It is the exponent wich all pixels will be powered.

4.1.7 General operations

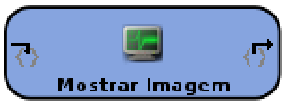
Acquistion

Creates a new image from the filename selected on the block's preferences.

Save

Saves the input image to a file.

Show



Creates a terminal for viewing the image at that branch of the processing tree. Click Twice to see image.

Histogram

Plots the normalized histogram for the selected channel(s)

Histogram equalizer

Equalizador de Histograma

The algorithm normalizes brightness and increases contrast of the image. This block equalizes the histogram of the input image using the following algorihm (one time for each channel):

- Calculate histogram H for src.
- Normalize histogram, so that the sum of histogram bins is 255.
- Compute integral of the histogram: $H'(i) = \int_{j=0}^{i} H(j)$
- Transform the image using H' as a look-up table: $dst(x,y) = H'(src(x,y))$

Color conversion

Conversao de Cores

It converts the color format to another one.

Decomposition RGB

Decomposicao RGB

Extract the R, G and B channels.

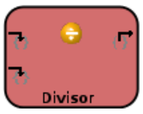
Composition RGB

Composicao RGB

Join the R, G and B channels into a unique image.

4.1.8 Arithmetical operations

Division

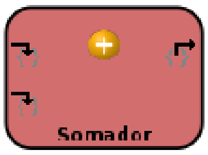


It takes the intensity of each pixel of both images and divide them.

Multiplication

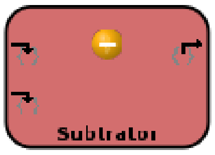
Performs matrix multiplications with the two input images/matrix. It takes the intensity of each pixel of both images and makes the multiplication between them.

Sum



It takes the intensity of each pixel of both images and sum them.

Subtractor



It takes the intensity of each pixel of both images and subtract them.

4.2 Interface overview

This section gives a small overview about the *Graphic Interface* of Harpia.

4.2.1 Toolbar overview

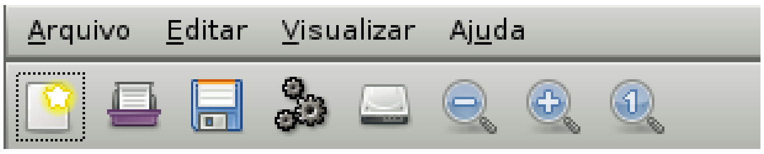


Figura 4.1: Toolbar screenshot.

This button allows to create a new work area.

This button allows to open a image processing chain inside a new diagram.

This button allows to save the current diagram blocks. The diagram that already have been saved does not have the [*] signal after the name showed at the top of the current tab.

This button send a request to the Harpia server properly configured in properties in order to process the current image processing chain. After click this button, it is possible to see the results which can include images and the source-code created.

This button creates a zip package with the files from the server. Inside it is possible to find the generated C++ source-code, the Makefile, the original image, the process images along others files like the image processing chain in its XML format.

By clicking this button, it is possible to zoom out the entire work area.

By clicking this button, it is possible to zoom in the entire work area.

Returns the document to the original size.

4.2.2 Short keys

The complete list of short-keys include:

Ctrl+N Create a new work area

Ctrl+O Open a new image processing chain

Ctrl+W Close the work area

Ctrl+S Save the current image processing chain

Ctrl+Q Close the application

Ctrl+X Cuts the selected region of the work area

Ctrl+C Copy the selected area to a temporary buffer

Ctrl+V Paste the contents of the temporary buffer

Ctrl+P Process the current image processing chain