

Image Knowledge Builder

Image Learning and Recognition
powered by NeuroMem network

Version 4.2
Revised 06/04/2019



A. CONTENTS

1	INTRODUCTION	4
1.1	WHAT CAN I DO WITH IMAGE KNOWLEDGE BUILDER?	4
1.2	A COMPLETE IMAGE LEARNING AND VALIDATION FRAMEWORK	5
	A SIMPLE USER INTERFACE	5
2	GETTING STARTED.....	6
2.1	INSTALLATION.....	6
2.2	PRACTICE WITH EXAMPLE PROJECTS	6
2.3	METHODOLOGY STEP-BY-STEP	6
3	EXAMPLE PROJECTS	7
3.1	GENERAL INSTRUCTIONS.....	7
3.2	BGA INSPECTION	7
3.3	FACE RECOGNITION ON SOCCER TEAMS AND THE FERRET DATABASE	8
3.4	INKJET CHARACTER RECOGNITION.....	8
3.5	GLASS SURFACE ANOMALY DETECTION.....	9
3.6	WAFER TEMPLATE INSPECTION.....	9
3.7	INTRODUCTION TO IMAGE COMPRESSION WITH NEUROMEM	10
3.8	KANJI CHARACTER RECOGNITION	11
4	EXPERT SETTINGS	12
4.1	DEFINE YOUR CATEGORIES AND ASSOCIATED COLOR TAGS	12
4.2	ROI SIZE	13
4.2.1	Default 16 x16.....	13
4.2.2	Example #2: Part inspection.....	13
4.2.3	Example #1: Face detection.....	13
4.3	FEATURE EXTRACTION	14
4.3.1	SubSamples.....	15
4.3.2	Histograms.....	16
4.3.3	Horizontal, Vertical or Composite Profiles	16
4.4	AMPLITUDE NORMALIZATION (OPTIONAL).....	17
4.5	MINIMUM AND MAXIMUM INFLUENCE FIELDS	17
5	SUPERVISED LEARNING.....	18
5.1	ANNOTATE DISCRETE OBJECTS	18
5.2	ANNOTATE SURFACE	18
5.3	AUTOMATIC COUNTER EXAMPLES	18
5.4	LEARN THE ANNOTATIONS.....	19
5.4.1	Undo Learning.....	19
5.4.2	Automatic Learning	19
5.4.3	Automatic re-scaling.....	19
5.5	REVIEWING ANNOTATIONS	19
5.6	ANNOTATION FILE MANAGEMENT	19
5.7	ANNOTATION FILE FORMAT	20
5.7.1	Header	20
5.7.2	Line entry.....	21
6	KNOWLEDGE BUILDING	22
6.1	REVIEWING THE KNOWLEDGE CONTENT	22
6.2	UNDO LEARNING	22
7	RECOGNITION AND VALIDATION.....	23
7.1	LOCAL CLASSIFICATION	24

7.2	REGIONAL CLASSIFICATION	24
7.3	OBJECTS, CLUSTERS OR ANOMALIES.....	25
7.3.1	<i>Result files</i>	25
7.4	MAP OF CATEGORIES, DISTANCES OR IDENTIFIERS	25
7.5	MAP OF MODELS	25
7.6	COPING WITH CHANGE OF SCALE.....	25
7.6.1	<i>Alternative #1</i>	25
7.6.2	<i>Alternative #2</i>	26
8	SEMI-SUPERVISED LEARNING	27
8.1	CODEBOOK GENERATION.....	27
8.2	INFLUENCE OF THE MAXIF.....	28
9	PROJECT AND KNOWLEDGE FILES.....	29
9.1	COGNISIGHT PROJECT FILE.....	29
9.2	PREFERENCE FILE	29
10	TIMINGS	30
11	APPENDIX B: WHAT IS NEW?.....	31
11.1	IKB VERSION 3.3	31
11.2	IKB VERSION 2.7	31
11.3	IKB VERSION 2.6	31
11.4	IKB VERSION 2.5.4	31
11.5	IKB VERSION 2.5.1	32
11.6	IKB VERSION 2.4.....	32
11.7	IKB VERSION 2.3	32
12	TROUBLESHOOTING	34
12.1	UNABLE TO LOAD DLL “COGNISIGHT_XXX.DLL”	34

Image Knowledge Builder is a product of General Vision, Inc. (GV)

This manual is copyrighted and published by GV. All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of GV.

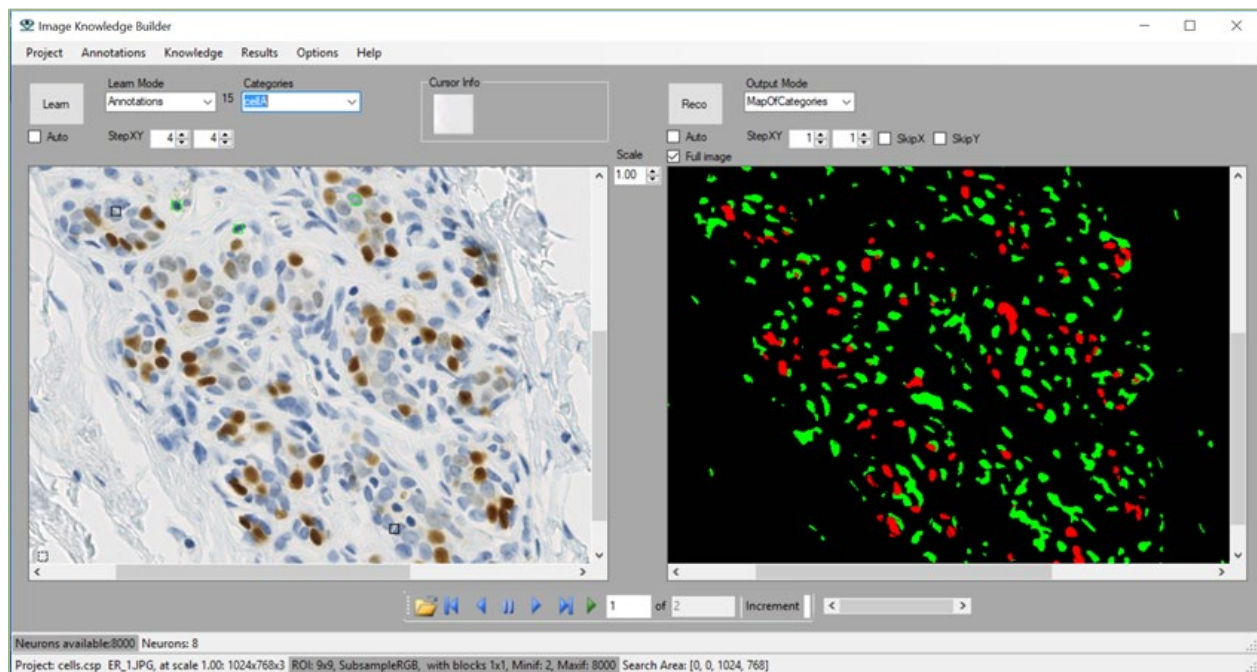
For information about ownership, copyrights, warranties and liabilities, refer to the document [Standard Terms And Conditions Of Sale](#) or contact us at www.general-vision.com.

1 INTRODUCTION

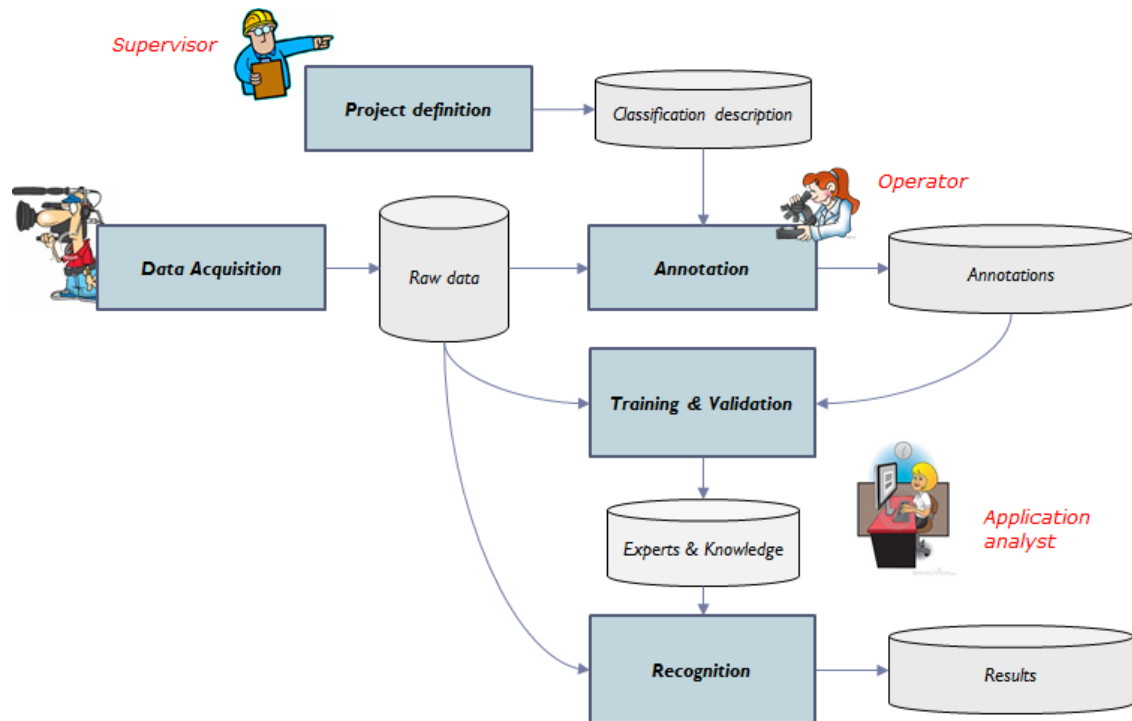
Image Knowledge Builder is a simple framework to train and validate a NeuroMem network to classify the content of your images.

1.1 What can I do with Image Knowledge Builder?

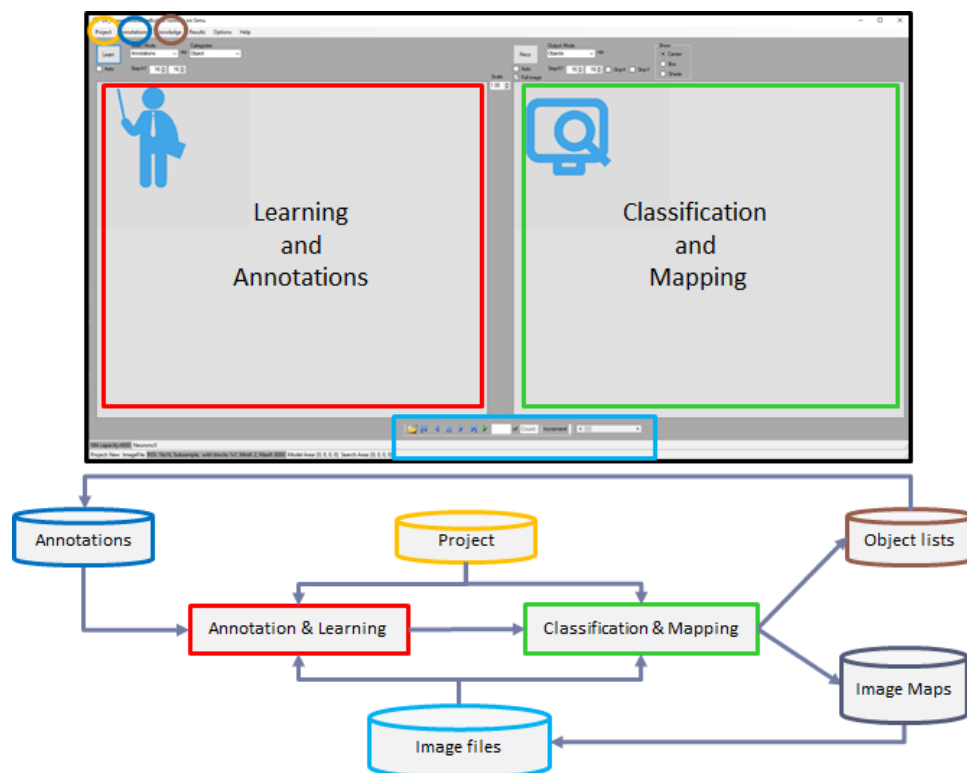
- Load and annotate objects and surfaces for a supervised learning
 - o Edit and color label your categories of objects
 - o Edit and save annotations in your images
 - o Select features of interest to classify visual objects
- Train the neural network
 - o Edit learning settings such as level of conservatism of the neurons, use of automatic counter examples, etc.
 - o Start a training session from scratch or add knowledge to an existing neural network
 - o Train in a step by step or batch mode
- Classify images and generate a variety of outputs
 - o Tag the recognized objects in the image
 - o Export X,Y location and categories of the recognized objects in text format
 - o Display maps of the hit locations, category locations, confidence levels
- Save the knowledge of the neurons for future use or portability to other NeuroMem platforms



1.2 A complete Image Learning and validation framework



A simple User Interface



2 GETTING STARTED

2.1 [Installation](#)

- Double click at the setup.exe file and follow the instructions on the screen.
- After the display of the panel “Installing GV Image Knowledge Builder...”, you may get a pop-up message asking you to confirm the installation of the software from an Unknown Publisher. Click Yes to proceed.
- Upon termination, click at Start menu \All Programs\General Vision\Image Knowledge Builder

2.2 [Practice with example projects](#)

Image Knowledge Builder is supplied with example projects and their accompanying images and videos. They are usually organized per topic and the name of the project file matches the name of the folder where you will find the images used for the training and the verification. Our example projects are described in a [next chapter](#).

2.3 [Methodology Step-by-Step](#)

- Select a folder of images to use for training
 - o Default location c:\MyDocuments\General Vision\Images
- Select a project file if any already exist for your application
 - o Default location c:\MyDocuments\General Vision\Projects
- Observations:
 - o The status bar at the bottom of the screen reports the settings of the project and the size of the knowledge (i.e. number of committed neurons)
 - o If any annotation file has the same root name as the project in the image folder, the annotations associated to the image on screen are displayed.
- Possible actions on the first image
 - o Learn existing annotations
 - o Edit and learn new annotations
 - o Recognize a region or the entire image
- Go to next image to observe the recognition results obtained with the current knowledge
- Possible actions on the next image
 - o Learn existing annotations to teach the neurons with new examples
 - o Edit and learn new annotations
 - o Recognize a region or the entire image
- Observations:
 - o If the scanning steps are too high, the report may skip some locations which should be recognized positively
 - o Check Auto recognition and Full Image option allow to automate the search across multiple images
 - o Annotations are saved to a log file as soon as they are learned. This log file can be saved for later or further use
- Try building a new knowledge delivering better accuracy using the same annotations, but a different feature
 - o Go back to the first image
 - o Clear the current knowledge
 - o Open the Project/Settings menu and change your parameters
 - o Proceed again with the learning of annotations

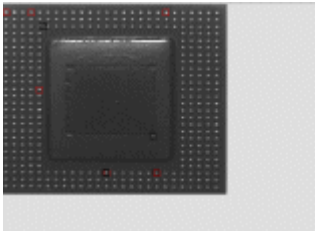
3 EXAMPLE PROJECTS

3.1 [General Instructions](#)

The proposed examples can be reviewed in a few minutes as follows:

- Use the navigation bar to open an image stored in C:\\yourUser\\Documents\\General Vision\\Images**Topic**.
- Select File\\Open Project from the Main menu and look for an existing *.prj file with the **Topic** label.
- A summary of the project definition is displayed in the Status bar at the bottom of the screen: size of the knowledge, type of feature extraction, image scanning mode.
- If neurons are committed, their models can be viewed in the View/Knowledge menu, or you can click directly at the Reco button to view the results of the recognition.
- If annotations for the project exist in the selected folder of images, they are overlaid in the image.
- If no annotations exist for the selected image, you can edit your own manually.
- You can click at the Learn button and when done at the Reco button.

3.2 [BGA inspection](#)



Observe how the powerful non-linear classifier of a NeuroMem network is a simple remedy to learn a diversity of examples in a few mouse clicks.
Learn a variety of good solder balls and get their total count immediately.
Try to achieve the same results in less than a minute with a threshold technique!

1. Click Browse in the Navigation toolbar and load the image stored in Images/BGA folder.
 - a. Assuming that the Learn method is set to the default "Annotations", the program should automatically find and load the annotations associated to this image and stored in the file BGA_GT.txt stored in the same folder.
 - b. Note that the annotations include "background" annotations (in black). They behave as counter examples to ensure that the neurons learning solder balls (in red) do not over generalize too much.
2. Select Load project in the File menu and load the file bga.prj stored in the Projects folder
 - a. In the status bar at the bottom of the screen, notice that the knowledge is empty. The size of the region to learn and recognize is 12x12 pixels and the feature extraction a monochrome and subsample.
3. Checked the Auto checkbox to automatically recognize the image when a Learn operation or Load image is executed. Verify that the Reco method is set to Objects.
4. Click Learn to learn the annotations seen on in the learning panel. Verify that all the solder balls are highlighted in the right panel. Some appear more reddish than other when a solder ball is recognized at 2 adjacent positions.

3.3 [Face recognition on soccer teams and the ferret database](#)

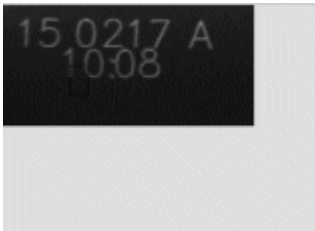


Observe how the model generator of a NeuroMem network can automatically over-generalize and correct mistakes if applicable.

The network taught with 7 faces of soccer players from a team can detect the faces in four other teams but also the faces from the ferret database!

1. Click Browse in the Navigation toolbar and load the image stored in Images/Faces_SoccerPlayers folder
2. Select Load project in the File menu and load the file faces_soccerplayers_kn7.prj stored in the Projects folder
 - a. In the status bar at the bottom of the screen, notice that the knowledge is already built and consists of 7 neurons. The size of the region used to learn and recognize the characters is 27x 36 pixels. The feature extraction a monochrome normalized subsample.
3. The knowledge can be displayed by clicking "View Knowledge" button.
4. Click the Reco button to inspect the entire image. Note that this can take a few seconds in simulation.
5. Select the Auto checkbox next to the Reco button. Go to the next image by clicking at the ">" button in the navigation bar. Review the results. Proceed to next image.
6. Using the navigation bar, select the first image supplied in the Images/faces_ferret folder. The Ferret database is a known repository of faces looking at the camera from approximately the same distance. The region of interest is clearly too small in this case to cover the eye and nose area of the person. In the navigation bar, reduce the image by a scale 1/3 and you should observe that the face is then detected.
7. Go through several images of this same folder by clicking at the ">" button in the navigation bar.

3.4 [Inkjet character recognition](#)



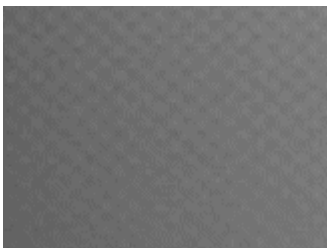
Observe how the feature extraction can impact the ease of deployment of an application, and particularly reduce the number of examples to teach without compromising the accuracy of the recognition.

Image Knowledge Builder has been designed so you can easily find the best feature extraction for your application using the same images and annotations.

1. Click Browse in the Navigation toolbar and load the first image stored in Images/OCR_inkjet folder: Im_110217A_1008.jpg.
 - a. Assuming that the Learn method is set to the default "Annotations", the program should automatically find and load the annotations associated to this image and stored in the file Im_110217A_1008_GT.txt stored in the same folder.
2. Select Load project in the File menu and load the file ocr_inkjet.prj stored in the Projects folder
 - a. In the status bar at the bottom of the screen, notice that the knowledge is empty. The size of the region to learn and recognize is 36x60 pixels and the feature extraction a monochrome and normalized subsample.
3. Checked the Auto checkbox to automatically recognize the image when a Learn operation or Load image is executed. Verify that the Reco method is set to Objects.
4. Click Learn to learn the annotations seen on in the learning panel. Verify that the digits are highlighted with the proper categories in the right panel. The color legend can be seen and changed in the Options/Settings menu.

5. Click at the Next arrow in the navigation bar to proceed with the next image. Notice that on the third image, a new annotation appears for the digit "6" which has still not been taught. Click Learn to add it to the current knowledge composed of 7 neurons as reported in the status bar. You can review the results on the next and previous images navigating through the various images supplied in the OCR_inkjet folder.
6. Now, let's repeat the same routine, but with a slightly different feature extraction:
 - a. Erase the knowledge by selecting the Clear command in the Knowledge menu
 - b. Open the Options/Settings panel and uncheck the "Normalize" checkbox. This means that the amplitude of the subsample will not be expanded to the range [0, 255].
7. Load the first image again with its annotations, learn and review the results of the recognition:
 - a. Select Options/Details at cursor and move the cursor over the image
 - b. Observe the number of misfiring such as in (320,90)
 - c. Observe how the number of uncertainties such as in (54,70)

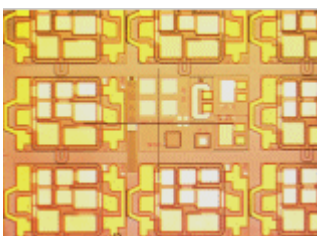
3.5 Glass surface anomaly detection



Observe how the neurons can autonomously model a good texture, even a patterned texture such as a solar glass panel, and later be used to detect anomalies in new images which are basically locations which do not look similar to the models they hold in memory.
A simple illustration of the power of the NeuroMem RBF classifier to detect novelty which in this case represents the occurrence of an anomaly.

1. Change the data source in the Navigation toolbar to "Image" and load the Glass0_good.png file stored in Images/Surface_Glass folder
2. Select Load project in the File menu and load the surface_glass.prj file stored in the Projects folder
 - a. In the status bar at the bottom of the screen, notice that the knowledge is empty. The size of the glass area to learn and recognize is 16x16 pixels. The feature extraction a monochrome subsample.
3. Select the "Codebook" learning mode to build the knowledge and click the Learn button. Using the Maxif of 1500, and a learning step of 16 along the horizontal and vertical axes, 12 neurons are committed.
4. Select the "Anomalies" recognition mode, check the Auto checkbox and click the Reco button.
5. No anomaly is reported in the good image.
6. Go to the next image by clicking at the ">" button in the navigation bar. Anomalies should be reported in the other three images.
7. You can practice with the same routine using bigger or smaller values of the maximum influence field and observe the differences in the number of committed neurons and number of discrepancies identified in recognition.

3.6 Wafer template inspection



Observe how a large NeuroMem network can be used to store an entire reference image (wafer, printed circuit, printed material), and later be used to report locations in new images which do not match the right pattern at the right position.
A simple illustration of an unsupervised learning and high-speed template matching.

1. Click Browse in the Navigation toolbar and load the image Good.tiff stored in Images/Wafers folder
2. Select Load project in the File menu and load the file wafer_inspection.prj stored in the Projects folder

- a. In the status bar at the bottom of the screen, notice that the knowledge is empty. The size of the region to learn and recognize is 16x16 pixels and the feature extraction with the value 3 means that the subsample includes the color information. The maximum influence field of 2000 is an important parameter in this example and tunes the conservatism of the neurons.
3. Select the "Golden template" learning mode to build the knowledge and click the Learn button. Using the Maxif of 2000, and a learning step of 16 along the horizontal and vertical axes, 674 neurons are committed.
4. Select the "Anomalies" recognition mode, check the Auto checkbox and click the Reco button.
5. No anomaly is reported in the good image.
6. Go to the next image by clicking at the ">" button in the navigation bar. Anomalies should be reported in the other three images.
7. You can practice with the same routine using bigger or smaller values of the maximum influence field and observe the differences in the number of committed neurons and number of discrepancies identified in recognition.

3.7 [Introduction to image compression with NeuroMem](#)



This example illustrates the use of an unsupervised learning method to build a codebook of reference codes over one or multiple images and reconstruct simpler images by replacing the original blocks by their closest matches.

1. Click Browse in the Navigation toolbar and load the first image stored in Images/MiscBWImages folder
2. Select Load project in the File menu and load the file compression.prj stored in the Projects folder
 - a. In the status bar at the bottom of the screen, notice that the knowledge is empty. The size of the region to learn and recognize is 9x9 pixels and the feature extraction is a monochrome subsample includes the color information.
3. Select the "Codebook" learning mode to build the knowledge and click the Learn button. Using the Maxif of 2000, approximately 50 neurons are committed.
4. Select the "MapOfModels" recognition mode, check the Auto checkbox and click the Reco button.
5. A simplified image based of only 50 patches is generated.
6. Go to the next image by clicking at the ">" button in the navigation bar and observe the same compression effect.
7. If blocks appear in black in an image, click Learn. A few neurons should be added to the knowledge. They correspond to the blocks not recognized in the previous codebook.
8. You can practice with the same routine using smaller values of the maximum influence field and observe the differences in the number of committed neurons and the reduction of the error between the source and its Transform image.
9. You can practice with the same routine using smaller patches of pixels and observe a finer granularity and lesser error in the Transform images.
10. You can practice with the same routine using the images of the MiscColorImages folder. Change the feature to Color Subsample in the Options/Settings menu if you wish to build a codebook of color patches.

3.8 Kanji Character recognition



This example demonstrates that the NeuroMem network is (1) a good classifier which can easily discriminate between many models, and also (2) the recognition time is not impacted by the number of models which constitute the knowledge base (360 in this case).

Warning! If using a simulation of the NeuroMem network, the recognition may take several seconds. On the small image, the window of interest is moved across 68,432 positions and at each position, the block of pixels (256 bytes) is compared with 360 models. Assuming that each comparison between an input pixel and the same pixel stored in a neuron takes 4 operations, the recognition of the entire image is equivalent to 25.2 billions of operations. The CM1K chip will execute this in 2 seconds or 12.5 Giga ops.

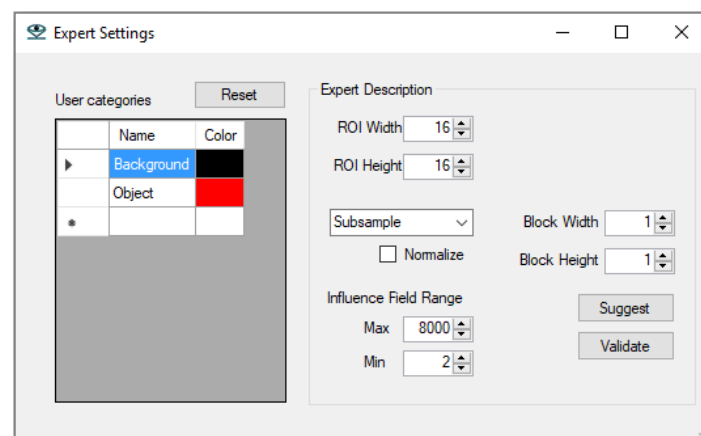
1. Click Browse in the Navigation toolbar and load the image stored in Images/Kanji folder
2. Select Load project in the File menu and load the file kanji360.prj stored in the Projects folder
 - a. In the status bar at the bottom of the screen, notice that the knowledge is already built and consists of 360 neurons. The size of the region used to learn and recognize the characters is 16x16 pixels. The feature extraction with the value 0 corresponds to a monochrome subsample.
3. Select Options/Overview at Cursor. Move the cursor over a character and view the result of the recognition in the Cursor Info box of the main panel. Note that positive recognition of a character only occurs when well centered over it.
4. The knowledge can be displayed by clicking “View Knowledge” button. Note that in this case, each character is represented by 3 models shifted by one pixel each. This training was done intentionally to allow the detection of a character at different offsets.
5. Click the Reco button to inspect the entire image. Note that this can take a few seconds in simulation, since the image is entirely scanned with a step of 1 pixel per column and 2 pixels per row, and at each position the block of 16x16 pixels is matched against 360 models.
6. Review the results in the Options/Results window

4 EXPERT SETTINGS

This version of Image Knowledge Builder allows to define and validate a single expert at a time.

An expert is defined in the Settings panel and described with the following parameters:

- Nominal size of the region of interest (ROI) to learn and recognize at different locations and across the images
- The signature or feature to extract for the characterization of the RO
- Its calculation parameters if applicable
- The option to normalize the amplitude of the resulting signature so it is distributed between values 0-255
- The Min and Max Influence field of the neurons trained by this expert. These 2 values depend on the dimension represented by the feature vector.



4.1 [Define your categories and associated color tags](#)

A new project starts with a single generic category labeled “Object”, the other option being the “Background” category or “null” category. Names can be changed and validated by typing the Enter key. Depending on the colors of your images, you can change the colors which will be used to highlight your objects. For example, if your image has a red background, you may want to select a blue or green color to highlight your objects. Click at the color cell and select a new color in the color wheel.

Examples of categories for a part inspection

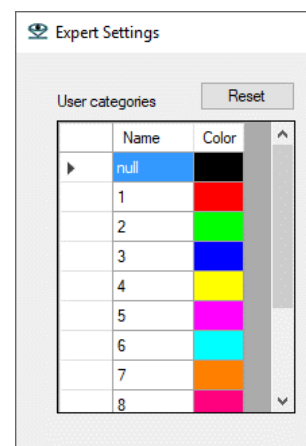
- Acceptable (category 1)
- Recyclable (category 2)

Or

- Grade A (category 1)
- Grade B (category 2)
- Grade C (category 3)

Examples of categories for a traffic monitoring application:

- Car (category 1)
- Truck (category 2)
- Van (category 3)



REMARK: If a project uses the BuildCodebook function to build a knowledge, the neurons automatically assign category values to the codes. As a consequence, any category you edit under this Settings panel will be discarded except for the “Background” category or category value 0.

4.2 [ROI Size](#)

The Region Of Interest (ROI) is the smallest rectangular area including your object or a discriminant portion of your object. It should be limited to the smallest discriminant portion of the object to recognize for the following reasons:

- The smaller, the less need to compress the information or extract a feature
- The smaller, the faster the readout of the pixel values
- Exclude as much as variability as possible

4.2.1 [Default 16 x16](#)

The **default ROI size** is 16x16 which is a reasonable size for an iconic representation of simple object.

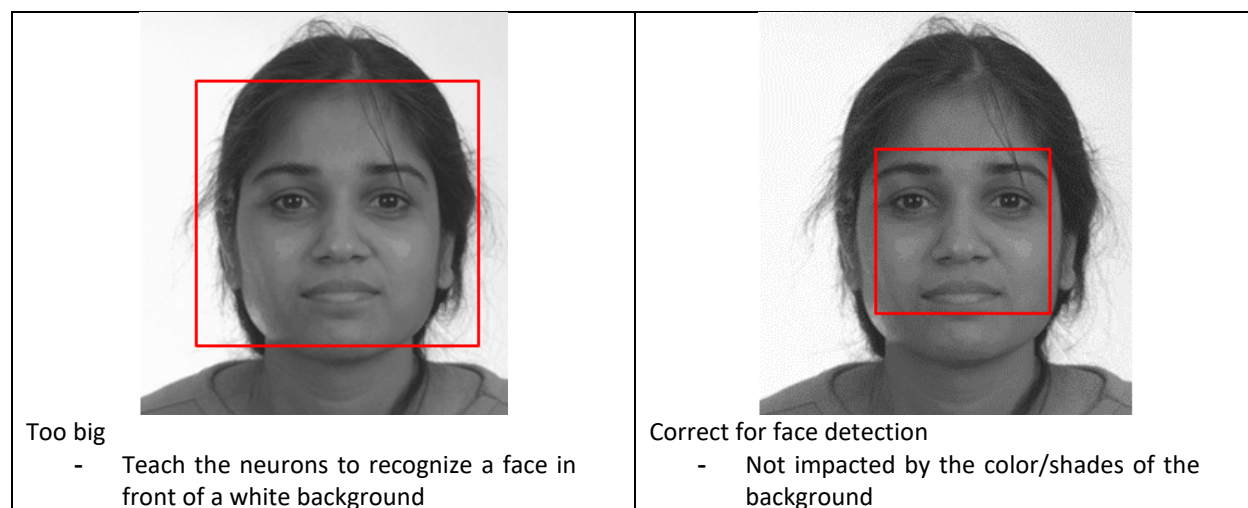
Indeed, 16x16 pixels can be assembled “AS IS” in a vector of 256 monochrome values. In this case, the subsample feature simply uses the original intensities of all the pixels, with no need to subsample them.

4.2.2 [Example #2: Part inspection](#)

If a mechanical part can be classified as good or defective based on the presence or absence of a screw at a given position in the part, the region of interest should not be defined as the whole part but rather limited to the rectangle where the screw is expected.

The BGA project supplied with the IKB uses an ROI of 12x12. If you load this project and its annotations, clear the knowledge (Knowledge/Clear menu) and change the ROI to 16x16 (Project/Settings menu), you will observe that additional annotations are required to obtain the same satisfactory results.

4.2.3 [Example #1: Face detection](#)



<ul style="list-style-type: none"> - Recognition will fail if this person is placed in front of a painting on the wall. - Impacted by haircut, presence of earrings, beard, etc. 	<ul style="list-style-type: none"> - Not impacted by type of haircut, presence of a beard or earrings
--	--

4.3 [Feature extraction](#)

The feature extraction is the method used to convert the pixel data within the region of interest into a vector of 256 bytes ready for broadcast to the neurons:

Expert Description

ROI Width: 16

ROI Height: 16

Subsample (selected)

Block Width: 1

Block Height: 1

Suggest

Validate

Subsample	Monochrome subsample, or average intensity of up to 256 blocks fitting inside the ROI
Histogram	Grey histogram, or the number of pixels per 256 grey-level values.
HistoCumul	Grey histogram cumulative
SubsampleRGB	Color subsample, or average Red, Green and Blue intensities of up to 85 blocks fitting inside the ROI
HistogramRGB	Color histogram, or sequence of the red, green and blue histograms, each of 85 values.
HistoCumulRGB	Color histogram cumulative
Composite profile	Average of the lines of pixels and columns of pixels
Horizontal profile	Average of the lines of pixels
Vertical profile	Average of the columns of pixels

Some feature extraction functions use input parameters.

The **Suggest** button will suggest values for these parameters. They can be overwritten. In which case the **Validate** button verifies that their values are acceptable and display then in red otherwise.

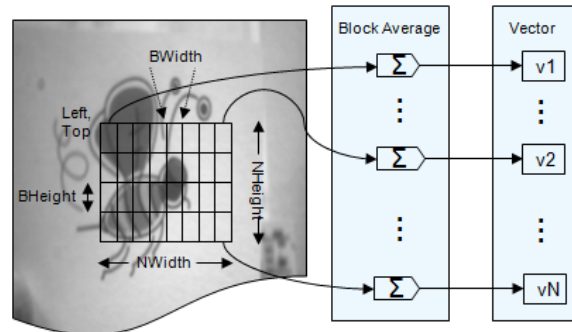
4.3.1 SubSamples

Subsample is a vector which appends the average intensity of blocks of pixels extracted from the region of interest. The blocks are all the same size, but not necessarily square. They are surveyed in a raster displacement and their average intensity is assembled into vector.

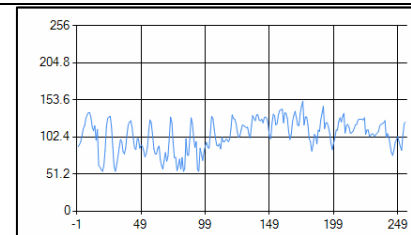
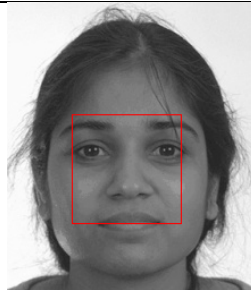
The size of the blocks (BWidth x BHeight) can be calculated by the program or edited manually if you prefer to edit non square blocks or obtain a number of blocks lesser than 256.

The **Suggest** button calculates the maximum size of square blocks which can fit inside the ROI.

The **Validate** button verifies that the extracted feature will fit in 256 bytes for the neurons.



The feature called Subsample extracts the average of the intensity value of each block regardless if the image is monochrome or color. The region must contain less than 256 blocks so the output vector fits in the 256 bytes of memory of the neurons

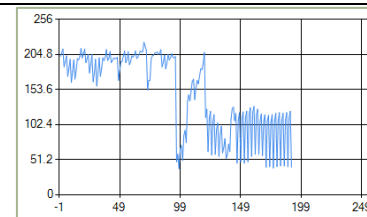


Feature vector reconstructed as a monochrome image



Since the SubSample function generates 1 average value per block and the final vector can only have 256 values, the blocks must be sized such that their number inside the ROI is less than or equal to 256.

The feature called SubsampleRGB extracts the average of the Red, Green and Blue intensities of each block in the ROI, so three average values per block. The ROI must contain less than 85 blocks so 3×85 values fits in the 256 bytes of memory of the neurons.



Feature vector reconstructed as an RGB image



Since the SubSampleRGB function generates 3 average value per block and the final vector can only have 256 values, the blocks must be sized such that their number inside the ROI is less than or equal to 85.

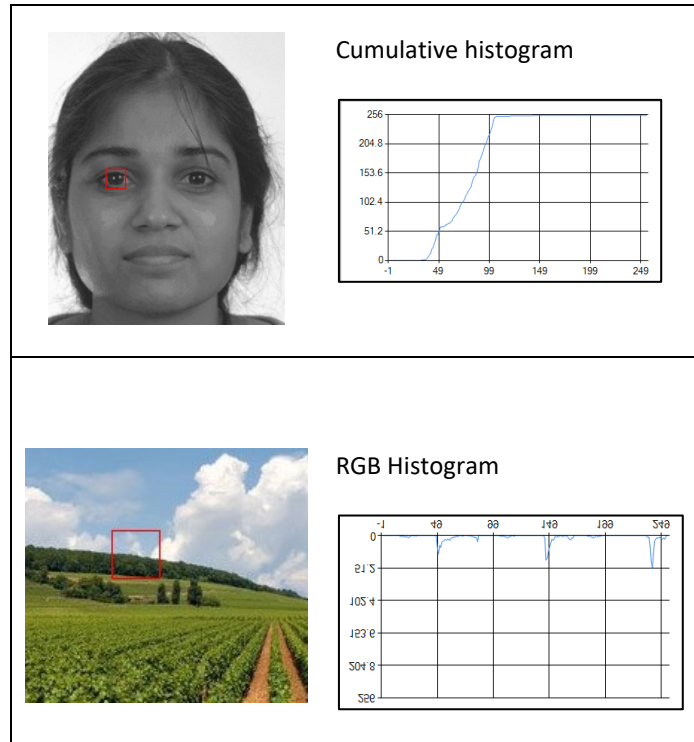
4.3.2 Histograms

Histogram is a vector which gives the distribution of the grey-level values in the region of interest. It indicates the number of shades in the ROI, the presence of noisy pixels and more.

HistoCumul or cumulative histogram is a mapping of the standard histogram which counts the cumulative number of pixels in all of the bins up to the specific bin.

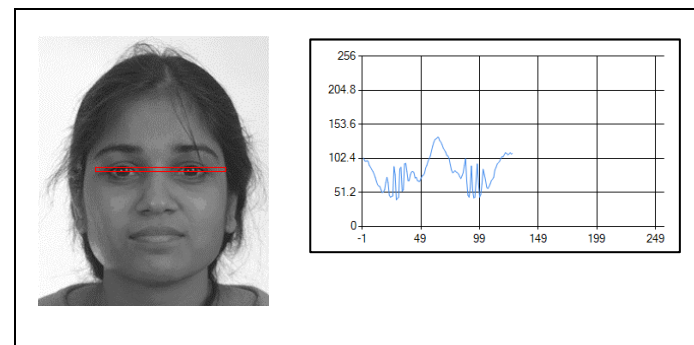
HistogramRGB is a vector which gives the distribution of the Red, Green and Blue intensities in the region of interest. It is assembled as a series of 3 histograms of 85 bins each representing 85 bins for the Red, 85 bins for the Green and 85 bins for the Blue.

HistoCumulRGB or cumulative color histogram is a mapping of the standard histogram which counts the cumulative number of pixels in all of the bins up to the specific bin.



4.3.3 Horizontal, Vertical or Composite Profiles

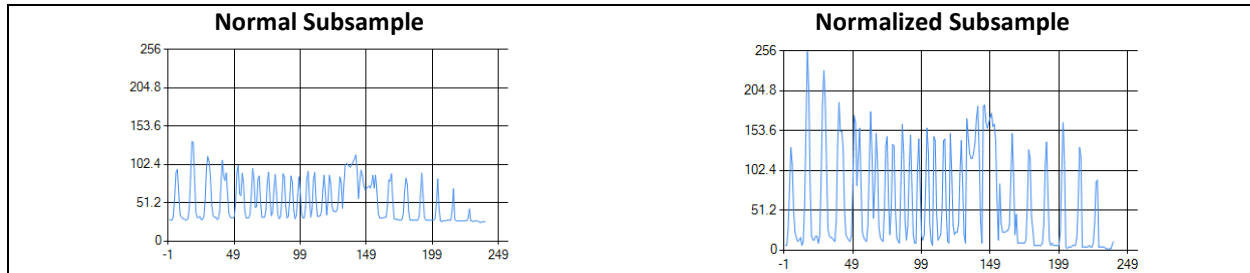
The average intensity along a column or row of pixels, or both appended in a same feature vector. The composite profile can be helpful to classify the alignment of objects. The vertical and horizontal profiles to classify edges and geometric transitions.



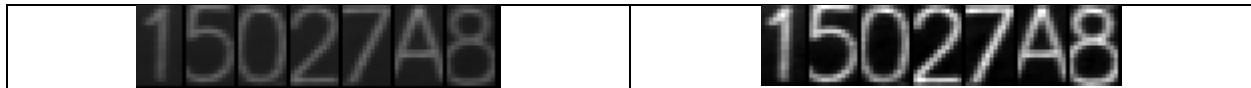
4.4 [Amplitude Normalization \(optional\)](#)

Regardless of the feature you select, you have the option to normalize the amplitude of the feature vector or not.

The normalization amplifies the dynamics of the entire feature vector its values range between [0,255].



The table below shows the subsamples stored in neurons trained to recognize inkjet characters and reconstructed as 2D image. The fact the subsample is normal or normalized changes the contrast of the feature vector.



Positive effect of a Normalization: Practice with the OCR_inkjet project supplied with the IKB.

Negative effects of the Normalization: noisy pixel amplification and therefore signature deterioration.

4.5 [Minimum and Maximum Influence Fields](#)

The Minimum and Maximum Influence Fields are also global variables of the NeuroMem network and their values should be set in relation to the dimension represented by the feature vectors.

MAXIF:

Control the initial level of moderation of the neurons. This limit applies to the neurons which are not yet committed, but ready to learn. It is a maximum value assigned to a newly committed neuron if it does not fall within the influence field of already committed neurons. Otherwise its influence field is set to its distance to the closest committed neuron.

- Range [2, 65534]; Default 16384
- The larger the MAXIF, the more generalization or over-fitting. Tendency to create lesser liberal neurons.
- The lesser the MAXIF, the less generalization. Tendency to create many conservative neurons.

MINIF:

Control the final level of uncertainty of the committed neurons. This limit applies to the neurons which are not yet committed, but ready to learn. It is the minimum value below which the influence field of a newly committed cannot shrink.

- Range [2, MAXIF]; Default 2
- The larger the MINIF, the more uncertain in the decision space. Tendency to create degenerated neurons which overlap on one another.

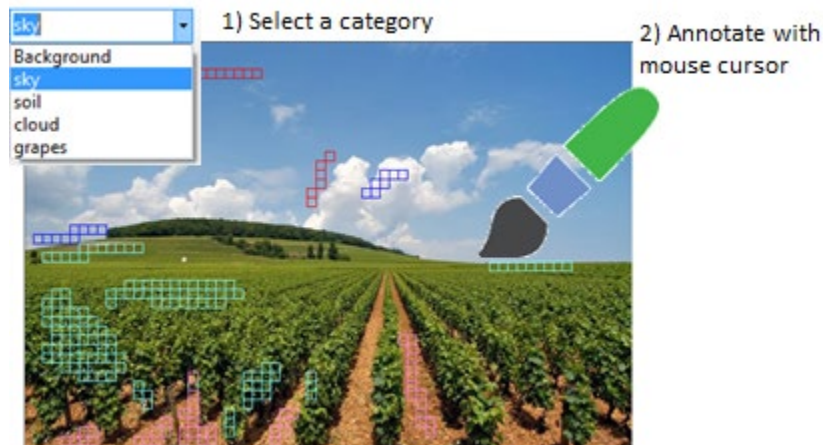
5 SUPERVISED LEARNING

5.1 [Annotate discrete objects](#)

- Select the “Annotations” learning mode
- Select the category of the object in the dropdown list above the Teaching panel
- Move the cursor over the object and Left Click with the mouse button
- To delete, the annotation, press Shift +Left click

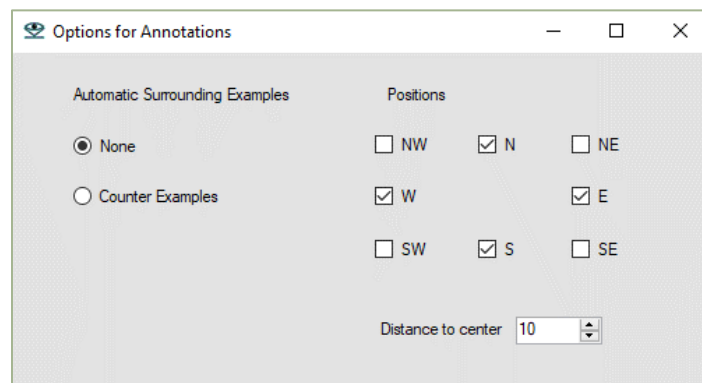
5.2 [Annotate surface](#)

- Select the “Annotations” learning mode
- Select the category of the area in the dropdown list above the Teaching panel
- Draw the area pressing down the right mouse button
- Select the stepX and stepY you want to use when learning an area
- To delete, the annotation, press Shift +Right click



5.3 [Automatic counter examples](#)

When learning discrete objects, you have the option to automatically annotate surrounding background examples to limit overgeneralization. The positions and distance of the counter examples can be defined in the Annotations\Options menu.



5.4 [Learn the annotations](#)

Annotations are learned by clicking the Learn button.

Monitoring of the object recognition in the same image is immediate by clicking the Recognize button.

5.4.1 [Undo Learning](#)

Undo Learning is accessible under the Knowledge menu and allows to cancel the last Learn function by restoring the image of the knowledge prior to that.

5.4.2 [Automatic Learning](#)

If the “Auto” checkbox next to the Learn button is marked, the annotations loaded with an image are automatically learned. Combine this setting with the use of the Play mode in the navigation bar and you can learn automatically all the “annotated images within a given folder.











5.4.3 [Automatic re-scaling](#)

The scale at which the annotations are learned is automatically updated as soon as they are learned. For example, if an annotation entry was recorded for a scale of 1 but the current scale factor is 50%, the Image Knowledge Builder automatically converts the annotation to the 50% scale. This will become the new scale stored in the annotation file after you save the Annotations.

5.5 [Reviewing annotations](#)

The Annotations\Review menu displays all the annotations associated to the current project and images of the current image folder.

This display allows to screen quickly and pinpoint visually inconsistencies. They can be even filtered per category.

	File	Index	Scale	Type	Left	Top	Width	Height	Category	CatValue	Example
►	crop.png	-1	1	surface	103	31	16	16	sky	1	
	crop.png	-1	1	surface	159	34	16	16	sky	1	
	crop.png	-1	1	surface	391	363	16	16	soil	2	
	crop.png	-1	1	surface	323	371	16	16	soil	2	
	crop.png	-1	1	surface	363	134	16	16	cloud	3	
	crop.png	-1	1	surface	520	114	16	16	cloud	3	
	crop.png	-1	1	surface	147	264	16	16	grapes	4	
	crop.png	-1	1	surface	356	310	16	16	grapes	4	
	crop.png	-1	1	surface	444	352	16	16	grapes	4	
	crop.png	-1	1	surface	212	172	16	16	grapes	4	

5.6 [Annotation file management](#)

Annotations edited in an image are validated when learned or if you save them manually when loading a new image or project. If none of these operations is executed, the annotations are lost.

Annotations accumulated across multiple images located in a same folder can be saved to a single file in this same folder. The file saved with the *.txt extension and named by default after the current project with the suffix _GT.txt (GT stands for Ground Truth).

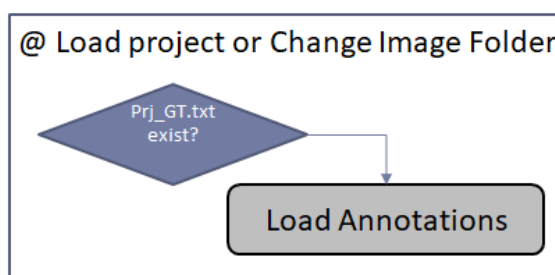
The loading of annotations is executed automatically by the Image Knowledge Builder when

1. A new project is loaded
2. An image is selected in a new folder

If the following rule is verified:

1. Given that the current project is *projectName*
2. Given that the folder where the current image is stored is named *imageFolder*
3. If there is a file *projectName_GT.txt* in *imageFolder* (GT stands for Ground Truth), the annotations of this file are loaded for all the images found present in imageFolder
4. If within this file there are entries related to the current image, the later are displayed on screen after adjustment if applicable to match the scale and object size of projectName.

Annotation files can also be selected manually, but only the annotations of images found in the same folder as the annotation file will be loaded.



Annotations can also be imported manually at any time using the Annotations menu.

5.7 [Annotation file format](#)

5.7.1 [Header](#)

The header is composed of comma delimited lines as follows. The words in bold shall exist AS IS so the file format can be decoded properly by the IKB:

Version, xyz	
CatNbr, xxx	xxx includes the category 0
Name of category 0	
Name of category 1	
...	
Name of category xxx	
CatColors, xxx	
R,G,B of category 0	
R,G,B of category 1	
...	
R,G,B of category xxx	
Annotations, yyy	

Optional

After this end of header, each line represents an annotation with the format described in the next paragraph.

Example of GT annotation file header

Version, 0317

CatNbr,3

Nobody

John

Debbie

CatColors,3

0,0,0

255,0,0

0,255,0

Annotations,8

5.7.2 Line entry

1 line per annotation:

[ImageFileName, user defined index, scale, Annotation Type, Left, Top, Width, Height, Category number]

string imagefileName;

long indexFile = -1; // a user defined index, a frame number in case of movie file, etc.

decimal Scale = 1; // scale at which annotation is edited

int Type = 0; // 0=object, 1=area

int Left = 0;

int Top = 0;

int Width = 0;

int Height = 0;

int Category = 1; //index referring to the list of category names in the header

Example of GT annotation file header

Im_110217A_1008.jpg,0,1,37,28,36,60,1

Im_110217A_1008.jpg,0,1,88,28,36,60,5

Im_110217A_1009.jpg,0,1,160,28,36,60,10

Im_110217A_1016.jpg,0,1,388,32,36,60,11

Im_110217A_1016.jpg,0,1,286,86,36,60,8









Im_110217A_1016.jpg,0,1,126,99,36,60,0

6 KNOWLEDGE BUILDING

6.1 [Reviewing the Knowledge Content](#)

The View Knowledge menu allows reviewing the contents of the neurons, the pattern they hold, their category and active influence field (AIF). This utility can be helpful to understand why some erroneous or uncertain classifications occur.

The AIF is a dynamic attribute entirely controlled by the neuron itself. From the time the neuron is created, its influence field can be reduced as new and contradicting examples are taught.

	NeuronID	Category	Model	Active IF	Degenerated
▶	1	cloud		2686	<input type="checkbox"/>
	2	grapes		12268	<input type="checkbox"/>
	3	soil		6649	<input type="checkbox"/>
	4	soil		6948	<input type="checkbox"/>
	5	soil		8322	<input type="checkbox"/>
	6	sky		8936	<input type="checkbox"/>
	7	soil		8993	<input type="checkbox"/>
	8	soil		7395	<input type="checkbox"/>

6.2 [Undo Learning](#)

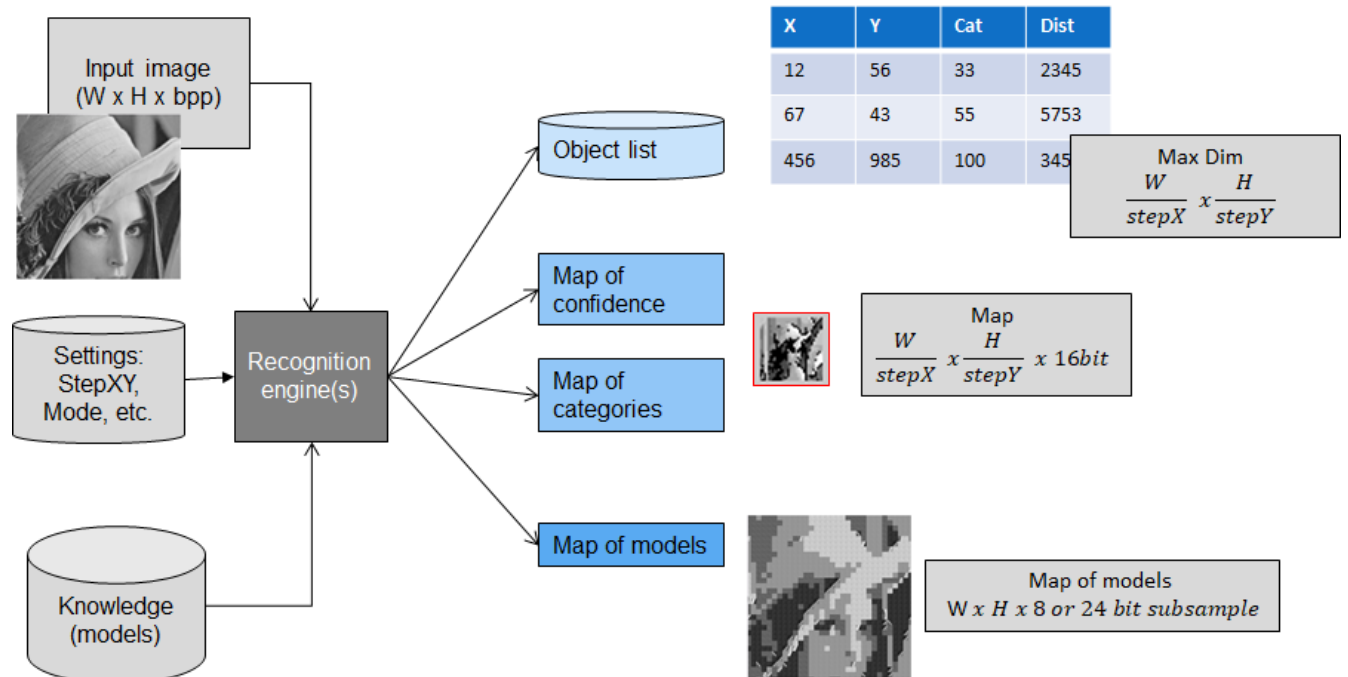
Undo Learning is accessible under the Knowledge menu and allows to cancel the last Learn function by restoring the image of the knowledge prior to that.

7 RECOGNITION AND VALIDATION

The recognition is executed by moving the ROI over the ROS using a raster displacement and a given step along the horizontal and vertical axis.

The IKB can format the results of the scanning with different format:

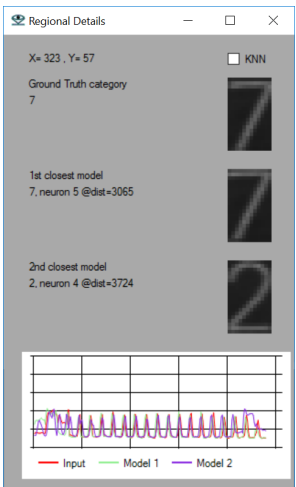
- Objects
- Clusters
- Anomalies
- Map of the categories
- Map of the closest models (useful when the neurons contain a codebook)
- Map of the distances (inversely proportional to the confidence of the neurons)



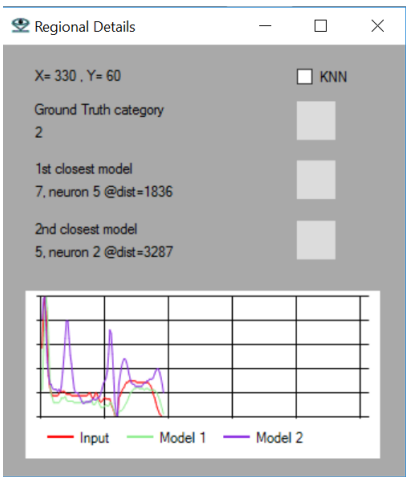
7.1 Local Classification

To help understand the recognition at specific locations, select Options/ Details at Cursor. As you move the cursor, the feature extracted from its location is displayed and, if applicable, the models of the first and second closest firing neurons along with the distance between the feature and their models. This utility can be useful to correct mis-firing neurons which recognize a feature vector with an incorrect category at a specific location.

Report of subsample vectors



Report of composite profile vectors


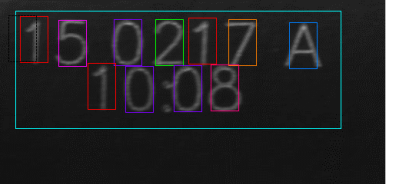
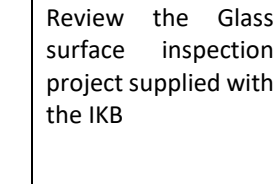


7.2 Regional Classification

The region to survey called Region of Search or ROS is defined manually with the left mouse cursor in the image to the right of the main panel. It can be set to the full image by checking the box "Full image".

In the case of a regional classification, the nominal region of interest is displaced across the entire ROS in a raster pattern and with a scanning step which can be different from 1 to accelerate the process. Note that the higher the step, the faster the scanning, but the higher the probability to miss some positive identifications.

The Recognition method can be one of the following: Objects, Clusters or Anomalies

Objects	Clusters	Anomalies
Objects, or all locations with a positive recognition.	Clusters, or groups of locations recognized with a same category.	Locations which are not recognized by the neurons
		

The reporting can be displayed in the form of a hit point, a box or shades:

- Hit point = center pixel of the ROI is highlighted if the ROI is recognized
- Box = Adjacent hit point are grouped and a box with the size of the ROI is placed at their centroid
- Shade: Hit point but with a transparency so the original pixels remain visible

7.3 [Objects, Clusters or Anomalies](#)

- The Object list reports all the locations where the ROI has been recognized. Depending on their density, they can be displayed as a single pixel or with a bounding box, and the color defining in the Project Settings.
- The Cluster List reports all the groups of adjacent objects contained within the ROI size and which have been recognized with the same category. They are represented with a bounding box centered on the cluster.
- The Anomaly list reports all the locations where the ROI has NOT been recognized. They are displayed as a single pixel and the color associated to the Background or category 0 in the Project Settings.

7.3.1 [Result files](#)

Objects or anomalies highlighted in an image are appended to a Results file whenever you select a new image.

The Results file can be cleared, saved or loaded from the Results menu. It has the same format as the Annotations files which is described in the paragraph Supervised Learning.

7.4 [Map of Categories, Distances or Identifiers](#)

A Map is a transform image of a Region of Search (which can be the entire image) showing the spatial distribution of the recognized objects in term of category value, distance/confidence value or neuron identification value.

The size of the map is the dimension of the ROS divided by the selected scanning step.

The amplitude of the CatMap array can range between [0,32,364].

The values of the DistMap array can range between [0, 65535].

The display of these maps uses a color lookup table.

7.5 [Map of Models](#)

This transform is only available if the selected feature extraction is a monochrome or color subsample

Each recognized ROI is replaced with its closest model. The effect is a posterization/simplification of the ROS using the models stored in the neurons.

Such display is useful to apprehend if the current knowledge is rich enough to compress the image without significant loss of information. In such case, it is usually assumed that the knowledge has been built using the Codebook learning method and composed of primitive pixel subsamples.

If the map of models does not render the important information present in the original image, the alternatives are to (1) train the neurons on more annotations, or (2) build a new knowledge with different value of the Maxif or different ROI size (Project/Settings menu).

7.6 [Coping with change of scale](#)

7.6.1 [Alternative #1](#)

Learn at a given ROI size (as specified in the Options/Settings menu). In recognition, the image scale can be changed in the navigation bar at the bottom of the screen.

The project faces_soccerplayer is a good illustration of this approach: the ROI has been sized to learn on the images of the soccer players. However, if you load an image from the folder faces_Ferret, you can notice that the ROI size is too small, but if you scale down the image by 3, then it becomes suitable and the face is recognized. Refer to manual paragraph 4.2.3.

7.6.2 *Alternative #2*

If the feature extraction can be a subsampling, it is scale invariant if the number of blocks within the ROI remain the same. It is then possible to learn objects at different compatible ROI sizes (as specified in the Options/Settings menu). For example, an ROI of 64x64 with blocks of 4x4 creates a vector of 16x16 components, but similarly an ROI of 128x128 with blocks of 8x8 also creates a vector of 16x16 components. In recognition, the neurons holding models of the objects at the proper scale will fire.

8 SEMI-SUPERVISED LEARNING

8.1 Codebook generation

The Build Codebook function lets the neurons decide when a ROI is sufficiently different from the models already stored in the committed neurons and should be learned under a novel category.

The input to this function can be one or more annotated areas. For each ROI scanned within these areas, the neurons are queried to decide if the ROI should be learned and with which category.

Parameters to verify prior to using the BuildCodebook:

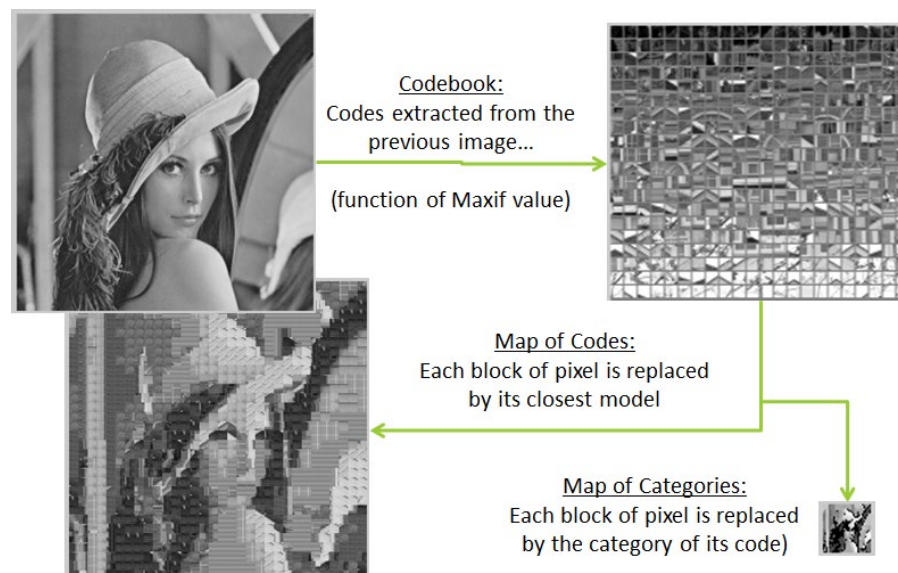
- Block size
- Feature extraction
- Maximum Influence Field
- Option to clear the current knowledge
- Scanning Step X and Y

Select the method to automatically allocate the categories:

- A constant value
- An incremental value
- Max Delta Amplitude
- Average Amplitude
- Raster position of the region in the image

The knowledge or contents of the neurons is then called a codebook, or a book of codes (or features) which the neurons have learned and detected as significant.

The codebook can be used to generate simpler transform images such as a map of models, a map of code IDs, a map of confidence, etc.

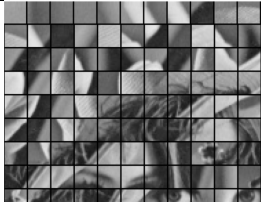
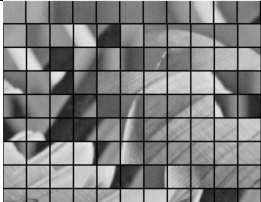
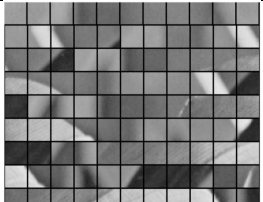





REMARK: Since the neurons automatically assign category values to the codes, any category edited under the Settings panel will be discarded except for the “Background” category or category value 0.

8.2 [Influence of the Maxif](#)

The smaller the Maxif, the more detailed the codebook, the less noise in the transform image.

ROI= 16x16
Codebook built with StepX=16 and StepY=16

Maxif 4000 => neurons=210	Maxif=2000 ➔ neurons=360	Maxif=1000 ➔ neurons=507
First models stored in the neurons		
		
Transform image obtained by replacing each ROI by its closest model stored in the neurons		
		

9 PROJECT AND KNOWLEDGE FILES

Projects defined under Image_Knowledge_Builder can be saved to disk in a project file or to the Flash memory of the camera.

A project is composed of two files associated to the same rootname:

- A project file (*.csp) containing the description of the recognition engine
- A settings file (*.prf) with the description of the GUI preferences

9.1 [CogniSight Project file](#)

- The *.csp file describes the recognition engine. It contains all the information necessary to resume the project including the knowledge, the region of interest, the region of search and sensor settings

9.2 [Preference file](#)

- *_settings.csv describes the settings of the graphical user interface including the names and colors associated to the category values, as well as the preferred learning and recording settings.

Example of preference file

```
ImageSource,Image
LearningMode,Annotations
RecoMode,Clusters
Scale,1
StepLX,1
StepLY,1
StepRX,1
StepRY,1
SkipX,1
SkipY,0
CatAlloc,Increment
CatNbr,2
Background
ball
CatColors,2
0,0,0
255,0,0
FrameIncrement,1
```

10 TIMINGS

The latencies of the instructions executed by the NeuroMem network can be reported in the status bar shown at the bottom of the main panel.

Such reporting adds an overhead to the real overall latencies of execution, so it can be turned ON/OFF under the Options menu.

The timings are calculated using the exact number of clock cycles spent in the NeuroMem chips times a clock cycle of 55 ns or 18 Mhz which is the recommended clock for a NeuroMem network composed of multiple NM500 chips.

These timings do not include the time spent to transmit the USB or SPI packets back and forth between the host and the platform hosting the NeuroMem chips.

11 APPENDIX B: WHAT IS NEW?

11.1 [IKB Version 3.3](#)

Addition of an Annotations\Options menu to define Automatic Counter examples when annotating discrete objects.

11.2 [IKB Version 2.7](#)

Updated the chapter of this manual describing how to manipulate Annotations for supervised learning.

Auto Learn: Fixed a memory error problem when learning an entire folder of images automatically using the Play button.

Review Annotations: Improved the display of the annotations and optimized the refresh rate when dealing with a large number of records.

11.3 [IKB Version 2.6](#)

The cursor color can be changed in the Options menu. The default color is black, but this can be an inconvenience to locate dark features.

Annotations loaded from a new file are automatically sized to match the current Scale selected in the main panel.

The Codebook learning method has been extended to support the use of annotations. This means that you can generate a knowledge of primitive codes based of specific regions selected in a single or multiple images. Since the annotations are saved, you can easily test various settings to build different codebooks and observe the transforms they can produce. The settings include the feature to extract and the value of the Maximum Influence Field.

The Codebook function has been tuned to better generalize and use the value of the Maximum Influence Field edited under the Settings menu.

A bug has been fixed to produce a proper display of the maps generated in recognition when the scaling was changed.

Applying a recognition to the entire image is now made through a simple checkbox (instead of a double key entry with mouse click).

Learning a Golden Template based on the entire image is now made through a simple checkbox (instead of a double key entry with mouse click). Note that this option is not available when the learning method is "Annotations".

The format of the Preference file (*.prf) saved with a Project file (*.csp) has been changed so that both the learning and recognition method are described with their text label and not their numeric code. This is easier to read if you open the file in a text editor. The older format is still supported.

11.4 [IKB Version 2.5.4](#)

The program can load annotations saved in comma delimited and tab delimited format.
The annotations loaded from file are automatically converted to the current scale selected on the main panel.

When scaling an image, the location of existing annotations and/or results are updated accordingly. Their size is not since this is related to the project settings.

Automatic option to fit the region of search to the entire image

11.5 [IKB version 2.5.1](#)

Fixed bug preventing to edit steps and scales with more than 1 digit

Addition of a Settings button to access the project settings directly from the main panel

Addition of 3 display options to report the recognized objects and anomalies: a single point, a box, a shade

Bug fix and Improved display of the maps of the categories, distances, and identifiers: The maps are shown with their true dimension in pixel after the decimation by the selected scanning steps. The maps are pasted into a background image to clearly show their position relative to the original image.

The outline of the region of search is always visible even if the region is set to the entire image. This prevents the mis-interpretation of results due to an incorrect ROS selection.

11.6 [IKB version 2.4](#)

The Arrow cursors can be used to move the Region of Search in the recognition window.

The Details at Cursor panel has been improved so it can report more than the first 2 firing neurons if applicable. The plot display has been improved.

The last project can be retrieved in the C:\MyDocuments\General Vision\Project folder. This project is saved automatically before each learning operation.

The learning mode through Annotations now supports the annotations of single objects (ROI, Region of Search) but also areas (ROS, Region of Scan). The key controls have slightly been changed as described under the [Learning and Utilities](#) paragraph.

11.7 [IKB version 2.3](#)

- Menus have been re-arranged
- Data logging functionalities have been added
- Fix the bug with the Knowledge\Undo last learning command
- The Settings panel allows to edit the Minimum Influence Field
- The Auto-Learn check box is active and triggers the automatic learning of the annotations associated to the newly loaded image.
- The Auto-Reco check box is active and triggers the automatic savings of the recognized objects (or clusters or anomalies) to a "results" file (see below).
- Annotations are now appended in a single file which can be saved and reviewed in a new panel. This file must reside in the same folder as the images it relates to.
- Results are now appended in a single file which can be saved and reviewed in a new panel. This functionality replaces the former "View Results" menu.

- The feature vectors extracted from annotations and/or results can be exported in a csv format compatible with NeuroMem Knowledge Builder

12 TROUBLESHOOTING

12.1 Unable to load DLL “CogniSight_xxx.dll”

Download [Visual C++ Redistributable for Visual Studio 2015](#) from the Microsoft Download Center.

Apply (execute) both files:

vc_redist.x86

vc_redist.x64