INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Specifications of the software videomatch 1.1

Matthieu Personnaz — Peter Sturm

N° 0260

Mars 2002

THÈME 3

Rapport technique

ISSN 0249-0803

Specifications of the software videomatch 1.1

Matthieu Personnaz , Peter Sturm

Thème 3 — Interaction homme-machine,
images, données, connaissances
Projet MOVI

Rapport technique n° 0260 — Mars 2002 — 52 pages

Abstract: *videomatch* is a software which allows to define matches between primitives belonging to frames of a video sequence. In particular, it allows the definition of a video sub-sequence, the manual or automatic definition of the primitives, the manual or automatic matching of these primitives.
This document is made up of the analysis of *videomatch*, according to the OMT (Object Modelling Technique) method. It aims at understanding and modelling the application under three main features: the object model, the functional model and the dynamic model.
The conception of the objects and their implementation as well are based on these specifications.

Key-words: video, segmentation, match, tracking, OMT

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 Montbonnot-St-Martin (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Specifications du logiciel videomatch 1.1

Résumé : *videomatch* est un logiciel qui permet de définir des correspondences entre primitives appartenant à certaines images d'une séquence vidéo. Il permet notamment la définition d'une sous-séquence video, la définition manuelle ou automatique des primitives, la définition manuelle ou automatique de correspondences entre ces primitives.
Ce document constitue l'analyse de *videomatch*, selon la methode OMT (technique de modélisation par objet). Son objectif est donc de comprendre et de modèliser l'application sous trois principaux aspects: le modèle objet, le modèle fonctionnel et le modèle dynamique.
La conception des objets ainsi que leur implémentation s'est effectuée sur la base de cette analyse.

Mots-clés : vidéo, segmentation, mise en correspondence, suivi, OMT

Contents

GLOSSARY

videomatch: It is the name of the software system currently designed to provide tools allowing specific operations on a video steam. In particular, it allows the matching of specific features of images contained in a video stream.

POI: Points of interest. If they are automatically detected, these poits have a high rate of repeatability under the following different transformations: image rotation, scale change, illumination and viewpoint change. Otherwise, a POI may be any point interactively defined by the user.

To match: To define a set of points of interest. Each point of interest associated to a given match should correspond to the same 3D world point observed from several points of view. The images associated to each point are mutually distinct.

A match: A set of points of interest so the images associated to each point are mutually distinct. In other words, any POI can be a part of one match at most.

A video sequence: Set of images gathered by a camera.

A working sequence: A subset of the video sequence. The most processes of the software are applied to it.

A project: A project expresses the results of the processes applied to a working sequence. A project can be saved at any time. It may possibly bereset to the last saved state. Moreover a new one can be opened and updated, or created.

A navigator: A navigator allows to navigate through sequences. The user has also the ability to view specific frames and to manually point specific features.

A mode: A mode specifies a set of functionalities. For instance, the POI mode allows to define POI, the user refines the working sequence with the sequence mode, and the matching is done under the match mode.

The console: It allows to handle a project, to select and to parameterize any mode.

INTRODUCTION

Purpose The software **videomatch** aims at providing tools for:

- the definition of a sub-sequence of a given video sequence.
- the manual or automatic detection of points of interest.
- the manual or automatic matching of points of interest.

Scope The document performs an analysis of **videomatch**, according to OMT (Object Modelling Technique). The version of the document is the version 1.1.

Overview The users requirements are expressed in a user manual. Then, they are used to produce an OMT model: an object model, a dynamic model and a functional model.
Also, The first section describes a manual of the subsystem. The three following sections define respectively the three models of OMT: the object model, the dynamic model and the functional model.

1 The users requirements

The requirements are defined through a user's guide. They can be divided into five main modules:

- the project.
- the navigator.
- the definition of an intended sub-sequence from the provided video sequence.
- the manual and automatic detection of the POI.
- the manual and automatic matching of the detected POI.

Also, the document shows how to handle a project and a navigator. It explains how to define a working sequence as well, the detection of the POI, and the building of the matches. The showed figures of the software are issued from the implementation. They replaced the drawings which were initially used to design the graphic user interface.

1.1 The project

A project corresponds to the results of processes the user apply to a video stream. This section explain how to create, save, reset and open a project.

1.1.1 The main features

A project identifies a work associated to a particular video sequence. It corresponds as well to the a directory whose name is the project's name and which contains the results files.
The user has the ability to load and update a project. The saving of the intermediate results is possible and corresponds to the saving of the project.
A project takes as input a set of parameters which may be read, updated and saved, whether the application is running or not.
Let's add that a project is associated to a single working sequence and several projects can be issued from a same video sequence.

1.1.2 Handling

A particular graphic interface called **console** allows to handle a given project.

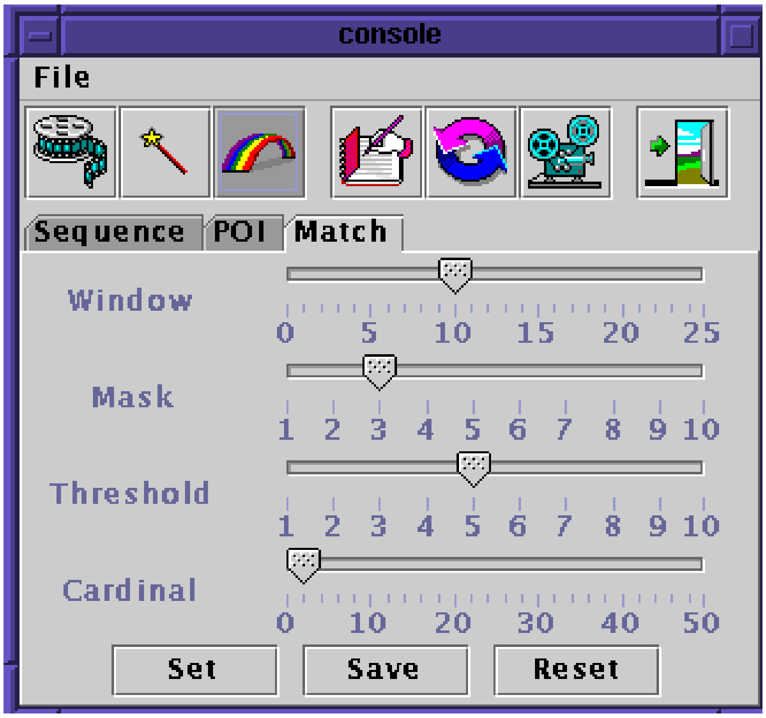

Figure 1: **The console**

Creation A project is created in two steps. First the user names its project after clicking on the menu item **New**.

Figure 2: **Dialog to name the project**

Then, the user selects a video file through a file chooser dialog. The CODEC of the video must be MJPG or JPEG.

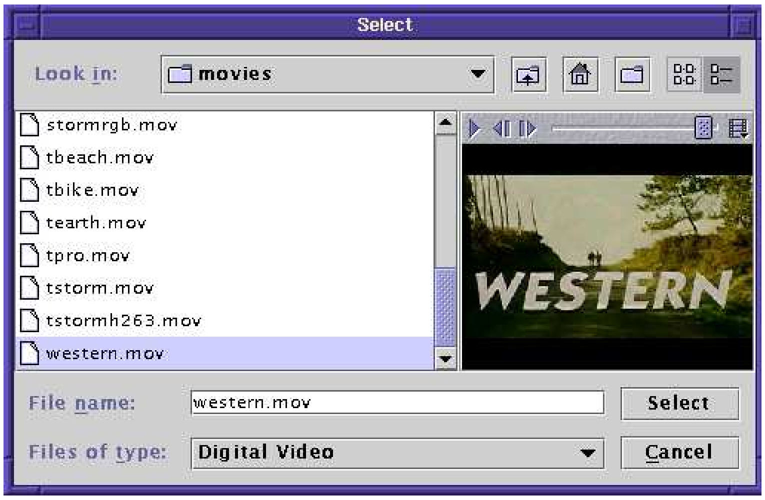

Figure 3: **Select the video file**

Loading To open an existing project, the user has to click on the menu item **Open**. A file chooser pops up, allowing to select the intended project.

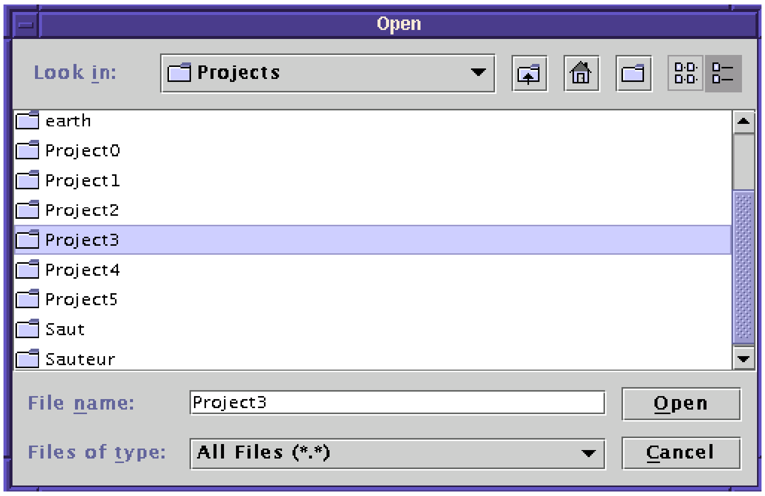

Figure 4: **Open an existing project**

Parameterization The parameterization of a project is defined in an understanding ASCII file **parameters** and contained in the project directory. This file is read when the application is loaded. The application running, this file may be updated through the parameters panels.

Saving The user saves the results of the project by pushing on the **save** icon. The stored results are made up of four files located in the project directory. The **working sequence** file expresses the definition of the working sequence. The **poi** file contains the coordinates of the points of interest. The **match** file describes the matches. The **parameters** file allows to access to the parameters of the application: name of the video sequence and parameters associated to the modes.

Reset The user has the ability to retrieve and work with the last saved state by pushing onto the **Reset** icon.

Leaving The user leaves the project by clicking on **Quit** icon.
If the user wishes the last results were saved, the user has to push on the button **Save** icon before.

1.2 Handling of a navigator

1.2.1 The main features

From the moment a project has been defined or loaded, a navigator automatically pops up, allowing the user to navigate through the video or the working sequence.
Many navigators can be created and the functionalities provided by each one are identical.

A navigator provides as well a mean to interact with specific frames by pointing out, for instance, some points of interest. The possible interactions and the functionalities these interactions induced are submitted to the current mode of the project.

1.2.2 Handling

Creation The user may possibly create a new navigator by pushing onto the **Navigator** icon of the console.

Dismiss A navigator can be discarded either by clicking on the **Close** button located on the window's navigator.

1.2.3 Navigation

Selection of the sequence The label **Video** suggests that the navigator deals with the video sequence. The label switches to **Working** if the user pushes the previous **Video** button. In this case, the navigation is associated to the working sequence.

Figure 5: **A navigator**

Navigation functionalities The first image displayed may be different of the first one if the navigator has been duplicated. The buttons labeled **Forward** and **Backward** allows to display respectively the next and the previous image belonging to the sequence. The **Reset** button allows to get back the first image.
An other way to navigate in the sequence is to move the slider located at down the window. The buttons **Play** and **Stop** manages the periodic display of the images.
The number of the shown is visualized. The user can access to a specific frame by typing its number in the text area and pressing the **Enter** key.

Zoom By keeping pressed the **Crtl** key, the user has the ability to zoom the frames. If the user clicks with the mouse button1 or button2 on any point, the image respectively zoom up or zoom down. The center of the computed image is, only when it is possible, the point previously clicked.
The **Crtl** key being kept pressed, the user can move the zoomed image by dragging it with the mouse button3.
Let's add that a zoom is associated to a navigator: the value of the scale is kept, even if the image or the navigator change.

Figure 6: **A zoomed frame**

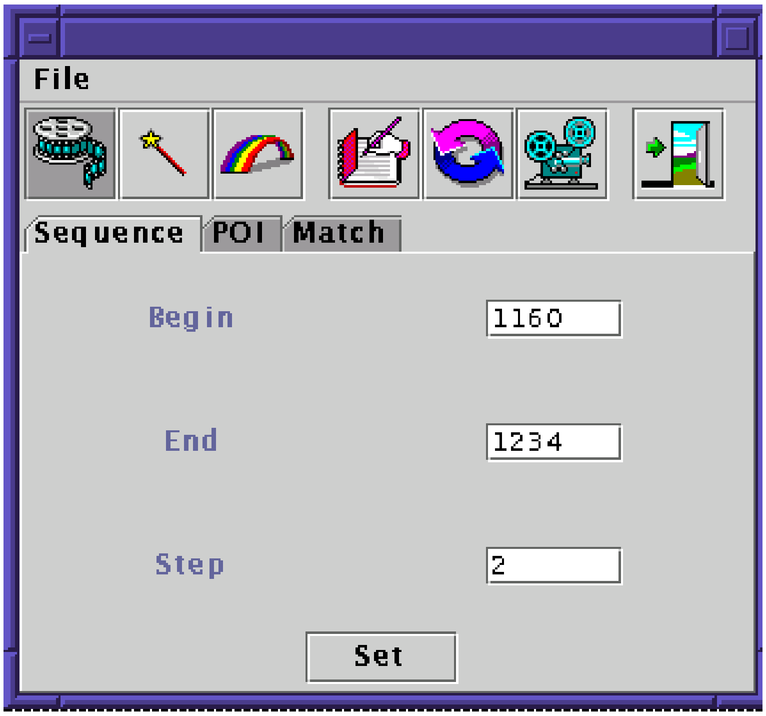

Figure 7: **The sequence parameters tabbed pane**

1.3 To define a working sequence

The whole data retrieved from a video sequence are not necessary for the system. Furthermore, the useless images make the processes longer. So the user should first define a subset of images from the video sequence. This subset is referred to as the working sequence and is initialized to the video sequence.
Given a new created or an opened project, the section gives also a way to define a particular working sequence from the video sequence associated to the project.

1.3.1 Automatic definition

After selecting the **Sequence** tabbed pane, the user can define a subset of the video sequence.
Thanks to this panel, the user sets three values:

- **Begin**: the index of the starting picture.
- **End**: the index of the end picture.
- **Step**: the step.

If we identify each frame of the video sequence to its index, a first subset of the video sequence is defined as following:

$$R = \{\textbf{Begin} + k\textbf{Step} : k \in \mathbb{N} \text{ and } 0 \leq k \leq E\left(\frac{\textbf{End} - \textbf{Begin}}{\textbf{Step}}\right)\}$$

where $E(x)$ is the integer part of the real number x.
The application takes account of the values associated to the labels **Begin**, **End** and **Step** as soon as the button titled **Set** is pushed.

1.3.2 Refinement of the working sequence

The previous section has explained a way to automatically define a working sequence. Nevertheless, the user may possibly refine this subset by adding or deleting specific frames.

Starting of the sequence mode The user has the ability to modify the content of the working sequence when the sequence mode is invoked. By pushing on the **Sequence** icon located on the console, the user switches the application to the sequence mode.

Adding The final look of a navigator depends on the selected mode. Also, the sequence mode being set, each navigator contains a new button labeled **Add** if the video sequence has been chosen.
By pushing the **Add** button, the user adds the displayed frame from the video sequence to the working sequence.

Delete The working sequence being selected in a navigator, the label of the **Add** button switches to **Remove**. After clicking on the **Remove** button, the current frame of the navigator is removed from the working sequence.

1.4 The detection of the POI

A point of interest (POI) is a point in the image for which the signal changes two-dimensionally. For instance, the corners such as L-corners, T-junctions and Y-junctions satisfy this. The localization of the POI is used for tasks like camera calibration or 3D reconstruction. However, in the case of a POI interactively defined by the user, a POI may be any point of the frame.
Given a working sequence, this part explains how to detect automatically and interactively some points of interest.

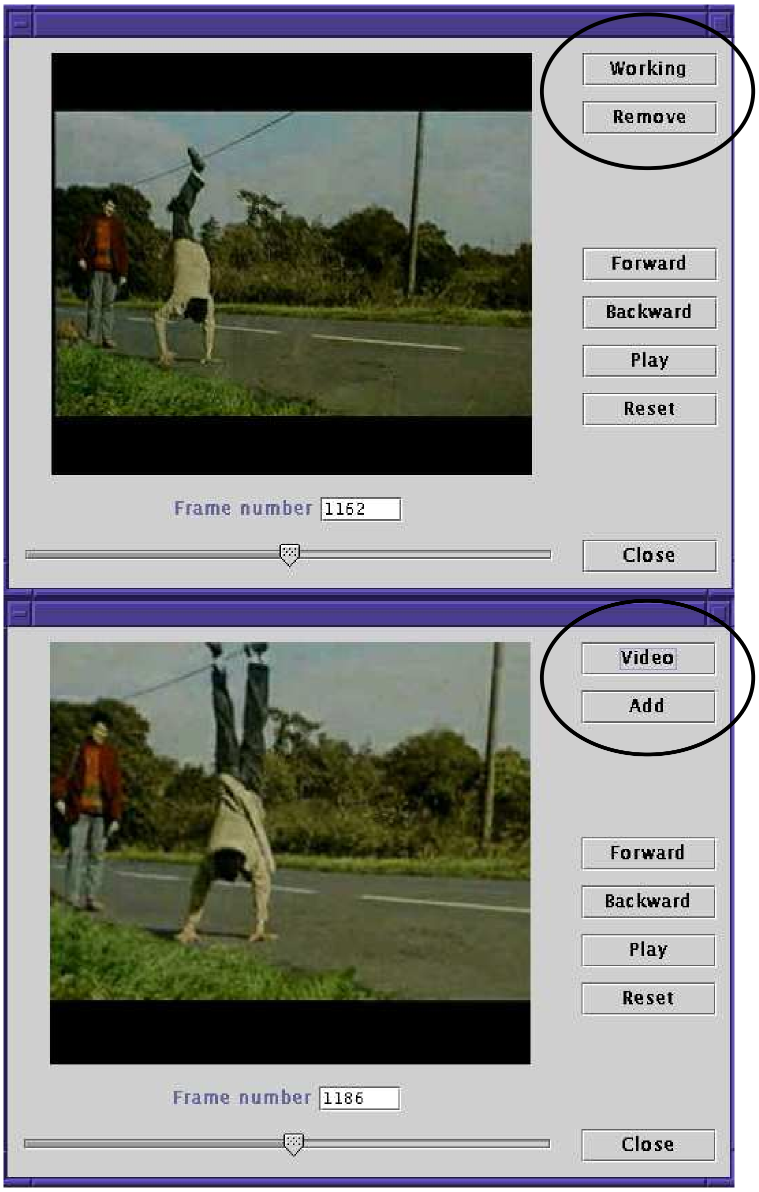

Figure 8: **Add or remove the displayed frame**

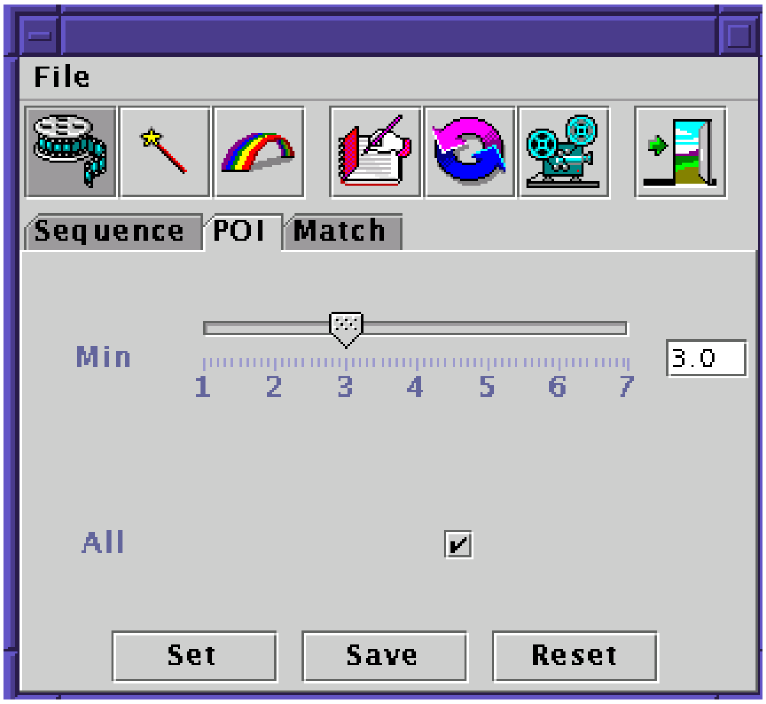

Figure 9: **The poi parameters tabbed panel**

1.4.1 Automatic detection

The POI mode The user may define some points of interest only when the POI mode is activated. To invoke the POI mode, the user pushes onto the **POI** icon located in the console.

Parameterization The parameters allows to parameterize the automatic detection, detection performed by an improved version of the detector of Harris ([2]).
To parameterize the detection, the user first selects the **POI** tabbed pane belonging to the console.

By dragging the slider with the mouse, the user defines the intended value of the detection threshold. By checking the check box **All**, the user decides whether the detecion has to be applied to the viewed frame or to the whole sequence.
The values are taken into account after the user confirms his choice by pushing onto the **Set** button. The **Save** button allows to save the set value. At last, the **Reset** button permits to return to the last saved values.

Detection The look of the navigator is related to the activated mode. In the case of the POI mode, the navigator has a new button just above the button labeled **Working**: a **Auto** labelled button.
If the user pushes onto **Auto**, POI of the viewed frame or POI of the whole working sequence, according to the checking of the **All** check box, are automatically detected. At last, a new detection erases the automatically detected POI of the previous one.

1.4.2 Manual detection

This section explains how to manually define some POI.

Starting of the POI mode As in the automatic detection, the user may define some points of interest only when the POI mode is activated. To invoke the POI mode, the user pushes onto the **POI** icon located in the console.

Detection and removing Let's suppose that a working sequence is displayed thanks to a navigator. By clicking with the mouse button1 onto points on the current image, the users defines new points of interest.
The user has too the ability to erase a POI by clicking on it with the mouse button2. This is the only way to remove a POI manually defined: a such POI is not modified by the results of the possible automatic detections.

1.5 Building of the matches

This section describes how to define matches of points of interest. In this document, a match means a set of points of interest whose the associated frames are mutually distinct.

1.5.1 Automatic building

Starting of the matching mode The matching mode must be active in order to define matches. The matching mode may be called by clicking onto the button titled **Match** icon located on the console.

Parameterization The automatic matching is based on a crooss-correlation algorithm. Is is parameterized by four values:

- **Window**: Given a POI, the algorithm tracks its correspondant in the next frame. The research of the matching point is bordered by a square window whose half size of an edge is **window**.
- **Mask**: It is the half size of the mask used for the SAD correlation measure.

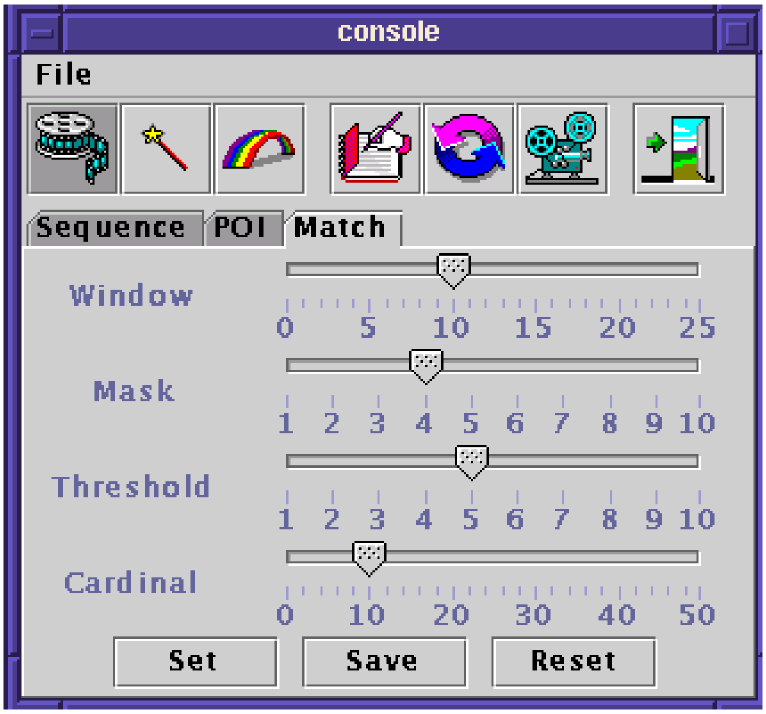

Figure 10: **The match parameters tabbed pane**

- **Threshold**: Two points are matched if their SAD correlation measure is inferior to the **Threshold**. The similarity between two patches decreases as the value of the **threshold** grows up.

- **Cardinal**: Only the matches whose the cardinal exceeds **Cardinal** are kept.

The user has the ability to specify these values after selecting the tabbed pane titled **Match**.

By dragging the slider with the mouse, the user defines the intended values. The values are taken into account after the user confirms his choice by pushing onto the **Set** button. The **Save** button allows to save the set value. At last, the **Reset** button permits to return to the last saved values.

Tracking The current mode of the application being the match mode, the user runs the automatic matching of the POI by pushing onto the **Auto** button.

1.5.2 Manual definition of a match

In this section, the POI are supposed to belong initially to no match and the match mode had been activated, as it is previously described . Furthermore, each POI is expressed thanks to an empty square.

Starting By a click on a point of interest with the button1 and the **Shift** button being simultaneously pressed, the user starts a specific match.
Moreover, the clicked point belongs to the new created match.
A green square containing the starting POI is drawn.

Adding Let's consider an image whose no POI belongs to the new created match. If the user clicks on a POI of a such image, with mouse button1, he adds it to the match. A green square overlaps the added POI.
By repeating the procedure as many times as is required, the user defines a specific match. For instance, let's suppose that the POI are numbered for each frame. Let's supposed too that any POI is referenced by the pair (number of the POI in the frame, number of the frame containing the POI). The following set is a match:

$$\{(10,30),(10,2),(1,20)\}$$

To produce it, the user has first clicked, keeping the **Shift** button pressed, on the POI 10 of the image 30 , then he has clicked on the POI 10 of the image 2 and on the POI numbered 1 of the image 20. The set of the POI being implicitly arranged by pictures numbering, a match is rather considered as a family of POI.

Creation of a POI and adding Given an activated match, the user has the ability to create and add a POI which was not defined during the POI mode. He just has to click on the POI with the mouse button3.

Removing By clicking on a particular green square with the mouse button2, the user removes the associated POI from the match. If the match contains a single point, to remove it involves the end of the procedure.

Closing By a click with the button1 on a POI, with the **Shift** button pressed, the user finishes the building. The clicked POI belongs too to the current match.
As it is previously described, during the building of a specific match, the POI belonging to this match are overlapped with a green square. When a match is closed, the color of the green squares switches to red. The color of the squares also points out which is the current building match.
A match is called a built match when it exists, i.e. it is not empty, and when it was closed.

Figure 11: **Match the top of the shoe**

Activation If the user wishes to update a specific match, he has first to click, the **Shift** key being pressed, on any POI belonging to the match.
The color of the red squares which overlapped the POI switches to green. That means the match is active.
Let's notice that a match is kept active during its definition. Furthermore, no more than one match can be active at the same instant.

Correction Given an image containing a matched POI, the user may change the POI belonging both to the image and to an active match. He just have to click with the mouse button1 on an other POI in the given image. The new pointed POI takes then the place of the last one in the active match.
For instance, let's suppose that an active match contains the POI 213 of the image 354. Let's suppose too that the user wants to change this POI with the POI 105 of the same image 354. The user clicks then with the mouse button1 onto the POI 105 in the image 354 and the replacement is done.

Closing The update is closed by clicking with the mouse button1, the **Shift** key being pressed, on a POI of the active match.

Visualization To check the content of a match corresponding to a particular POI, the user may click with the mouse button1 on this POI, no match being active.The red squares corresponding to the pointed match switch to the cyan color. In this case, only the images which containing an element of the match are displayed thanks to the navigator.
A second click with the mouse button1 on a POI of the highlighted match stops the visualization: the blue color of the squares corresponding to the match switches back to green.
The user has as well the ablity to control several matches by clicking on a representative POI of each one.

2 The object model

The object model is the first model of OMT. The object model provides the static structure of the software by describing the classes of objects and their connections. Attributes and operations may be added to the model in further analysis.

2.1 Identification of the classes

The classes issued from the software's requirements are defined in this section. They are classified in modules, a module being a basic functional component of the application. For each class, the definition includes its purpose, its connections with other objects, its operations and its attributes.

2.1.1 The project module

The project module allows to cretae, load and save a project. Tools to program and to control the processes are given too.

Working sequence data

It contains data describing the working sequence of a given class. Then, this class allows the user to keep a track of his work. The file associated to the **Working sequence data** are readable when the application doesn't work.
The **Working sequence data** contains the attribute *file*: the file where the data are stored. The operations are:

- *create*: to create a new project.
- *load*: to load the data associated to a given project.
- *save*: to save the whole results each time it is required.

POI data

The purpose of the class **POI data** is similar to the one of **Working sequence data**. But instead of defining the working sequence, **POI data** describes the POI of the images belonging to the working sequence.
Attributes and operations correspond to the ones of the class **Working data**.

Match data

The purpose of the class **POI data** is similar to the one of **Working sequence data**. **Match data** keeps a track of the defined matches.
Attributes and operations are those of the class **Working data**.

Parameters

The class **Parameters** programs the application. It contains the parameters associated to the programming of the working sequence, and those which have an implication for the POI detection. It contains too the parameters associated to the color which are involved in the

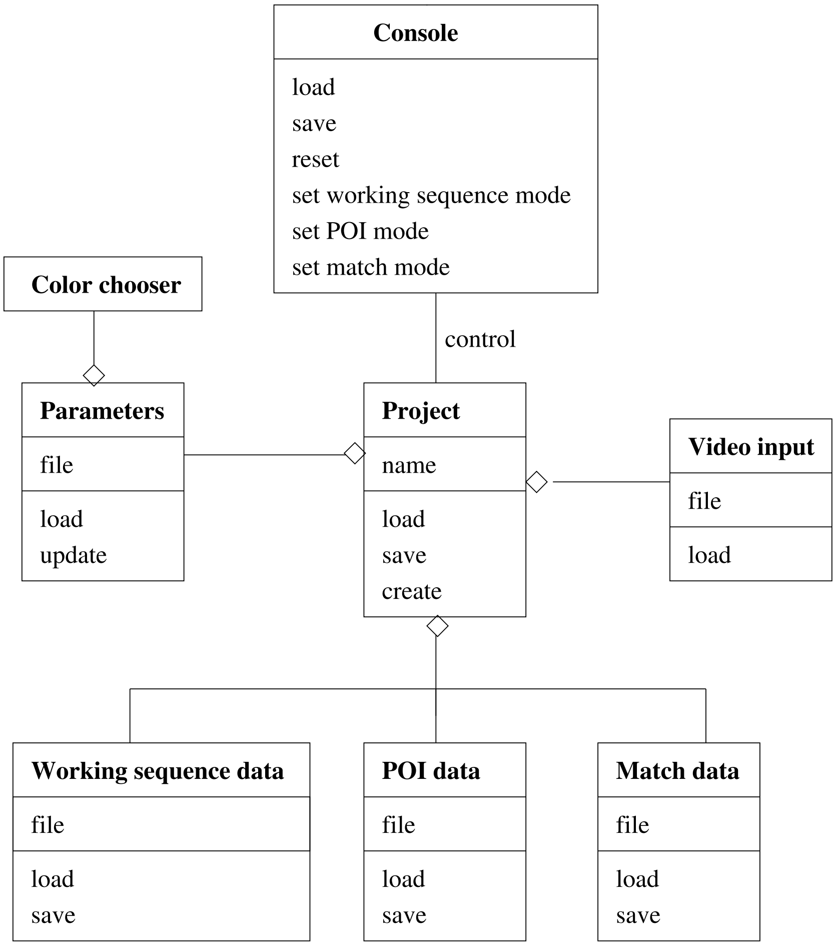

Figure 12: **The project module**

representation of the POI.
The class contains the attribute *file* which is the name of the file storing the parameters.
The methods are the following:

- *load*: to load the content of the file.
- *update*: to update the content by replacing the last saved values by the new ones.

Color chooser
This class provides tools to program the colors used to draw some specific graphic features such the POI.

Video input
This class provides an access to the recorded video, i.e. the output of the camrecorder.
It contains the attribute *file* which is the file corresponding to the video recording.
The operation *load* allows the software the reading of the video.

Project
The **Project** class describes the progress of the processes applied to a video input.
In particular, it contains the classes **Video input**, **Parameters**, **Match data** and **Working sequence data**.
The operations are:

- *load*: to load the project. This operation is required before to apply any process.
- *save*: to save the current working sequence, the matches and the set of POI.

Console
The **Console** controls the class **Project**, contains the class **Parameters** and determines the mode of process.
The operations are:

- *load*: to load a project.
- *save*: to save a project.
- *reset*: to reset data associated to current mode. Some not saved data are not reset: the data which are obtained in a mode distinct of the current mode.
- *set working sequence mode*: to switch to the working sequence mode.
- *set POI mode*: to switch to the POI mode.
- *set match mode*: to switch to the match mode.

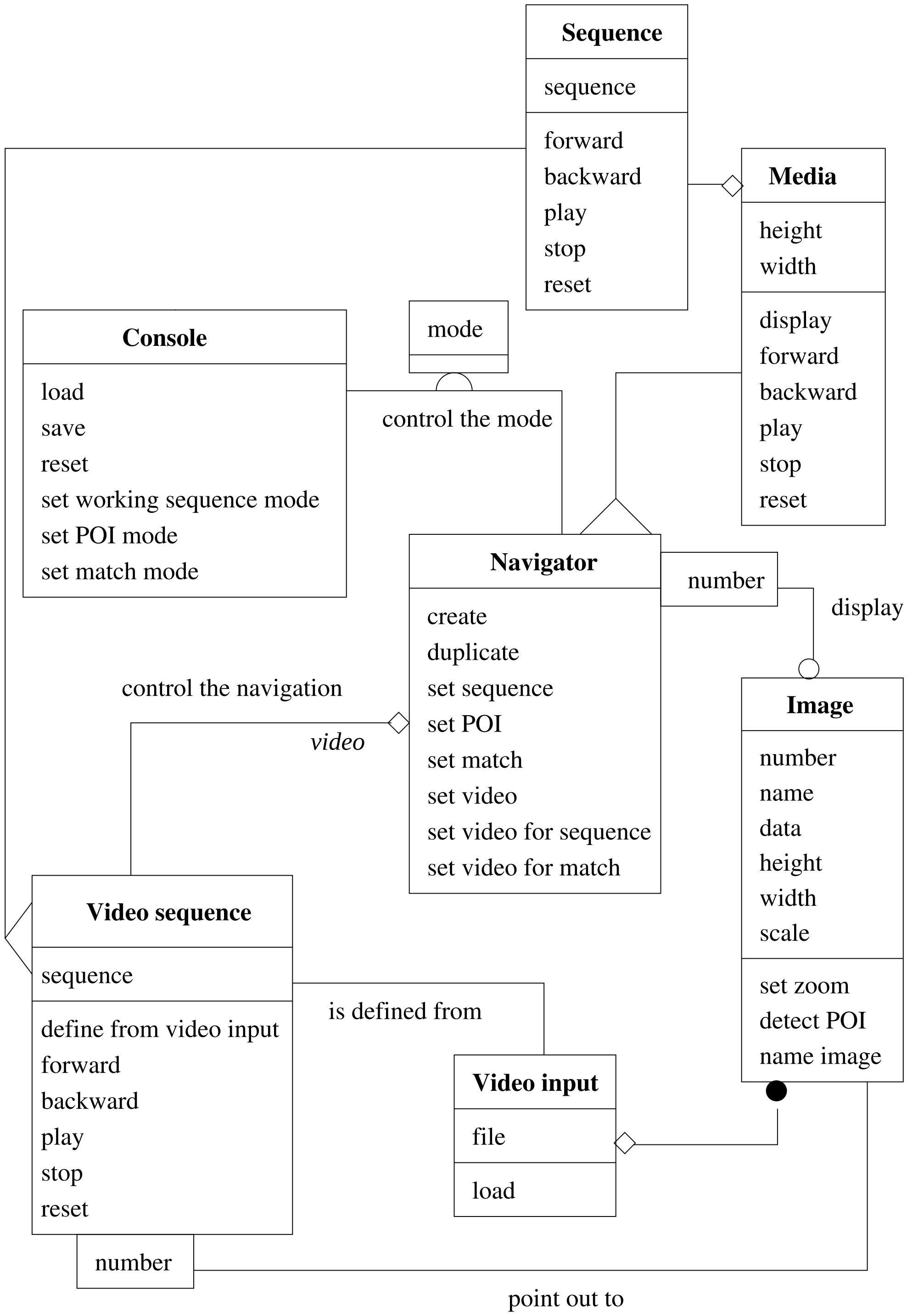

Figure 13: **The navigation module**

2.1.2 The navigation module

The navigation module is made up of functionalities allowing the navigation through a sequence. According to the active mode, it is a video, a working or a match sequence. It provides too an interface to achieve and control the processes associated to each mode.

Image

The class **Image** describes a particular image retrieved from the video input. Furthermore, it provides tools related to the displaying and the detection of features associated to a single image. Let's add that an instance of a such image contains too the content of an image so it can be managed "as a provider" by the others objects.
The attributes are:

- *index*: the number of the image.
- *data*: the content of the image.
- *height*: the height of the image.
- *width*: the width of the image.
- *scale*: the parameter of the zoom.

The operations are:

- *set zoom*: to define the scale of the zoom.
- *detect POI*: to detect the so called POI of the image.

Video sequence

The class **Video sequence** allows to navigate through a video sequence. It is defined from the video input and may point out on any image of the video, thanks to the number of the image.
It contains the attributes:

- *sequence*: a family of numbers, each number pointing out to an image.
- *current*: current is the number of an image subject to processes such the displaying or the detection of POI.

The class contains the following operations:

- *define from video input*: to define an instance of **Video sequence** from an instance of the class **Video input**.
- *forward*: to point out to the next image of the video sequence.
- *backward*: to point out to the previous image of the video sequence.
- *play*: to automatically navigate forward.

- *stop*: to break the automatic forward.
- *reset*: to point out to the first image of the sequence.

Sequence
The **Sequence** class describes a set of numbers and gives ways to navigate through the set. It is an abstract class. **Video sequence** inherits of **Sequence**.
Sequence has the *sequence* attribute, i.e. a set of numbered which may possibly point out to images.
It provides the operations:

- *forward*.
- *backward*.
- *play*.
- *stop*.
- *reset*.

Media
The **Media** class is an abstract class which displays images associated to the **Sequence** class. It provides too the control of a such instance.
The attributes are:

- height: height of the window allowing the display of the images.
- width: width of the displaying window.

The provided operations are:

- *display*: to display an image.
- *forward*: to navigate forward the set.
- *backward*: to navigate backward the set.
- *play*: to automatically navigate forward.
- *stop*: to break the automatic forward.
- *reset*: to point out to the first image of the set.

Navigator
The class **Navigator** allows to achieve the processes related to the different modes. It inherits of the **Media** class. Then, it provides tools to control the displaying and the processes of a particular set of images as the video sequence, the working sequence and the match sequence. So each instance of the class may be have several parts. The parts depend on both the active mode and the kind of sequence. The possibly parts of an instance of **Navigator** are:

- sequence: the navigation deals with the working sequence and the active mode is the sequence mode.
- POI: the displayed pictures are those of the working sequence. The active mode is the POI mode.
- match: the navigation deals with the working sequence. The active mode is the match mode.
- video: the navigation is allowed through the video sequence. The active mode may be either the POI mode or the match mode.
- video for sequence: the navigation deals with the video sequence. The active mode is the sequence mode.
- video for match: Given a shown match, the navigation deals only with the pictures associated to this match. The active mode is the match mode.

Let's notice that the part of a navigator being partially determined by the active mode, the **Console** class partially controls a navigator.
The properties inherited from the **Media** class are not rewrote.
Also, the operations are:

- *create*: to be created, taking account of the active mode.
- *duplicate*: to be cloned.
- *set sequence*: to switch to the sequence part.
- *set POI*: to switch to the POI part.
- *set match*: to switch to the match part.
- *set video*: to switch to the video part.
- *set video for sequence*: to switch to the video for sequence part.
- *set video for match*: to switch to the video for match part.

2.1.3 The working sequence module

This module provides tools to define and manage a working sequence.

Sequence parameters

The **Sequence parameters** class allows to program the working sequence. It is also a component of the **Parameters** class.

It contains the attributes:

- *begin*: index of the first image of the working sequence.

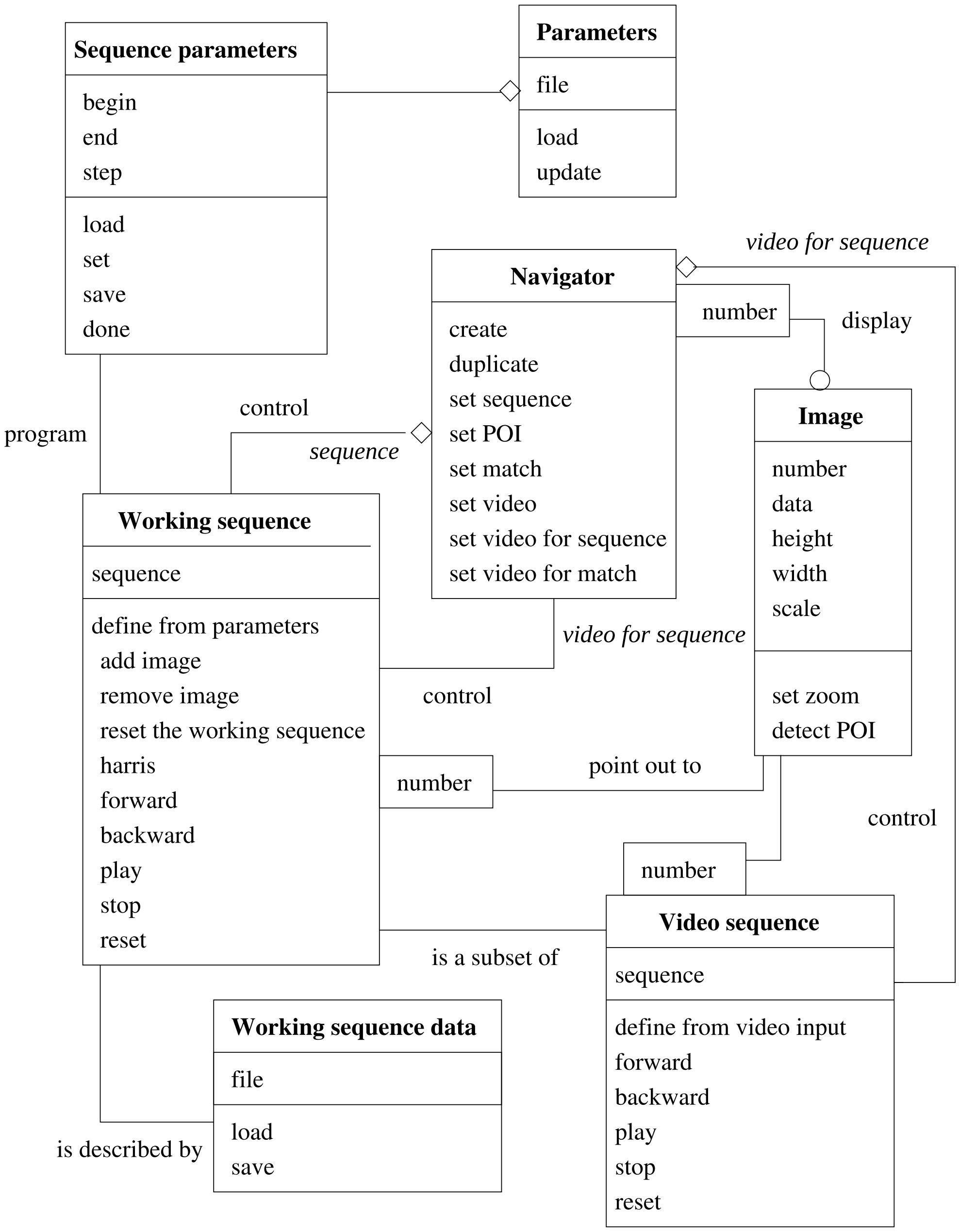

Figure 14: **The working sequence module**

- *end*: index of the last image of the working sequence.
- *step*: given an image of the working sequence, the step defines the index of the next one.

The operations are:

- *load*: to load the saved parameters.
- *set*: to set the parameters.
- *save*: to save the set parameters.
- *done*: to leave the programming and keep for parameters the last saved ones.

Working sequence

The class **Working sequence** provides tools to define and manage a working sequence, i.e. the subset of the images extracted from the video sequence and providing the data inputs for further processes. An instance of the **Working class** points out to a specific image thanks to the number of the image.
Let's add that the **Working sequence data** class keeps a track of the built working sequence.
It contains the attributes *sequence*: a family of numbers pointing out to the selected images of the working sequence.
The class contains the following operations:

- *define from parameters*: to define an instance of **Working Sequence** from an instance of the class **Sequence Parameters**.
- *add image*: to add an image to the working sequence.
- *remove image*: to remove an image from the working sequence.
- *reset the working sequence*: to get the last saved sequence.
- *detect POI*: to automatically detect some POI.
- *forward*: to point out to the next image of the video sequence.
- *backward*: to point out to the previous image of the video sequence.
- *play*: to automatically navigate forward.
- *stop*: to break the automatic forward.
- *reset*: to point out to the first image of the sequence.

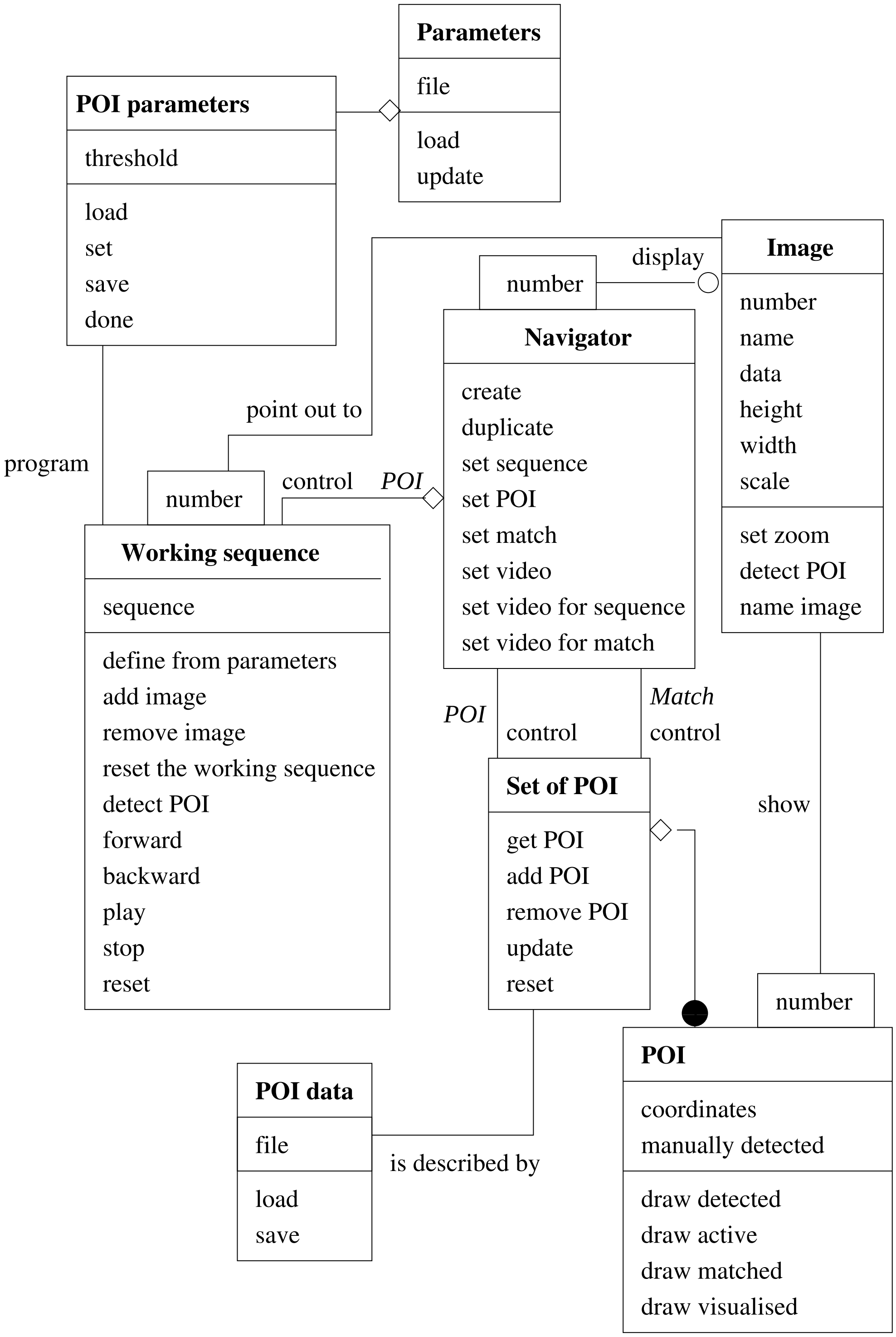

Figure 15: **The POI module**

2.1.4 The POI module

The POI module is a set of classes allowing the building of POI.

POI parameters
The **POI parameters** class allows to program the POI mode. Then, the class is linked to the **Image** class and the **Working sequence** class in the order to program the automatic detection of some POI. **POI parameters** is a component of the **Parameters** class as well. It contains the attribute *threshold*, a parameter of the Harris' detector.
The operations are:

- *load*: to extract the POI parameters from the parameters of the application.
- *save*: to save the POI parameters.
- *set*: to set the parameters.
- *save*: to save the set parameters.
- *done*: to leave the programming and keep for parameters the last saved ones.

POI
The **POI** class describes a POI. It provides tools to show them as well.
An instance of **POI** may be then a component of the **Image** class.
The attributes are:

- *coordinates*: the coordinates of the POI.
- *manually detected*: if the value is true, the point had been manually detected.

The class contains the following operations:

- *draw detected*: to show a POI belonging to no match yet.
- *draw active*: to draw a POI belonging to a match, when the match is active.
- *draw matched*: to draw a POI belonging to a match, the match being active.
- *draw visualized*: to draw a POI belonging to a visualized match.

Set of POI
The class **Set of POI** describes the set of POI and provides tools to manage it. The building of a such set is performed through an instance of the **Navigator** class. It contains then instances of the **POI** class and the **POI data** class allows to keep a track of the saved POI. Its operations are:

- *get POI*: to retrieve a particular POI.

- *add POI*: to add a manually detected POI.
- *remove POI*: to manually remove a POI.
- *update*: to add a set of automatically detected POI.
- *reset*: to get the last saved set of POI.

2.1.5 The match module

The match module allows to construct a set of matches.

Match
The **Match** class describes a particular match. An instance of the **Match** class may contains several instances of the **POI** class. It may be used too for the definition of a match sequence.
The class contains the following operations:

- *add POI*: to add a POI to an activated match.
- *remove POI*: to remove a POI from an activated match.
- *change POI*: to change a POI of the activated match; the new and the old POI belong to the same picture.
- *initialize*: to create a new match by adding its first POI.
- *destroy*: to remove the active match by removing its last POI.
- *set active*: to make active a particular match.

Set of matches
The **Set of matches** class contains and manages the whole built matches. The building of the matches is achieved through a navigator. The **Match data** allows to track the constructed matches.
It contains the following operations:

- *add match*: to add a new match to the set of matches.
- *remove match from POI*: to remove a match from the set of matches, given a POI of the match.
- *get match from POI*: to get a specific match from one of its POI.
- *reset*: to get the last saved set of matches.

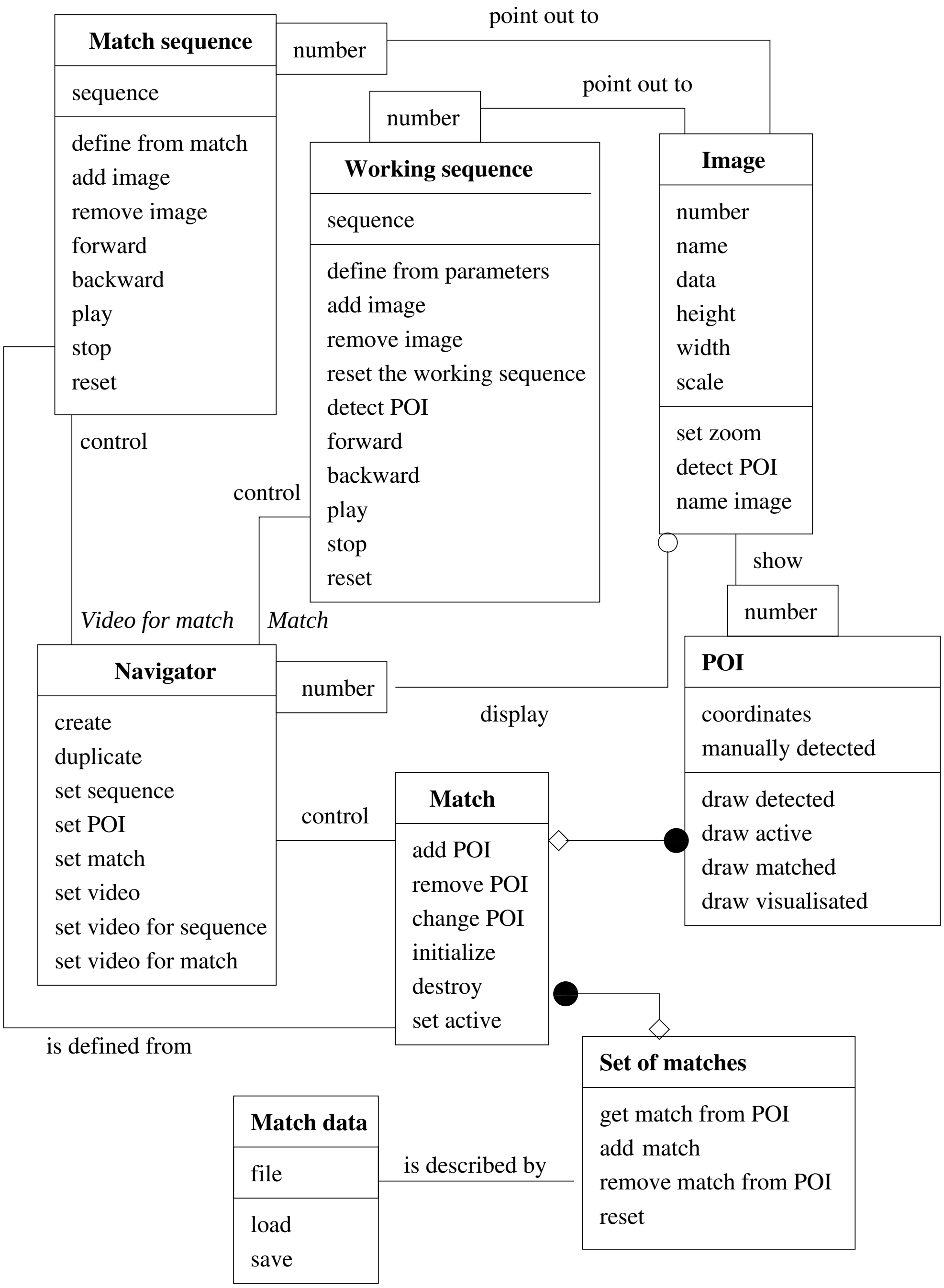

Figure 16: **The match module**

Match sequence
The class **Match sequence** provides tools to manage a match sequence, i.e. a set of images associated to a given match. An instance of **Match sequence** is controlled through a navigator.
It contains the attributes *sequence*: a family of numbers, each number pointing out to an image.
The class contains the following operations:

- *define from match*: to define an instance of **Match Sequence** from an instance of the class **Match**.
- *forward*: to point out to the next image of the match sequence.
- *backward*: to point out to the previous image of the match sequence.
- *play*: to automatically navigate forward.
- *stop*: to break the automatic forward.
- *reset*: to point out to the first image of the sequence.

2.1.6 Generalization

Some generalizations may be deduced from the previous object models.

Data
The classes titled **Working sequence data**, **POI data** and **Match data** inherit of the abstract class **Data**.
Data contains the attribute *file* and the following operations:

- *load.*
- *save.*

Sequence
The classes named **Working sequence**, **Video sequence** and **Match sequence** inherit of the **Sequence** class.

Program
Sequence parameters and **POI parameters** inherit of **Program**.
Program provides the operations:

- *load.*
- *save.*

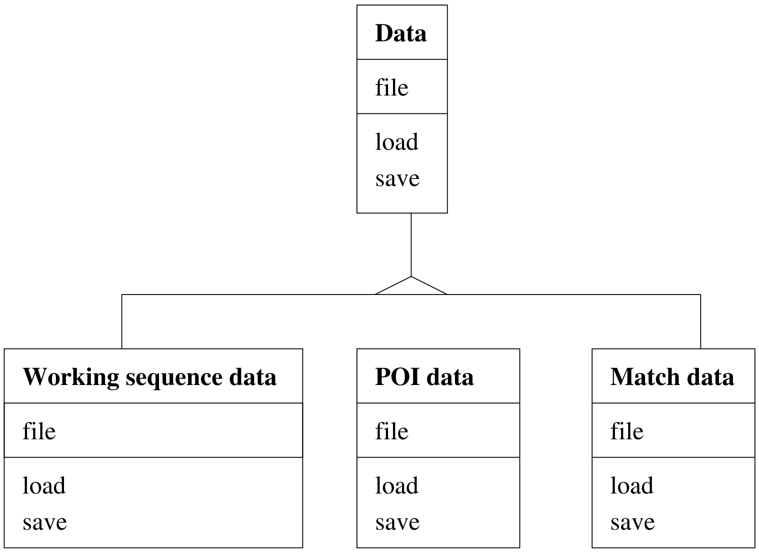

Figure 17: **The Data class**

3 The dynamic model

The dynamic model is made up of several diagram of states. Each diagram is associated to the different states of an instance of a specific class.

3.1 POI

This section deals with the states of an instance of **POI** only when the match mode is the active mode. Four states are associated to an instance of **POI**. Each state corresponds to a particular drawn of the instance.
The states are:

- hidden: the POI is not shown.
- detected: the POI has been detected but doesn't belong to any match.
- matched: the POI belongs to a match.
- activated match: the POI belongs to an activated match.
- visualized match: the POI belongs to a visualized match.

The events which lead to change the state of an instance of **POI** are:

- *hide*: the part of the navigator switch to "sequence", "video" or "video for sequence".

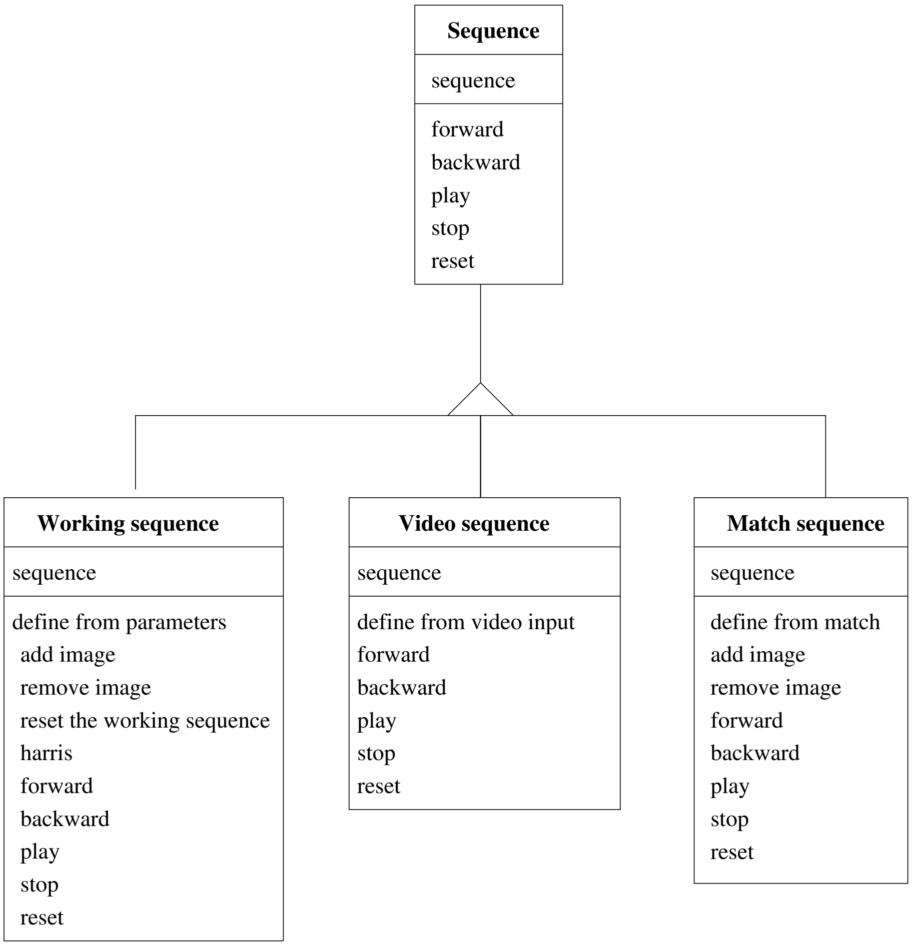

Figure 18: **The Sequence class**

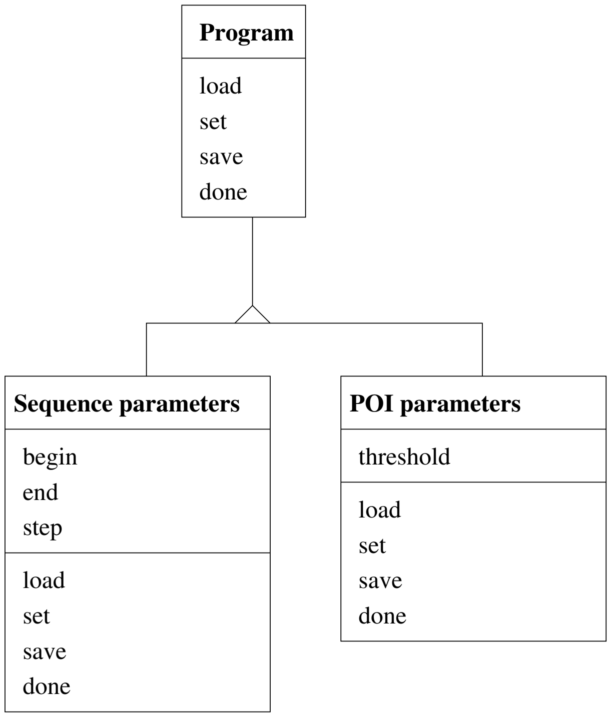

Figure 19: **The Program class**

- *show*: the part of the navigator switch to “POI”, “match” or “video for match”.
- *1*b1*: the user clicks with the mouse button1 on a POI belonging to the same image.
- *2*b1*: the “double clicks” with mouse button1 on a POI belonging to the same image.
- *1*b2*: the user clicks on a POI with the mouse button2 on a POI belonging to the same image.
- *1*b3*: the user clicks with the mouse button3 on a POI belonging to the same image.

The click on a POI usually deals with the studied instance. If the clicks happens on a different POI, the condition “different POI” is added.

3.2 Set of POI

In this section, we focus on the states of an instance of the **Set of POI** class, only when the active mode is the POI mode.
The events which leads to a change of the states are the following:

- *1*b1*: the user clicks on the mouse button1.
- *1*b2*: the user clicks on the mouse button2.
- *1*b3*: the user clicks on the mouse button3.
- *reset*: the user click on the **Reset** button of the console.
- *detect POI*: the user requests to automatically detect the POI.

3.3 Navigator

To each part of the navigator is associated a state. Also, the states of an instance of **Navigator** are:

- sequence: the part of the navigator is “sequence”.
- POI: the part of the navigator is “POI”.
- match: the part of the navigator is “match”.
- video: the part of the navigator is “video”.
- video for sequence: the part of the navigator is “video for sequence”.
- video for match: the part of the navigator is “video for match”.

The state of an instance of the **Navigator** class may possibly modified when one of the following events occurs:

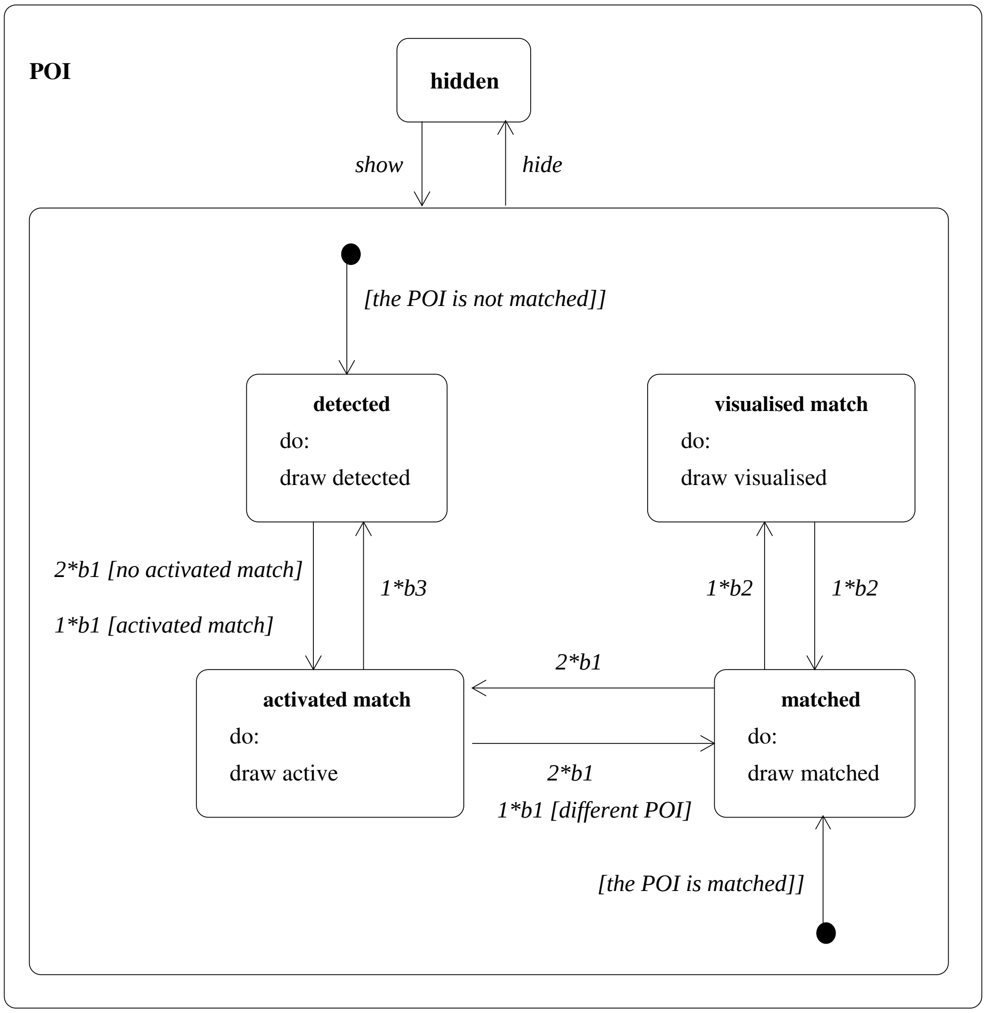

Figure 20: **POI states diagram**

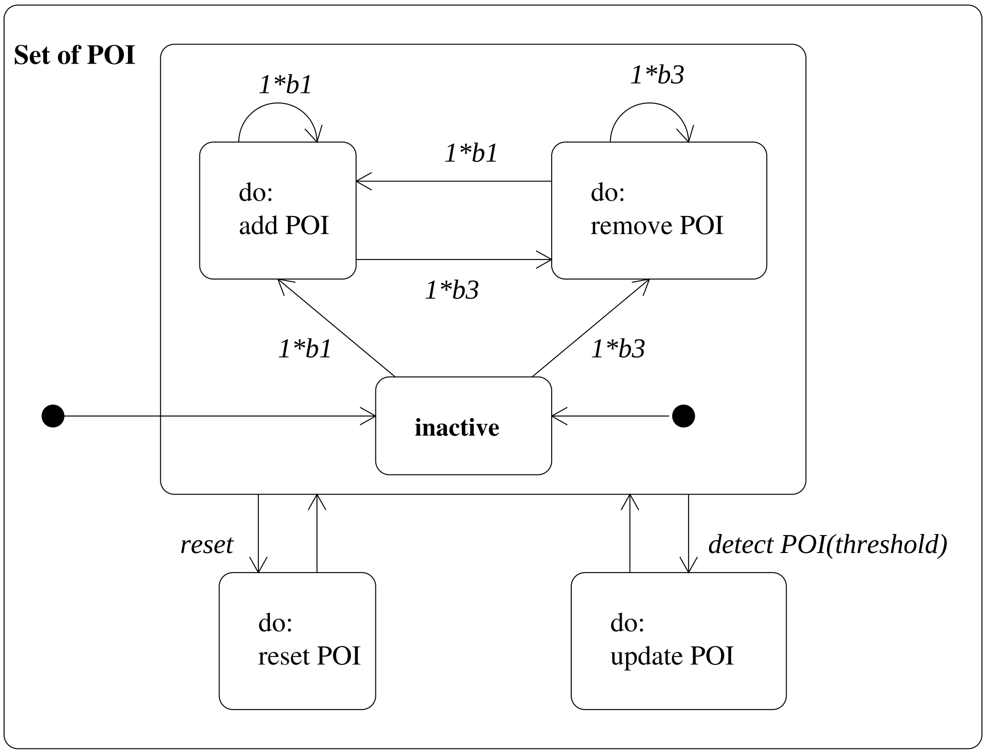

Figure 21: **Set of POI states diagram**

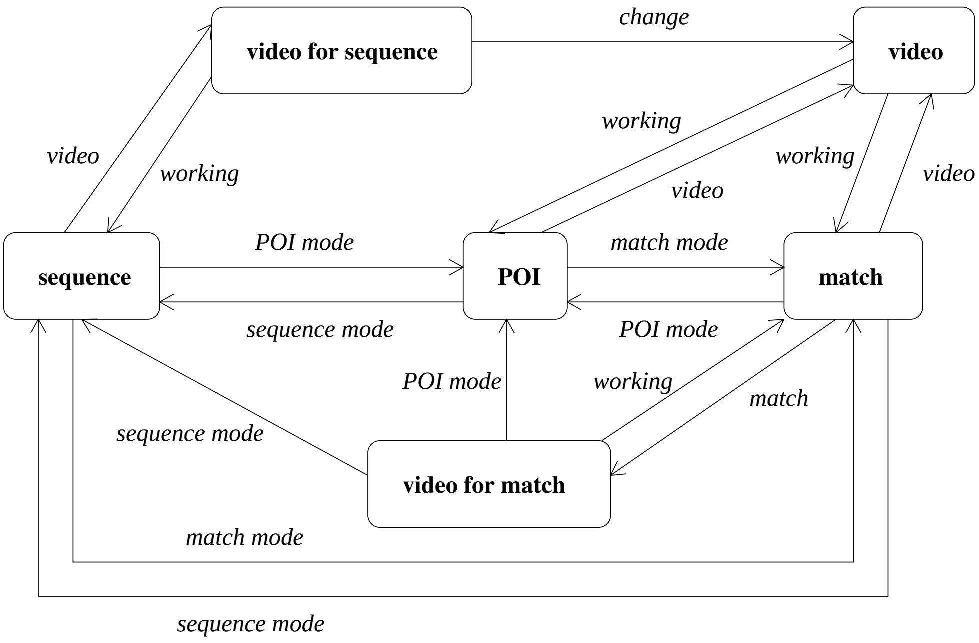

Figure 22: **Navigator states diagram**

- *change*: the current mode of the application switches to a different mode.
- *sequence mode*: the application switches to the sequence mode.
- *POI mode*: the application switches to the POI mode.
- *match mode*: the application switches to the match mode.
- *video*: the users requests to navigate through the video sequence.
- *working*: the users asks for the control the working sequence.
- *match*: the users wishes to visualize a match.

3.4 Match

This section deals with the states of an instance of the **Match** class when the match is defined or updated. We suppose too that the match mode is the active mode.
The events which leads to a change of the states are the following:

- *1*b1*: the user clicks on a 2D point with the mouse button1.

- *2*b1*: the user "double clicks" on a 2D point with the mouse button1.
- *1*b3*: the user clicks on a 2D point with the mouse button3.

The states diagram uses the conditions:

- *matched image*: the image contain a POI of the active match.
- *no matched image*: the image contains no POI of the active match.
- *matched POI*: the POI belongs to a match.
- *no matched POI*: the POI belongs to no match.
- *the POI is not still defined*: the previous clicks are supposed to be clicks on defined POI. Also, this condition is occurs when a POI is not still defined.

3.5 Sequence

This section describes the different states of an object whose the super class is the **Sequence** class. The used events are:

- *forward*: the users requests to point out to the next image of the sequence.
- *backward*: the users requests to point out to the previous image of the sequence.
- *play*: the users requests to automatically point out forward.
- *stop*: the users requests to stop the automatic navigation.
- *reset*: the user request to point out to the first image of the sequence.

To the previous events, we associate corresponding actions with a similar name: for instance, the name forward expresses both the events **forward** and the action forward which increments the number of the pointed image.

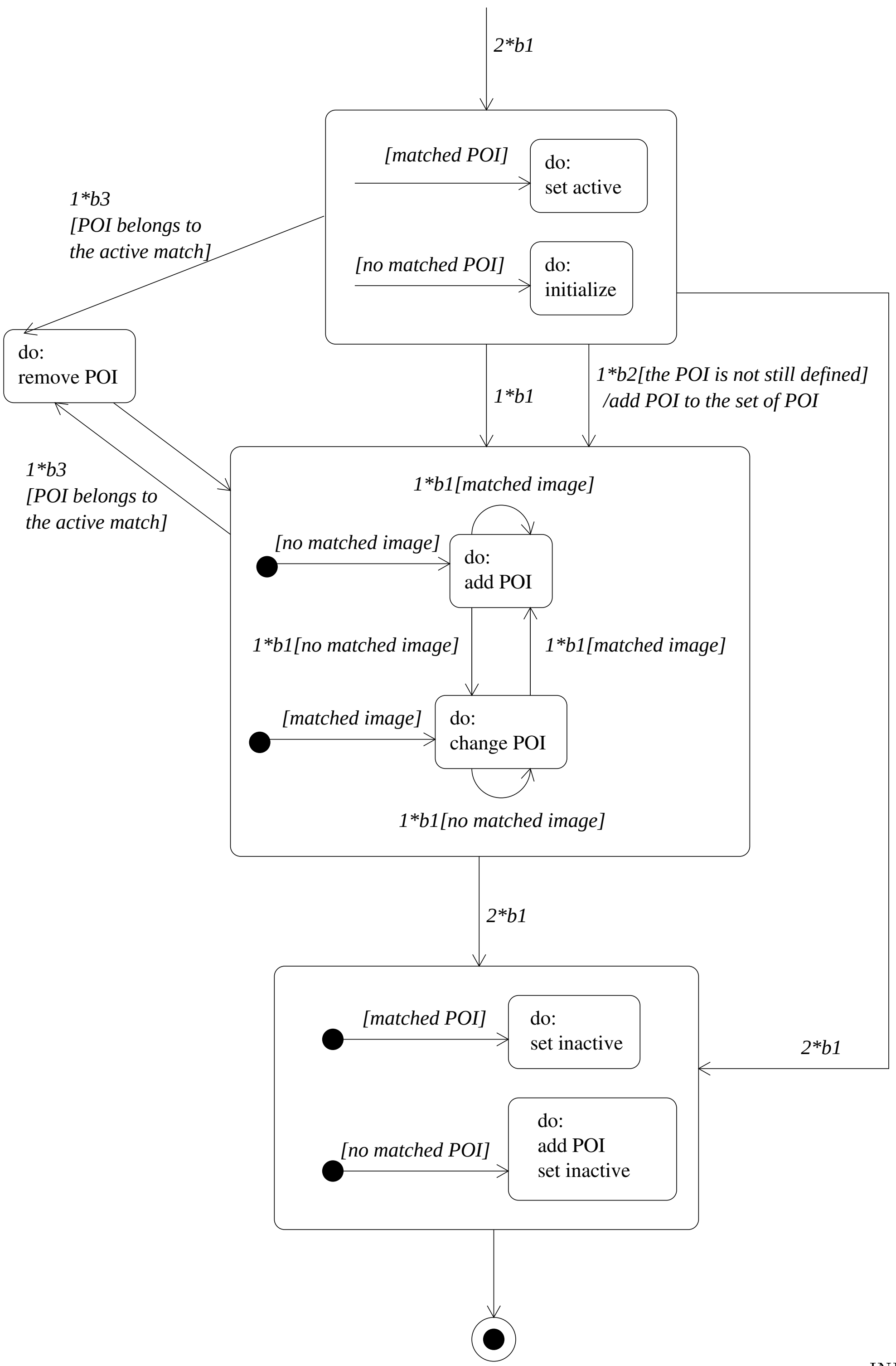

Figure 23: **Match states diagram**

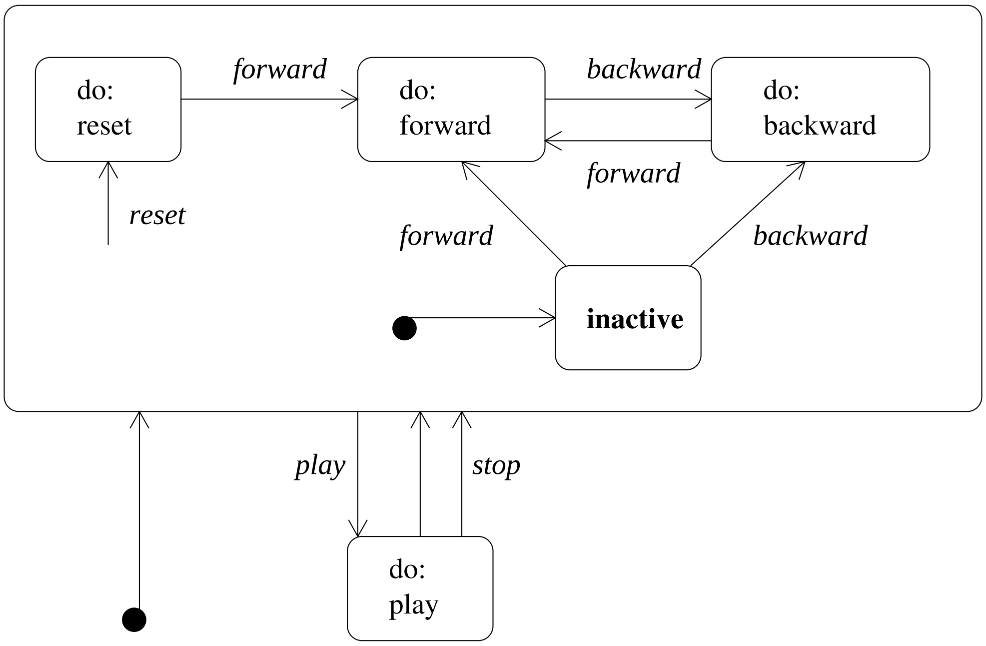

Figure 24: **Sequence states diagram**

4 The functional model

The functional model describes the processes achieved the running of the application. It is made up of several data flows diagrams.

4.1 Main diagram

The main diagram is a data flow diagram describing the high level processes.
The data input and output are the 2D coordinates of some selected points and instances of the classes **Image**, **Match**, **POI**, **Video Input**, **Set of POI**, **Set of Matches**, **Working Sequence**, **Video sequence** and **Match sequence**.
The instances of the classes **Video input**, **Set of matches** and **Set of POI** store the data for later accesses; these are data tanks.
The instances of the classes**Working sequence**, **Video sequence**, **Match sequence**, and the user are actors because they produce or consume data, directing them.

4.2 Select

The process "select" of the main data flow diagram is defined by the flow diagram of the figure 26.
The actor "sequence" is, according the part of the navigator, an instance of one of the three classes titled **Working sequence**, **Video sequence** and **Match sequence**.

4.3 Define working sequence

The figure 27 corresponds to the diagram of the high level process "define working sequence".

4.4 Define POI

The high level process "define set of POI" is described by the diagram of the figure 28.

4.5 Define matches

The high level process "define match" corresponds to the diagram of the figure 29.

4.6 Define match sequence

The high level process "define match sequence" is described by the figure 29.

4.7 Update parameters

The high level process "update parameters" corresponds to the diagram of the figure 31.

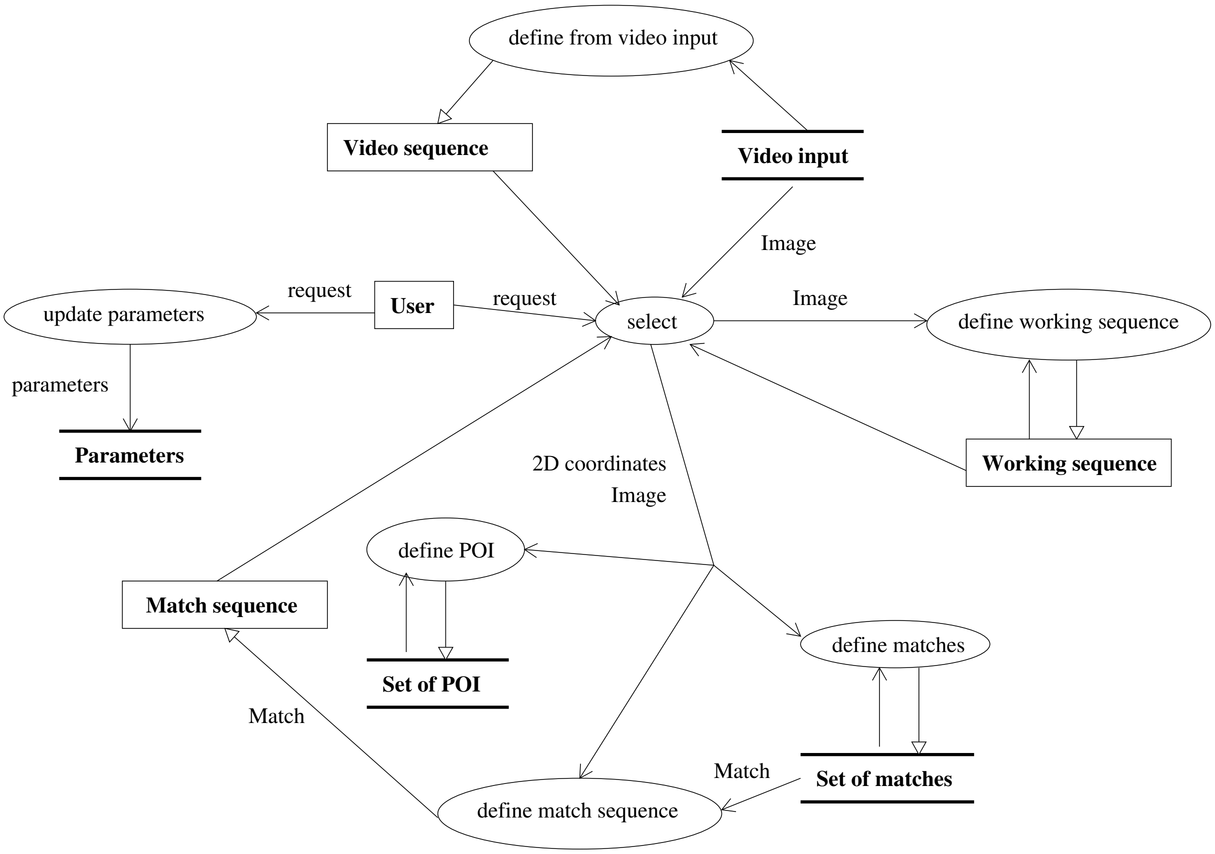

Figure 25: **Main data flow diagram**

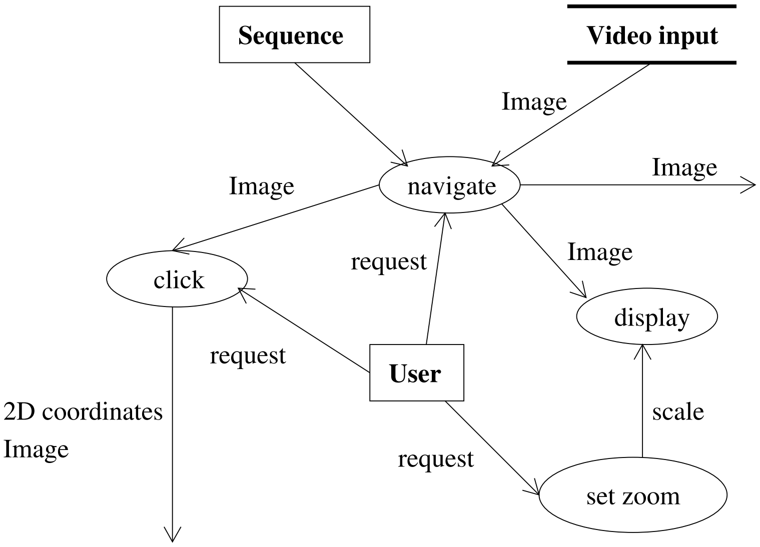

Figure 26: **select**

References

[1] J. Rumbaugh. Object oriented modeling and design. 1997.

[2] C. Schmid, R. Mohr, and C. Bauckage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):pp151–172, 2000.

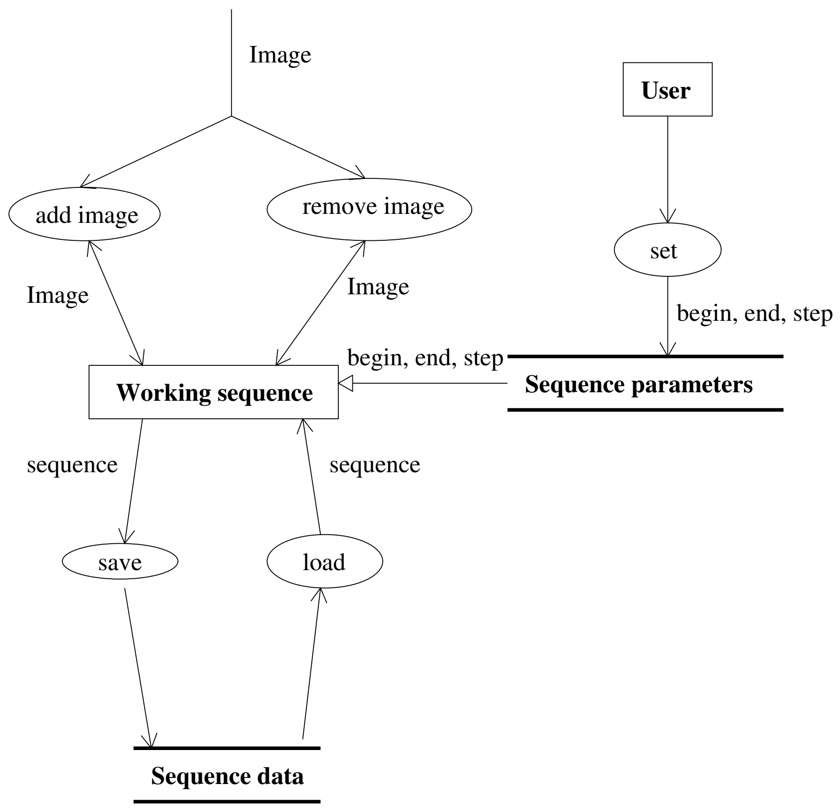

Figure 27: **define working sequence**

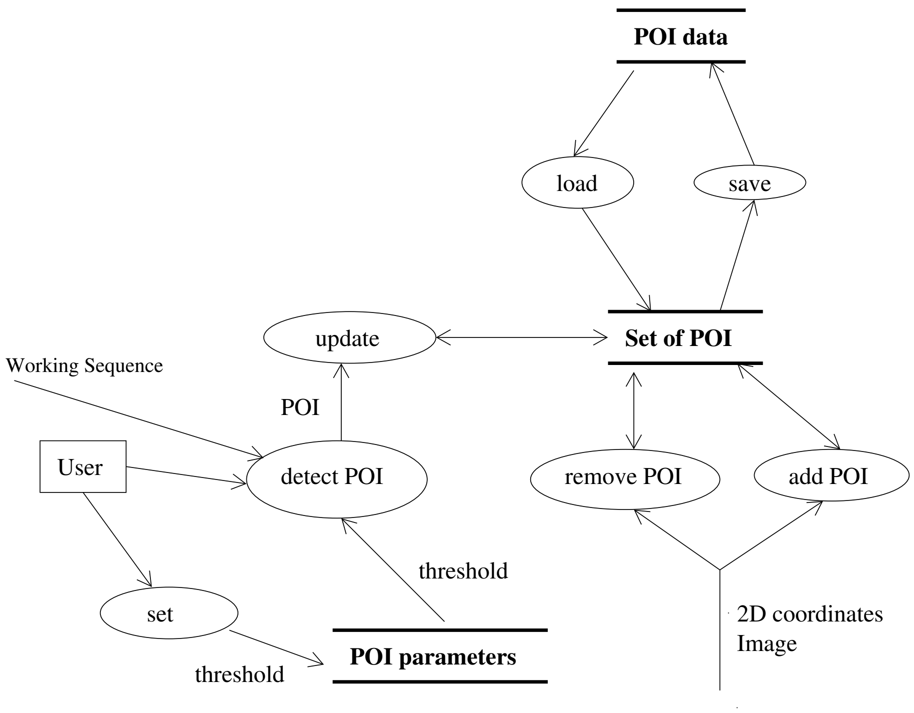

Figure 28: **define POI**

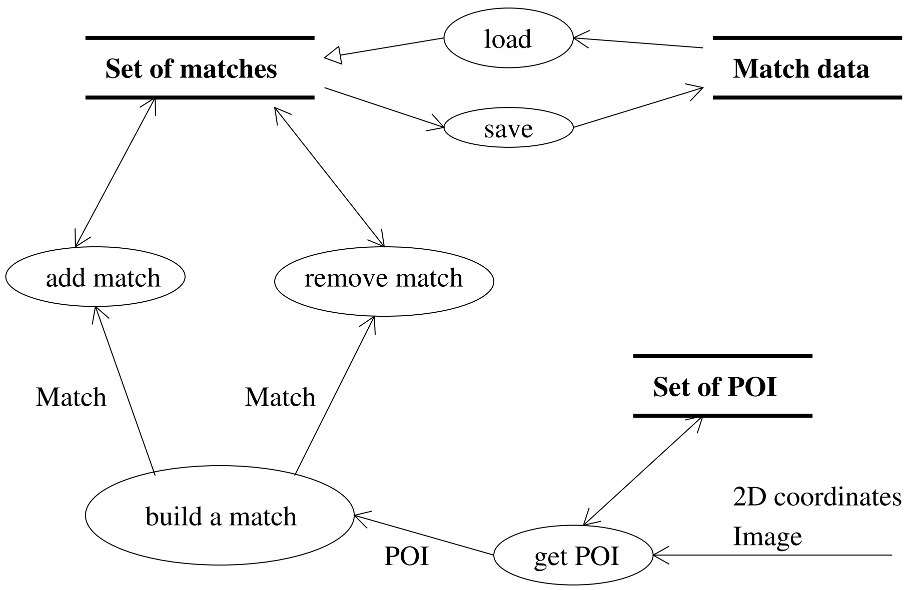

Figure 29: **define matches**

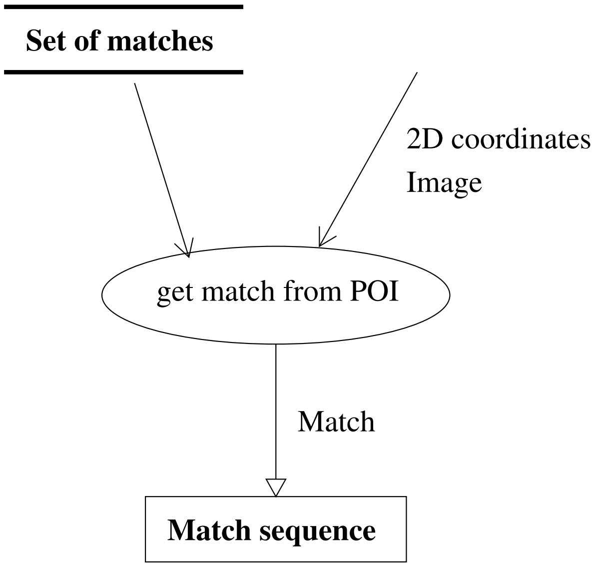

Figure 30: **define match sequence**

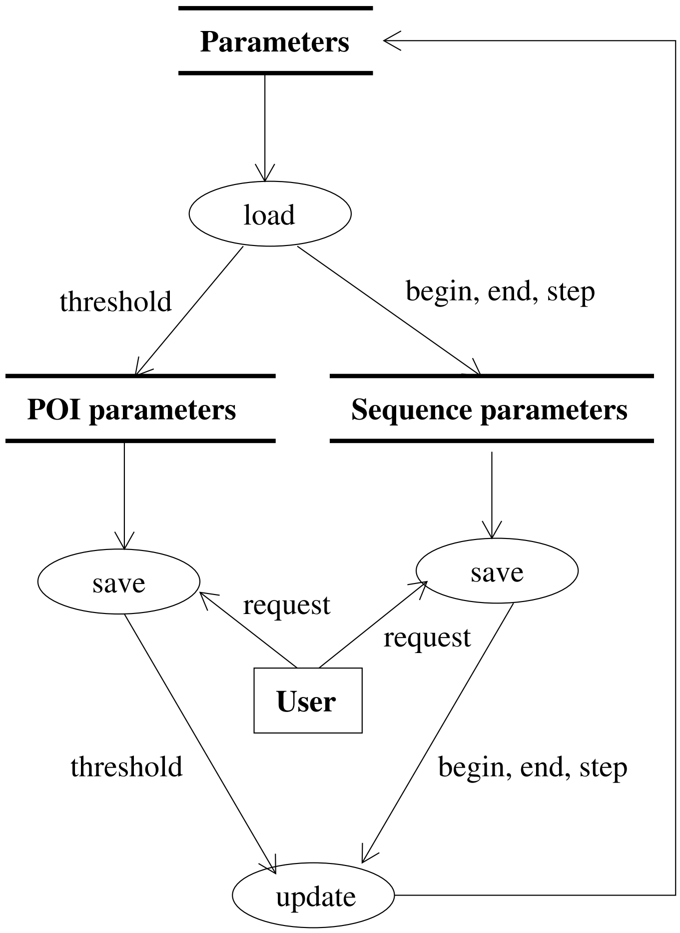

Figure 31: **update parameters**

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-0803