Design of a Library for Dense Matching

Matthieu Personnaz, Peter Sturm, Frédéric Devernay

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

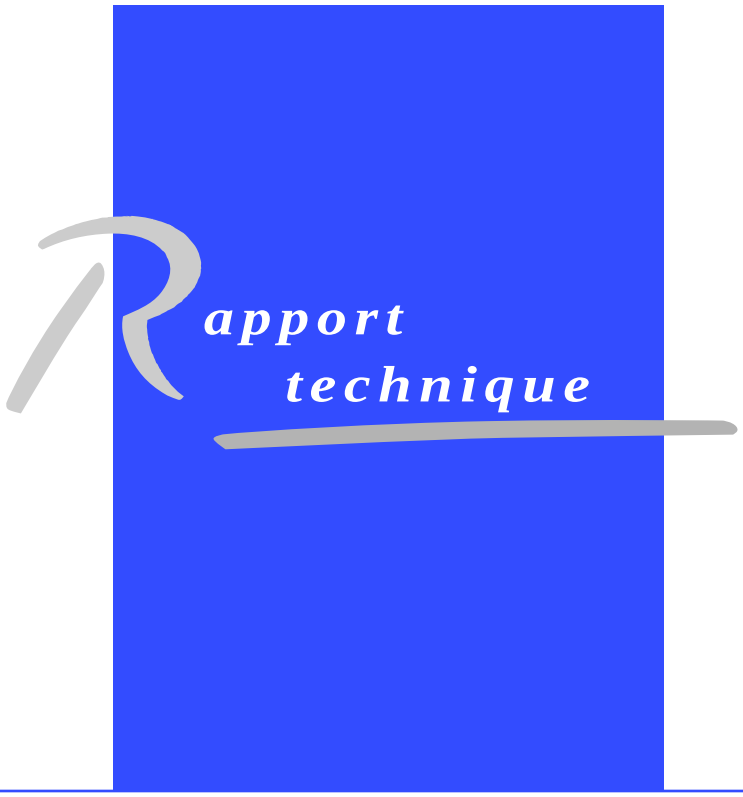
Design of a Library for Dense Matching

Matthieu Personnaz — Peter Sturm — Frédéric Devernay

N° 0278

April 2003

THÈME 3



ISSN 0249-0803

Design of a Library for Dense Matching

Matthieu Personnaz , Peter Sturm , Frédéric Devernay

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

Rapport technique n° 0278 — April 2003 — 34 pages

Abstract: The algorithms involved for a dense 3D reconstruction from two frames are highly dependent of the viewed scene. Furthermore, the user goals themselves induce should induce a specific choice. Indeed, the user focuses alternatively on the rendering, the running speed or the metrologic quality.
Also, the aim of this paper is the design of a software library which allows to perform a dense matching between two images, by taking account both of the various user intents and world scenes. This leads first to the integration of several performant and possibly complex dense matching algorithms. The multiplicity and the complexity of the algorithms should however not be at the expense of the ergonomy of the software system. A futher goal is therefore to keep an easy and safe access to the library components.
An iterative and object-oriented process is applied in order to cop with the design of a such library.

Key-words: dense matching, 3D reconstruction, design patterns, UML, XP process

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

Architecture d'une librairie dédiée à la mise en correspondence dense

Résumé : Les algorithmes utilisés pour la reconstruction 3D à partir d'une paire d'images stéréo dépendent fortement de la scène observée. Les objectifs de l'utilisateur constituent également un critère de choix. En effet, ce qui importe généralement à l'utilisateur de tels algorithmes alterne entre le rendu visuel de la reconstruction, la vitesse d'éxecution ou encore la qualité métrologique.
Aussi, l'objectif de ce document est de définir l'architecture d'une librairie logicielle tenant compte à la fois des intentions de l'utilisateur et de la variété des scènes 3D réelles. La multiplicité ainsi que l'éventuelle complexité des algorithmes ne doivent cependant pas empiétér sur l'ergonomie de la librairie. Une seconde priorité de la librairie est donc de préserver un accés simple et fiable à ses composants.
Un procédé itératif et orienté objet est mis en oeuvre afin de spécifier une telle architecture.

Mots-clés : mise en correspondence dense, reconstruction 3D, architecture orientée objet, UML, XP process

Contents

INTRODUCTION

The quality of the algorithms involved for a dense 3D reconstruction from several frames is highly dependent of the viewed scene (see [8]). The dense stereo correspondence algorithms are in particular subject to this property. To provide a flexible solution to this scene-dependent problem, this document designes a software library whose the main features are:

- Provide a qualitative evaluation, mainly based on the image rendering, of different methods.
- The set of algorithms includes the dense stereo mathing, the dense reconstruction and the rectification.
- The user may run most of the processes with a minimal knowledge in software development and computer vision.
- Any process may be quickly integrated in a program written in C, C++, python or java langage.
- The development tools are free and allow a free distribution of the library.

In order to address the achitecture of the library, the paper is organized into the four subsequent parts:

- The glossary captures definitions.
- The requirements are the capabilities and the conditions the library must conform.
- The domain model is an illustration of the noteworthy abstractions, dense matching vocabulary, and information content of the software library.
- The design model specifies the software classes that participate to the software solution.

Thus, the ultimate goal of this document is the production of diagrams which specify the library and may be translated with a low cost in a programmation langage.
The described analysis and design of the library focuses on the inception and the elaboration phases of an iterative development based on the XP (Extreme Programming) process (see [4]). That means that the four parts of the document had been incrementally introduced and refined through many iterations. However, the paper doesn't deal neither with the implementation details nor with the tests performed at the end of each iteration.
At last, the design choices are essentially issued from Patterns described in [2], i.e on elements of reusable object-oriented software. The drawing of the diagrams is based on the UML convention. The definition of some UML artefacts may be found in [2].

1 Glossary

The glossary defines terms associated to the dense matching. These definitions have raised during the requirements specification and they are used in the whole document.

Rectification the process of resampling pairs of stereo images in order to produce a pair of projections in which the epipolar lines run parallel to the x-axis and match up between views. Hence the correspondence between a pixel (x,y) in the reference image and a pixel (x',y') in the matching image is given by:

$$\begin{cases} x' &= x + sd(x,y) \\ y' &= y \end{cases} \quad (1)$$

where for a given pixel (x,y), s is a sign chosen so that the disparity d(x,y) is positive. Let notice that a similar equation stands by transposing the x-coordinates with the y-coordinates in the equation 1. To distinguish the two possible rectifications we may apply to a $(referenceimage, matchingimage)$ pair, the first (defined by 1) and the second rectifications are respectively referred to as the rectification according to the x-axis and the y-axis.

Disparity map Set of disparity values d(x,y) (see equation 1) associated to a stereo pair, (x, y) being pixels of the reference image. The disparity map may as well be considered as a depth map (see [6]).

Disparity interval : The minimal length interval which contains all the disparities associated to a rectified pair.

Dense matching : Process of two images, the reference frame and the matching frame, which estimates the disparity value at each pixel of the reference frame. Also, a dense correspondence produces a dense disparity map.

WTA for Winner Takes All. In the context of the dense correspondence, the WTA algoritm associates to each pixel the density associated to the minimum cost value over all disparity values, assuming the unicity of this value.

Local method dense matching method which is mainly made up of two steps: the matching cost computation and the "winner takes all optimization" (WTA) optimization at each pixel. A possible intermediary step may be added in order to filters the costs (see [8]).

Global method dense matching method which is generally formulated in an energy-minimization framework. Global method may also denote the optimization algorithm used to solve the minimization problem.
A set of costs C((d(x,y), x,y) being computed after a matching cost computation step, a global method aims at estimating the disparity as the application which minimizes the global energy:

$$\begin{cases} E(d) &= E_{data}(d) + \lambda E_{smooth}(d) \\ E_{data}(d) &= \sum_{(x,y)} C(x,y,d(x,y)) \end{cases} \quad (2)$$

where the E_{smooth} encodes the smoothness assumptions made by the algorithm.
The E_{smooth} term chosen is defined by Veksler [9]. It corresponds to the piecewise constant assumption and takes advantage of contextual information by increasing the smoothness cost for low intensity gradient.
E_{smooth} is written:

$$E_{smooth}(d) = \sum_{(x,y)} \rho(d(x,y) - d(x+1,y)) + \rho(d(x,y) - d(x,y+1)) \tag{3}$$

with:

$$\rho((d(x,y) - d(x+1,y)) = \begin{cases} 0 & \text{if } d(x,y) - d(x+1,y) \neq 0 \\ \rho_I(|I(x,y) - I(x+1,y)|) & \text{if not} \end{cases} \tag{4}$$

A similar expression stands for $\rho((d(x,y) - d(x,y+1))$. The contextual information ρ_I is defined by:

$$\rho(\Delta I) = \begin{cases} \lambda.p & \text{if } \Delta I < threshold \\ \lambda & \text{if not} \end{cases} \tag{5}$$

with $p \geq 1$ and $\lambda \geq 1$. The last system means the smoothness cost is increased for low intensity gradient.

Matching cost Given a stereo pair of images, a matching cost is the restriction to $I_r \times I_m$ of a distance on pixels, where I_r and I_m are respectively the pixels of the reference image and the pixels of the matching image. A matching cost between two pixels generally measures the similarity beetwen the neghborhood of the both pixels.

Costs map A discrete finite set of disparities being given, the image formed to the whole provided disparities may be referred to as the costs map.

AD for Absolute Difference. Matching cost which is defined by:

$$C((x,y),(x',y')) = |I_r(x,y) - I_m(x',y')| \tag{6}$$

where (x,y) and (x',y') are pixels which belongs respectively to the reference image and the matching image of a stereo pair. I_r and I_m are the intensities functions associated to the images.

SD for Squared Difference. By keeping the notations of the AD definition, SD is written:

$$C((x,y),(x',y')) = (I_r(x,y) - I_m(x',y'))^2 \tag{7}$$

SAD for Sum of Absolute Differences. By keeping the notations of the AD definition, SAD is written:

$$C((x,y),(x',y')) = \sum_{(i,j)\in[1,s]\times[1,t]} |I_r(x+i,y+j) - I_m(x'+i,y'+j)| \tag{8}$$

SSD for Sum of Squared Differences. By keeping the notations of the AD definition, SSD is written:

$$C((x,y),(x',y')) = \sum_{(i,j)\in[1,s]\times[1,t]} (I_r(x+i,y+j) - I_m(x'+i,y'+j))^2 \qquad (9)$$

ZSAD for Zero-mean Sum of Absolute Differences. This cost is insensitive to the local brightness bias between the two images. By keeping the notations of the AD definition, ZSAD is written:

$$C((x,y),(x',y')) = \sum_{(i,j)\in[1,s]\times[1,t]} |(I_r(x+i,y+j)-\bar{I}_r(x,y))-(I_m(x'+i,y'+j)-\bar{I}_m(x',y'))| \qquad (10)$$

where $\bar{I}_r(x,y)$ and $\bar{I}_m(x',y')$ represent the average of the respective intensities I_r and I_m on the (x,y) and (x',y') centered windows defined by the product $[1,s]\times[1,t]$.

ZSSD for Zero-mean Sum of Squared Differences. This cost is insensitive to the local brightness offset between the two images. By keeping the notations of the ZSAD definition, ZSSD is written:

$$C((x,y),(x',y')) = \sum_{(i,j)\in[1,s]\times[1,t]} [(I_r(x+i,y+j)-\bar{I}_r(x,y))-(I_m(x'+i,y'+j)-\bar{I}_m(x',y'))]^2 \qquad (11)$$

ZNSSD for Zero-mean Normalized Sum of Squared Differences. This cost is insensitive to the local brightness gain and offset between the two images. By keeping the notations of the ZSAD definition, ZNSSD is written:

$$C((x,y),(x',y')) = \frac{\sum_{(i,j)\in[1,s]\times[1,t]}[(I_r(x+i,y+j) - \bar{I}_r(x,y)) - (I_m(x'+i,y'+j) - \bar{I}_m(x',y'))]^2}{\sigma_r(x,y)\sigma_m(x',y')} \qquad (12)$$

with:

$$\begin{cases} \sigma_r(x,y) &= \sqrt{\sum_{(i,j)\in[1,s]\times[1,t]}(I_r(x+i,y+j) - \bar{I}_r(x,y))^2} \\ \sigma_m(x',y') &= \sqrt{\sum_{(i,j)\in[1,s]\times[1,t]}(I_m(x'+i,y'+j) - \bar{I}_m(x',y'))^2} \end{cases} \qquad (13)$$

Dynamic Programming Dynamic programming determines a class of optimization algorithms. The algorithm described by Bobick and Intille in [1] is applied to minimize the energy function defined by the equation 2. It assumes the ordering constraints is checked.

The algorithm defines a correspondence between the pixels of two matching scanlines by computing a minimum-cost path in a specific dynamic programming graph. The nodes of this graph are the pairs (x_i, d_j) where the x_i $(1 \leq i \leq n)$ are the ordered pixels of a given scaline in the reference image and d_j $(1 \leq j \leq m)$ are the possible disparities. Each node is in one of the following three steps:

- (x_i, d_j) corresponds to a match. The transition to a next node get charged the matching cost associated to (x_i, d_j).
- (x_i, d_j) does not correspond to a match and the matching pixel defined by $x'_i = d_j + sx_i$ (see the equation 1) is not visible in the reference image. The cost of the transition to an other node is the occlusion cost k.
- (x_i, d_j) does not correspond to a match and x_i is not visible in the matching image. The cost of the transition to an other node is the occlusion cost k.

For a node in a given state, a few transitions to a neighborhood node is allowed. To the cost of some of them is added the smoothness cost so the disparity jumps are related to the intensity gradient in the reference image (see equation 5). The cost to each possible transitions being specified, the dynamic programming algorithm computes the minimum-cost path between (x_1, d_1) and (x_n, d_1).
Let notice the value of the penalty k may dramatically modify the behavior of the algorithm. If the penalty value is too low, only the reliable pairs, i.e. whose the matching cost is near zero, are matched. If k is too high, the most pairs are matched but the occlusions are not detected anymore. Also the penalty value should be carefully choosen in order to obtain a meaningful disparity map.

Graph cuts Graph cut is a global optimization method which aims at minimizing the energy defined by the equation 2. The chosen algorithm corresponds to the α-β swap move algorithm described in [9].

Scanline Optimization This is an optimization technique described in [8]. Like the dynamic programming, scanline optimization may be seen as a global method and operates on matching scanlines. It solves the optimization problem defined by the system 2 without taking account of the vertical smoothness term.

Bayesian Diffusion Bayesian diffusion is a global method made up of a probalistic model of the stereo matching problem and an optimization technique to solve it. The both are described in [7].
A bayesian model of the stereo matching leads to estimate the disparity function d which minimizes the following energy function:

$$E(d) = \sum_{i,j} \rho_P(d_{i+1,j} - d_{i,j}) + \rho_P(d_{i+1,j} - d_{i,j}) + \sum_{i,j} \rho_M(C_{i,j}) \tag{14}$$

with:

$$\begin{cases} d_{i,j} & = & d(x_i, y_j) \\ \rho_M(x) & = & -log((1-\epsilon_M)e^{-x^2/2\sigma_M^2} + \epsilon_M) \\ \rho_P(x) & = & -log((1-\epsilon_P)e^{-x^2/2\sigma_P^2} + \epsilon_P) \end{cases} \tag{15}$$

and where $C_{i,j}$ denotes the matching cost of the pixel (x_i, y_j) of the reference image with the pixel $(x_i + d_{i,j}, y_j)$ of the matching image.

The ρ_M and ρ_P functions characterizes the use of contaminated gaussian models for the respective distributions of the matching cost and the disparity gradient.
The final disparity function is estimated after successively performed an algorithm based on diffusion of the energies of each site and a WTA algorithm.
The values of ϵ_P, σ_P, ϵ_M, σ_M and the number of itererations connected to the diffusion process define the behavior of the method:

- σ_M is connected to the matching cost noise.
- ϵ_M should be the likelihood of outlier measures or occlusions.
- Small values of σ_P favor fronto-parallel surfaces.
- ϵ_P characterizes the discontinuities of the surface.
- The number of iterations control the size of the support of the energy aggregation inherent to the diffusion process.

2 Requirements

Requirements envision the library scope and the user cases. Now, it is usual that the requirements are imperfect and they generally evolve through the iterations of the XP process. Nevertheless, there may remain some differences between the final software library and the requirements. Indeed, the requirements are updated only if the new informations are implied in a further design.

2.1 Use case model

The use case model defines the requirements of the user. It is illustrated by the use case diagram. This section aims at describing its components.

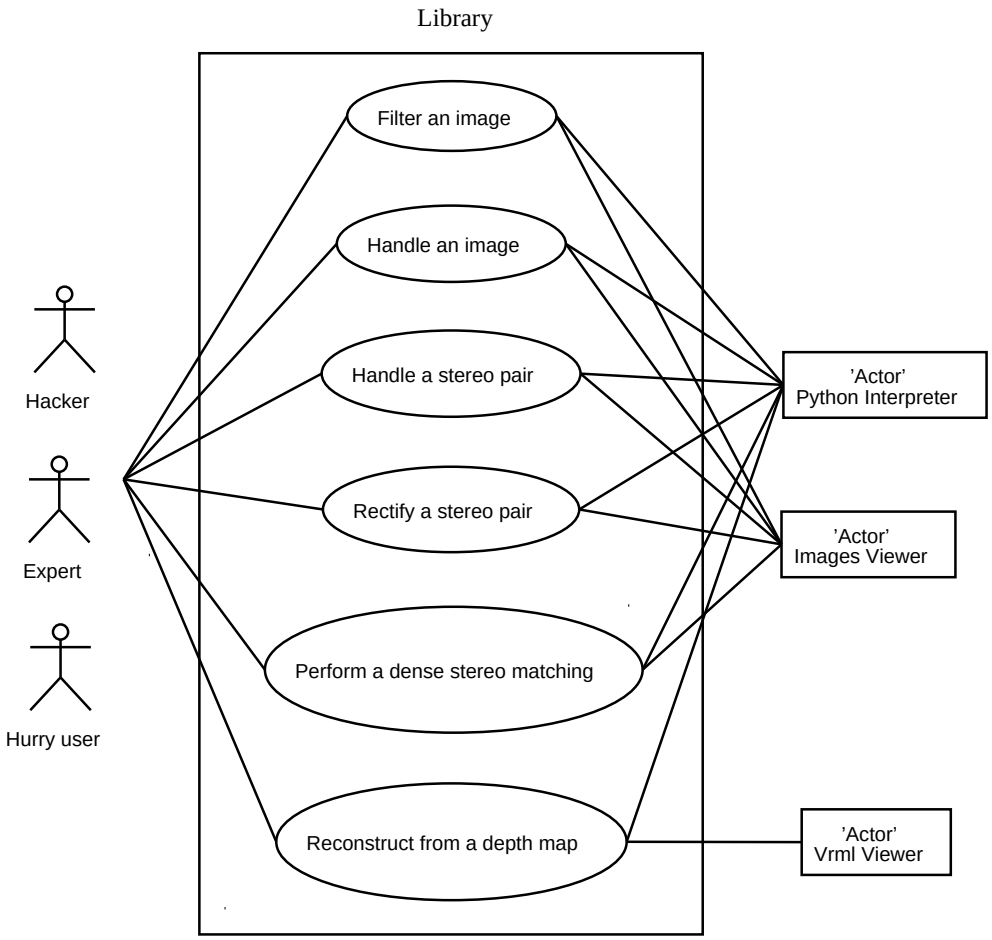


Figure 1: The Use Case Diagram summurizes the behavior of the library and its actors.

2.1.1 System boundary and primary actors

The system is a software toolkit, i.e. a set of reusable objects designed to provide useful, general-purpose functionalies. The goals are fulfilled through composition or derivation of the elements of the toolkit. Thereby, the nature of the system leads us to introduce very early in its design some programming aspects and further technical features.

Let's define now the primary actors, i.e. the users of the services provided by the library:

Hurry user The intent of the Hurry user is to obtain a 3D reconstruction of a scene and maybe some intermediary results by spending minimal efforts in the understanding of the library or the topic of the 3D dense reconstruction.

Expert The Expert wishes to use some specific and possibly optimal algorithms related to the dense matching of a given stereo pair of images.

Hacker The Hacker wants to integrate some features of the software into his own system. He may also wish to contribute to the developement of the library.

The previous classification is not a partition of the users: we may want to be simultaneously Expert, Hurry user and Hacker.

2.1.2 Filter an image

Primary actor: Expert

Stakeholders and interests:

- Expert: Wants to process images by invoking some basic linear filters, non-linear filters or morphological filters. Wants to pre-process images for bias-gain or histogram equalization. Wants to process the disparity map. Wants to filter the cost map.
- Hurry user: Wants to filter images with minimal efforts.
- Hacker: Wants to experiment some properties of the toolkit before to integrate them in his own system.

Preconditions: An Image object is available.

Postconditions: The return result is a filtered image stored and defined by a new Image object.

Main success scenario:

1. The user creates an object filter, i.e an element of the library, designed to perform the wished filtering.
2. The system returns a filter object.
3. The user run the process by invoking the filter command. The filtering object previously returned and the image object are passed as parameter of the command.
4. The system returns a new filtered image, i.e. a new instance of the Image class.
5. The user visualizes the result, treats it by repeating the Main success scenario, or leaves.

Extensions:

1-2a. The default configuration of the filter object does not satisfy his goal:

1. The user modifies the configuration of the returned filter object or invokes the filter constructor by passing additional parameters.

Technology and data variations list:

4a The data belongs to float or byte types.

4b The handled images may be color images or grey level images.

4c The results are components of an existing image library. For instance, they may derive from the image class of the Python Imaging Library (PIL).

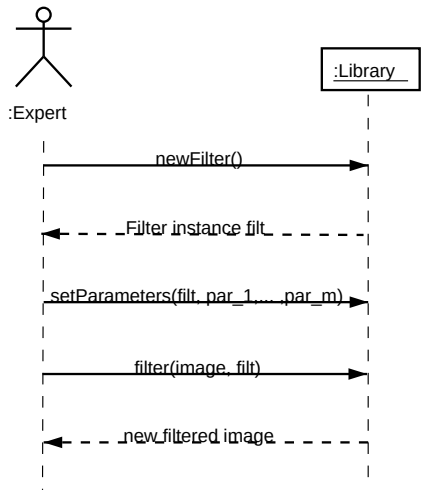


Figure 2: Toolkit Sequence Diagram for image filtering

2.1.3 Handle an image

Primary Actor: Expert

Stakeholders and Interests:

- Expert: Wants to handle images for Input/Output purposes, or some basic operations as rotation or cutting an in image. By opposition to the use case "Filter animage", the operations are not necesseraly correlated.

Preconditions: data images are available through an instance of the Image class, unless the scenario is an input request.

Postconditions: The return result may be a transformed image, or the vizualization of an image.

Main success scenario:

1. The user creates a new operation object.
2. The system returns a new operation object.
3. The user run the process by associating the previous operation object to the running command. The image object is as well associated to the command.
4. The system returns a new instance of the Image class.
5. The user visualizes the results, processes them by repeating the Main success scenario, or leaves.

Extensions:

1-2a. The default configuration of the operation object does not satisfy his goal:

1. The user modifies the configuration of the returned filter object or invokes the filter constructor by passing additional parameters.

1-5a. The operation is a loading operation:

1. The user calls the load operation by passing as parameter the location of the image.
2. The system returns an Image object.

1-5b. The operation is the visualization of an image:

1. The user calls the visualization operation by passing as parameter the location of the image file or the Image instance itself.
2. The system shows the image through a given 2D viewer.

Technology and data variations list: They correspond to the use use case "Filter an image".

2.1.4 Handle a stereo pair

Primary Actor: Expert

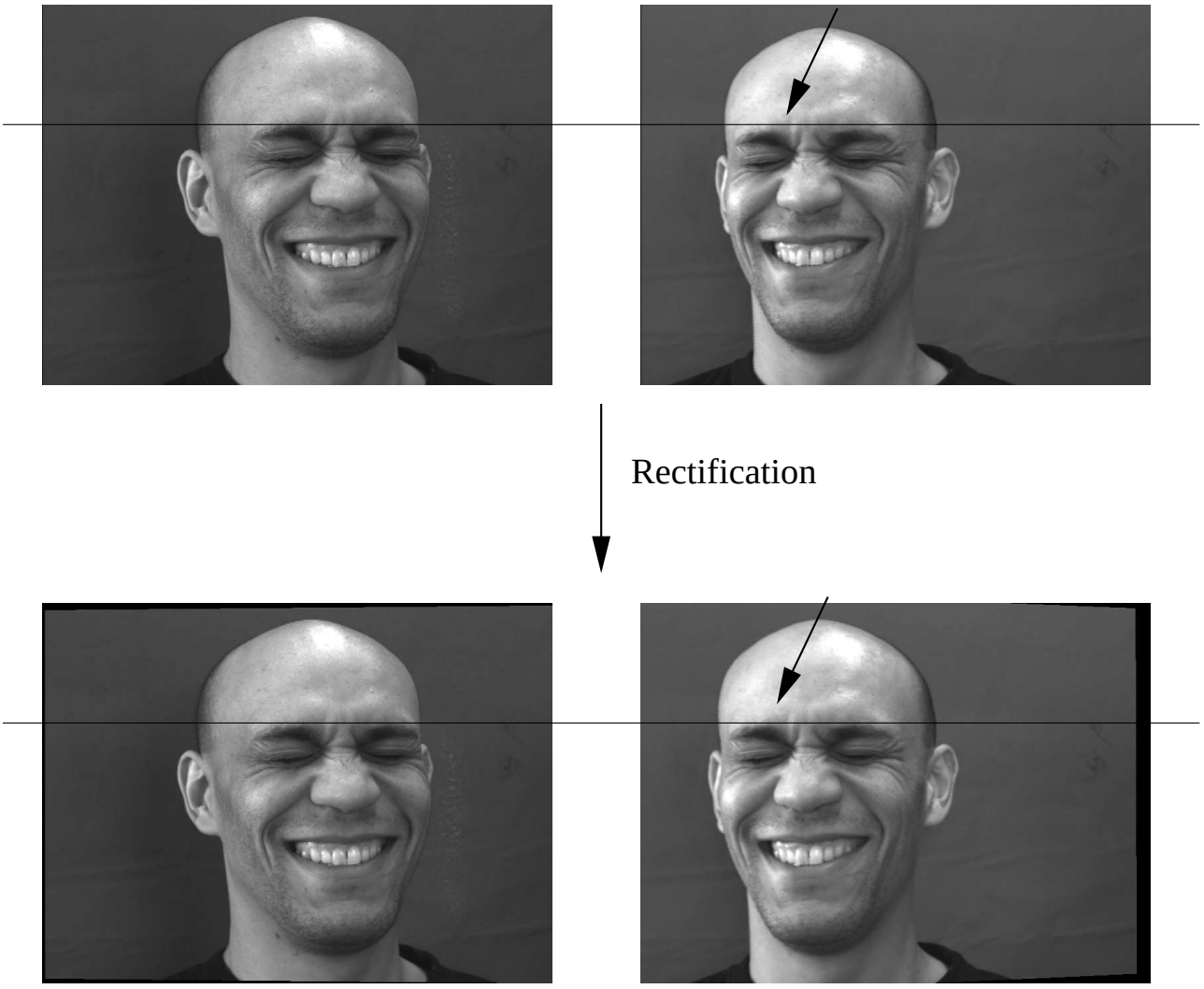


Figure 3: The rectification of the stereo pair of images may be notice by considering the distance between the line and the top of the Ben's eyebrows.

Stakeholders and Interests:

- Expert: Wants to apply to a stereo pair some treatments designed to a single image: I/O purposes or filtering. Wants to perform geometric transforms specific to a stereo pair, as the rectification.

Preconditions: a stereo pair of images is available through a stereo pair object. According to the kind of process, other objects has to be defined. These additional objects may be the rectification matrices for the rectication treatement.

Postconditions: The return result is a new stereo pair object with modified images.

Main success scenario:

1. The user creates a new operation.
2. The system returns the operation object.
3. The user runs the process by giving the operation object and the stereo pair as parameters of the command.
4. The library returns a new instance of StereoPair.
5. The user visualizes the results, process them by repeating the Main success scenario, or leaves.

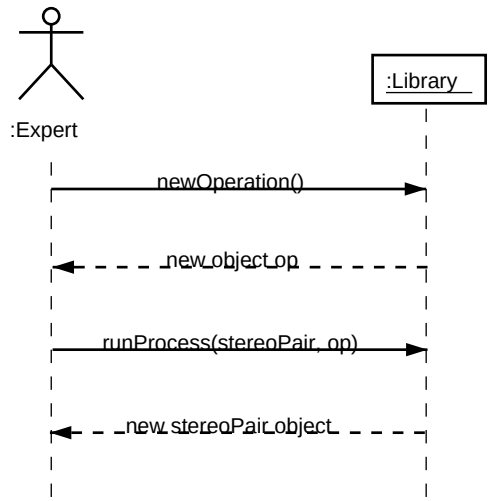


Figure 4: System Sequence Diagram for the Main stereo pair handling

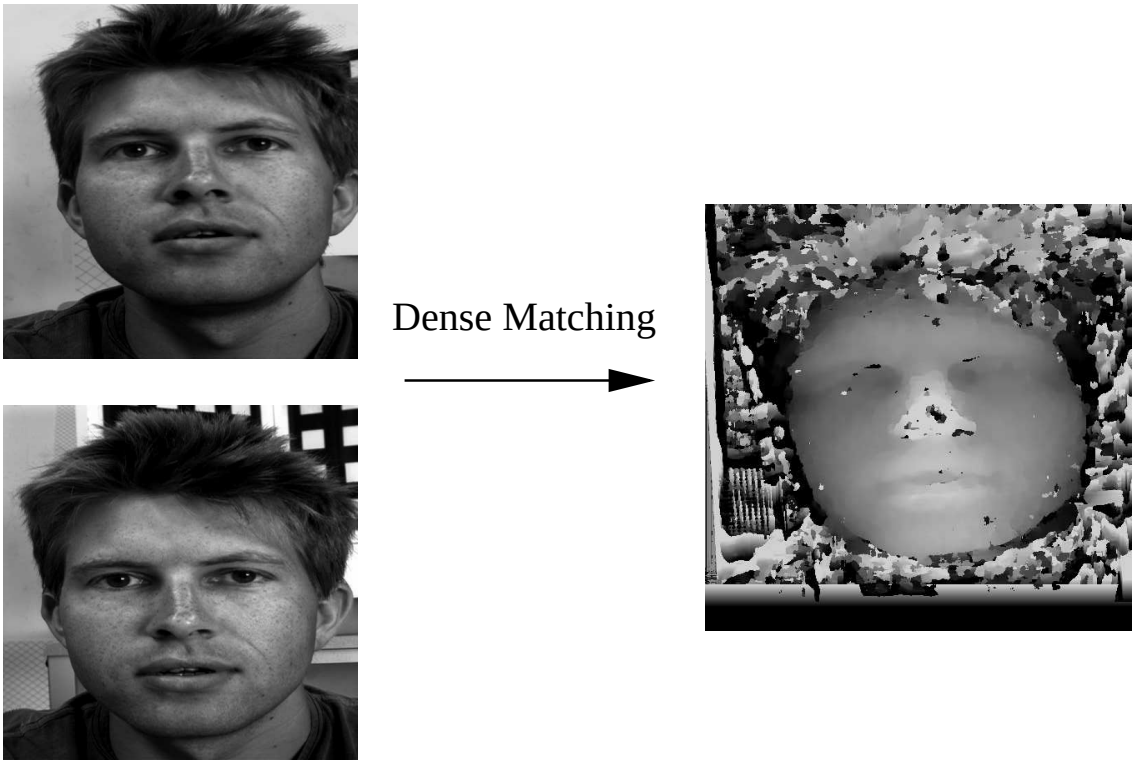


Figure 5: Production of a disparity map from a rectified stereo pair of the Hervé face.

Special requirements:

- The rectification process is perfomed in less than 1s for a 700x500 image.
- The rectification takes account of the possible cameras distortion parameters.

Technology and data variations list:

*a The expression and the handling of the matrices must be done in a similar way than the ones defined by Matlab or the Numerical Python Library.

4a The same of the use case "Filter an image".

2.1.5 Perform a dense stereo matching

Stakeholders and Interests:

Primary Actor: Expert.

- Expert: Wants to apply a particular dense stereo algorithm. Wants to specify sub processes. Wants to visualize the disparity map result.
- Hurry user: Wants to quickly perform a dense matching by using the default configuration of the tool. Wants to be guided in the choice of a specific dense matching algorithm thanks to the learning documentation.

Preconditions: A rectified stereo pair is available. A disparity interval associated to the rectified stereo pair has been estimated.

Success Guarantee: A depth map is generated.

Main success scenario:

1. The user specify a matching algorithm.
2. The system returns a DenseMatch object.
3. The user runs the matching by passing as parameter the DenseMatch object to the command. The command takes likewise the stereo pair object as parameter..
4. The system returns the depth map.
5. The user visualizes the depth map or processes it by calling other tools.

Extensions

1a. The default matching algorithm does not satisfy his goal:

   1. The user specifies a particular dense stereo matching algorithm by refining the definition of the DenseMatch object. He may also choose a particular matching cost function, set the filtering of the costs and specify a particular optimization method in order to estimate the disparity map.

Technology and Data Variation List:

4a The depth map derives from the image class of an existing library. For instance, it may derive from the image class of the Python Imaging Library (PIL).

4b The data belongs to a float type.

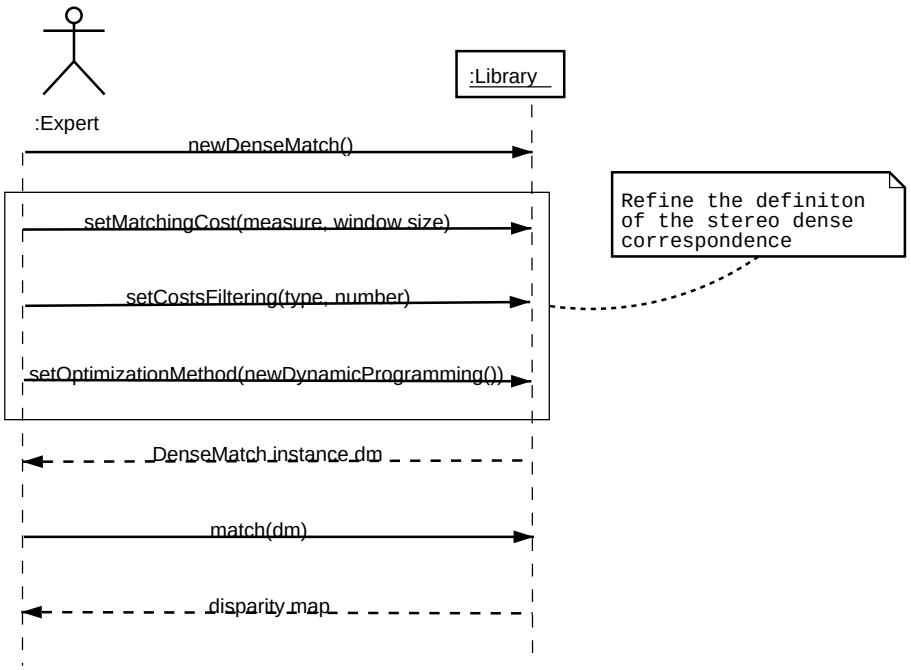


Figure 6: Sequence Diagram associated to a specific dense stereo matching.

2.1.6 Reconstruct from a depth map

Primary Actor: Expert

Stakeholders and Interests:

- Expert: Wants to perform the 3D reconstruction from the depth map and the cameras parameters. Wants to visualize the 3D view.

Preconditions: A depth map is available, connected to the cameras parameters and rectification matrices.

Success Guarantee: A dense 3D reconstruction is generated.

Main success scenario:

1. The executes the process by giving as parameters of the command the depth map and the camera parameters.

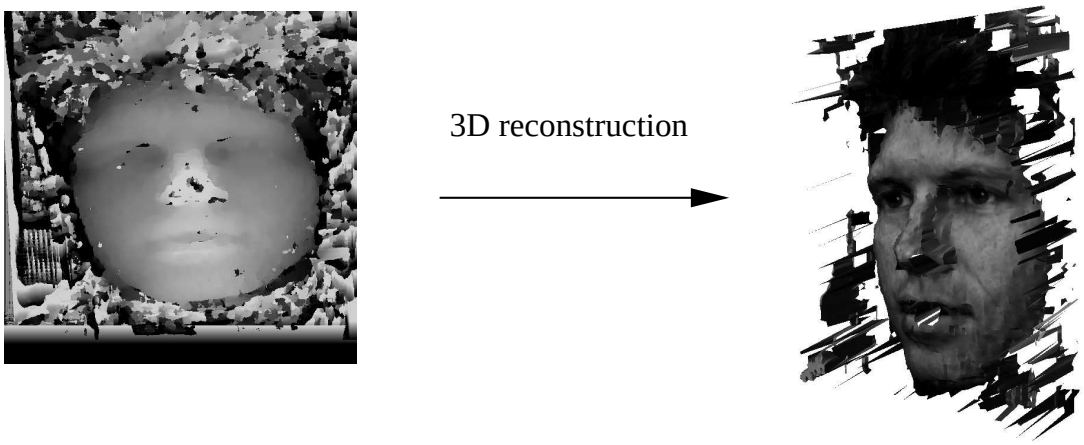


Figure 7: 3D reconstruction of the Hervé face from the disparity map.

2. The system generates and stores a 3D file.
3. The user visualizes the 3D generated file.

Technology and Data Variation List

*a As the use case "handle a stereo pair", the expression and the handling of the matrices must be done in a similar way than the ones defined by Matlab or the Numerical Python Library.

2a The results may be stored in a vrml format and be readable by a free vrml viewer.

2.2 Supplementary Specification

2.2.1 Functionality

- Image format: The software system supports the image object defined in the PIL (Python Imaging Library) library. It supports as well format files such the PPM, PGM or JPEG formats.
- Parameters format: According to the their level of complexity, the parameters of a method may either be instances of a specific class or belong to basic types.
- Error handling in the methods argument call: The system signals the errors in the input formats or type. The user may then follow the instruction of the sending message in order to correct the format or the type of the input data.

- Error handling when the system fail: The system does not handle failures which generates a core dumped. Most of them must be issued from corrupted data. In this case, the user reexecutes the intended process with new data or leaves after forwarding the data to the support of the library.

- Error handling when the process does not return: The library does not handle errors corresponding to a process which doesn't seem to return. Although this execution time depends in particular of the running algorithms and the clock frequency, the user may kill the process and check if the data are not corrupted before reexecuting the wished algorithms.

- Results hard to interpret: Let assume the results vizualization is difficult to understand. In this case, the user should apply or enhance its knowledge in the using of the object by reading the learning documentation.

2.2.2 Usability

- Competitive algorithms are intended to experts in the area of the dense correspendence or the dense 3D reconstruction.

- The commands are executed through a command line interface similar to those provided by Python, Maple or Matlab.

- A student in computer vision must be able to understand and use all the components.

- Any user may execute the whole reconstruction process by spending a minimal time in understanding dense correspondence or software development.

- Experts in the area have an exhaustive information about the implemented algorithm.

- Hackers may introduce tools in their own code.

- Hackers may retrieve the native code of a specific algorithm essentially to enhance the performance of the algorithm.

- A learning documentation decribes how to perform the whole process with a minimal knowledge concerning the dense correspondence.

- A software documentation allows either to collaborate in the development of the library or to hack it.

2.2.3 Reliability

- Failures caused by corrupted input data may cause unrecoverable failures.

- According to the used process and the runtime environment, the time processing or the lack of memory may cause unrecoverable failures.

- A command line interface, like the Python or Matlab interface, restricts the data types to the specified ones.

2.2.4 Performance

- The whole reconstruction process applied for a 700x500 stereo pair of images can be executed in less than 5s, by taking account of the subsequent hardware constraints (see the subsection hardware and software constraints) and using the fastest algorithms.
- The dense correspondence algorithms, given a convenient set of parameters, has to be able to reconstruct reference scenes.
- The intermediary results are not necesseraly stored in files to avoid the I/O cost between two successive processes.
- The algorithms may be called through a command line interface in order to perform a specific or a complex processing.
- The distribution of the library is facilitated by a convenient packaging.

2.2.5 Supportability

- The functional cohesion associated to the elements of the library should remain high, in the aim to supportcoupling with a possible graphic user interface.
- The portability towards system such as Solaris or Windows do not require a too costly programming effort.
- The components may be written in C, C++ or Python langage. The integration of new algorithms is independent of the previous programming langages and just requires the production of a wrapper.
- The library components may be integrated in a C, C++, Java or Python application.
- The camera calibration results issued from the **Tele2** (see [5]) software may be loaded.

2.2.6 Interface

The tools of the library may be executed by the python interpreter following three different ways. They are a straightforward exploitation of the python services:

- The user embeds his instructions in a python script. He also creates its own program.
- The user interactively processes images by running the python interpreter in its interactive mode. The invocation of the processing steps through a command line interface corresponds to the most common use of the library.
- The python interpreter is called to run python code embedded in a C, C++ or java program.

2.2.7 Implementation constraints

The python langage is chosen as "glue" code. Its feature of script langage and the interactive mode of its interpreter provide the wished interface to the library.
Furthermore, python allows to answer to the supportability constraints:

- The ability to embed the python code in C, C++ or java allows hackers to embed the components in their own application.
- Conversely, C, or C++ code may be integrated in the library by extending the python interpreter.
- Porting to system as Solaris or Windows is supposed to require minimal efforts.

2.2.8 Hardware and Software constraints

- The software development use free APIs (application programmer interface) and the algorithms can be called under the Linux system.
- The SWIG development tool (Simplified Wrapper and Interface Generator) should be used in order to embed C++ and possibly C into python classes.
- The hardware requirements are at least 128M for the RAM and at least 450 Mhz for the clock frequency.

2.2.9 Legal Issues

We must involve in the library only open source components whose possible licensing restrictions allow resale of products that include open source software.

2.2.10 Packaging

The distribution of the tools is facilitated by a convenient packaging.

2.2.11 Information in Domain of Interest

Parameters and specific processings involved in this section are defined in the glossary of the present document.

Dense Stereo Matching A dense stereo matching method may generally require the following five steps:

1. Filtering of the stereo pair of images.
2. Computation of the matching costs.
3. Filtering of the matching cost.

4. Estimation of the disparity map.

5. Filtering of the disparity map.

A large number of algorithms for stereo correspondences exists. The library design focuses on at least five of them for their competetive performance according to the criterias of the taxonomy of stereo methods presented in [8].

- Local method.
- Dynamic Programming method.
- Scanline Optimization.
- Graph-cut Optimization.
- Bayesian Diffusion.

The quantities measured in the taxonomy give informations about the behavior of the algorithms in particular regions such as the textureless regions, the occluded regions or the depth discontinuity regions. The time to produce a dense disparity map count as well among the criterias.
The design of the five specific methods we adress in this paper deals only with step 2, 3 and 4. Steps 1 and 5 may be solved by the functionalities of other components. The three selected steps are specified by three sets of parameters

In order to help the user in fixing the parameters, a default process is suggested for each step. The whole parameters are likewise set to default values. Now, if the library defines default parameters for all the treatments, the user should nevertheless set some of them. Also, for the method:

- Local method: the user has to define the squared window size of the support associated to the default matching cost ZSSD (zero-mean sum of squared difference).
  The user has to precise if he wants shiftable windows (see [1]) during the filtering matching cost step: a moving average square filtering following by a square min-filtering are used to simulate this filtering. In this case the size of ZSSD window is used to define the size of the shiftable-windows.
  The disparity estimation step is a WTA (winner takes all) algorithm.
- Dynamic programming: the user has to define the smoothness weight λ and the occlusion cost. Default values are given for the gradient-dependent cost. The window size of the default matching cost ZSAD has to be set. There's no filtering of the matching cost.
- Scanline optimization and graph cut: the user has to define the smoothness weight λ and the window size of the default matching cost ZSAD. Default values are given for the gradient-dependent cost. The matching cost filtering step is skipped.

- Bayesian Diffusion: The user has to define the number of iterations of the cost map filtering phase and the window size of the default matching cost ZSAD. The default disparity map estimator is WTA.

The default dense matching corresponds to the ZSSD local method with a default window size. The library has however to let the user refine his matching specifications at each step of the process. This must be done without loosing the benefit of predefined processing.

3 Domain Model

The domain model illustrates the meaningful conceptual classes associated to the scenarios of some use cases.

3.1 Domain model of the image filtering

This section vizualizes the conceptual classes related to the main scenario of the use case titled "filter an images".

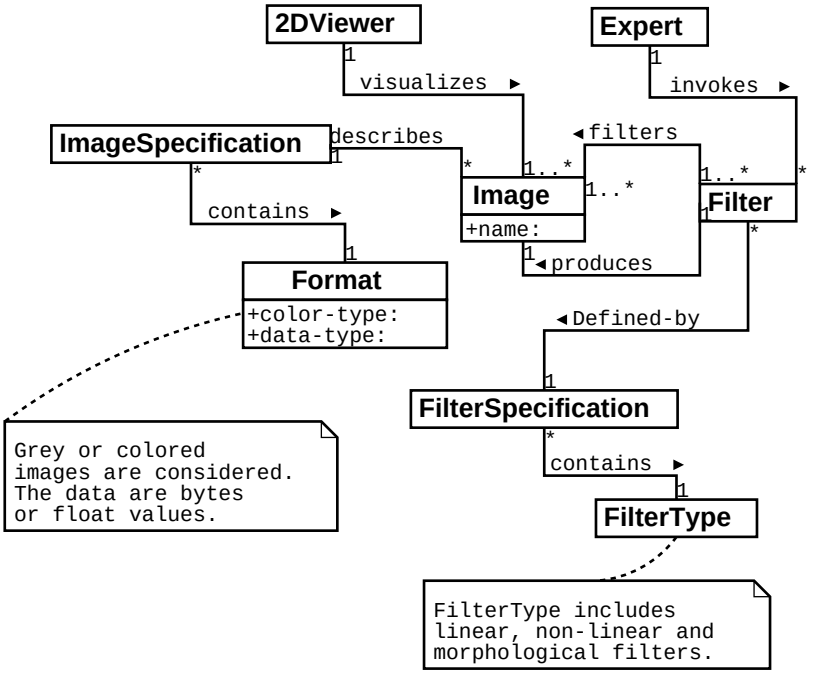


Figure 8: Domain model corresponding to the image filtering main scenario.

3.2 Domain model of the stereo pair handling

This section vizualizes the conceptual classes related to the main scenario of the use case titled "handle a stereo pair".
The Expert invokes the whished operation which acts on a given StereoPair instance.

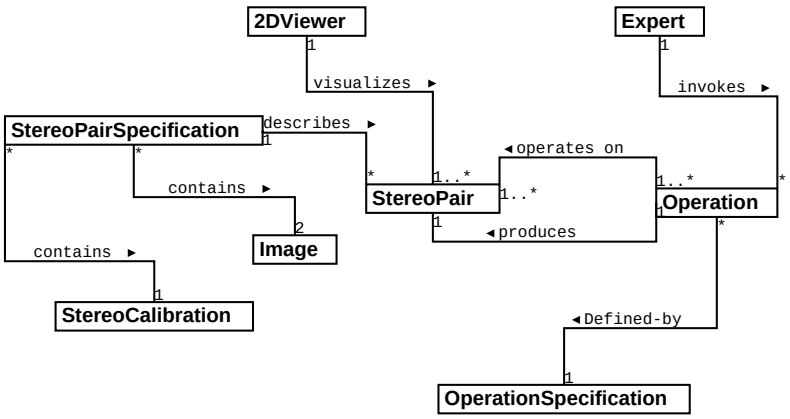


Figure 9: Domain model corresponding to the stereo pair handling main scenario.

3.3 Domain model of the dense stereo matching

This section depicts the concepts associated to the alternatice flows adressed in the use case titled "Dense stereo matching".

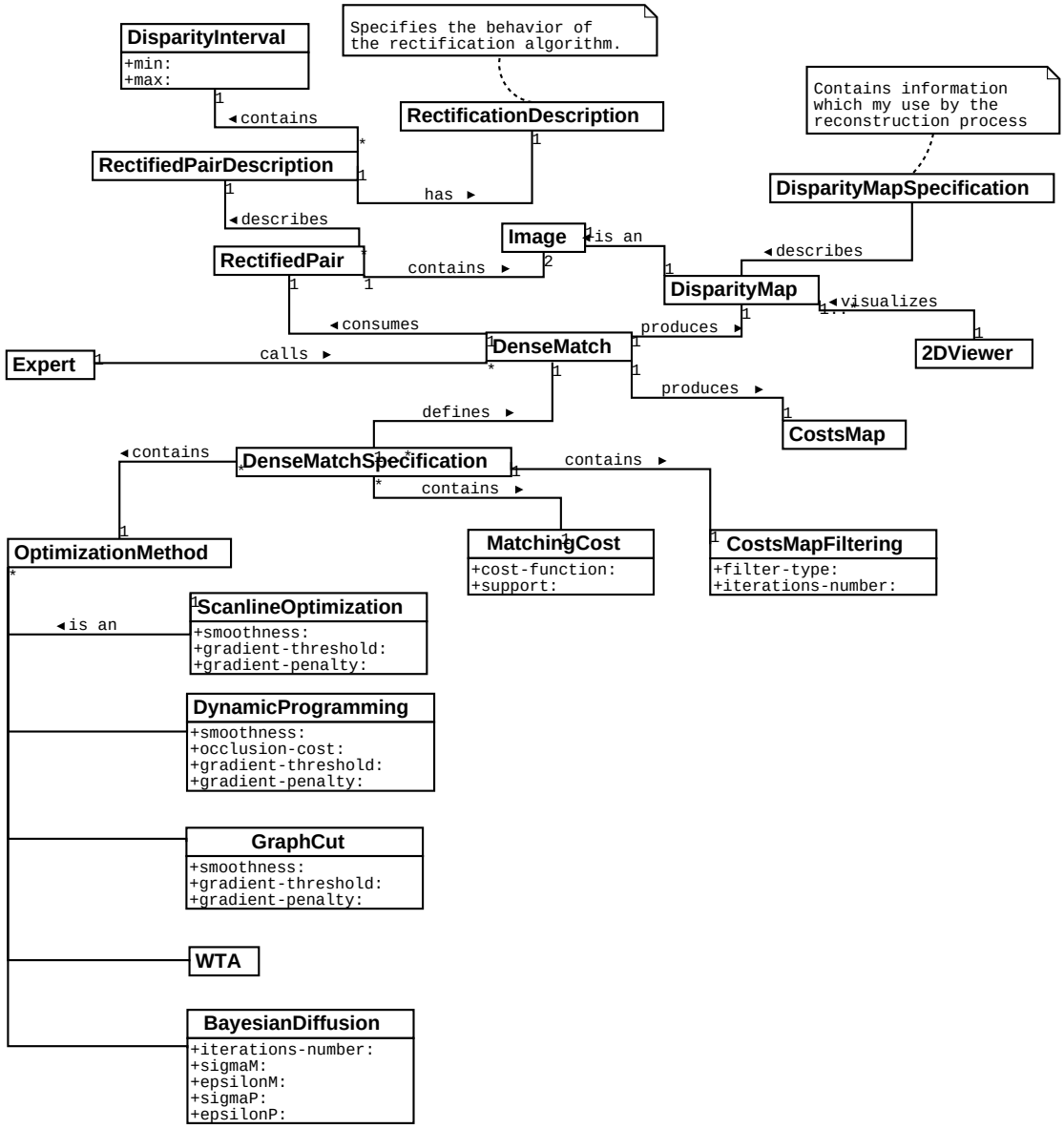


Figure 10: Domain model corresponding to the alternative scenarios of the dense stereo matching.

4 Design Model

This section designs use-case realizations. The result is the specification for software classes and their collaborations. This is illustrated both by design class diagrams and interaction diagrams. The design mainly consists in assigning responsabilities to software classes. These responsabilities may be deduced from use cases or collaboration diagrams. They may be as well be identified and assigned to classes by applying some of the patterns defined in [2] and [4]. The basic patterns *Expert*, *Creator*, *Low Coupling* and *High Cohesion* are continuously used in the whole design. Thereby their use will generally not be quoted anymore. Conversely, the use of more specific others patterns will be referred, in order to understand the motivations of the design choices.

4.1 Design model of the image model filtering

The filtering needs has to be strictly related to the dense matching. However, low level and powerful subsystems associated to image processing may provide some of the required algorithms. This is the case of the Python Imaging Library. So, to fullfill the use-case scenarios, the design model has to:

- define a set of filtering algorithms and make them interchangeable.
- convert the interface of existing filtering code into our interface client expect.

The *Strategy* pattern provides a solution to design the set of filtering since the behavior of filtering depends essentially on the algorithm. This is depicted in the following design class diagram (DCD).

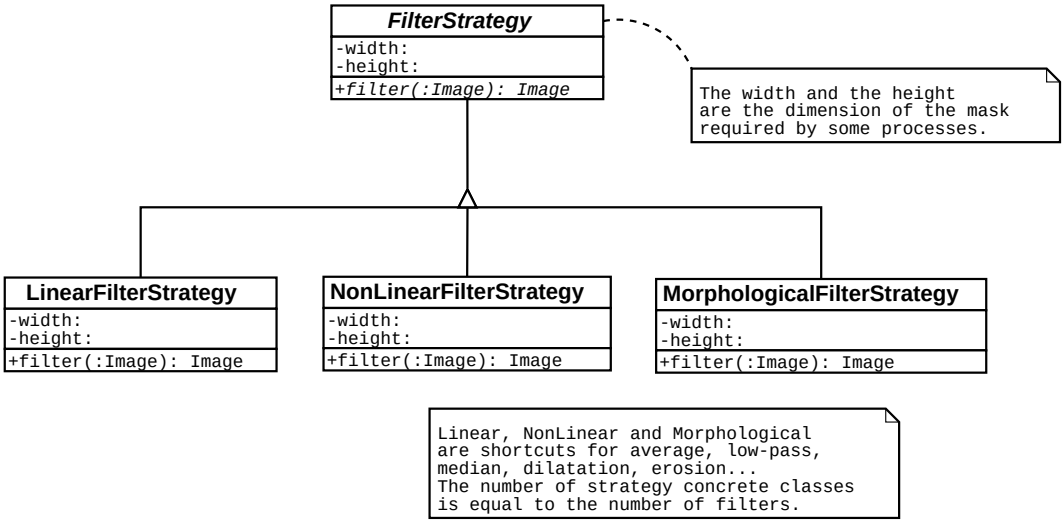


Figure 11: Design class diagram deduced from the *Strategy* pattern.

A filtering strategy is attached to a context object, an instance of the Image class, to which it applies the filtering. The next sequence diagram shows the collaboration between to the Image object and a strategy.

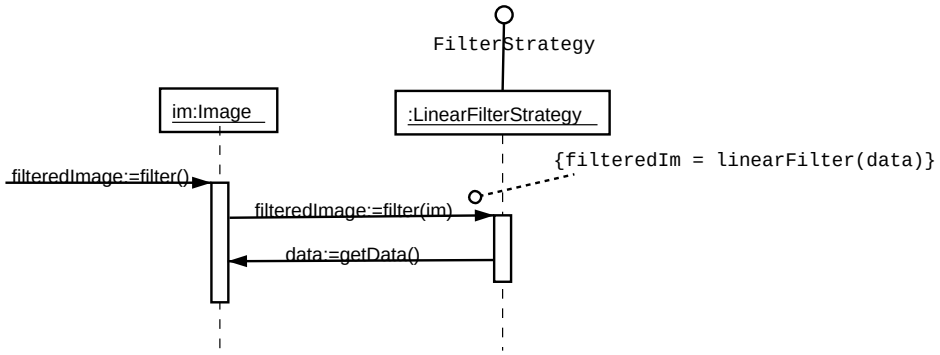


Figure 12: StrategyFilter in collaboration.

The *Adapter* pattern gives a way to adapt the PIL interfaces to our requirements. The object Adapter is chosen versus the class Adapter: the class Adapter adapts the Image class but none of its subclasses which precisely implement the wished filtering operations. The subsequent DCD depicts the software classes resulting from the *Adapter* pattern.

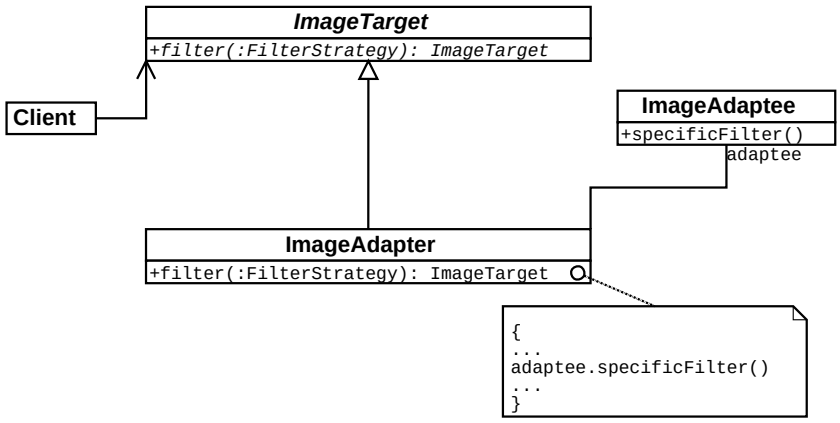


Figure 13: Design class diagram deduced from the *Adapter* pattern.

The ImageAdapter class adapts the interface of ImageAdaptee, a PIL class, to the ImageTarget interface. A Client, a primary actor for instance, call filtering operations on an ImageAdapter instance. The ImageAdapter calls in turn ImageAdapee filtering operations that carry out the request.

4.2 Design model of the image handling

The design of the handling differs slightly from the design of filtering. The operation are not related beetween themselves and the *Strategy* pattern may be not not convenient. Nevertheless, the *Adapter* pattern allows again to exploit the features of an existing imaging library.

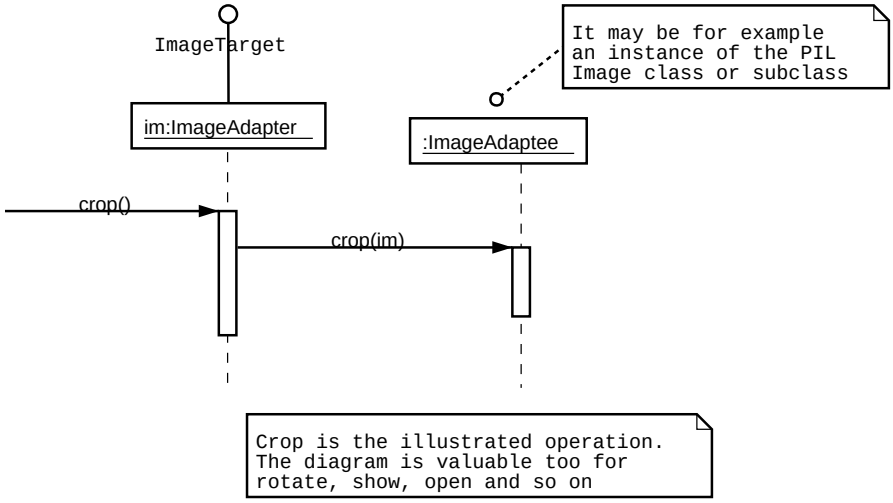


Figure 14: Sequence diagram for the image handling use case.

4.3 Design model of the stereo pair handling

The design of the "stereo pair handling" attempts to retrieve some the "image handling" abilities by exploiting the obvious aggregation structure of a StereoPair object. Indeed, a stereo pair contains two images and the operations associated to a stereo pair are chosen among those applied to a single image.
The *Composite* pattern gives a flexible design solution by creating an abstract class Stereo-Component that represents both the stereo pair and a single image.

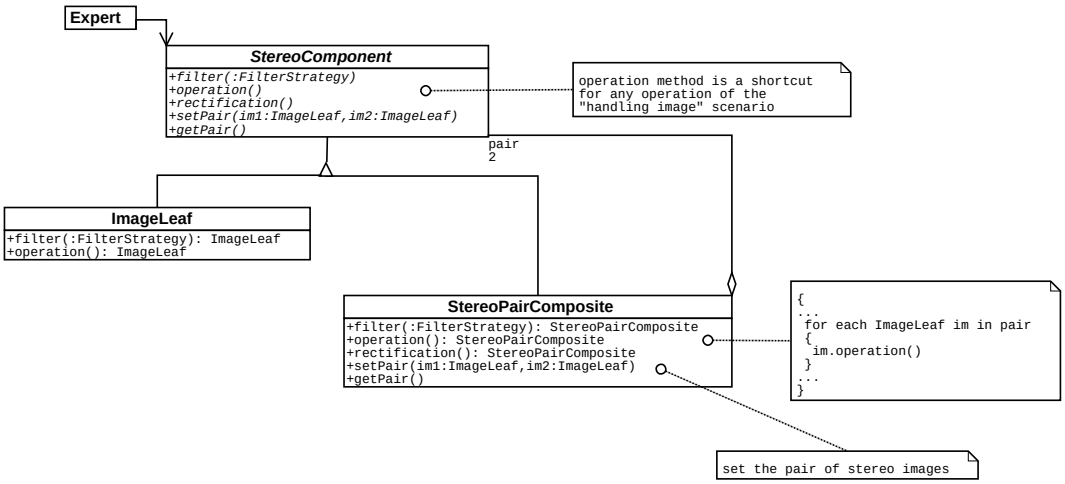


Figure 15: Design class diagram for the stereo pair handling use case. The suffixes Component, Leaf and Composite aim at precising the participants of the pattern.

The Expert uses the StereoComponent class to interact with objects in the composite structure. If he interacts with an ImageLeaf object, the request is handled directly. Otherwise, if the recipient is an instance of the StereoPairComposite, the request is forwarded to the child component, i.e. the pair of images.

4.4 Design model of the dense match

Many objects are involved in the dense match process as the stereo pair of images, the cost map or the disparity map. Furthermore, many distinct and unrelated operations, as the filtering or the energy minimization, need to be performed on these objects. The *Visitor* pattern gives a design solution, especially if we have to develop a such process "from scratch". In practise, many part of this object structure still exists, issued from various libraries and the *Adapter* pattern is a way to adapt their interface to our need. The section "filtering an image" illustrates the application of a such pattern. A drawback comes from the limitation of the atomization of the operations and the objects of the existing tools: the adapted structures may not be elementary enough to be included in our *Visitor* pattern components. Also, the *Visitor* pattern seems not a solution to our problem, unless we have to develop the whole components of a dense match process.
Now, the dense match processes differ only in their behavior: all these processes aim at producing a disparity map bu using distinct algorithms. It appears as well that different variants of a same algortihm may be involved: the expert, for instance, wants only to change the matching cost. At last, the use case scenario do not necesseraly exposes complex and

algorithm-specific data structures. Indeed, according to their level of use and understanding of the matching process, the clients should not know data used in some operations. The *Strategy* pattern give a design solution to these three problems.

If the *Strategy* pattern provides a solution to invoke several algorithms in a uniform way, it leads as well to a great number of strategies. Indeed, each way used to specify a dense match may correspond to a particular *Strategy*! This number may be reduced by exploiting the part-whole hierachies of objects (see the domain model figure 7) through the *Composite* pattern.

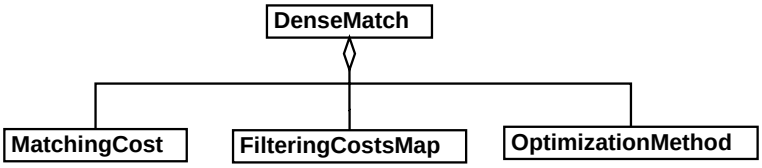


Figure 16: Illustration of the composite structure of the dense match specification.

The *Composite* pattern allows the client to treat composite and primitive structures in a uniform way. The client may invoke too child-related operation implemented by a composite. The ability to use the dense match object at different understanding level is also obtained, without to have to cop with numerous strategies.
The use of the *Strategy* and the *Composite* patterns leads to the following design class diagram.

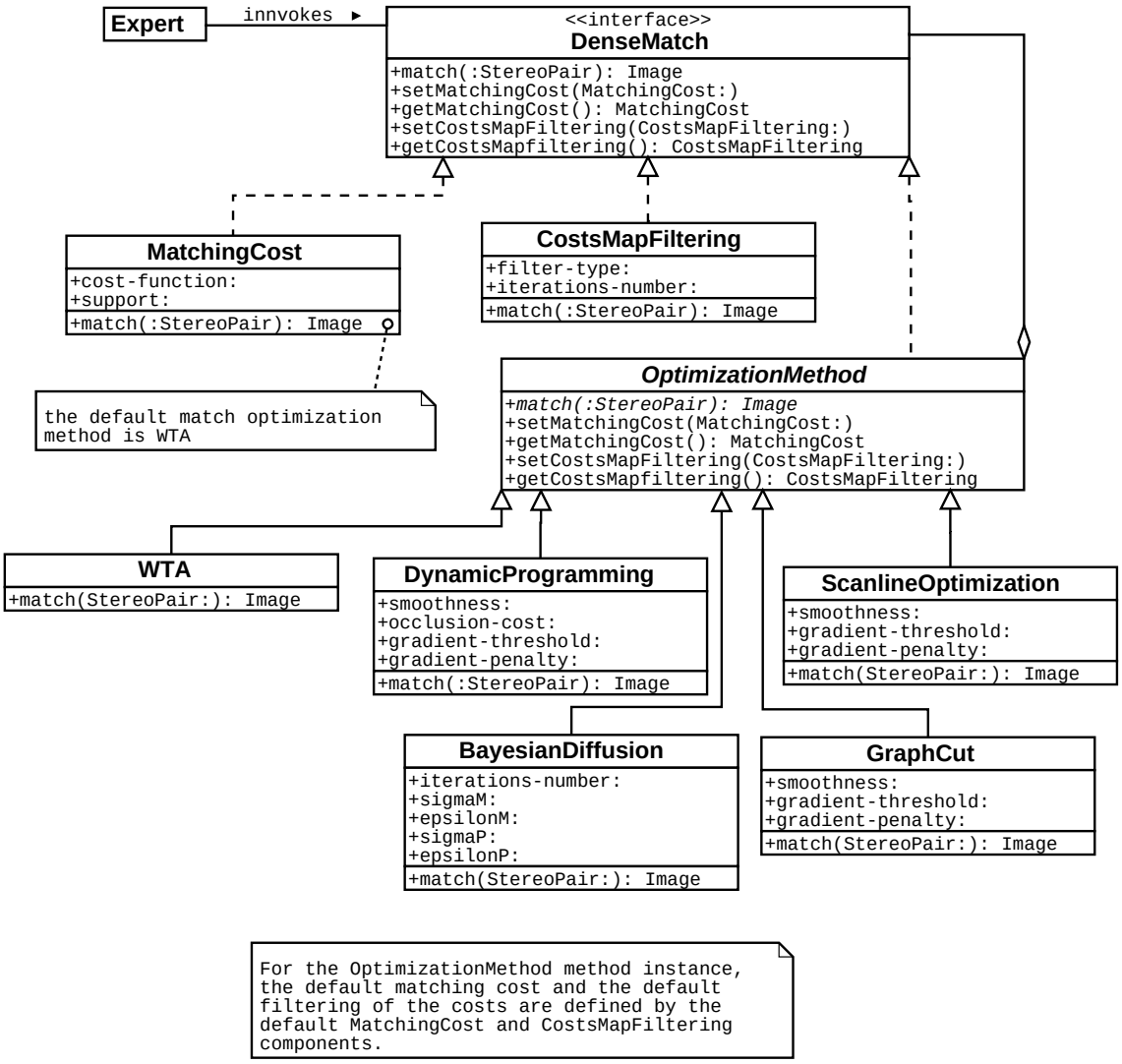


Figure 17: Design class of the dense match which combines both the *Strategy* and the *Composite* patterns.

CONCLUSION

This document and the software implementation of the design model of the library result from several elaboration iterations. Nevertheless the whole development cycle is not finished. To refer to the scheduling of the XP process, we may consider the inception and the elaboration phase has been achieved. Indeed, the high risk issues are now mitigated and the sofware development can get in its construction phase. This next phase deals with the iterative implementation of the remaining lower risks and the preparation of the software deployement.

References

[1] A. Bobick and S. Intille. Large occlusion stereo. *IJCV*, 33(3):180–200, 1999.

[2] R. Johnson E. Gamma, R. Helm and J. Vlissides. *Design Patterns.* Addison Wesley, 1995.

[3] R.I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2001.

[4] C. Larman. *Applying UML and Patterns.* Prentice Hall, second edition, 2002.

[5] M. Personnaz and R. Horaud. Camera calibration: estimation, validation and software. Technical Report RT-258, INRIA, 2002.

[6] H. H. Baker R. C. Bolles and D. H. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *IJCV*, 1:7–55, 1987.

[7] D. Scharstein and R. Szeliski. Stereo matching with non-linear diffusion. *IJCV*, 28(2):155–174, 1998.

[8] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, 2001.

[9] O. Veksler. Efficient graph-based energy minimization methods in computer vision. PhD thesis, Cornell University, 1999.

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