Project 2007

October 1, 2019

Contents

Part I

Generalities

Chapter 1

My first command : set editing

1.1 Introduction

This chapter presents the first commands of MMVII. It uses a plan that will be almost systematic in many other chapter :

- a section relative to algorithmic and photogrammetric aspect of the chapter, generally this section may exist [1] almost totally independantly of MMVII, but it is pre-requisite as there is obviously no interest to know the command and the code if the fundamentalls are not understood;
- a user's guide section, relative to MMVII at user level, including the syntax of the command;
- one or more programmers section, relative to `C++` code implemanting the command, it will be a presentation of general organisation [2], as the detail are to be found in *doxygen*pages;

This chapter will be a bit specific as the part or user's guide and programming will be much more important than other for a single command, as many concept common to all command will be explained here, conversely the algorithmic part will be very short.

1.2 Algorithms/Photogrammetry

This command is useful for editing a set of files. Almost all commands of MMVIIrequire as parameter one or more set of file (i.e. the subset of images that we are considering for a given computation). For single case, this set of file can be simply specified by a regular expression : for example `".*JPG"` to specify all the file with a `JPG` extension.

However for more complex case we may want to :

- create a set from a single pattern;
- add or substract an interval, a pattern ...
- memorize the result and reuse it.

This is what does the `EditSet` command, piece by piece create a `XML` file that memorize a "complex" set of file that can be used instead of a pattern.

1.3 User's side(1)

1.3.1 Basic notion

MMVII is a command line programm. There is unique programm which name is `MMVII`. Any command, `OneCmd`, of MMVII will be called with the syntax `MMVII OneCmd Args` where `Args` are the arguments of the command. To know what are the existing command there is two way :

[1] i.e. may be of interest for the reader, hopefully

[2] as link between concept and classes

- a basic one just enter `MMVII`;
- a more sophisticated one , to be written, `MMVII help` described in 3.1;

For the basic one we get:

```
MMVII
...
Bench => This command execute (many) self verification on MicMac-V2 behaviour
Cpp11 => This command execute some test for to check my understanding of C++11
TBS => This command execute some experiments en boost serrialization
MPDTest => This used a an entry point to all quick and dirty test by MPD ...
EditSet => This command is used to edit set of file
EditRel => This command is used to edit set of pairs of files
...
```

We get the list of all command and short commentary on the service given by the command.

1.3.2 Getting help

Very currently, user will know what the command does, but will not remember the exact syntax. The `help` key word can be used at any position for requiring this information, for example :

```
MMVII EditSet help

*********************************
*   Help project 2007/MMVII      *
*********************************

  For command : EditSet
   => This command is used to edit set of file
   => Srce code entry in :../../MMVII/src/Appli/cMMVII_CalcSet.cpp

 == Mandatory unnamed args : ==
  * string [FDP] :: Full Name of Xml in/out
  * string :: Operator in (+= *= -= = =0)
  * string [MPI0] :: Pattern or Xml for modifying

 == Optional named args : ==
  * [Name=Show] int :: Show detail of set before/after , (def) 0->none, (1) modif, (2) all
  * [Name=Out] string :: Destination, def=Input, no save for NONE
```

We get three part :

- first part give the short comment, and the name of the `C++` file where the entry point of the command is implemented (may be of interest to programmers);
- second part contains the description of mandatory args, we see that here we have three mandatory args; for each args is indicated the type (here all strings), and after `::`, the semantic of the parameter; sometime it is inserted inside square bracket (like `[FDP]`) some "predefined semantics" that will be described later (1.4.5);
- third part contains the description of optional args, as for mandatory args, the type and a short command is given, before this is added the name the optional parameter in the form `[Name=TheName]`;

As said before, `help` can appear at any position after `OneCmd`, this can be usefull when one has begin to edit a command, and dont want to loose it, for example with parameter of next section, the following line is perfectly valide to obtain help about `EditSet` :

```
MMVII EditSet File.xml = "F[0-3].txt" help
```

1.3.3 basic usage

For example, if we go in the folder `MMVII-MainFolder/MMVII-TestDir/Input/Files`, we can test :

```
MMVII EditSet File.xml = "F[0-3].txt"
```

Here we have used only the mandatory paramaters. As there is no naming for these parameters, the order is used to make the correspondance between parameters and value, so here :

- `File.xml` correspond to first parameter described as `"Full Name of Xml in/out"`;
- `=` correspond to second paramater, described as `Operator ...`;
- `"F[0-3].txt"` correspond to third paramater, described as `Pattern ...`;

Some comment on the effect of this parameter :

- `File.xml` is the name of the `XML` file that contain the initial list of name, it's pefectly acceptable that this file does not exist, in this case an empty list is created;
- `=` correspond to second paramater, is describe the operator that will be used to modify the file with the value $S3$ of third parameter, its value must belong to an enumarated list with the following meaning

  $=$, $S3$ ovewrite `File.xml` ;

  $+=$, $S3$ is added to `File.xml` ;

  `-=` , $S3$ is subsbracted from `File.xml`

  `*=` , `File.xml` is the intersection of $S3$ and its previous value;

  $=\mathbf{0}$, `File.xml` is empty, whatever may be in $S3$;

- `"F[0-3].txt"` correspond to third paramater, described as `Pattern ...`;

We can now inspect the file `File.xml` which contains the name of the files *present in the folder* and matching the regular expression `"F[0-3].txt"`:

```
cat File.xml
<?xml version="1.0" encoding="ISO8859-1" standalone="yes" ?>
<MMVII_Serialization>
   <SetOfName>
      <Nb>4</Nb>
      <el>F0.txt</el>
      <el>F1.txt</el>
      <el>F2.txt</el>
      <el>F3.txt</el>
   </SetOfName>
</MMVII_Serialization>
```

As always when a regular expression is used to specify set of file, it is understood as a filter on existing file. So if one had used `"F([0-3]|[a-z]).txt"`, given the file present in `MMVII-MainFolder/MMVII-TestDir/Input/Fi` we woul have obtained exactly the same result.

1.3.3.1 Exercices

Try the following command and inspect the result , after each :

```
MMVII EditSet File.xml  = "F[0-3].txt"
MMVII EditSet File.xml += "F[7-9].txt"
MMVII EditSet File.xml -= "F8.txt"
MMVII EditSet File.xml *= "F[02468].txt"
MMVII EditSet File.xml =0 ".*"
```

1.3.4 Optional paramaters

1.3.4.1 `Out` paramater

Optional parameter are given after the mandary one in a list of string `Name=Value`. For example until now we have use the file `File.xml` both as input and output, but sometime we don't want to modify the input file, we can the use the optionnal `Out` parameter. For example if we enter :

```
MMVII EditSet File.xml = "F[0-3].txt"
MMVII EditSet File.xml += "F[7-9].txt"  Out=File2.xml
```

After first line `File.xml` contains t4 names. After second line, the `File.xml` is unchanged while `File2.xml` contains 7 names.

An interesting option, for this commans as each time a pattern is expected, is that if the file is `XML` file, created by MMVII and with main tag `<SetOfName>`, then name used will not be the pattern itself but the name contained in the file, for example :

```
MMVII EditSet File1.xml = "F[0-3].txt"
MMVII EditSet File2.xml = "F[7-9].txt"
MMVII EditSet File1.xml += File2.xml  Out=File3.xml
MMVII EditSet File3.xml += File2.xml  Out=File4.xml
```

After this command `File3.xml` contains the sum of `File1.xml` and `File2.xml`, here 7 name. All the operation are set operation, in the mathematicall sense, so there is no duplicate, dans `File4` contain still 7 names.

1.3.4.2 `Show` paramater

The `Show` allow to visualize the result of the operation.

```
MMVII EditSet File.xml =0 ".*"
MMVII EditSet File.xml = "F[0-4].txt"
MMVII EditSet File.xml += "F[0-6].txt" Show=1
-+ F5.txt
-+ F6.txt
MMVII EditSet File.xml *= "F[02468].txt" Show=2
 ++ F0.txt
 ++ F2.txt
 ++ F4.txt
 ++ F6.txt
 +- F1.txt
 +- F3.txt
 +- F5.txt
```

The third command use the parameter `Show`, as the value is 1, only the modification are shown :

- `-+ F5.txt` means that the file was inially absent ($-$) and is present at end ($+$)

It is also possible to show all the result, including the names that present before and after the modification :

- `++ F0.txt` : means that the file is present before and after the operation;
- `+- F1.txt` : means that the file is present before and absent after the operation;

1.3.5 More help

In MMVIIthere exists many optional parameter. There are not shown by default in the help mode, but it is possible to show :

- the standard common parameter by setting `Help` instead of `help`
- all the common parameter, including the *internal* common parameter by setting `HELP` instead of `help`; the internal parameter are used by MMVIIto communicate information to sub-process when MMVIIcalls itself; for example the parameter `LevCall` allow MMVIIto know if it was called by the user or by MMVIIand to which level of imbrication; obviously it is generally a bad idea to fix yourself the internall parameter;

Here is an example with `EditSet` :

```
MMVII EditSet File.xml *= "F[02468].txt" Show=2 HELP
...
  * [Name=Out] string :: Destination, def=Input, no save for NONE
  * [Name=FFI0] string [FFI0] :: File Filter Interval, Main Set   ### COMMON
  * [Name=FFI1] string [FFI1] :: File Filter Interval, Second Set   ### COMMON
  * [Name=NumVOut] int :: Num version for output format (1 or 2)   ### COMMON
  * [Name=DirProj] string [DP] :: Project Directory   ### COMMON
  * [Name=StdOut] string :: Redirection of Ouput (+File for add,NONEfor no out)   ### COMMON
...
  * [Name=LevCall] int :: Internal : Don't Use !!   ### INTERNAL
  * [Name=ShowAll] bool :: Internal : Don't Use !!   ### INTERNAL
...
```

As some command have many option, it possible to filter the optionnal parameter using a regular expression , with a syntax `help=expr` (or `Help` or `HELP`), for example :

```
MMVII EditSet File.xml *= "F[02468].txt" Show=2 HELP=F.*
...
 == Optional named args : ==
  * [Name=FFI0] string :: File Filter Interval, Main Set   ### COMMON
  * [Name=FFI1] string :: File Filter Interval, Second Set   ### COMMON
```

1.4 User's side-2, global parameter

1.4.1 Fixing project directory `DirProj`

In MMVII the notion of project is closely related to the folder where are stored a given set of data, basically one can consider for universall the rule "one project/one folder". MMVII uses the following rule to determine the directory of project :

- many command have a parameter that fix the project folder, for example with `EditSet` the first parameter fix the project directory, they are indicated by `[FDP]` (see 1.4.5);
- when there is no command to fix the folder, by default MMVIIfix the project folder to `"./"`.;
- it is also possible to fix this directory with the optionnal parameter `DirProj`.

For example, if we go in the folder `MMVII-MainFolder/MMVII-TestDir/Input/`, we can test :

```
MMVII EditSet Files/FileX.xml = "F[0].txt"
MMVII EditSet Files/FileX.xml = "F[0].txt"  Show=2
 ++ F0.txt
MMVII EditSet FileX.xml = "F[0].txt"  Show=2  DirProj=Files/
 ++ F0.txt
```

In the first two command, the project folder is computed from `Files/FileX.xml`. In the last command, it is computed from `DirProj=Files/`.

1.4.2 Filtering by interval FFI0, FFI1

Intervall can be used for different ordered type, for string the order is the standard lexicographic order. Interval are describe on command line usign square barcket, "[" and "]", separated by a comma ",". Rather than a formal definition, explain by example :

- one can use closed interval : [a100.jpg,a150.jpg] filter the string S such that $a100.jpg \leq S$ and $S \leq a100.jpg$
- one can use open interval :]a100.jpg,a150.jpg[filter the string S such that $a100.jpg < S$ and $S < a100.jpg$
- interval can be semi open as [a100.jpg,a150.jpg[with obvious interpretation;
- interval can be semi finite : [a100.jpg,[filter the string $a100.jpg \leq S$, and no upper bound;
- finally one can create union of intervall by simply concatening the string:],a110jpg[[a140.jpg,[filter the string such that $S < a110.jpg$ *or* $a140.jpg \leq S$

The common optional parameter FFI0 (and FFI1) can be used to do this filtering, for example , if we go in the folder MMVII-MainFolder/MMVII-TestDir/Input/Files, we can test :

```
MMVII EditSet File.xml = "F.*txt" FFI0="[F1.txt,F3.txt[]F7.txt,["
...
      <el>F1.txt</el>
      <el>F2.txt</el>
      <el>F8.txt</el>
      <el>F9.txt</el>
```

In this example, the parameter FFI0 has been used to filter "F.*txt", and gives the result described with the 4 names. Of course, the question is "How the user can knows that the filter FFI0 will apply to this parameter ?". This here where comes the "predefined semantics" [MPI0] that is shown in the help (see 1.4.5).

1.4.3 Redirecting message with StdOut

By defaut MMVII print several messages on the console. When user want to print the messages in a file File.txt, it is possible to :

- just append the messages at the end to the possibily existing file by StdOut=File.txt;
- just append the messages to the possibily existing file and still print the messages on the console by StdOut=+File.txt;
- print the messages in this file, and reset if it exist, StdOut=0File.txt;
- print the messages in this file, and reset if it exist, and still print the messages on the console by StdOut=0+File.txt;
- print nothing by StdOut=NONE.

This has for consequences that the name of the file of redirection cannot begin by + or 0.

1.4.3.1 Exercices

Try the following command and inspect the result , after each :

```
# File Mes.txt grows
MMVII EditSet File.xml  = "F[0-3].txt" StdOut=Mes.txt Show=2
MMVII EditSet File.xml  = "F[0-3].txt" StdOut=+Mes.txt Show=2
MMVII EditSet File.xml  = "F[0-3].txt" StdOut=Mes.txt Show=2
```

```
# File Mes.txt reiniliazed
MMVII EditSet File.xml  = "F[0-3].txt" StdOut=0Mes.txt Show=2
MMVII EditSet File.xml  = "F[0-3].txt" StdOut=0+Mes.txt Show=2
MMVII EditSet File.xml  = "F[0-3].txt" StdOut=+0Mes.txt Show=2
# No output
MMVII EditSet File.xml  = "F[0-3].txt" StdOut=NONE Show=2
```

1.4.4 Fixing MicMac version for export `NumVOut`

As versions 1 and 2 of MicMac will coexist for several (many ?) years, it is usefull that new tools are able to import/export. For import, the solution is easy, MMVII, recognize by analyzing the first tag which version is it (if any). For export the rule are more complicated but quite logical, they use the common optionnal parameter `NumVOut` :

- if `NumVOut` is set (to 1 or 2) this fix the num version for export;
- else if there at least one file of $V2$ was imported, the export will be in $V2$;
- else if there at least one file of $V1$ was imported, the export will be in $V1$;
- else the export will be in $V2$;

1.4.5 Predefined semantics

1.4.5.1 Generalities

Many parameters of many command of MMVIIcorrespond to the same meaning/semantic, this is the case for "main set of images", "main orientation", ... These predefined semantic are indicated in square bracket after the types, for example with command `EditSet` we can see `[FDP], [MPI0], [FFI0],[FFI1] [DP]` :

```
MMVII EditSet HELP
...
  * string [FDP] :: Full Name of Xml in/out
  * string [MPI0] :: Pattern or Xml for modifying
...
  * [Name=FFI0] string [FFI0] :: File Filter Interval, Main Set   ### COMMON
  * [Name=FFI1] string [FFI1] :: File Filter Interval, Second Set   ### COMMON
  * [Name=DirProj] string [DP] :: Project Directory   ### COMMON
..
```

We desribe after this semantic (but not for common parameter, as they have already been described).

1.4.5.2 Main pattern image `[MPI]`

Many command have a parameter which is the main set/pattern of files (generally images). This parameter is described by the predefined semantic `[MPI0]`. If this parameter exists, then it is possible to use `[FFI0]` to filter the set (if there is no `[MPI0]` then use of `[FFI0]` is forbidden).

Some command have several main sets, in this case one of their parameter will have the predefined semantic `[MPI1]`, which can be filtered by `[FFI1]`. See the command `EditRel`.

1.4.5.3 File of Directory Project `[FDP]`

The notion of project directory was introduced in 1.4.1. Generally there is no need to specify it, as there is one "main" file parameter that fix this directory. This parameter can be recognized by the predefined semantic `[FDP]`, in `EditSet` command, this is first parameter that corresponds to this.

1.5 User's side-3, most frequent error

1.5.1 Generality

When a command fails, it generates an error message and generally wait for the user to press "return key". The first part of the message contains the type of error, it can be :

- `Level=[Internal Error]` : this mean that some incoherence in MMVIIwas encontered, probably in this case user cannot do many thing but report to forum or devlopping team mentionning the complete message;
- `Level=[UserEr:XXXX]` : this means that the error is probably due to a bad manipulation of the user, where `XXX` is the reference of the error;

For example :

```
MMVII EditSet File.xml = "F.*txt" ShowAll=tru

Level=[UserEr:BadBool]
Mes=[Bad value for boolean :[tru]]
```

Let comment the message :

- `Mes=[Bad value for boolean :[tru]]` : this as short message, which will be generally sufficient to analyse the error, here the error occured because `ShowAll` is of type boolean and `tru` is not a valide string to create a boolean;
- `Level=[UserEr:BadBool]` : this line indicate the reference of the error, this reference can be used , if the short message is unsufficient, as an entry in this documentation to get more information on the error;

1.5.2 Error `BadBool`

This error occurs when a parameter of boolean type is initialized with an unvalid string. Valide string for boolean are : {`0,1,false,true`} (case unsensitive). Example, parameter `ShowAll` being boolean :

```
MMVII EditSet File.xml = "F.*txt" ShowAll=tru
```

1.5.3 Error `BadOptP`

This error occurs when an optionnal parameter name do not match any the expected paramater name. Example, typing `AllShow` instead of `ShowAll`.

```
MMVII EditSet File.xml = "F.*txt" AllShow=true
```

1.5.4 Error `MultOptP`

This error occurs when the same optional parameter was used several time. Example doubling the `NumVOut` :

```
MMVII EditSet File.xml = ".*txt" NumVOut=1 NumVOut=1
```

1.5.5 Error `OpenFile`

This error occurs when MMVIIcannot open a file, in read or write mode, several reason can exist : hard disk full, rights on the file system, directory do not exist. Example :

```
MMVII EditSet File.xml = ".*txt"  Out=o/o.xml
```

1.5.6 Error InsufP

This error occurs when the number of parameter is inferior to the number of mandatory parameters. Example, omiting the operator in EditSet:

```
MMVII EditSet File.xml "F.*txt"
```

1.5.7 Error BadEnum

This error occurs when a string cannot create a specific enum. Example, typing eq instead of = in EditSet.

```
MMVII EditSet File.xm eq "F.*txt"
```

1.5.8 Error FileSetN

This error occurs when a File a file was expected to be a set of name and : the file exist (else it would be just an empty set) but is not a correct xml file in V1 or V2 format. Exemple under MMVII-MainFolder/MMVII-TestDir/Inpu using the file BadFile.xml :

```
MMVII EditSet BadFile.xml = .*txt
```

1.5.9 Error IntWithoutS

This error occurs when a file filter image (FFI0,FFI1) were used but the corresponding main pattern is not member of the command, for example :

```
MMVII EditSet BadFile.xml = .*txt FFI1=[,]
```

1.6 Programmer's side, adding a new command (1)

In this chapter we will see the main roadmap to follow for adding a new command in MMVII.

1.6.1 Heriting from cMMVII_Appli

A first and easy principle is "One command/One class" and "this class must inherit from cMMVII_Appli. In our case the class corresponding to EditSet is the class cAppli_EditSet defined in file :

- MMVII-MainFolder//src/cMMVII_CalcSet.cpp

As cMMVII_Appli is pure virtual class, the concrete class must override 3 methods :

- int Exe(); : this method execute the action of the command;
- cCollecSpecArg2007 & ArgObl(cCollecSpecArg2007 &) this method communicate the specification of mandatory parameters;
- cCollecSpecArg2007 & ArgOpt(cCollecSpecArg2007 &) this method communicate the specification of optional parameters;

1.6.2 Link between name an class

The first thing MMVIIhas to do is to create the object heriting from cMMVII_Appli from the command name (here create a cAppli_EditSet from EditSet). To avoid huge compilation this creation is done without declaration of all the class in header; the philosophy is to have "hidden" derived class definition in ".cpp" files and just export an allocator. More precisely this is done via the class cSpecMMVII_Appli which is the specification of an application, it contains :

- an allocator function able to create a `cMMVII_Appli` from command line (see type `tMMVII_AppliAllocator`), here this is the function `Alloc_EditSet`;
- the name of the command, here `EditSet`;
- the comment of the command;
- three vector of specification : features (what main group the command belongs to), type of input, type of output; these specification are used in the `help` command to look for a given command (satisfying a requets of the user such that "which command deal with orientation and produce ply data?");
- the file where the spec is created (using macro `__FILE__`) to help recovering file from command name.

Once the spec is created, it must be added to vector containing all the specification, this is done simply by adding a line as

- `TheRes.push_back(&TheSpecEditSet);`

in file `src/Appli/cSpecMMVII_Appli.cpp`.

1.6.3 Specifying paramaters

The specification of parameters is done by the methods `ArgObl` and `ArgOpt`. They both return a `cCollecSpecArg2007` with are an agregation of `cSpecOneArg2007`. A `cSpecOneArg2007` contains the specification of one parameters, it is a virtual class that contains :

- the variable that will be initialzed, this variable which can be of different type as it is contained in the derived classes;
- a vector of predefined semantics, a predefined semantic is create from one enum `eTA2007` and an optional additional string;
- the comment associated to the parameter;
- the name of the parameter (aways empty string `""` for mandatory parameters);

Let's make a brief comment with class `EditSet`, first mandatory parameters :

```
cCollecSpecArg2007 & cAppli_EditSet::ArgObl(cCollecSpecArg2007 & anArgObl)
{
   return
      anArgObl
         << Arg2007(mXmlIn,"Full Name of Xml in/out",{eTA2007::FileDirProj})
         << Arg2007(mOp,"Operator in ("+StrAllVall<eOpAff>()+")" )
         << Arg2007(mPat,"Pattern or Xml for modifying",{{eTA2007::MPatIm,"0"}});
}
```

Let's comment :

- The function `Arg2007` is template and adapt to the type of the l-value , here all the parameter are string , but could be `int` ...or any type pre-instatiated in `cReadOneArgCL.cpp` using macrp `MACRO_INSTANTIATE_ARG2007`;
- the `StrAllVall<eOpAff>()` function is used to generate the string of all valid operators;
- the first parameter will fix the project directory, we indicate this by a having the semantic {`eTA2007::FileDirProj`}
- the third parameter will be the first main set of file, with indicate it by the semantic `eTA2007::MPatIm,"0"`;

```
cCollecSpecArg2007 & cAppli_EditSet::ArgOpt(cCollecSpecArg2007 & anArgOpt)
{
  return
    anArgOpt
       << AOpt2007(mShow,"Show","Show detail of set before/after , (def) 0->none, (1) modif, (2) all",{})
       << AOpt2007(mXmlOut,"Out","Destination, def=Input, no save for " + MMVII_NONE,{});
}
```

Here we use the template function `AOpt2007`, it is used with a `bool` (parameter `mShow`) and a `std::string` (parameter `mXmlOut`). We see also that there is one more parameter : the name of the parameter that MicMac user will see (here `Show` or `Out`).

1.6.4 Standard access paramaters

Many command have paramaters that have more or less the same meaning. It is possible to avoid code redundancy via standard acces function and/or use of predefined semantic.

1.6.4.1 Reading set of name from file with `SetNameFromString`

1.6.4.2 Main sets with `MainSetk`

Chapter 2

Programming organisation, style ...

2.1 Naming convention

2.2 Never use `std::cout, printf ...`

2.3 Encapsulation of boost, stl ..

2.4 Error handling

2.5 Memory check

2.6 Serialization

2.7 Shared pointer

2.8 Random number

MMVIIuse some pseudo random generator. As every such generator they must be initialized with a seed. By default , the seed is always the same to facilitate debuging. When user wants initialization from time this must be specified with a negative value of parameter `SeedRand`.

2.9 Enum to string

The enum/string conversion is a recurent problem of C++ which as far as I know is still an issue. A possible solution would be to use some code genration which from easy to read an write text file woul generate it. But I tried to limitate the code generation in MMVII.

In file `Serial/uti_e2string.cpp` is implemanted the used solution , it consist to create data for each enum for which we want to do the conversion `Serial/uti_e2string.cpp`.

Chapter 3

Project management command

3.1 Help command

Part II

Methodologies

Chapter 4

The "Aime" methods for Tie Points computation

4.1 Fast recognition

4.1.1 Motivation

For each image, we have computed tie points. A tie points is made of a vector $V \in \mathbb{R}^n$) . Typically V is invariant to the main geometric deformation . Le V_1 and V_2 be two tie points, we note :

- $H_{om}(V_1, V_2)$ the fact that two tie points are homologous;

Given V_1, V_2, there is of course no no oracle that can indicate if $H_{om}(V_1, V_2)$, and classically we want to compute a fast mathematical function Ψ that indicate if two vector V_1 and V_2 correspond to the same tie points . The ideal function would be such :

- $\Psi(V_1, V_2) \iff H_{om}(V_1, V_2)$

Of course this impossible, and we introduce the miss rate and fall out:

- miss rate , probability of $\Psi = 0$ knowing $H_{om} = 1$, we note p_m;
- fall out , probability of $\Psi = 1$ knowing $H_{om} = 0$, we note p_f;

As we cannot have the ideal function Ψ such as $p_m = 0$ and $p_f = 0$, we have to compromise, and depending on the circunstances, the price of the two error, will not be the same. Typically, in indexation step, we are especially interested to have a low p_f; converselly in recognition step we are especially interested to have a low p_m.

4.1.2 Bits vector

4.2 Gaussian pyramid

4.2.1 Computing σ_0

The gaussian pyramid is made from a succession of image at different scale that result from a gaussian filter. How can we justify it :

- image I_k must be at resolution $R_k = R_0 s^k$,
- if we assimmilat I_0 to a (sum of) gaussian of std dev σ_0 , I_k must be (sum of) gaussian $\sigma_0 * R_k$
- so we can write $I_k = I_0 * G(\sigma_k)$ with $\sigma_k^2 + \sigma_0^2 = (\sigma_0 R_k)^2$;

To compute the pyramid, we need an estimation of σ_0. Which is quite natural, if the initial image is very blured, it a high R_0, and the value $R_1 - R_0 = (s-1)R_0$ is also high, which require a high value for gaussian filter.

We need a way to estimate the initial value σ_0. Also it's quite arbirtrary, the way it is done in MMVII is :

- assimilate I_0 to a gaussian of std dev σ_0;
- suppose I_0 is well sampled (nor blurred nor aliased);
- we traduce it mathematically by :

$$\int_{-\infty}^{+\infty} I_0|x| = \frac{1}{2} \tag{4.1}$$

Due to symetry , we can replace by integral on $[0, +\infty]$ and supresse absolute value :

$$\int_{0}^{+\infty} \frac{x}{\sigma_0\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma_0^2}} = \frac{1}{4} \tag{4.2}$$

We integrate :

$$[\frac{-\sigma_0}{\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma_0^2}}]_0^{+\infty} = \frac{1}{4} \tag{4.3}$$

So :

$$\sigma_0 = \sqrt{\frac{\pi}{8}} \simeq 0.626 \tag{4.4}$$

4.3 Matching by relaxation

4.3.1 Notations

We have two images I and J, and two set of characteristic points in I and J :

- $A_i \in I$ for $i \in [1 \cdots M]$;
- $B_j \in J$ for $j \in [1 \cdots N]$;

We will note D_{Im} the size of the images, typically the length of the diagonal. It will be useful when we need to set some geometric thresholds.

For each characteristic point, we have both a vector descriptor and a position in the image. We note :

- A_i^v, B_i^v the vector descriptor, $A_i^v, B_j^v \in \mathbb{R}^K$ were the dimension K is typically some hundreds; however here we are only interested by the fact there exist some distance D^v between vector such that the lower the distance, the more likely is that points are homologous; we note $D^v(A_i^v, B_i^v) = D^v(A_i, B_i)$
- A_i^p, B_i^p the position ("pixels") in images of A and B , $A_i^p, B_i^p \in \mathbb{R}^2$

We suppose that we can convert the distance D^v in the likelihood L expressing that A and B are homologous. For example we use a threshold σ_v and set :

$$L(A_i, B_j) = e^{-\frac{D^v(A_i, B_i)}{\sigma_v}} \tag{4.5}$$

The transformation from D^v to L , or even the computation of σ_v admitting of formula like 4.5, may be an issue. By the way we will admit from now that we have the function L , probably a way to rationalize this fact would be to "learn" it from statistic of "real" homologous points.

4.3.2 Formulation

We want to compute a matching between A_i and B_j that use both the "radiometric context" informations and the "spatial consistency" information :

- radiometric context : A_i, B_j are matched if, as much as possible, $L(A_i, B_j)$ is high;
- spatial consistency : if A_i and B_j are matched, then for each A'_i and B'_j such that , A'^p_i is close to A^p_i and B'^p_i is close to B^p_i , then the likelihood to match A'_i and B'_j is higher (i.e. neighbors of my homologous are homologous of my neighbors);

The principle of relaxation is to compute a non-unique weighted matching and to use the two pieces of information to iteratively re-compute new value of the weighting. Generally, at the beginning, the "brute" information (=radiometric context) has a higher importance, and as we go further we give more importance to relational information (=spatial consistency).

To formalize the problem we consider that we want to compute simultaneously two functions :

- a discrete matching function ψ such that $\psi(A_i)$ is the homologous of A_i, with possibly $\psi(A_i) = 0$ when no match is found for A_i, and we set $L(A_i, 0) = L_0$ where L_0 is a constant;
- a geometric *smooth* function ϕ .

And the criterion on ψ and ϕ are to maximize global likelihood $\mathcal{L}(\psi)$ (4.6) while minimizing the spatial incoherence $\mathcal{S}(\psi, \phi)$ (4.7).

$$\mathcal{L}(\psi) = \sum L(A_i, \psi(A_i)) \tag{4.6}$$

$$\mathcal{S}(\psi, \phi) = \sum |\psi(A_i)^p - \phi(A^p_i)| \tag{4.7}$$

Of course, there must be some strong regularity constraint on ϕ so that the equation 4.7 is not trivial. Generally it belongs to a parametric space of low dimension (such as similitude, homography ...).

4.3.3 Simultaneous ψ and ϕ ?

In most general case, it is quite complicated to compute simultaneously ψ and ϕ.

4.3.3.1 Classical approach

In the most current case in photogrammetry, a first estimation ψ_0 is computed with a simple strategy as :

$$\psi_0(A_i) = \underset{B_j \in J}{\text{argmax}} L(A_i, B_j) \tag{4.8}$$

And then ϕ is computed to remove false match detected as "outliers" of tested ϕ. There is many tricks to have better result (like symmetric matching), but basically this is the idea.

A particular and very common case of this simultaneous computation is the case where ϕ represents epipolar geometry and is computed by Ransac. By the way in this case, the use of Ransac supposes that we know ψ and that it contains a reasonable proportion of false matches (typically no more than 50%). This case is quite common for example with images acquired the same time (no diachronism) and tie-points resulting from SIFT.

4.3.3.2 Approach by computing ϕ first

In the problem we want to tackle here, the proportion of false match we would get with equations like 4.8 is much too high . So we suppose that we can first have an *approximate* value ϕ_0 of ϕ. We will discuss just after how we can estimate ϕ_0.

We have to specify that what we mean by ϕ_0 is an approximation of ϕ. To understand equation 4.10 and we must imagine a possible scenario for ϕ and ϕ_0. Typicaly ϕ, the real correspondence, will the combination of epipolar geometry and $3D$ scene, so according to the scene, the ϕ can be continuous or

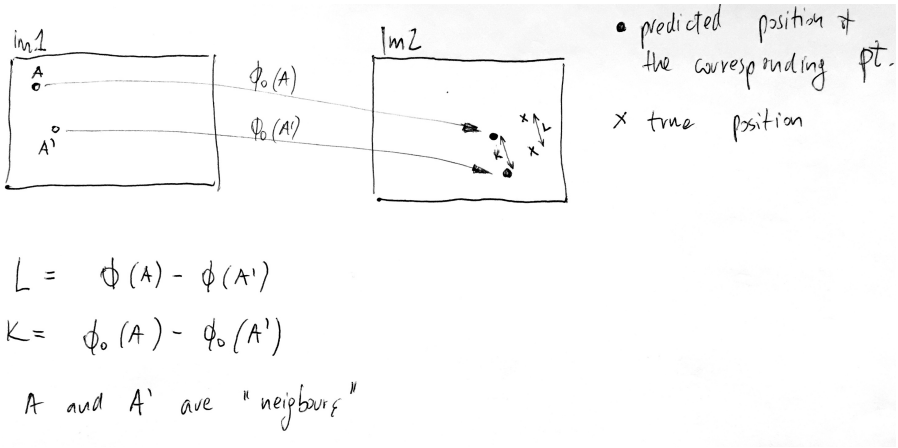

Figure 4.1: Illustration of equation 4.10.

only piece-wise continuous. For ϕ_0, in our context it will be a function with few parameters, typically a similitude plane computed from few points (at least 2).

We then formalize "ϕ_0 approximating ϕ" by (see also Fig. 4.1) :

$$|\phi_0(A) - \phi(A)| < D_A \tag{4.9}$$

$$|(\phi_0(A) - \phi_0(A')) - (\phi(A) - \phi(A'))| < D_g + \alpha|A - A'| \tag{4.10}$$

By extension, using equation 4.7 and criterion $\mathcal{S}$ we want to minimize, we will set :

$$|\phi_0(A) - \psi(A)| < D_A \tag{4.11}$$

$$|(\phi_0(A) - \phi_0(A')) - (\psi(A) - \psi(A'))| < D_g + \alpha|A - A'| \tag{4.12}$$

Let comment these equations :

- in equations 4.9, 4.11, 4.13 the value D_A can be relatively high as $\phi 0$ is extracted with few points and the model (similitude) is a rough approximation of the "real" model; it seems natural to have D_A proportional to D_{Im} (see 4.13);
- equation 4.10, 4.12 model the fact that once we "know" that $\psi(A) = B$ then ϕ_0 is a relatively accurate approximation of ψ around A; when ϕ is continuous, we don't need D_g, however when the $3D$ scene is discontinuous, it create discontinuities ; it seems also natural to have D_g proportional to D_{Im}, and obviously D_g significantly smaller than D_A (see 4.14)

So we can write :

$$|\phi_0(A) - \psi(A)| < \beta_A D_{Im} \tag{4.13}$$

$$|(\phi_0(A) - \phi_0(A')) - (\psi(A) - \psi(A'))| < \beta_g D_{Im} + \alpha|A - A'| \tag{4.14}$$

Of course, practically, an important question is the values of thresholds β_A, β_g, α. As a rule of thumb, values of these parameter can be fixed in the following interval :

- $\beta_A \in [0.05, 0.2]$, less than 0.05 would be very optimistic , and by the way this low value is probably already sufficient for basic approach like in 4.3.5; more than 0.2 would be very pessimistic and by the way difficult to use;
- similarly $\beta_g \in [0.01, 0.05]$ and $\alpha \in [0.05, 0.3]$

And in a first try I woud use $\beta_A = 0.1$ $\beta_g = 0.025$ $\alpha = 0.15$.

4.3.4 Estimation of ϕ_0

Obviously, if we cannot estimate ϕ_0, the previous stuff is useless. Basically, we can imagine 3 way to estimate ϕ_0 :

- the most trivial way is to have an "external" estimation, this can come from meta data ("tableau d'assemblage") or from operator measurements (operator select manualy two real homologous point and it is sufficient to estimate a similitude);
- another easy way is to use tradional Ransac on a solution like 4.8; with D2Net executed at full resolution, it will probably not work as the proportion of false match can be very high; however it is possible to run D2Net at a very low resolution, where 4.8 works not so bad, the accuracy is not so good, but it is probably not a problem;
- a third, more sophisticated way, consist to use a non-unique matching at high or medium resolution; it is described bellow .

Non-unique matching :

- for each point in one image select a number p of its potential matches in the other images, in other words for each point in A_i, select $\Psi(A_i)$ which contains the B_j corresponding to the p best scores of $L(A_i, B_j)$; to compute Ψ it is also possible to compute more globally the $p * M$ best pairs of all matches (A_i, B_j) where M is a predefined value; here, however, there might be a case where certain points are not assigned a match; to assure that all points have matches a mix of both approaches can be adopted (computing a global set of pairs, and also assure to have a minimal number of matches per point);
- generate solutions, by random selection, if we decide to use similitude, it is sufficient to use 2 pairs of (A_i, B_j) ; it is also possible to make a biased random selection that favors the set of pair having higher likelihood (a way to do that is, for each iteration, to generate a few random pairs, and finally select the the pair corresponding to the best likelihood);
- for each 2 pairs, we compute a similitude S and estimate it cost $C(S)$ by formula below;
- after many iteration, select finaly the S minizing $C(S)$. by formula above;

$$C(S) = \sum_i (\underset{B_j \in \Psi(A_i)}{\text{Min}} c(A_i, B_j, S)) \tag{4.15}$$

For $c(A_i, B_j, S)$, different formula can be tested, a basic formula proportional to distances, one using a threshold to limit the influence of outliers :

$$c(A_i, B_j, S) = Min(|S(A_i) - B_j|, D_A) \tag{4.16}$$

Or a smoother version :

$$c(A_i, B_j, S) = \frac{|S(A_i) - B_j|}{|S(A_i) - B_j| + D_A)} \tag{4.17}$$

It is also possible to use the likelihood $L(A_i, B_j)$ and merge it with the geometric term , however it is always complicated to mix valures of different kind.

4.3.5 Basic usage of ϕ_0

The easiest way to use ϕ_0 is to re-use the basic strategy of equation 4.8 but filtering the result with equation 4.9 :

$$\psi_1(A_i) = \underset{B_j \in \mathcal{D}(\phi_0(A_i), D_A)}{\text{argmax}} L(A_i, B_j) \tag{4.18}$$

Where $D(\phi_0(A_i), D_A)$ is the disc of center $\phi_0(A_i)$ and radius D_A.

4.3.6 Relaxation

Part III

Reference documentation

Part IV

Annexes

Appendix A

Bibliography

Bibliography

[Tomasi Kanabe 98] S. Roy, I.J. Cox , 1998, "Shape and Motion from Image Streams under Orthography: a Factorization Method", International Journal of Computer Vision, 9:2, 137-154 (1992)

[Cox-Roy 98] S. Roy, I.J. Cox , 1998, "A Maximum-Flow formulation of the N-camera Stereo Correspondence Problem", *Proc. IEEE Internation Conference on Computer Vision*, pp 492–499, Bombay.

[Fraser C. 97] C. Fraser, 1997, "Digital camera self-calibration", *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 52, issue 4, pp. 149-159,

[Penard L. 2006] L. Pnard, N. Paparoditis, M. Pierrot-Deseilligny. "Reconstruction 3D automatique de faades de btiments en multi-vues.", RFIA (Reconnaissance des Formes et Intelligence Artificielle), Tours, France, January 2006.