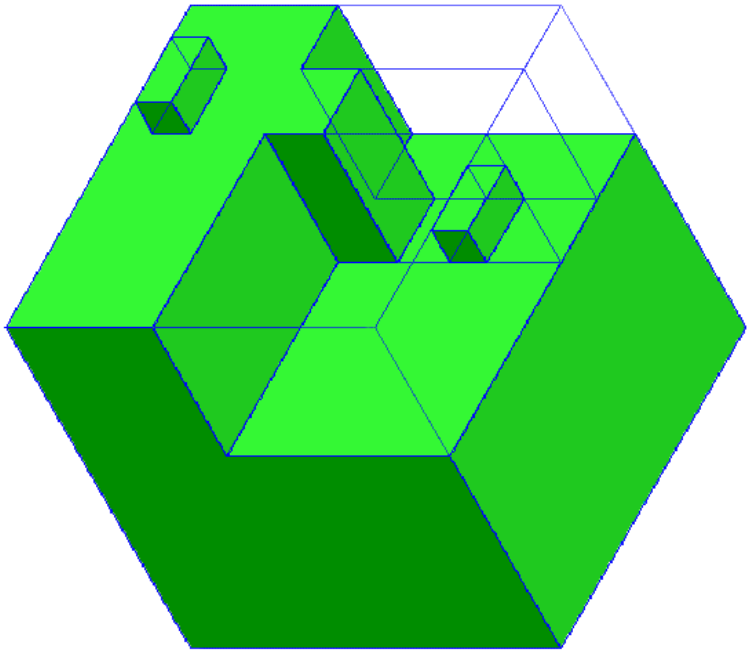


The Integrated Feature-based Modeller - The Completion Step

Francesco Robbiano
Genova, March 24th, 2004

Technical Report n.5/2004, CNR-IMATI, Genova

Content

THE INTEGRATED FEATURE-BASED MODELLER - THE COMPLETION STEP.....1

0) INTRODUCTION 3
0.1) The Integrated Feature-Based Modeller *3*
0.1.1) The recognition step 3
0.1.2) The completion step.....3
0.2) Where is the feature? *4*
0.3) Geometric characteristics of a feature *4*
1) VOLUME GENERATION PROCESS 4
1.1) Other decomposition criteria.....*7*
2) USAGE 10
2.1) Input.....*10*
2.2) Output *10*
Recognition 10
Completion.....11
A) APPENDIX A – COMPLETION EXAMPLES 12
Example 1 12
Example 2 13
Example 3 14
B) APPENDIX B – FILE FORMATS 15
SFOG++ format 15
FEAT format.....15
C) APPENDIX C – LOCATION OF FILES 16
D) APPENDIX D – COMPILING INSTRUCTIONS.....17
E) APPENDIX E – USING ACIS R10 WITH VISUAL C++.....17
F) APPENDIX F – USING INTEROP ACIS-STEP READER/WRITER 19
BIBLIOGRAPHY 21

0) Introduction

This document describes the upgraded completion process within the Intermediate Modeller module. Moreover, for what concerns the recognition and the completion steps, the usage, the results, and the choices that can be done thanks to the textual interface provided with the application are also explained.

0.1) The Integrated Feature-Based Modeller

The Integrated Feature-Based Modeller is the kernel of Integrated Feature-Based Prototype System, developed within a European cooperation between Fraunhofer-IGD, Darmstadt and IMATI, C.N.R., Genova. It is written in C++ and its specific characteristics are explained in [GAMetal96].
It includes the recognition and the completion steps.

0.1.1) The recognition step

Based on the boundary model of ACIS (see [ACIS01]), the feature recogniser performs a reasoning process in order to identify and classify characteristic regions. This process performs analysis of convex and concave regions on the solid model. It derives a shape feature based model, which contains a decomposition of the boundary model into facesets (one for each detected feature). Faces belonging to concave regions are grouped into depression features, faces belonging to convex regions are grouped into protrusion features, and all the other faces constitute the so-called mainshape. All the features' information (including the interaction among features) is kept in a SFOG model, and the hierarchy can be saved in a text file (see Appendix B, SFOG++ format).

0.1.2) The completion step

The completion step aims at determining the volumetric representations of the space occupied by the features of an object. The objective is to evaluate the volumes that can be associated to specific operations executed in an application context, such as design, assembling or machining.

The goal of the completion step is to compute these volumes, without bindings to any specific context: there is no unique output that fits every application context, therefore we look for generic classes of volumes while avoiding to produce an overfragmented decomposition, because in such way some expressiveness would be lost. It is important to notice that the goal is to produce positive volumes, even in the case that the actual volumes are negative (depressions). In Chapter 1 it will be shown how this situation is dealt with.

In the completion step the aim is to deduce volumetric descriptions of features by the information given by facesets. For example, in the machining application context, this process is used to go back up to the so-called "stock material", i.e. the original block that is modified through machining operations until the present 3D model is achieved.
That is the reason why it is called "completion step": the original object is "completed" in order to reconstruct the stock material.
On the other hand, using the inverse approach, it is possible to say that the original object can be obtained by the stock material through the application of Boolean operations: union for protrusions, subtraction for depressions.
Let:

- VS = Stock Volume

- VO =Object Volume
- VP = Volume of Protrusions (union of all the protrusions)
- VD = Volume of Depressions (union of all the depressions)

So it should be possible to say that:

VS = VO \ VP ∪ VD and
VO = VS ∪ VP \ VD

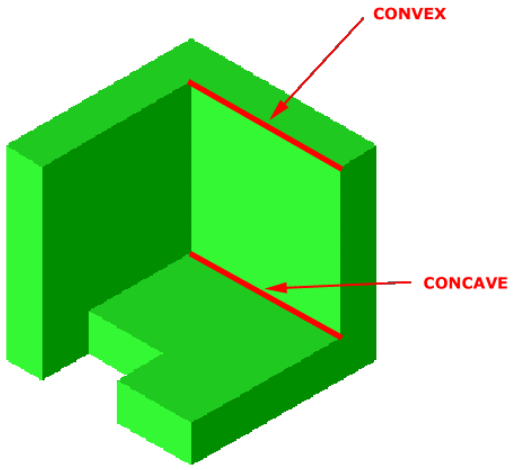
Actually this is not yet precise, as the sequence of the operations is fundamental.
Thus, the process has to identify a precise sequence of operations to succeed.

0.2) Where is the feature?

In order to calculate volumes it is important to understand "where" the identified feature is located. A protrusion feature is connected to the "full" part of the solid, while a depression feature is connected to its "blank" part.
We will call "intrinsic nature" of the feature this portion of space: the "blank" in case of a depression, the "full" in case of a protrusion.

0.3) Geometric characteristics of a feature

According to the recognition process, inside a feature the faces are mutually connected by adjacencies that are all concave or all convex: in other words they are subsets of the object's faces that constitute local convexities (for protrusions) or local concavities (for depressions).



1) Volume Generation Process

The completion step is applied to a SFOG model obtained by the shape feature recognition.
The specific structure of SFOG model is explained in [GAMetal96] and some introductive information can be found in [PET95].
It includes:

- The B-Rep of the object (through facesets)
- Information about all the detected features (including B-Rep of facesets)
- Information about all the mutual adjacencies (including B-Rep)

- Hierarchical structure of adjacencies between features (SFOG++ format, see Appendix B)

The completion step is iterative: the addition of the computed volumes to the existing object produces a new object. Recognition is performed on this new object and then completion can be performed again. The completion step is iterated until the recognition step yields no other feature than the mainshape or when the completion step doesn't produce any new volume.

For every feature, the portion of space to locate is determined by the *intersection* of the **internal region**, i.e. the region calculated through combination of the halfspaces induced by the faces of the feature, and the **external region**, i.e. the region calculated through combination of the halfspaces induced by the faces adjacent to the feature.

The **internal region** is determined by the intersection of all the halfspaces induced by the faces of the feature. Since depressions are associated to negative volumes, in order to compute the internal region of depressions the normals are inverted (as said before, the goal is to produce positive volumes). Therefore, in any case they will locate the intrinsic nature of the feature.

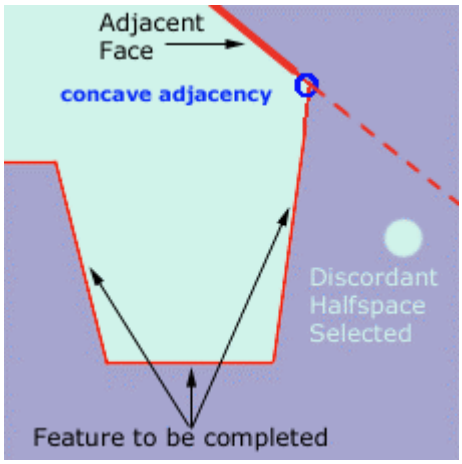
Assumptions:
- Every feature FF_k is defined by planar faces F_{ki}
- Every face F_{ki} can be associated to a plane, and to the two halfspaces induced by it
    - Let's call "concordal" the halfspace that shares the normal with the face (i.e. that is below the face according to its normal) and "discordant" the other halfspace

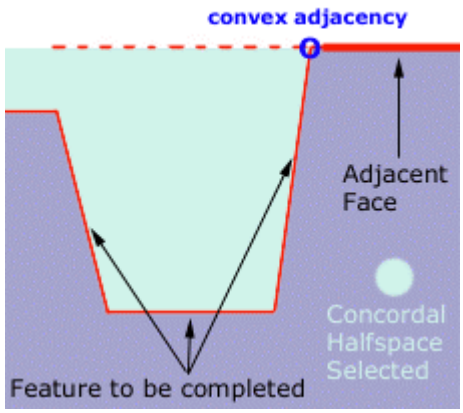
Algorithmic approach for computing the internal region I_{FFk} of feature FF_k:
- For every face F_{kj} of FF_k ($j \in J$):
    - If FF_k is a protrusion (or is the mainshape) let HS_{kj} be the concordal halfspace
    - If FF_k is a depression let HS_{kj} be the discordant halfspace (the normal must be inverted)
- $I_{FFk} = \cap_{j \in J} HS_{kj}$

The **external region** is obtained by considering the faces adjacent to the feature to be completed.
The process is more complex, because the proper halfspace induced by each external face has to be selected, and a key role is played by the kind of adjacency between the considered feature and the external face.

Non-algorithmic approach for computing the external region of feature FF_k:
Two main decisions have to be made:
- For each external face which of the two induced halfspaces to use
- How to combine these halfspaces.

If the adjacency between the feature and the external face is concave the discordant halfspace must be chosen, otherwise the concordal halfspace must be chosen.





For all the external faces belonging to the same feature the decision (concordal/discordant halfspace) will be the same, since the adjacencies between one feature and another are all concave or all convex. So it is possible to group all the halfspaces belonging to the same feature, and to combine them using either the intersection or the union operator.

According to the mutual adjacency (concave or convex) and to the nature of the adjacent feature (protrusion or depression), one out of four cases is matched.

Specifically, the cases in which the intersection operator has to be used are:
- Concave adjacency with depression
- Convex adjacency with protrusion

And the cases in which the union operator has to be used are:
- Convex adjacency with depression
- Concave adjacency with protrusion

The obtained portions of space will be later combined through intersection.

Algorithmic approach for computing the external region E_{FFk} of feature FF_k:
- The external region E_{FFk} must be computed as a combination of the completion volumes induced by the external features.
- Let FG_j be all the features adjacent to FF_k ($j \in J$)
- $CompVol_{kj}$ is the completion volume relative to feature FF_k induced by feature FG_j
- Computation of $CompVol_{kj}$:
    - For every feature FG_j adjacent to FF_k:
        - Selection of proper halfspace HS_{kji}: for every face G_{ji} of FG_j ($i \in I$)
            - If the adjacency between FF_k and FG_j is convex let HS_{kji} be the concordal halfspace
            - If the adjacency between FF_k and FG_j is concave let HS_{kji} be the discordant halfspace
        - If the adjacency between FF_k and FG_j is convex and FG_j is a protrusion
            - set OP = $\cap$ (intersection)
        - If the adjacency between FF_k and FG_j is concave and FG_j is a depression
            - Set OP = $\cap$ (intersection)
        - If the adjacency between FF_k and FG_j is concave and FG_j is a protrusion
            - Set OP = $\cup$ (union)
        - If the adjacency between FF_k and FG_j is convex and FG_j is a depression
            - Set OP = $\cup$ (union)

- $CompVol_{kj} = \mathbf{OP}_{i \in I} HS_{kji}$
- $E_{FFk} = \cap_{j \in J} CompVol_{kj}$

1.1) Other decomposition criteria

To avoid an inappropriate decomposition of the volumes, other criteria for determining the internal and external volumes can be used.
A high-level grouping can be induced: some features can be collapsed into one. It is necessary to discriminate features that induce a volume by themselves by features that can be considered part of other features (and therefore, in a certain sense, "inactive"). The features that need to be grouped with other features are named "degenerate".

Some kind of features can be considered degenerate:

- Features made up by just one face (do not induce a completion volume by themselves)
- Protrusions with all the faces that are adjacent at most to one face of another feature (they can be grouped with the related face)
- Protrusions that don't descend just from the mainshape

There is another particular case. It may happen that a depression FD_m is adjacent to another depression FD_n. If all the faces of FD_n are adjacent **just to one face** of FD_m it is possible to say that FD_n is "nested" inside FD_m (see image B.2).
An option is not to use faces of a depression that is nested inside a feature for the definition of its external region.

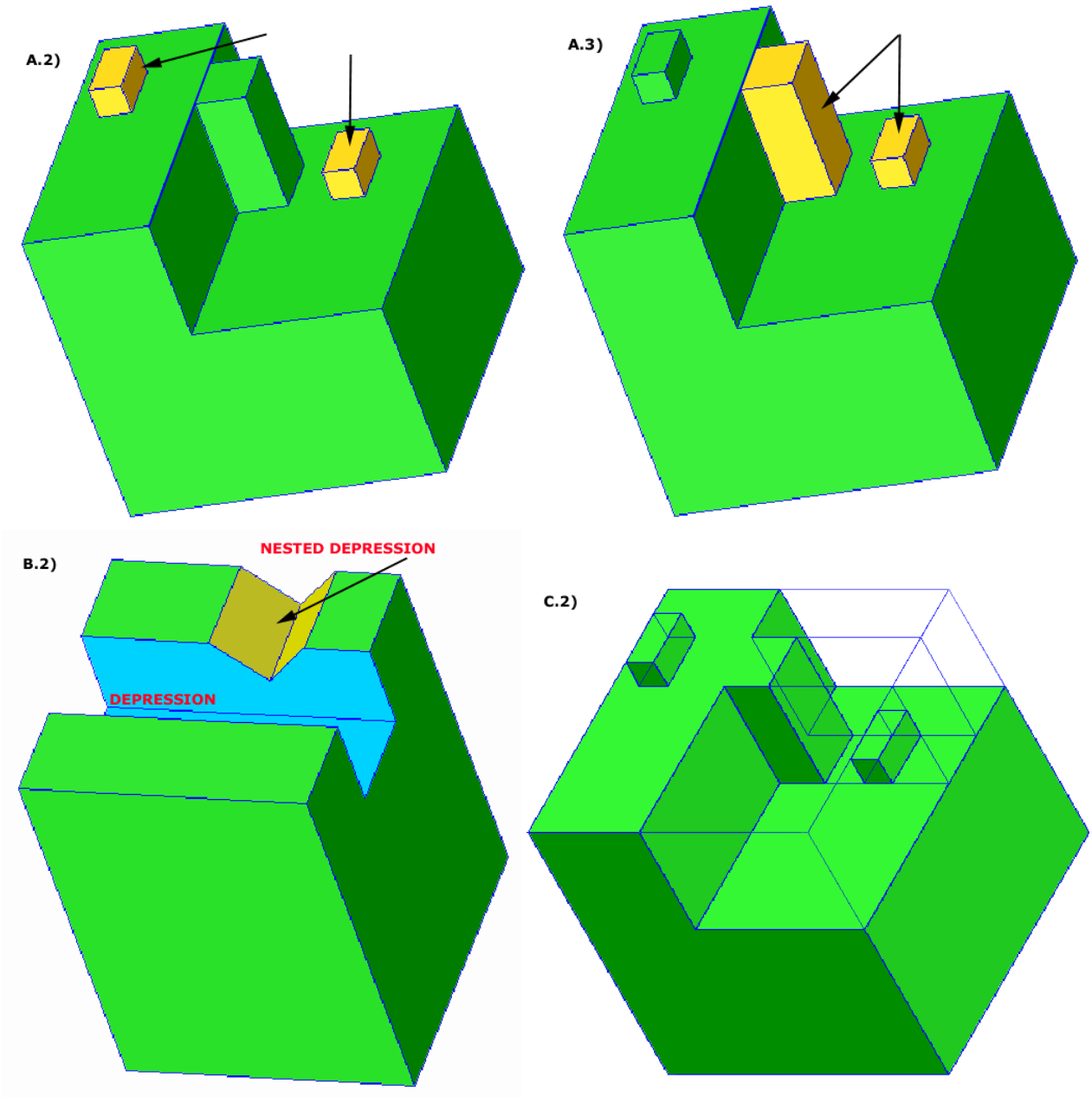
Moreover it is possible to choose to subdivide the completion volume of depressions along the coplanar planes of the internal regions: if two or more faces belonging to the internal region of the same feature are coplanar, it is possible to use the common plane as a subdivision plane.

These three choices are combined in one flag:

Criterion A) (on protrusions)
1) No added criterion
2) Protrusion is degenerate if its faces are adjacent to only one face of another feature
3) Protrusion is degenerate if it is adjacent to feature different from the mainshape

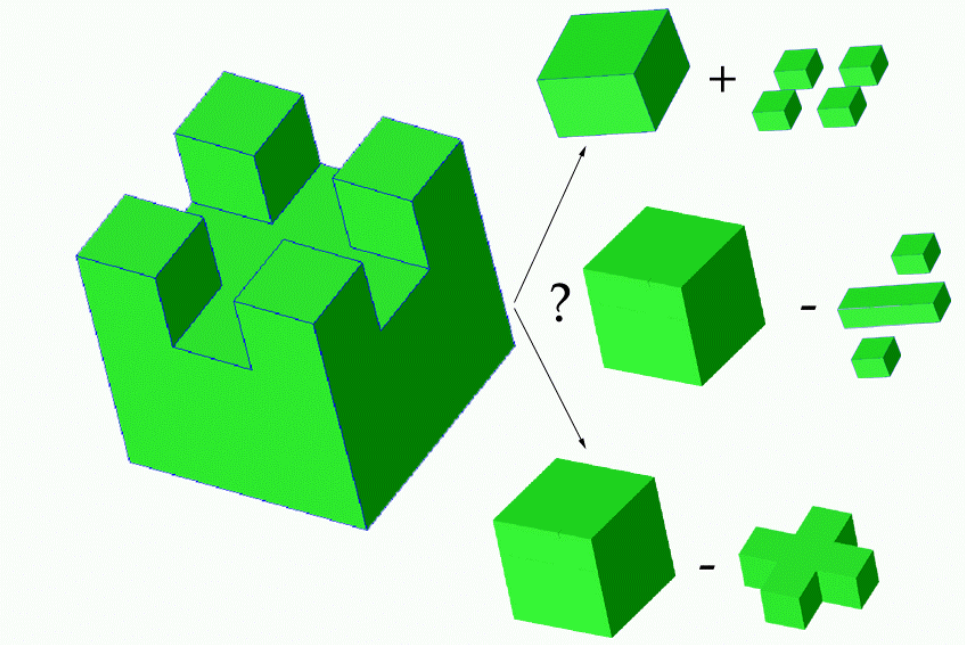
Criterion B) (on depressions)
1) No added criterion
2) Depressions nested inside FF_k are not considered in the definition of the external region of FF_k

Criterion C) (on decomposition)
1) No added criterion
2) Subdivision of the completion volume of depression along the coplanar planes of the internal region



The criteria A2 and A3 (on protrusions) are used when the isolation of protrusions is not very relevant in the decomposition process (for example, in a machining context).
The criterion A2 is used especially in an assembling context, in which isolated protuberances are often conceptually linked with the face they lie upon.
The aim of the criterion A3 is to give an absolute precedence to completing depressions, going on to protrusions just when they are connected only to the mainshape, i.e. they don't interact with any other depression.
The criterion B is used when the aim is to fragment depressions as less as possible.
The criterion C is used when there is a close relationship between generated volumes and cutting operations (again, especially in machining contexts) or in general when elementary volumes are looked for, in order to be rearranged to produce various volume alternatives.

These criteria are conceptually linked to the detection of interacting features configurations, which normally produce situations giving rise to multiple interpretations (see image below).
See Appendix A for examples on the application of these criteria.



Combining the 3 criteria the possible choices are 12:

Flag	Protrusion criterion	Nested depressions omitted	Internal decomposition
0	NONE	NO	NO
1	DEG ONE FACE	NO	NO
2	DEG NO MAIN	NO	NO
3	NONE	YES	NO
4	DEG ONE FACE	YES	NO
5	DEG NO MAIN	YES	NO
6	NONE	NO	YES
7	DEG ONE FACE	NO	YES
8	DEG NO MAIN	NO	YES
9	NONE	YES	YES
10	DEG ONE FACE	YES	YES
11	DEG NO MAIN	YES	YES

Flag 0 (yellow) **0** criteria applied
Flag 1,2,3,6 (green) **1** criterion applied
Flag 4,5,7,8,9 (orange) **2** criteria applied simultaneously
Flag 10,11 (violet) **3** criteria applied simultaneously

2) Usage

Run the executable file SfogProj.exe with one parameter, specifying the name of the file without extension (disregarding of the file format).

Example: C:\> SfogProj cross_slot

Then set the required values of some specific flags according to the interactive runtime instructions.

2.1) Input

Flags (/choices) are used in order to:

- Specify the format of the input model: SAT or STEP (thanks to the INTEROP translator STEP files are preliminarily converted into SAT files)
- Decide whether to perform the recognition process or the completion process.
- For recognition:
    - Decide whether to produce files for every feature and for every macrolink or not.
    - Decide whether to produce also STEP files or not.
- For completion:
    - Decide to produce a unique output file or a file for every completion step.
    - Specify which decomposition criteria has to be used.
    - Decide whether to produce also STEP files or not.

A string (a parameter for the execution) is requested to identify the file to process. The string <name> corresponds to the file <name>.sat or <name>.stp (according with the above-mentioned choice).

2.2) Output

Recognition

At the end of the recognition process the SFOG++ file and the FEAT file (see Appendix B for details) are obtained. Moreover, if requested, also each feature's faceset and each macrolink's faceset are produced. According with the choice of the user the format will be SAT or STEP.

If the name of the model (without extension) on which the recognition step is performed is <name>, the feature CronoNumber (see Appendix B) is <fcnum>, and there is a macrolink between a feature with CronoNumber <fcnum1> and a feature with CronoNumber <fcnum2>, then for every feature the output files will have the following names:

- <name>_<fcnum>.sat (and possibly .stp)
    - The faceset of the feature
- <name>_<fcnum1>_<fcnum2>.sat (and possibly .stp)
    - The faceset of the macrolink

The overall results will be stored in:

- <name>.sfog
    - The SFOG++ file

- <name>.feat
    - The FEAT file
- <name>.sfog.ff.sat (and possibly .stp)
    - The B-Rep of the original model, structured in one ACIS Body (see [ACIS01]) for each feature
- <name>.sfog.ml.sat (and possibly .stp)
    - The B-Rep of all the macrolinks, structured in one ACIS Wire Body for each macrolink

Completion

According to the user's choice, either a file for every completion step or a unique file at the end of the process can be obtained.
If the name of the model (without extension) on which the recognition step is performed is <name>, the value of the flag is <flag> and the current completion step is <step> then the output files for every completion step will have the following names:

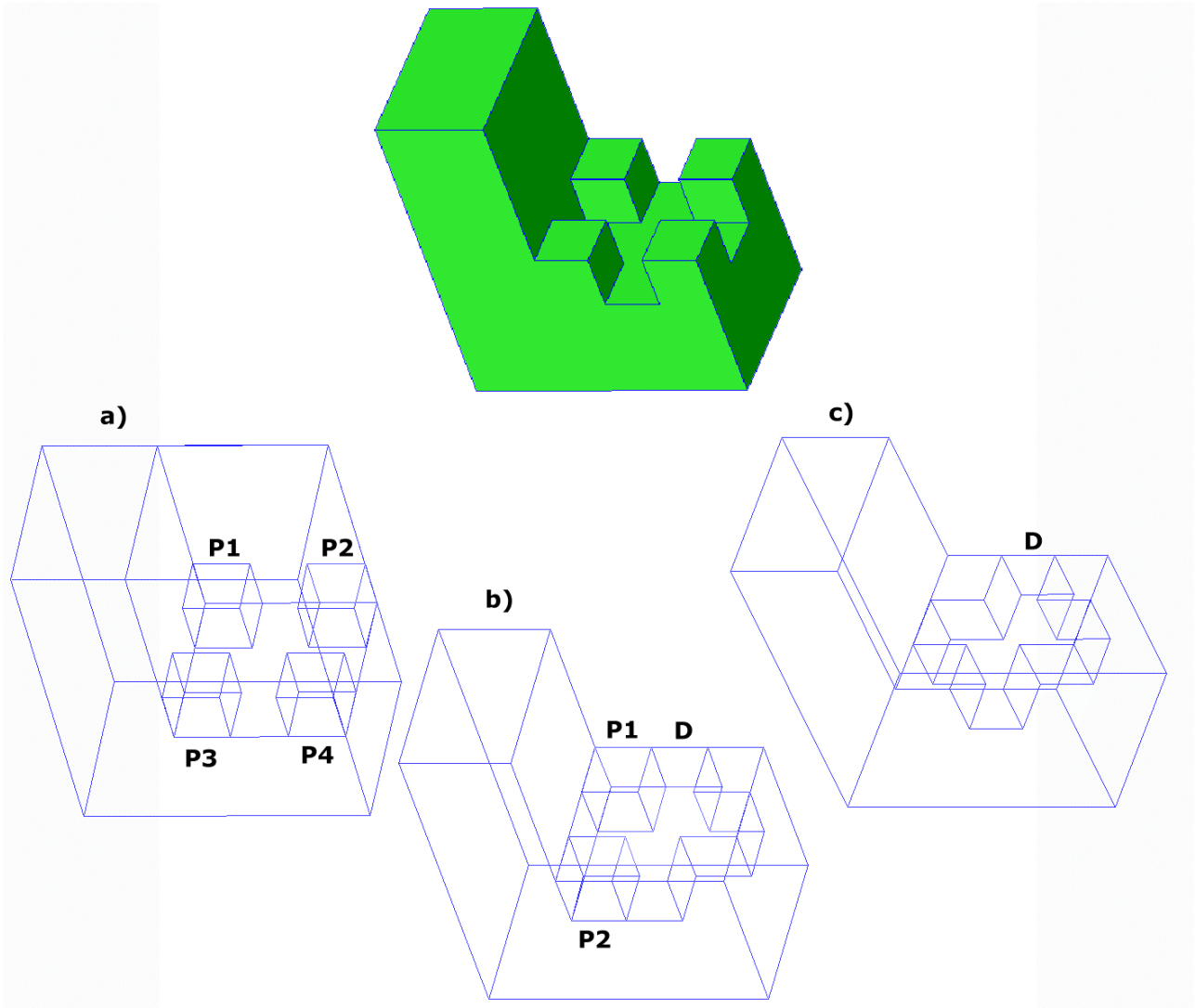
- OUT<name>_flag_<flag>_step_<step>.sat (and possibly .stp)

At the end of the process the SFOG model is stored in the files:

- <name>COMPLETED.sfog
    - The SFOG++ file
- <name>COMPLETED.sfog.ff.sat (and possibly .stp)
    - The B-Rep of the completed object, structured in one ACIS Body for each feature's added volume

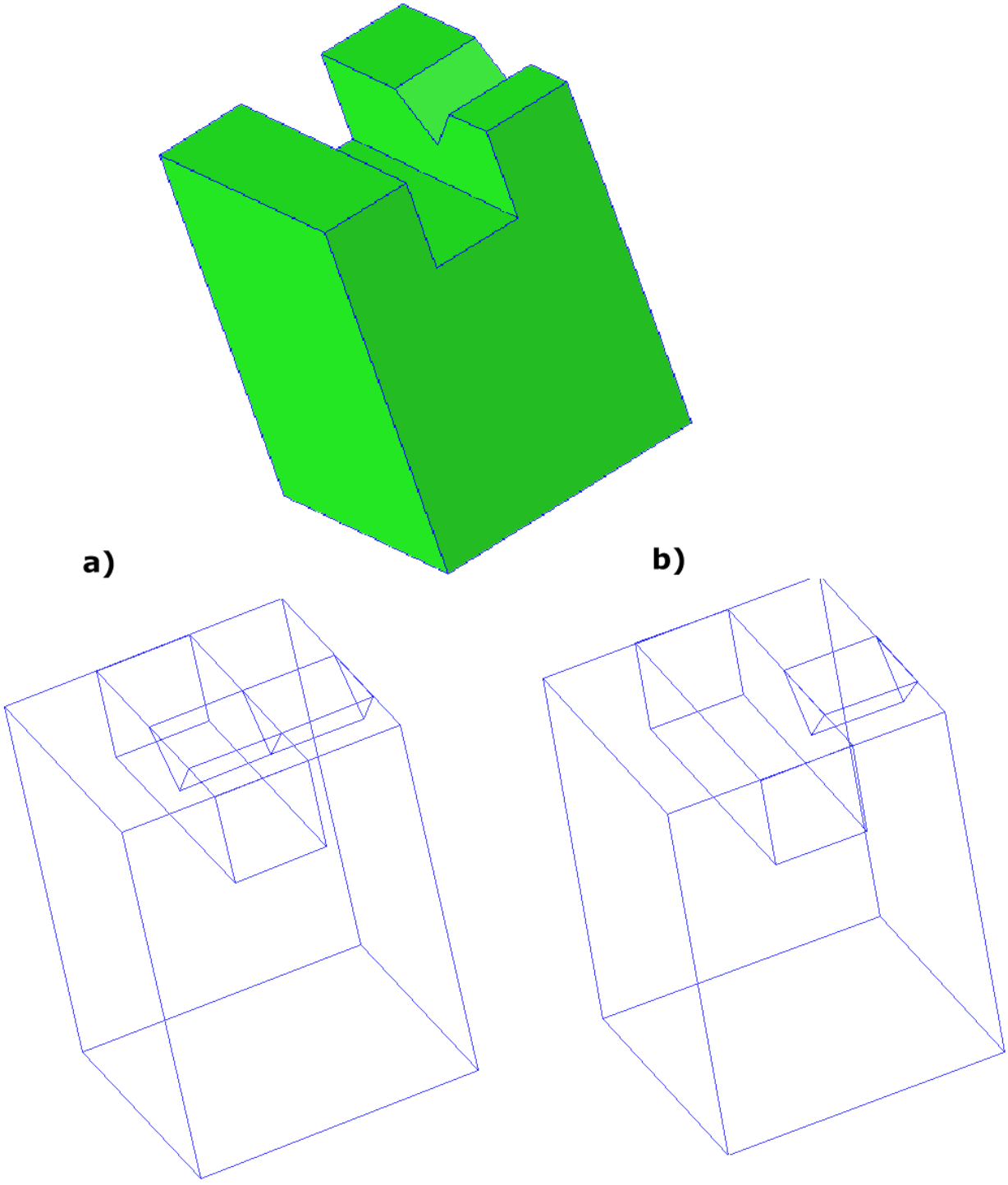
A) Appendix A – Completion examples

Example 1



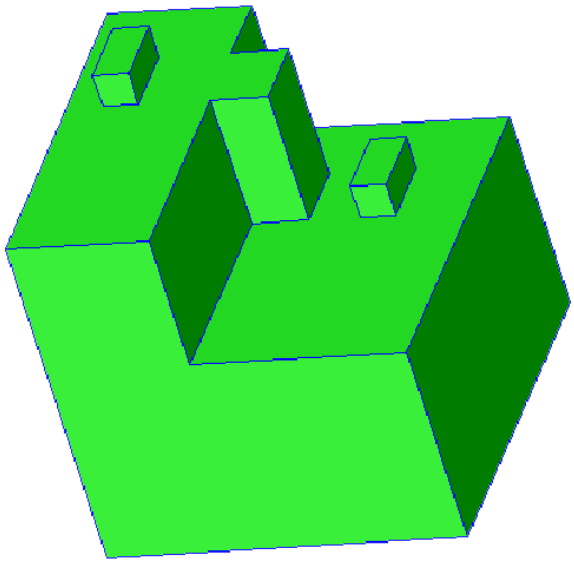
In the **example 1** the completion process takes place with different values of the above mentioned flag. The aim is to focus on the flag on protrusions, and so for *figure 1.a* it is used the flag 0 (no added criterion on protrusion), for *figure 1.b* it is used the flag 1 (protrusion degenerate if adjacent to one face), and for *figure 1.c* it is used the flag 2 (protrusion degenerate if adjacent to features different from mainshape).
In *figure 1.a* four protrusions are located: P1, P2, P3, P4. None of them is considered degenerate, so none is merged with the mainshape, and no depression is located.
In *figure 1.b* the former P3 and P4 are adjacent to one face, and so are considered degenerate. Therefore only P1 and P2 are located, and also the depression D is detected.
In *figure 1.c* all of the former protrusions are considered degenerate, as they are adjacent to features different from the mainshape. The depression D is detected.

Example 2

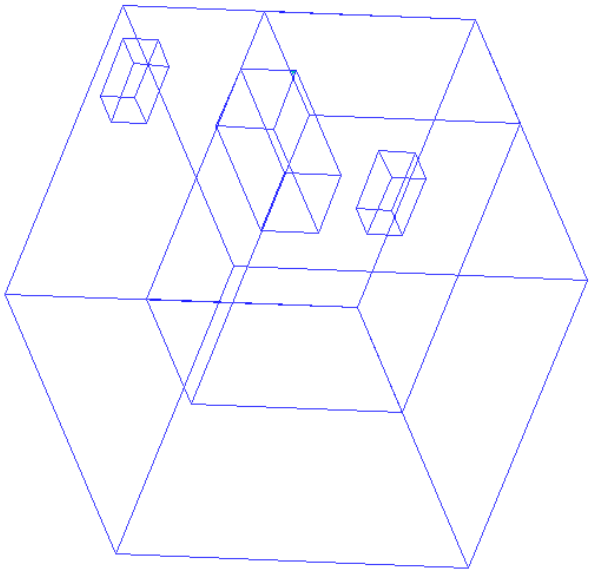


In the **example 2** the completion process takes place with different values of the above mentioned flag. The aim is to focus on the flag on depressions, and so for *figure 2.a* it is used the flag 0 (no added criterion on nested depressions), for *figure 2.b* it is used the flag 3 (nested depressions are not considered for the definition of the external region).
In *figure 2.a* in the calculation of the external region of the main depression the nested depression is considered, and so the calculated volume is fragmented.
In *figure 2.b* in the calculation of the external region of the main depression the nested depression is not considered, avoiding the internal fragmentation of the space.

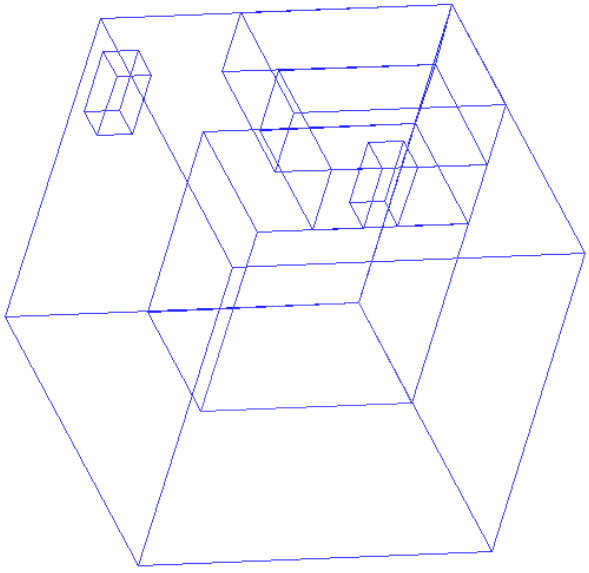
Example 3



a)



b)



In the **example 3** the completion process takes place with different values of the above mentioned flag. The aim is to focus on the flag on internal decomposition, and so for *figure 3.a* it is used the flag 0 (no added criterion on internal decomposition), for *figure 3.b* it is used the flag 8 (internal decomposition along coplanar planes + protrusion degenerate if adjacent to features different from the mainshape).
In *figure 3.a* the volume is not decomposed.
In *figure 3.b* two criteria are combined. The flag on protuberances (protrusion degenerate if adjacent to features different from the mainshape) makes the two protuberances to be merged in the same feature. Therefore, in the calculation of the internal region of that feature, two pairs of coplanar faces are detected, and the relative coplanar planes are used in the voulme's subdivision.
We can observe that in this particular case the result given in *figure 3.b* is obtained only when the two criteria are combined simultaneously (flag 8 and flag 11). Otherwise the protuberances are separated and no coplanar plane is detected (with all the other flag's values the situation represented in *figure 3.a* is obtained).

B) Appendix B – File formats

SFOG++ format

The **SFOG++** format is used to express the nature of every recognized feature and the mutual relations between features. To be more precise, it stores information about the Features and about the Macrolinks.

Extension: *.sfog*

```
<FN = Number of features>
For each feature
        <CronoNumber> <IsRoot> <Dimension> <Sign>
For each macrolink
       <CronoNumberA> <CronoNumberB> <Direction>
-1
```

CronoNumber is an integer going from 0 to FN-1
IsRoot is 1 when the referred feature is the root feature, 0 otherwise
Dimension specifies the dimension of the feature (2 for bidimensional, 3 for tridimensional)
Sign is –1 for depressions, 1 for protrusions, 0 for mainshape.
Direction specifies the direction of the macrolink, i.e. the arc of the SFOG++ graph. Its value can be:

- 0 – a2b – arc directed from a to b
- 1 – b2a – arc directed from b to a
- 2 – both – bi-directional arc
- 3 – none – undefined direction

FEAT format

The **FEAT** format is used to complete the information provided in the SFOG++ file with the characterization and the parametrization of every recognized feature.

Extension: *.feat*

First part:

Format as in the .sfog file

+

Second part:

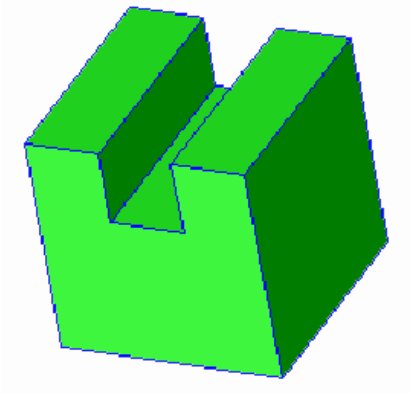
```
For Each Feature
  <Feature CronoNumber>
  <Name of matched feature>
  <Position> <x> <y> <z> (if computed)
  <Axes Vectors> (if computed)
  X <x> <y> <z>
  Y <x> <y> <z>
  Z <x> <y> <z>
  <Number of Parameters>
  For Each Parameter
    <Parameter Name> <Parameter Value> <Possible Axes on which it is computed>

-1 (end-of-file)
```

Example of SFOG++ and FEAT files produced by the recognition of a cube with a through slot.

SFOG++ file:

```
2
0 1 2 0
1 0 2 -1
0 1 0
-1
```



FEAT file:

```
2
0 1 2 0
1 0 2 -1
0 1 0
0
MainShape
1
Prism_Through_Slot
15.000000 15.000000 30.000000
X -1.000000 0.000000 0.000000
Y  0.000000 1.000000 0.000000
Z  0.000000 0.000000 1.000000
3
Width = 10.000000 X
Depth = 10.000000 Z
Length = 30.000000 Y
-1
```

C) Appendix C – Location of files

Let's take the folder "IntermediateModeler" as the current directory.

The main file *mainRec.cxx* is located in:

- Riconoscitore/recognition

The source files (C++ files) are located in:

- Sfog/src
- Ferg/src
- Riconoscitore/Completion
- Riconoscitore/Recognition

The header files (H files) are located in:

- Sfog/incl
- Ferg/incl
- Riconoscitore/incl/recognition
- Riconoscitore/incl/completion

The executable file (EXE file) is located in:

- Sfog/src/Demo/TestSfogPrj

The input models are located in:

- Sfog/src/Demo/TestSfogPrj/Input

The output files are located in:

- Sfog/src/Demo/TestSfogPrj

D) Appendix D – Compiling instructions

To compile and build the code a C++ project must be created.
In the folder Sfog/src/Demo/TestSfogPrj there is a Visual C++ workspace file, SfogPrj.dsw.
Anyhow in the project it is necessary to include all the header files (set the appropriate directories in the project settings) and all the source files listed in Appendix C.
The appropriate directories will include:

- The module's header files
- The header files for the ACIS software library
- The header files for the INTEROP software library

The *main* is included in the file MainRec.cxx.
For more information about using ACISr10-based applications see Appendix E, and for more information about using Interop ACIS-STEP Reader/Writer see Appendix F.

E) Appendix E – Using ACIS r10 with Visual C++

After **unzipping** the installation file, there will be the directory …XYZ/AcisR10 (from now on this directory will be called **ACISDIR**) where there is everything needed.

First of all make sure that version **6.0** (or later) of **Visual C++** is installed.
With earlier version of C++ the files will probably not even compile.

At the beginning of a C++ file based on ACIS there are some "include" commands. Obviously their paths are not absolute, because they depend on where the ACIS main directory is. Therefore it must be specified to the compiler where this directory is.

Tools – Options – Directories – Show Directories For: "Include Files" →

Set the absolute path of all the needed subdirectories (one level deeper of ACISDIR) of Acis.

For example it is possible to add ACISDIR/cstr for constructors utilities, ACISDIR/bool for Boolean operations, and so on.

This should be enough to **compile** correctly.

Now the target is to build a **console application** (basically a DOS program which executes in a Window shell).

1) **Accessibility of libraries**.

There are two methods to make the ACIS debug .dll and .lib files accessible to the application:

- Copy the ACIS debug dll library files from the lib directory (ACISDIR/lib/NT_DLLD) to the system directory (windows/system).
- Extend system Path environment variable to include the lib subdirectory.
- On WindowsNT:
    a. Right click on the "My Computer" icon
    b. Go to Properties – Environment
    c. Enter in Variable: Path

d. Enter in Values : %Path %;ACISDIR/lib/NT_DLLD
e. Press « Set »

2) **C++ Settings**

- File - New - Project →
  - Select "Win32 Console Application" and enter project name and location (as prompted)
- Project – Add to Project – Files →
  - "my_ACIS_example.cxx"
- Build – Set Active Configuration →
  - Select "Win32 Debug"
- Project – Settings – C++ - Category – Preprocessor →
  - Add "NT" and "ACIS_DLL" to the "Preprocessor Definitions"
- Project – Settings – C++ - Category – Code Generation →
  - Set "use runtime library:" to "Debug_Multithreaded_DLL"
- Project – Settings – Link →
  - Set "Category" to "Input"
  - Set "Additional Library Path" to ACISDIR/lib/NT_DLLD
- Project – Settings – Resources →
  - Add to "Preprocessor definitions" NT

F) Appendix F – Using Interop ACIS-STEP Reader/Writer

For detailed information about Interop ACIS-STEP Reader/Writer see [INTDOC]

Development Environment

The ACIS STEP Reader/Writer requires the following system settings:

Product Dependency
ACIS 8.0, ACIS R10SP2 or ACIS R11.

Integrated Development Environment
Microsoft Visual C++ 6.0 SP5 for Windows 98 2nd Edition, Windows NT 4.0 SP5 and Windows 2000 SP2.

Environment Variables
The list of environment variables required and the values to be set (for sample code and recommended for your application development) are as follows:

Environmental Variable	Value
X3DT	<Path to the InterOp installation>
A3DT	<Path to the ACIS Installation>
SPA_LOG	<Path to the Log File> (for sample application)
SPA_DATA_IN	<Path to the Input File> (for sample application)
SPA_DATA_OUT	<Path to the output file> (for sample application)
ARCH	Valid ARCH settings for this release are: NT_DLL - Windows release NT_DLLD - Windows debug NT_NET_DLL - Windows .NET release NT_NET_DLLD - Windows .NET debug Linux - linux_so HP - hp700_11_so SUN - solaris_so SGI - sgi_so AIX - aix4_so mac_carb - MAC release mac_carb_debug - MAC debug
PATH	%X3DT%\lib\%ARCH%;%A3DT%\lib\NT_DLL (This needs to be appended to the existing path)

How To Develop Your Application

This section describes the process that should be followed to plug the ACIS STEP Reader/Writer into your application.

Setup the environment
Set up the environmental variables as indicated in the above paragraph "Development Environment". The IDE uses the variables set. You may want to set the variables at user level to avoid system variables getting distended.

Setup the IDE
For the IDE Microsoft Visual C++:

- Create a console application in the IDE

Include directories

Set the additional include directories to:

- For ACIS R11
  - *$(X3DT)\include,$(A3DT)\include*
- For all previous ACIS releases
  - *$(X3DT)\include, $(A3DT)\eulr, $(A3DT)\oper, $(A3DT)\ga, $(A3DT)\fct, $(A3DT)\bool, $(A3DT)\kern, $(A3DT)\cstr, $(A3DT)\base, $(A3DT)\intr*

This could be reached in the IDE through

- Project -> Settings -> C/C++ tab -> Preprocessor (in category) -> Additional include directories: text field

Object/Library modules

Add *kernel.lib* and *acisstep.lib* to the Object/Library modules text field.

This could be reached in the IDE through

- Project -> Settings -> Link tab -> General (in category) -> Object/Library modules: text field

Code
To build a simple application, follow the sequence below:

- **Initialize the translator**
  - To make the translator APIs available for the application, it needs to be initialized by invoking api_initialize_xstep.
- **Translation**
  - Translate ACIS entity list to STEP file by invoking api_xstep_write methods respectively.
- **Terminate the translator**
  - Release all the resources and unload all the libraries specific to the translator by invoking api_terminate_xstep.

Build the application
Build the application using the IDE.

Running the application
Run the application by supplying the command line arguments depending on the application developed.

Bibliography

[ACIS01] J.Corney - T.Lim, “3D Modeling with ACIS” , 2001
[GAMetal96] F.Gamba – F.Petta – S.Haßinger – G.Brunetti , “Integrated Feature-Based Prototype System – Developer’s Guide”, European HCM Programme No. ERB CHBGCT 930380 , 1996
[ND] Nicola, “Il completamento volumetrico di feature”
[PET95] F.Petta , “Progetto e sviluppo di un interprete di feature funzionali” , Tesi di Laurea in Scienze dell’Informazione , AA. 1994/95
[INTDOC] “3D Interop ACIS STEP Reader/Writer version R11”, available in DOC folder of the software release by Spatial.

Francesco Robbiano
March 24th, 2004