Release Notes for
AIBAR: Angle-Independent Bundle Adjustment Refinement
Version 1.04

Thank you for using the AIBAR refinement software. This file contains important information regarding this release. We strongly recommend that you read the entire document.

CONTENT

Installation

INSTALLATION & SYSTEM REQUIREMENTS

SYSTEM REQUIREMENTS

Currently, AIBAR is only configured to run on Win32 architecture and can only be compiled using the Visual C++ 6.0 compiler.

INSTALLATION

1. UNZIP
   Unzip the file 'AIBAR_*.zip'. Aside from this text file, there is a directory called AIBAR_*. It may be placed anywhere.

2. COMPILE
   Move into the AIBAR_* directory. There should be a folder named 'pkg' and a makefile. Run 'make'. An executable named 'AIBAR' will be generated.

3. RUN
   To run AIBAR from the command line, type

   ```
   aibar [options] in_file out_file
   ```

   Input and output files are REQUIRED. See the table of contents for format specifications.

INSTALLATION TROUBLESHOOTING

1. COMPILE-TIME ISSUE: no compiler

If an error occurs because of a missing 'CL.EXE', the Visual C++ 6.0 compiler is probably not in your path. If you have the compiler installed, in can probably be found in one of the directories

"C:/Program Files/Microsoft Visual Studio/VC98/Bin"
or
"C:/Program Files/Microsoft Visual
Studio/Common/MSDev98/Bin"

2. RUN-TIME ISSUE: syntax error

Syntax errors occur because of the format of the input file. The syntax demanded of *.feat files is very strict, and any significant deviations result in a syntax error. For format of *.feat files, please see the table of contents.

Command-Line Options (Flags)

COMMAND-LINE OPTIONS (FLAGS)

1. BASIC OPTIONS

-c

INSIDE-LOOKING-OUT IMAGE SEQUENCES,
CONTIGUOUS PARTITIONING
Cannot be used with -i.

-i

OUTSIDE-LOOKING-IN IMAGE SEQUENCES,
INTERLEAVED PARTITIONING
Cannot be used with -c.

-e NUM

ERROR or NOISE INTRODUCTION
Introduces absolute (non-percentage-based) error into the best guess 3-D reconstruction provided by the user.

2. ADVANCED OPTIONS

-s NUM

SIZE of IMAGE GROUPS
(drives grouping)
Images will be grouped into groups of NUM size.

-f NUM

MAX FEATURES PER GROUP
(drives grouping)
Necessitates that at most NUM features be associated with each image group.

This exists simply to prevent one group from taking all the features, for if one group uses all the features, we still reconstruct 3-d points in O(N^2) time, thus partitioning was pointless.

-l NUM

MINIMUM FEATURE LIFE
(for culling)
NUM of consecutive images in a group that must see a feature in order for that feature to be associated with that image group.

-m NUM

NUMBER OF TIMES TO RUN and MERGE
(for extra refinement)
Runs AIBAR once, then doubles the group size and runs AIBAR again... a total of NUM times. Used for extra refinement of data sets. Default is 1.

3. COMPARE FUNCTIONALITY

-d TRUTH_FILE FILE_TO_TEST DIFF_FILE

COMPARE RECONSTRUCTED OBJECTS
Assuming two input files are of *.feat format and contain reconstructed objects (3-D points), the difference in location of each 3-D point is output to the DIFF_FILE. It is assumed TRUTH_FILE will be used as the ground truth
(so DIFF_FILE = TRUTH_FILE - FILE_TO_TEST)

Note that by using this flag, the AIBAR executable does not do its regular task of refining datasets. This is merely extra functionality provided for convenience.

*.feat Files

INTRODUCTION

How to refine scenes using Angle-Independent Bundle Adjustment Refinement

Since camera orientation has been removed from our bundle-adjustment formulation, the only information necessary to refine a scene is the camera position, the scene points and a best-guess 3-D reconstruction (and naturally some of the camera's properties). Provided below is a description of how to organize this data into a format digestible by the AIBAR executable.

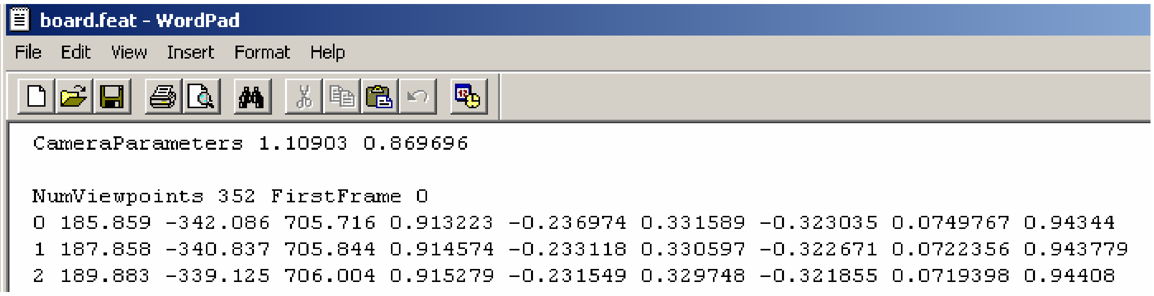
---

*.FEAT FILE FORMAT

1. Camera Parameters
2. Camera/Viewpoint Locations
3. Features List and Locations
4. 3-D Reconstructions or Objects

Sample *.feat files are available for download online.
In *.feat files, there are four sections, each separated by an empty line.

- **Camera Parameters**



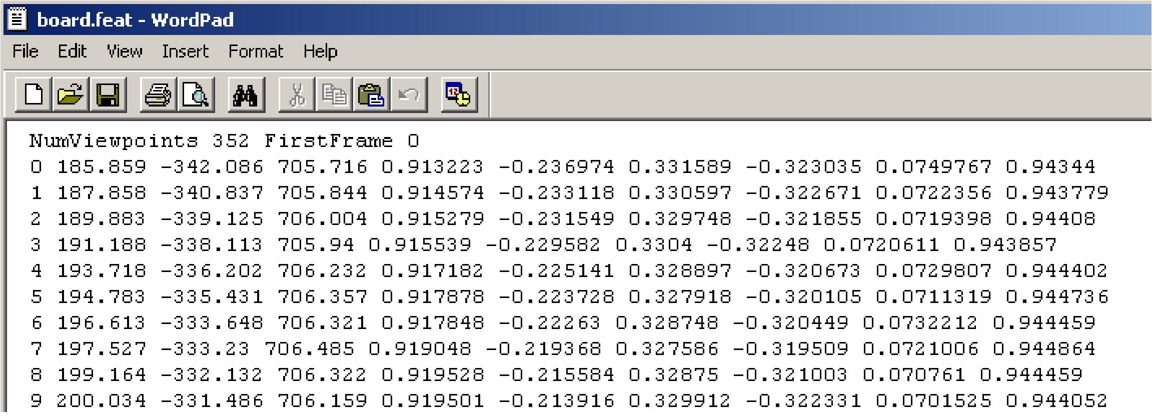
```
CameraParameters 1.10903 0.869696

NumViewpoints 352 FirstFrame 0
0 185.859 -342.086 705.716 0.913223 -0.236974 0.331589 -0.323035 0.0749767 0.94344
1 187.858 -340.837 705.844 0.914574 -0.233118 0.330597 -0.322671 0.0722356 0.943779
2 189.883 -339.125 706.004 0.915279 -0.231549 0.329748 -0.321855 0.0719398 0.94408
```

Format:
"CameraParameters "   XANGLE   YANGLE

-XANGLE and YANGLE are both floats detailing the camera's viewing field in x and y direction.

-It is assumed the camera has focus of length 1.

- **Camera/Viewpoint Locations**



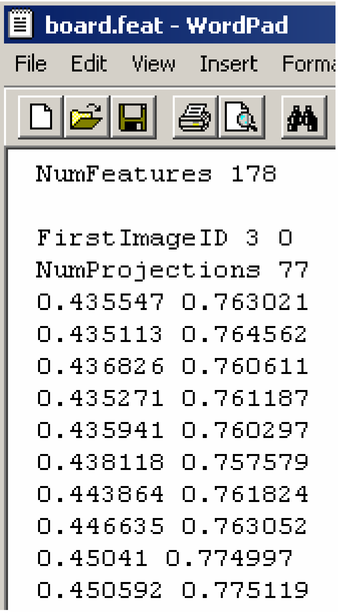
```
NumViewpoints 352 FirstFrame 0
0 185.859 -342.086 705.716 0.913223 -0.236974 0.331589 -0.323035 0.0749767 0.94344
1 187.858 -340.837 705.844 0.914574 -0.233118 0.330597 -0.322671 0.0722356 0.943779
2 189.883 -339.125 706.004 0.915279 -0.231549 0.329748 -0.321855 0.0719398 0.94408
3 191.188 -338.113 705.94 0.915539 -0.229582 0.3304 -0.32248 0.0720611 0.943857
4 193.718 -336.202 706.232 0.917182 -0.225141 0.328897 -0.320673 0.0729807 0.944402
5 194.783 -335.431 706.357 0.917878 -0.223728 0.327918 -0.320105 0.0711319 0.944736
6 196.613 -333.648 706.321 0.917848 -0.22263 0.328748 -0.320449 0.0732212 0.944459
7 197.527 -333.23 706.485 0.919048 -0.219368 0.327586 -0.319509 0.0721006 0.944864
8 199.164 -332.132 706.322 0.919528 -0.215584 0.32875 -0.321003 0.070761 0.944459
9 200.034 -331.486 706.159 0.919501 -0.213916 0.329912 -0.322331 0.0701525 0.944052
```

Format:
“NumViewpoints ” NUMVIEWPOINTS “ FirstFrame ” FIRSTFRAME
-NUMVIEWPOINTS is the number of camera positions (number of images)
-FIRSTFRAME is the first camera position seen (usually 0)

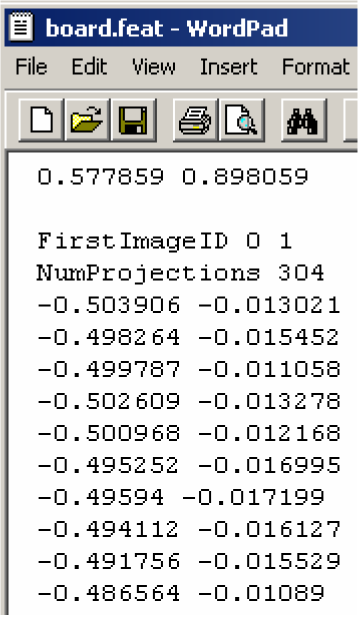
for each viewpoint (so a total of NUMVIEWPOINTS times)
{
in one line:
“position# viewLocX viewLocY viewLocZ viewDirX viewDirY viewDirZ upDirX upDirY upDirZ”

-'position#' is '0' for the first viewpoint, '1' for the second... 'n-1' for the nth viewpoint
}

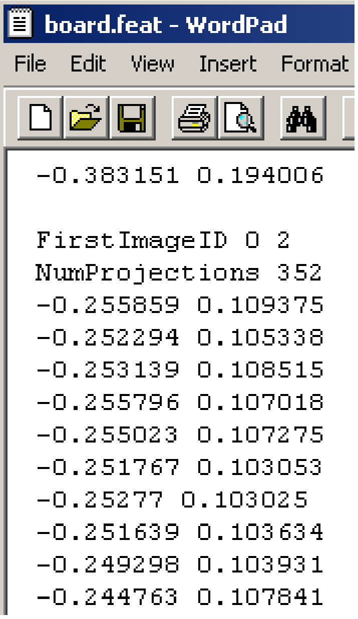
- **Feature List**



```
NumFeatures 178

FirstImageID 3 0
NumProjections 77
0.435547 0.763021
0.435113 0.764562
0.436826 0.760611
0.435271 0.761187
0.435941 0.760297
0.438118 0.757579
0.443864 0.761824
0.446635 0.763052
0.45041 0.774997
0.450592 0.775119
```



```
0.577859 0.898059

FirstImageID 0 1
NumProjections 304
-0.503906 -0.013021
-0.498264 -0.015452
-0.499787 -0.011058
-0.502609 -0.013278
-0.500968 -0.012168
-0.495252 -0.016995
-0.49594 -0.017199
-0.494112 -0.016127
-0.491756 -0.015529
-0.486564 -0.01089
```



```
-0.383151 0.194006

FirstImageID 0 2
NumProjections 352
-0.255859 0.109375
-0.252294 0.105338
-0.253139 0.108515
-0.255796 0.107018
-0.255023 0.107275
-0.251767 0.103053
-0.25277 0.103025
-0.251639 0.103634
-0.249298 0.103931
-0.244763 0.107841
```

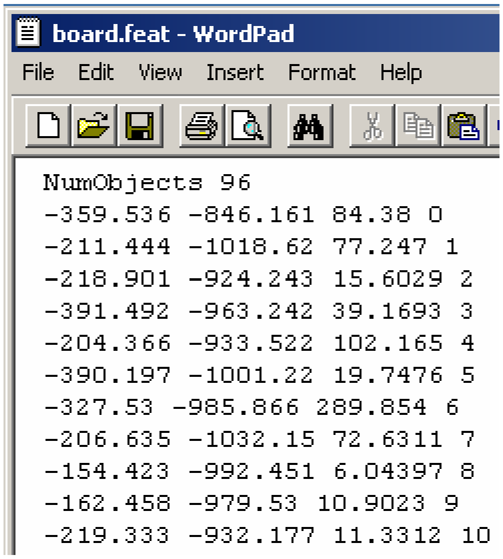
Format:
“NumFeatures” NUMFEATURES
-NUMFEATURES is the total number of features observed

for each feature (so a total of NUMFEATURES times)
{
“FirstImageID” FIRSTIMAGEID feature#
-firstImageID is the position# of the first viewpoint location that sees this feature
-‘feature#' is '0' for the first feature, '1' for the second... 'n-1' for the nth feature

“NumProjections” NUMPROJECTIONS
for each projection (so a total of NUMPROJECTIONS times)
{
X-coord, Y-coord
}
}

- **3-D Reconstructions (or objects)**

  Note: This is necessary for input files.



```
NumObjects 96
-359.536 -846.161 84.38 0
-211.444 -1018.62 77.247 1
-218.901 -924.243 15.6029 2
-391.492 -963.242 39.1693 3
-204.366 -933.522 102.165 4
-390.197 -1001.22 19.7476 5
-327.53 -985.866 289.854 6
-206.635 -1032.15 72.6311 7
-154.423 -992.451 6.04397 8
-162.458 -979.53 10.9023 9
-219.333 -932.177 11.3312 10
```

Format:

"NumObjects" NUMOBJECTS

-NUMOBJECTS is the total number of objects, or reconstructed 3-D features (NUMOBJECTS should be equal to the number of features)

for each object (so a total of NUMOBJECTS times)

{

X-coord Y-coord Z-coord object#

- object# is '0' for the first object, '1' for the object... 'n-1' for the nth feature

}

PARTITIONING MECHANISMS

INTRODUCTION

AIBAR uses a new bundle-adjustment formulation which exhibits noticeably better numerical behavior at the expense of an increased computational cost. For instance, regular bundle adjustment runs in time O(J*N), and AIBAR runs in time O(J*N^2), where J is the number of images and N is the number of features.

To alleviate the cost of the 'N^2' above, we break the data down into several smaller subsets and run AIBAR on those subsets. For instance, if N=4, then N^2 will be 16, but doing two groups of N/2 will result in 2^2 + 2^2 = 8, thus cutting the time in half. In other words, we compensate for our increased computational time by partitioning features and images into subsets and running AIBAR on those subsets.

There are two types of partitioning mechanisms in use: contiguous partitioning (for inside-looking-out sequences) and interleaved partitioning (for outside-looking-in sequences. Explanations of each mechanism and the command line flags it uses are below.

Note: AIBAR uses contiguous partitioning by default.

CONTIGUOUS PARTITIONING MECHANISM

In contiguous partitioning, image groups consist of adjacent images. For instance, if images are grouped into groups of size 6, the first group would consist of images 0-5, the second group images 6-11, the third 12-17, etc.

INTERLEAVED PARTITIONING MECHANISM

In interleaved partitioning, image groups consist of evenly spaced images that span the entire original image sequence. An example might help. If group size is 5 and there are 100 images, then we will have groups like
{1, 21, 41, 61, 81}, {2, 22, 42, 62, 82}, {3, 23, 43, 63, 83}