The Argus Runtime Environment

Guidelines on Installation and Use

Stefan Aarninkhof, WL | Delft Hydraulics
Kenneth Kingston, University of Plymouth

Merged panoramic view, Olympic Harbour Barcelona (Mapfre station), Spain

Table of Contents

1 Introduction 3
2 General Argus conventions 4
  2.1 Argus co-ordinate system 4
  2.2 Argus time conventions 4
  2.3 Site, user & platform dependent settings 5
3 The Argus Runtime Environment (ARE) 6
  3.1 Initialization of the ARE 6
  3.2 Data sources Argus Runtime Environment 6
    3.2.1 Argus archive of video data 6
    3.2.2 Argus data base of meta information 6
    3.2.3 External field data 7
  3.3 The Argus Analysis Tools 8
    3.3.1 DataBase Organizer (DBO) 9
    3.3.2 geomtool2002 (FG) 9
    3.3.3 Argus Merge Tool (AMT) 11
    3.3.4 Argus Stack Tool (AST) 13
    3.3.5 Intertidal Beach Mapper (IBM) 13
4 Advanced Argus programming 16
  4.1 Argus time processing 16
  4.2 Accessing argusDB information 16
  4.3 Argus image processing 17
  4.4 Accessing external fielddata 18
  4.5 Argus pixel processing 19
5 References 20

Appendix A: Setting up your Argus environment
Appendix B: Input file Argus Merge Tool
Appendix C: DEM on the basis of shorelines
Appendix D: Manual Argus PIXel Toolbox

1 Introduction

This note is meant for use as a starting point to explore the Argus Runtime Environment (ARE). Sensible and efficient analysis of Argus video images demands various data sources to come together: video data, Argus meta information (geometry solutions, etc) and supporting field data (see Fig. 1.1). The ARE provides a framework for that, by offering a set of licensed, Matlab-based Argus analysis tools, which enable the user to quantitatively interpret video data, to perform post-processing on the raw data (like image merging) and to derive meaningful information (like shorelines) from the images.

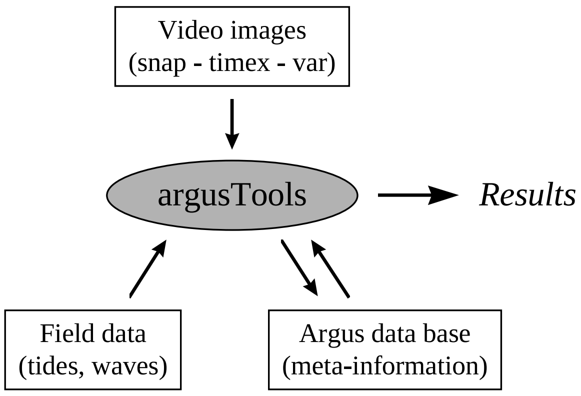


Fig. 1: *Layout of the Argus runtime environment (ARE)*

This note provides further background on each of the components of the ARE. Furthermore, it briefly discusses the use of the various argusTools and it gives an overview of Argus core routines, which are indispensable for sophisticated Argus programming. The information given here is based on the database format the Argus partners agreed on during the 4th overall Argus workshop, organized at Oregon State University in August 2001. Rob Holman (Oregon State University, USA), John Stanley (Oregon State University, USA) and Nathaniel Plant (Naval Research Lab, MS, USA) are gratefully acknowledged for their massive contribution to arrive at the operational level of Argus processing where we are today.

This manual is meant for use with the ARE version 2002b (release August 2002). It was written within the framework of the EU-sponsored COASTVIEW project under contract number EVK3-CT-2001-00054. Activities for maintenance and further development of the ARE receive co-funding from Delft Hydraulics, Plymouth University and the Dutch Ministry of Public Works Rijkswaterstaat.

2 General Argus conventions

2.1 Argus co-ordinate system

At every Argus site, the orientation of the x-axis is shorenormal, with the positive x-axis pointing in seaward direction. The y-axis is directed perpendicular to the x-axis, such that the co-ordinate system thus obtained is positive in mathematical sense. The latter means that the rotation from the x-axis towards the y-axis indicates the counter-clockwise (or 'positive') turning direction. As an example, Fig. 2.1 shows the Argus co-ordinate system for Miyazaki, Japan:

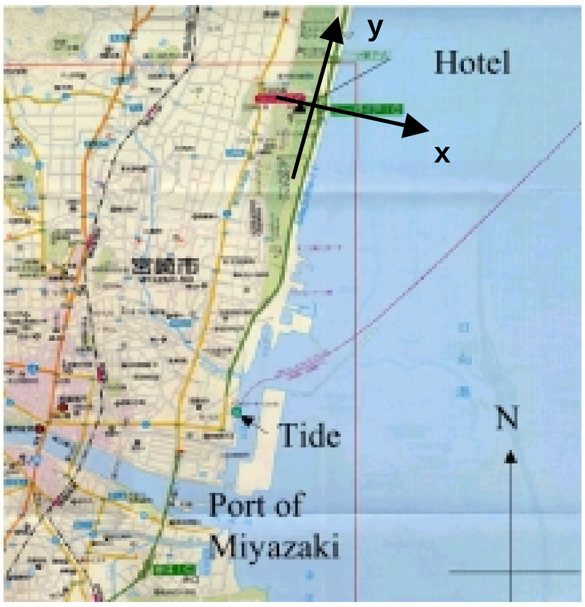


Fig. 2.1: *Argus co-ordinate system at Miyazaki, Japan*

The vertical reference level (z = 0) is generally set to match the mean tidal level or a commonly used (often national) ordinance level (like NAP in the Netherlands).

2.2 Argus time conventions

Three different time frames are used within the ARE:

- Epochtime: represents the number of seconds since January 1, 1970, 00:00:00. This is the nine or ten-digit number an Argus image filename begins with. By nature, epochtimes are relative to GMT.
- Julian days: represents the serial number of a day within a year, e.g. Feb 1st = 32.
- Matlab's 'datenum' time frame: represents the number of days since January 1, 0000, 00:00:00. Matlab times are obtained from the standard Matlab function *datenum* (see help on datenum). Matlab provides a set of standard functions to convert datenumbers to more accessible formats, the most important ones being *datevec* and *datestr*. See the Matlab help on timefun for further details regarding Matlab time processing.

Generally, it is recommended to do any data processing in epochtime or Matlab's 'datenum' time frame. This avoids any troubles with interpolation across different years, leap year cycles etc. To convert between the different formats, the ARE provides a set of standard conversion routines. Reference is made to Chapter 4 for further details on the routines available.

Important note:

Notice that all Argus processing is performed in GMT times. For instance, geometries are selected on the basis of their 'whenValid time', which is in GMT, and fielddata need to be defined against GMT time. Local times show up in only 2 situations (of relatively minor importance), (1) in the non-used readable part of the image filename (see below) and (2) in the whenDone field of the geometry solution.

2.3 Site, user & platform dependent settings

Site, user & platform dependent settings are specified in the file *argusOpt.m*. Reference is made to Appendix A for a detailed description of the contents of argusOpt.

3 The Argus Runtime Environment (ARE)

This chapter provides further background on the components of the ARE (Section 3.2), as well as an introduction on the use of the argusTools (Section 3.3). In anticipation of that, Section 3.1 discusses the initialization of the ARE.

3.1 Initialization of the ARE

In anticipation of the analysis of Argus video data, the ARE needs to be initialized. Initialization of the ARE involves two aspects:

1. Identification of the Argus meta-database of interest. This implies that the ARE reads the appropriate station, camera, gcp, geometry (etc) information from file.
2. Inclusion of all appropriate Matlab paths to the directories, which contain the argusTools source code.

Running the function *argusInit*, which is located in the argusRoot (cf. Appendix A), initializes the ARE. Make sure that argusInit.m is on your Matlabpath! The function argusInit requires a single input argument, which represents the name of the database of interest (eg. *argusInit('delft'), argusInit('plymouth'),* etc.). See Appendix A for an overview of the demands regarding the naming of Argus databases.

3.2 Data sources Argus Runtime Environment

3.2.1 Argus archive of video data

Ideally, the ARE reads the Argus video data directly from the Argus image archive. If impossible (for technical reasons), a local copy of the image archive is to be created. Notice the following aspects:

- The directory structure of the local image archive has to be an *exact copy* of the image archive structure designed by OSU. This implies that the ARE expects the following directory layout: imageRoot\site\year\camera\julianDay\imagename
  for example: d:\argusArchive\egmond\1998\c1\293_Oct.20\908878700.Tue.Oct.20_10_18_20.GMT.1998.egmond.c1.timex.jpg
- de imageRoot is to be defined in the function *argusOpt*. In the example given above, the imageRoot is 'd:\argusArchive'.
- Generally, the imageRoot is named 'argusImages', 'argusArchive', 'imageArchive', etc.
- If the ARE is used in a unix environment, all backward slashes mentioned here are to be replaced by forward slashes. Or even better: use Matlab's *filesep* command as much as possible!
- Image filenames are requested to have the long Argus names as cited above. Based on the information given in the filename, the ARE traces the associated Argus meta information including the most recent geometry solution. Notice that only the epochtime number is used for time referencing! The time string given in the middle is just meant to facilitate the Argus user who may otherwise have a hard time interpreting epoch numbers.

3.2.2 Argus data base of meta information

The Argus data base of meta information (hereafter referred to as argusDB) is a key-component of the ARE. Without argusDB, no quantitative interpretation of image features would be possible. Notice the following aspects:

- The argusDB contains all information needed to quantitatively interpret Argus video images. This concerns the characteristics of the local site, video station, image processor, cameras applied, the available ground control points and geometry solutions, etc.
- argusDB can be viewed (and edited – which may be dangerous!) with the help of DBOrganizer
- All meta information is accessible on the basis of the Argus image filename (that is: epochtime, name of the station and camera number).
- The different components of argusDB are all inter-connected. For instance, on the basis of the site identifier, all video stations available at that site can found. On the basis of the station identifier, all cameras of that station can be found. On the basis of a camera identifier, all geometry solutions available for that camera can be found. Etc. Using the epochtime of interest as an additional constraint yields a unique geometry solution, etc.
- An argusDB is initialized through the function *argusInit*. Example calls are argusInit('delft'), argusInit('plymouth'), etc. See the section above on the initialization of the ARE.
- See Appendix A for an overview of the directory structure of the argusDB's and the contents of the contents of the binary Matlab files found there.

3.2.3 External field data

Many techniques to derive information from video imagery require the availability of external field data, like the tidal level and/or the wave conditions. The present version of the ARE provides easy (fully-automated) access to external data sources, be it that the degree of flexibility is rather limited yet. It is important to understand that only the meta information on the fielddata is included in the argusDB. The fielddata itself are stored in separate data files (ascii format). Reading fielddata from file, the ARE follows a three-step approach:

1. On the basis of a single time or a longer period of interest, the ARE determines from the meta information in the argusDB what fielddata are available
2. Only the files containing fielddata for the time or period of interest are read from file. Initially, only 1st preference data (see below) are taken into account.
3. To find the data at the time or period of interest, the raw field data are interpolated.
4. If data gaps exist, an attempt is made to fill these in with data obtained from the 2nd preference source (see below), following a similar procedure.

Notice the following aspects:

- The ARE presently accounts for tidal levels, wave conditions (wave height, period and direction) and meteorological conditions (atmospheric pressure, wind)
- Tide, wave and meteorological data are located in separate directories under the fieldDataRoot (which is defined in argusOpt.m). The sub-directories under fieldDataRoot have to be named 'Tide', 'Wave' and 'Meteorological'. Field data are provided as text fiels, which obey a fixed format that cannot be changed.
    - In the case of tide data, this consists of a [Nx7] numeric matrix, which columnwise represents the years, months, days, hours, minutes, seconds and the tidal elevation in m.
    - In the of wave information, the data files consists of a [Nx{6+*n*}] numeric matrix, columnwise representing years, months, days, hours, minutes, seconds, and *n* wave parameters. The set of wave parameters can consist of as many parameters as are required. A recommended set of wave parameters consists of the *rms* wave height H_{rms} (in meters), the peak period T_{peak} (in seconds) and the wave angle of incidence (degrees) with respect to the north (positive angles in clockwise direction).

  - In the case of meteorological information, the data files consists of a [Nx{6+*n*}] numeric matrix, columnwise representing years, months, days, hours, minutes, seconds, and *n* meteorological parameters. The set of meteorological parameters can consist of as many parameters as are required. A recommended set of meteorological parameters consists of the atmospheric pressure ρ_{atm} (mB), wind speed W_{speed} (m/s) and the Wind Direction W_{dir} (degrees) with respect to the north (positive angles in clockwise direction).
- To speed up ARE processing, ascii data files can be converted into binary Matlab files. If available, ARE uses the mat files. Notice that the mat files use the epochtime time frame rather than the 6 column time definition given in Ascii files. Standard routins to convert ascii files into mat files are euroTideAscii2Mat, euroWave Ascii2Mat and euroMetAscii2Mat.
- Since all Argus processing occurs in GMT time, field data have to be given against GMT times. Correct for a possible time zone offset before adding new fielddata to the ARE!
- Each data file is represented in argusDB by a Matlab structure which summarizes meta-information on that file (start time, end time, columnorder, etc). On the basis of this meta information, the ARE determines the availability of field data for a specific time or period of interest.
- Thus, adding new fielddata to the ARE also means that the Matlab structures of meta information need to be updated (otherwise, the ARE will not find the new data). Being an integral part of the Argus database of meta-information, DBO ('Data Base Organizer', see below) provides the functionality to do so.
- The ARE enables the user to specify a 1st preference as well as a 2nd preference data source for each Argus site. The 1st preference source of tide data is often a locally measured water level, whereas the 2nd preference source may be an astronomically predicted level. The preference of wave data sources generally decreases with increasing distance from the Argus site of interest. The preference of each data source is also stored as part of argusDB.
- External tide, wave and meteorological data can be read from the argusDB through the routines *DBGetEuroTide*, *DBGetEuroWave,* and *DBGetEuroMet,* respectively. See the help on those routines for further details.

3.3 The Argus Analysis Tools

At present, six applications are part of the standard set of Argus analysis tools. These are summarized in Table 3.1:

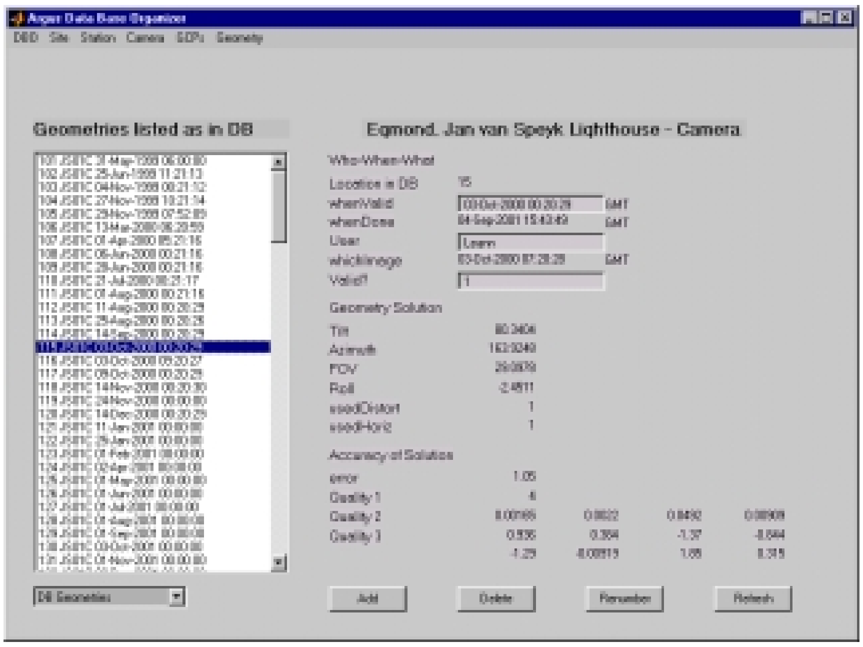
No.	Application	Call	Which means:
1	DBOrganizer	DBO	Date Base Organizer
2	geomtool2002	FG	'Find Geometry'
3	ArgusMergeTool	AMT	Argus Merge Tool
4	ArgusStackTool	AST	Argus Stack Tool
5	IBMapper	IBM	Intertidal Beach Mapper
6	ArgusDesignTool	ADT	Argus Design Tool

Table 3.1: *Overview of standard Argus Analysis Tools*

In the next few sub-sections, detailed comments & guidelines are given on each of six applications mentioned above.

3.3.1 DataBase Organizer (DBO)

Initially, DBO was developed to view existing geometry solutions and – if necessary – to remove erroneously stored geometries from the argusDB. Later on, DBO was extended to operate on other database elements as well: sites, stations, cameras & GCP's. Allthough DBO allows for the modification of each of these structures, it is strongly recommended not to change any field of the site, station, camera or GCP structure, since this may strongly violate the functioning of the ARE.


Argus Data Base Organizer
DBO Site Station Camera GCPs Geometry
Add
Delete
Renumber
Refresh
DB Geometries

Geometries listed as in DB

```
101 JS81C 31-May-1999 06:00:00
102 JS81C 25-Jun-1999 11:21:13
103 JS81C 04-Nov-1999 00:21:12
104 JS81C 27-Nov-1999 10:21:14
105 JS81C 29-Nov-1999 07:52:09
106 JS81C 13-Mar-2000 06:20:59
107 JS81C 01-Apr-2000 05:21:16
108 JS81C 06-Jun-2000 00:21:16
109 JS81C 28-Jun-2000 00:21:16
110 JS81C 21-Jul-2000 00:21:17
111 JS81C 01-Aug-2000 00:21:16
112 JS81C 11-Aug-2000 00:20:29
113 JS81C 25-Aug-2000 00:20:28
114 JS81C 14-Sep-2000 00:20:29
115 JS81C 03-Oct-2000 00:20:29
116 JS81C 03-Oct-2000 09:20:27
117 JS81C 08-Oct-2000 00:20:29
118 JS81C 14-Nov-2000 00:20:30
119 JS81C 24-Nov-2000 00:00:00
120 JS81C 14-Dec-2000 00:20:29
121 JS81C 11-Jan-2001 00:00:00
122 JS81C 29-Jan-2001 00:00:00
123 JS81C 01-Feb-2001 00:00:00
124 JS81C 02-Apr-2001 00:00:00
125 JS81C 01-May-2001 00:00:00
126 JS81C 01-Jun-2001 00:00:00
127 JS81C 01-Jul-2001 00:00:00
128 JS81C 01-Aug-2001 00:00:00
129 JS81C 01-Sep-2001 00:00:00
130 JS81C 03-Oct-2001 00:00:00
131 JS81C 01-Nov-2001 00:00:00
```

Egmond. Jan van Speyk Lighthouse - Camera

```
Who-When-What
Location in DB     15
whenValid          03-Oct-2000 00:20:29   GMT
whenDone           04-Sep-2001 15:43:49   GMT
User               Learn
whichImage         03-Oct-2000 07:28:29   GMT
Valid?             1

Geometry Solution
Tilt               80.3404
Azimuth            163.0240
FOV                26.0879
Roll               -2.4811
usedDistort        1
usedHoriz          1

Accuracy of Solution
error              1.05
Quality 1          4
Quality 2          0.00165    0.0022     0.0492     0.00909
Quality 3          0.936      0.384      -1.37      -8.044
                   -1.29      -0.00915   1.05       0.315
```

Fig. 3.1: *User Interface of the Argus DataBase Organizer (DBO)*

The relevant functionality of DBO can be described the next listing:

1. Make sure the ARE has been initialized
2. Start DBO by typing DBO at the Matlab prompt. DBO automatically loads the entire current Argus database (that is sites, stations, geometries, eurotides, etc).
3. View the information available. You can swap between data elements with the help of the 'Select DB' button in the lower left corner. Some of the elements provide sorting functionality to facilitate access to the data (eg. geometries can be sorted by camera, whenDone and whenValid).
4. Modify the data base as requested. This generally concerns the removal of erroneous geometry solutions, modification of the whenValid field of geometry solutions or the inclusion of new eurotide/eurowave structures. Notice that the date format for the whenValid field is very strict!
5. Once again: It is strongly recommended not to change any field of the site, station, camera or gcp elements!
6. Save the updated database with the help of the DBO | Save option.

3.3.2 geomtool2002 (FG)

Geomtool2002 was originally developed by Nathaniel Plant (Naval Research Lab, MS, OSU), making use of Argus core routines developed by John Stanley at Oregon State University. The application is meant for the generation of new geometry solutions. For a detailed background on image quantification and geometry solutions, reference is made to Lippmann and Holman (1989) and Holland et al (1997).

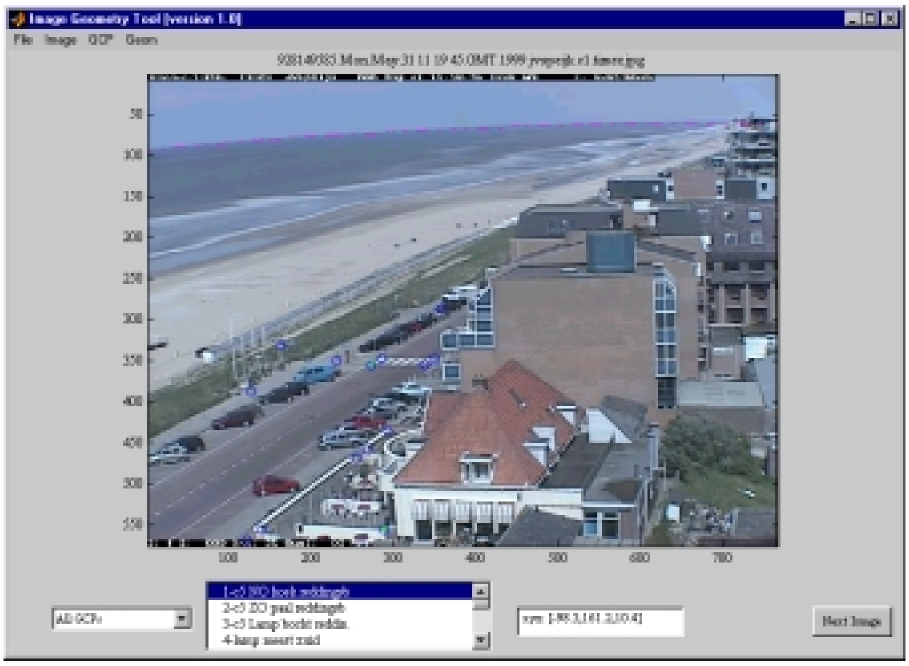


Fig. 3.2: *User Interface of geomtool2002 (FG)*

Use of geometool2002 typically involves the following steps:

1. Load an Argus image for which the most recent geometry solution is no longer valid (menu file | load image)
2. Load the most recent geometry solution (menu geometry | load geom), to get a first order impression of the location of the different GCP's. At the same time, all GCP's available at the site and time of interest are imported.
3. Label the GCP's on the screen (menu image | show labels).
4. Choose a clearly visible GCP, which can be used to determine a new geometry solution. Zoom in somewhat within the area of this GCP (click on left mouse button).
5. Select the GCP of interest from the listing of GCP's (make sure the lower left button is set to 'All GCPs').
6. Pick the location of the GCP on the screen (menu GCP | Pick gcp).
7. Zoom out again (right mouse button)
8. Repeat steps 4-7 for the next GCP. You need at least 2 to be able to determine a geometry solution.
9. A very clear horizon can be used as a GCP as well. To that end, select GCP | horizon gcp from the menu and follow the instructions (click just a few pixels above the horizon, use left mouse button, finish by clicking the right button). Decide whether you accept all horizon gcp's
10. If any of the GCP's does not satisfy, than select the option 'Picked GCPs' from the selection button in the lower left corner. Select the erroneous GCP and click 'Clear UV' in the lower right corner. Do so for each GCP which should not be used for the determination of a geometry solution.
11. Determine the new geometry solution (menu Geometry | solve geom). This is done by following an efficient iteration procedure. If this procedure is successfully (typically after about 30-80 iteration steps), the result is summarized on screen (separate window) and the computed GCP's are shown in the image.

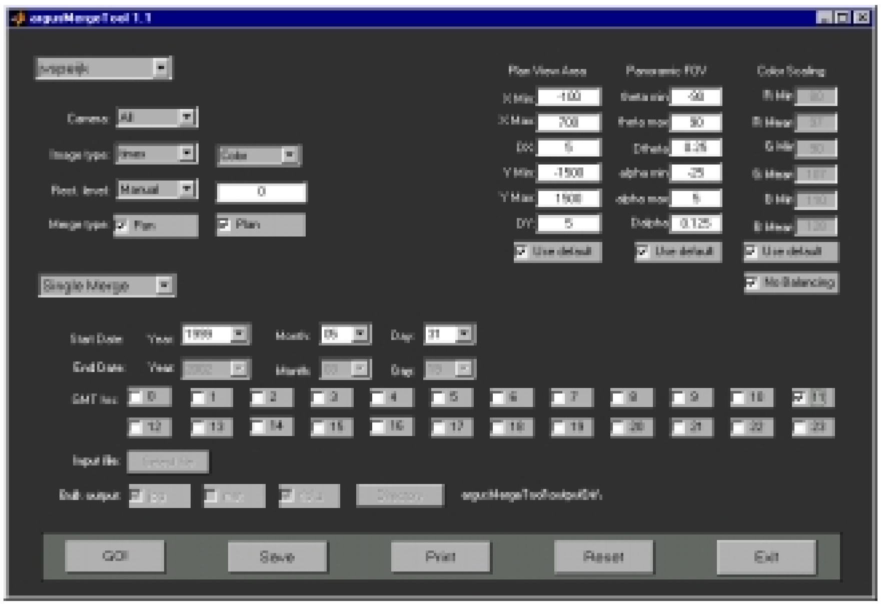
12. Modify the whenValid field? If not, the time in the whenValid field will be set to match the epochtime of the image you are presently working on. However, the situation may occur that the new solution was valid already during a certain time ahead of the present image (e.g. during the morning of the same day). In that case, response with Yes. Geomtool2002 will invite to select an Argus image, starting from which the new geometry solution will be valid.
13. Judge on the quality of the result by comparing the computed location of the GCP's to their real-world location as observed from the underlying image. If you are not satisfied yet, re-determine a geometry solution by excluding some picked GCP's from the analysis or by picking additional GCP's.
14. If you are happy with the result, save the new geometry to file (menu geometry | save geom).
15. Refresh the image (menu image | refresh) and remove all picked GCP's manually (button select DB, picked GCP's, clear UV for each of them). Load the next image
16. Repeat steps 2 – 15.

Notice that geomtool2002 offers some additional functionality which is worth mentioning here:

- Option rectification (menu Image | Plan view). This option invites the user to indicate the 4 corners of an area of interest by clicking the left mouse button. After the fourth point, geomtool will project the image area of interest on the ground plane (a process called 'image rectification' and the present the result in a separate window.
- Option 'Virtual GCP' (menu GCP | virtual gcp). This option allows for the definition of virtual GCP's. This can be very useful if a site provides clearly visible image objects which are however hard to survey in the field (for instance, corners of windows or street lights). For those cases, additional, virtual GCP's can be generated on the basis of an existing geometry solution. Notice that this should be a high-quality solution, to prevent the situation that errors strongly accumulate. Newly created virtual GCP's are immediately added to the argusDB and stored in an update file. It is strongly advised to make only very limited use of this option.
- Image modification. Occasionally, poor image contrast or brightness may obscure the proper identification of GCP's. For those situations, the image menu provides a few options to improve image quality
- Option 'Initial Angles' from the selection button in the lower left corner. With the help of this option, initial values can be defined for each of the 4 angles (azimuth, roll, fov & tilt) that are involved in the iteration process. In that way, the control over this iteration process is being improved. Alternately, one can even fix the value of each of the 4 angles. Fixed values do not participate to the iteration process, hence the number of GCP's needed may decrease (less unknown variables)
- Option 'Pie Pan Pick' (PPP). This option was included to determine the intra-pixel location of a GCP on the basis of center of mass considerations. It is particularly useful at Duck (NC, USA), where large, circular shields were erected to act as a GCP for the Argus station at that site. Besides, it may be helpful when lights are being used as night-time GCP's.

3.3.3 Argus Merge Tool (AMT)

AMT provides a user interface to merge images of multiple cameras into panoramic or plan view images. Also, it can be used to rectify images from a single camera. The tool was specifically developed for the merging of large series of images, to generate plan view movies or for display at an Argus website.


argusMergeTool 1.1

Camera: All

Image type: timex; Color

Rect. level: Manual; 0

Merge type: ✓ Plan; ✓ Plan

Single Merge

Plan View Area: X Min -100; X Max 700; DX 5; Y Min -1500; Y Max 1500; DY 5; ✓ Use default

Panoramic FOV: theta min -90; theta max 90; Dtheta 0.25; alpha min -25; alpha max 5; Dalpha 0.125; ✓ Use default

Color Scaling: R Min; R Mean; G Min; G Mean; B Min; B Mean; ✓ Use default; ✓ No Balancing

Start Date: Year 1999; Month 05; Day 31

End Date: Year; Month; Day

GMT hrs: 0 1 2 3 4 5 6 7 8 9 10 ✓ 11 12 13 14 15 16 17 18 19 20 21 22 23

Input file:

Bulk output: ✓ jpg; mat; ✓ grid; Directory; argusMergeTool\output\Dir\

Fig. 3.3: *User Interface of Argus Merge Tool (AMT)*

The AMT user interface features 3 different sections. The upper left section allows the user to specify the type of merging he is interested in. This involves specification of:

- The site of interest (egmond, nordzee1, miyazaki, argus02a, etc)
- The cameras of interest (‘All’ or a sub-set)
- The input image type (snap, timex, variance or daytimex)
- The output image type (colour or grayscale)
- The rectification level. So far, only the option ‘manual’ is operational
- The type of merging (Panoramic view, or plan view, or both)

On the basis of the site of interest selected here, default settings for the plan view area, panoramic field of view and color scaling are read from file argusMergeTool/gui/getSiteSpecifics and filled in the upper right section of the database. Notice that variation of the spatial (DX, DY) and angular (Dtheta, Dalpha) resolution directly affects the resulting image size. To remove sharp transitions between different cameras, an optional colour balancing can be applied. The method presently implemented is based on the work done by Jack Puleo (Naval Research Lab, MS, USA). It scales image intensities per colour band on the basis of a mean (Rmean, Gmean, Bmean) and offset value (Rmin, Gmin, Bmin). A side effect of the use of colour balancing may be that the image overall looses contrast. Further investigation over a range of Argus sites is needed to arrive at optimum settings for the colour balancing parameters.

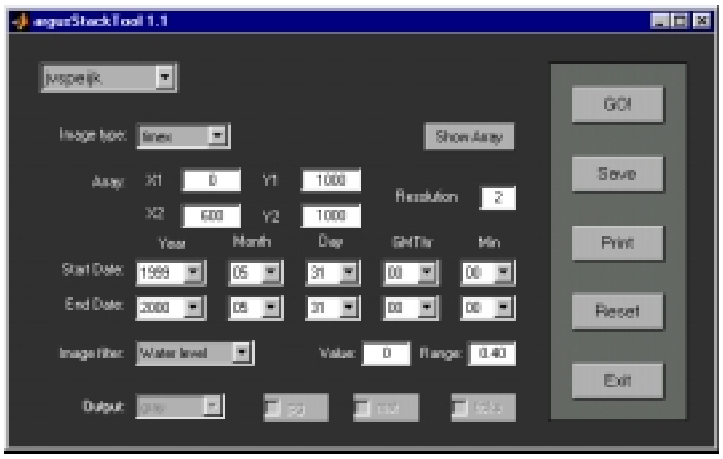
The lower section of the AMT user interface governs which images are to be merged. Merging can be performed in single image mode (output to screen), in bulk mode (output stored on disk) or from file. In the latter case, the user needs to create an input file (text format) which provides a listing of the images to be merged. Example input files are provided in Appendix B. In the case bulk merging, an arbitrary number of images per day can be merged during a longer period of time. In this specific case, the user can specify

- The type of output he prefers (jpg images, or mat files, or both)
- Whether or not to include gridlines & tickmarks at the merged images
- The output directory. This goes by selecting an existing file in the target directory.

NB: Notice that the times specified in Section 3 of the user interface represent GMT times! So, requesting AMT to generate a merge of Miyazaki at September 22, GMT 06:00 hr. will merge images collected at Miyazaki within half an hour of JST 15:00 hr. (local time).

3.3.4 Argus Stack Tool (AST)

AST was designed for the analysis of the morphodynamics of sand bars. To that end, image intensities are sampled along a user-specified array and stacked over time. On the basis of the evolution of the wave breaking patterns over time, the migration of sand bars can be quantified.



jvspeijk

Image type: timex

Array: X1 0 Y1 1000

X2 600 Y2 1000

Resolution 2

	Year	Month	Day	GMThr	Min
Start Date:	1999	05	31	00	00
End Date:	2000	05	31	00	00

Image filter: Water level Value: 0 Range: 0.40

Output: gray

Fig. 3.4: *User Interface of Argus Stack Tool (AST)*

AST requests the user to specify the site of interest, the end points of the array of interest and the period of interest. As before, the period of interest is to be specified in GMT times. With the help of the option 'Show Array', the user can verify whether the array co-ordinates entered do satisfy. Proper use of this application demands that image intensities are sampled at similar tidal levels (cf. Van Enckevort and Ruessink, 2001). To that end, AST applies an image filter, which selects the images that obey some user-specified filter criteria. In the example given in Fig. 3.4, only images which were sampled at a tidal levels between –0.20 m and +0.20 (that is the value '0' plus or minus half the range) qualify for analysis. If the user is interested in storm events only, a similar filter can be applied on the basis of wave height information. The fielddata (water level, wave height) needed to do so are read from the ARE fielddata archive, see Section 3.2.3.

3.3.5 Intertidal Beach Mapper (IBM)

IBM is meant for the interactive mapping of shorelines. The application allows the user to load an Argus image from the video archive, specify a Region of Interest (ROI) and to determine the location of the shoreline within that ROI. For most cameras, a default ROI is suggested as well. The user interface of IBM is shown in Fig. 3.5.

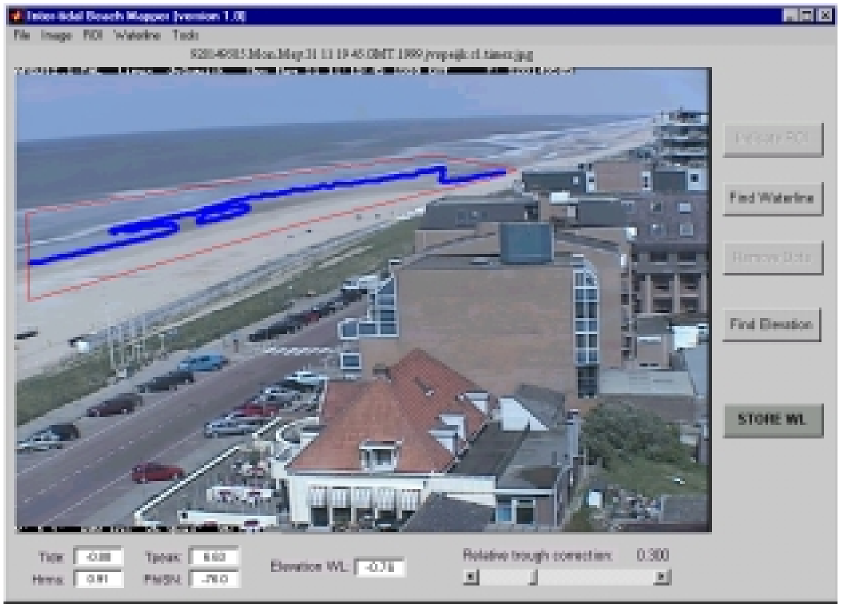

Inter-tidal Beach Mapper (version 1.0)
File Image ROI Waterline Tools
Indicate ROI
Find Waterline
Remove Dots
Find Elevation
STORE WL
Tide
Hrms
Tpeak
Elevation WL
Relative trough correction 0.300

Fig. 3.5: *User Interface of the Intertidal Beach Mapper (IBM)*

After loading an image from the video archive, IBM automatically reads the hydrodynamic conditions from the fielddata archive and the relevant meta-information from the argusDB. A user-specific ROI can be defined by clicking the button ‘Indicate ROI’. Use the left button of the mouse to specify the corners of the ROI, clicking the right mouse button closes the polygon and finishes ROI definition. Hitting the ‘Find Waterline’ button starts the model to determine the shoreline. This is done by clustering the dry and wet pixels within the ROI, in both colour and grayscale space. The shoreline is found at the interface of both clusters. If the clustering appears to be successful, IBM opens a second window showing a 2-dimensional histogram of the two clusters for the criterion (colour or grayscale) that provides the best distinction (Fig. 3.6). Besides, the identified shoreline is also shown in the Argus image in the IBM main window.

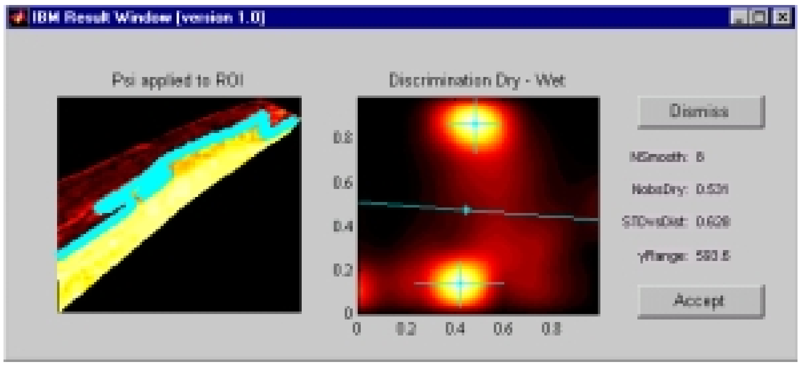


Fig. 3.5: *IBM Result Window, showing the distinction between dry and wet pixels*

On the basis of a visual inspection of the results, the user decides whether to accept the solution provided by the model, or to dismiss it. If the result largely satisfies, apart from a few outliers, the user is advised to select ‘Accept’. After that, the outliers can be removed manually with the help of the ‘Remove Dots’ option. Once the result fully satisfies, it can be stored by hitting the ‘Store WL’ button. Results are saved in the directory argusTools\IBMapper\outputDir, in anticipation of further post-processing. The latter may involve the mapping of intertidal beach bathymetry on the basis of set of waterlines, sampled over a tidal cycle. An example of that is given in Appendix C.

For a detailed background on the model to identify shorelines, reference is made to Aarninkhof and Roelvink (1999). Model accuracy was investigated in Aarninkhof et al (2000). Extension of the IBM functionality by means of the inclusion of relevant post-processing techniques (shoreline evolution over time, mapping intertidal beach bathymetry) is anticipated in the future.

4 Advanced Argus programming

Chapter 4 provides an overview of Argus core routines, that can be applied for advanced Argus programming. It strongly recommended to use these routines as much as possible, since they are well-tested, it strongly facilitates the exchange of code within the Argus users group and – last but not least – it avoids 're-inventing the wheel'. The routines are catogorized in four different clusters: Time processing, Argus database access, image processing, inclusion of fielddata and pixel processing. The functioning of each routine is briefly indicated, for further details reference is made to help on each function.

4.1 Argus time processing

Table 4.1 provides an overview of the Argus core routines for the conversion between different time frames of use within the ARE:

Name	Application
epochtime.m	Returns present epochtime (by definition against GMT time)
epoch2Matlab.m	Converts epochtime to Matlab's datenum format
matlab2Epoch.m	Converts Matlab's datenum format to epochtime
epoch2GMTString.m	Converts epochtime to a string definition wrt GMT
epoch2LocalString.m	Converts epochtime to a string definition wrt local time
julian2Matlab.m	Converts Julian day number to Matlab's datenum format. Accounts for leap year cycle.
matlab2Julian.m	Converts Matlab's datenum format to Julian day number. Accounts for leap year cycle.
argusDay.m	Routine to compose Julian day from epoch time or Argus struct with field 'time'. Overlapping with matlab2Julian.m

Table 4.1: *Overview of Argus routines for time processing*

Associated Matlab functions of interest: now, date, datenum, datestr, datevec

4.2 Accessing argusDB information

Table 4.2 provides an overview of the most important routines to retrieve Argus meta-information from the argusDB:

Name	Application
DBConnect.m	Connect to argusDB (msql only)
getDBname.m	Get name of argusDB presently in use
DBCreateEmptyStruct.m	Create empty argusDB structure
DBGetAllGeometries.m	Get all geometries for a specific station (shortName like nordzee1) and camera number
DBGetAllSites.m	Returns all sites of argusDB presently in use
DBGetCameraByID.m	Get camera structure given cameraID (eg. 'NO01C'). Output includes meta-information on station & lens
DBGetCameraModelByID.m	Get camera model given modelID (eg. 'DC50')
DBGetCamerasByStation.m	Return data about cameras for a specific stationID at a given time
DBGetCurrentGeom.m	Return the most recent geometry solution for a specific cameraID (eg. 'NO01C') at a given time
DBGetGCPBySiteID.m	Return all GCP's at a site for a specific siteID (eg. 'YAQUINA') at a given epochtime
DBGetImageData.m	Returns Argus meta-information (geom, cam, ip, GCPs, usedGCPs, station, site) given an image filename
DBGetLatestGeom.m	Get most recent geometry for a specific station (shortName like 'nordzee1') and camera number
DBGetLensModelByID.m	Get lens model information by lensID (like 'R12MI')
DBGetStationTZOffset.m	Returns time zone offset given station shortName (eg. 'nordzee1')
DBGetStationsByName.m	Get argusDB station information given station shortName (like 'nordzee1') and epochtime of interest
DBGetStationsBySite.m	Get all stations of a site given siteID (eg. NOXXX) or site structure
DBGetUsedGCPS.m	Returns usedGCPs from argusDB given sequential number of geometry solution of interest
DBGetIPFromImage.m	Creates an Argus IP structure for images of non-regular size.
DB2Geometry.m	Converts geom structure with Walton elements A…L stored in separate fields into geom structure with Walton vector m stored in single field
geometry2DB.m	Converts geom structure with Walton vector m stored in single field into geom structure with Walton elements A…L in separate fields

Table 4.2: *Overview of Argus routines for database access*

The latter 2 routines are needed since the msql database requires the 11 elements A…L of the Walton vector to be stored in separate fields, while geomtool needs the [11x1] Walton vector m. Notice furthermore that geometries returned by DBGetCurrentGeom contain a field m rather than the individual elements A…L.

4.3 Argus image processing

Table 4.3 provides an overview of Argus routines that are typically used for image processing:

Name	Application
FTPPath.m	Find imagepath on the basis of image filename
argusFilename.m	Create Argus image filename on the basis of relevant info (time, station, camera, type, format)
findArgusImages.m	Find names of Argus images in video archive. Several limits (closest, localDay) may be applied
grepImgRGB.m	Read Argus image from video archives
parseFilename.m	Interpret Argus image file name
inImage.m	Find out whether UV coords are in the image, or not
imageQuality.m	Determine image quality. Returns good (1) or bad (0)
rgb2mono.m	Convert color image to grayscale
distort.m	Re-apply radial lens distortion to undistorted image co-ordinates
undistort.m	Remove radial lens distortion from distorted image co-ordinates
findUV.m	Find undistorted image coords UV from field coords xyz
findXYZ.m	Find field coords xyz from undistorted image coords UV

Table 4.3: *Overview of standard Argus routines for image processing*

For transformation from image co-ordinates to field co-ordinates (= 'rectification') and vice versa, bear in mind the following scheme:

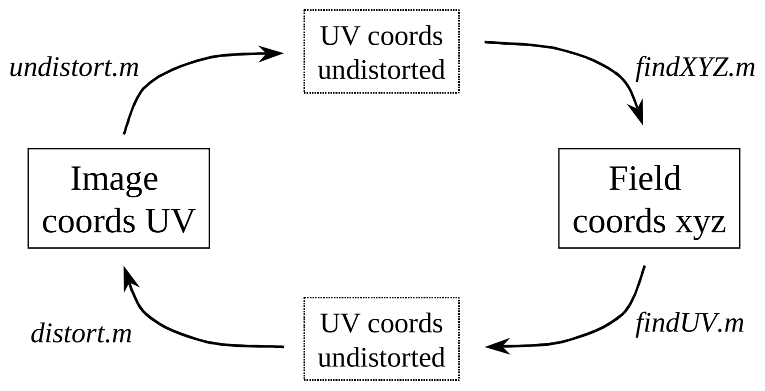


Fig. 4.1: *Image co-ordinates UV ←→ Field coordinates xyz*

Fig. 4.1 illustrates that image co-ordinates UV – being distorted – need to be undistorted first before being converted into field co-ordinates xyz. Oppositely, fieldcoordinates xyz can be converted into undistorted image co-ordinates UV directly. For projection on screen however, they need to be re-distorted.

4.4 Accessing external fielddata

Table 4.4 provides an overview of routines that are presently used to include fielddata information in the ARE:

Name	Application
DBGetEuroTide.m	get tidal elevation, on the basis of an Argus image filename, epochtime of interest or period of interest.
DBGetEuroWave.m	get wave conditions, on the basis of an Argus image filename, epochtime of interest or period of interest.
DBGetEuroMet.m	get meteorological conditions, on the basis of an Argus image filename, epochtime of interest or period of interest.

Table 4.4: *Overview of Argus routines for including field data*

Both DBGetEuroTide.m, DBGetEuroWave.m and DBGetEuroMet.m account for 1st preference and 2nd preference data sources.

4.5 Argus pixel processing

The collection and analysis of pixel time series and time stack images presently attracks considerable attention within the Argus Users Group. The code and tools developed to that end by Oregon State University are in the user-driven stage. Background information on the use of the Argus PIXel Toolbox is provided in a note by John Stanley and Rob Holman, which is included as Appedix D to this manual.

5 References

Aarninkhof, S.G.J. and Roelvink, J.A. (1999). Argus-based monitoring of intertidal beach morphodynamics. Proc. of Coastal Sediments Conf, Long Island (NY), USA, 2429-2444.

Aarninkhof, S.G.J., Caljouw, M. and Stive, M.J.F. (2000). Video-based quantitative assessment of inter-tidal beach variability. Proc. of Int. Conf. on Coastal Engineering, Sydney, Australia

Enckevort, I.M.J. and Ruessink, B.G. (2001). Effect of hydrodynamics and bathymetry on video estimates of nearshore sandbar position. Journal of Geophysical Research, 106(C8): 16969-16980

Holland, K.T., R.A. Holman, T.C. Lippmann, J. Stanley and N.G. Plant (1997). Practical use of video imagery in nearshore oceanographic field studies, IEEE Journal of oceanic engineering, Vol. 22, No. 1.

Lippmann, T.C., and R.A. Holman, Quantification of sand bar morphology: A video technique based on wave dissipation, Journal of Geophysical Research, 94 (C1), 995-1011, 1989.

Appendix A: Setting up your Argus Runtime Environment

This appendix discusses some issues, which are of relevance for setting up your personal Argus Runtime Environment (ARE).

Argus Runtime Environment: Directory structure & contents

Supply of the ARE by WL | delft hydraulics involves the provision of licensed Argus analysis software on a cd or through the Internet. The supply covers all components of the ARE, except for the Argus image archives, which are assumed to be available already. However, a subset of Argus images sampled at different sites world-wide can be delivered upon request. The ARE is supplied in a single directory named ‘argus’, which has the following contents:

- A directory ‘argusDB’, which contains all core routines that are used to access the Argus database of meta-information. These routines are supposed to be used by advanced Argus programmers.
- A directory ‘argusDB.matlab’, which contains the Argus databases of meta-information in Matlab format. The databases are organized in separate directories, which are typically called ‘delft’, ‘osu’, ‘plymouth’ etc. Each database directory contains a file named DB.XXXX.base.0.mat, which provides information on at least one site, station, image processor, camera model and lens model. Information on GCP’s, geometries, usedGCP’s and eurotide/eurowave is stored in separate files for each site. The database file of interest is identified on the basis of the siteID, which is EGXXX for Egmond, DUXXX for Duck, etc. The database is organized such that is allows for a merging with Oregon State University or any other Argus database.
- A directory ‘argusTools’, which contains the various Argus analysis tools discussed in this manual. Each application is located in a different subdirectory.
- A directory ‘CILTools’, which contains another set of Argus core routines that were originally developed at the Coastal Imaging Lab, Oregon State University. Many of these routines is used for Argus time processing and the interpretation of image filenames. None of these routines however is used for argusDB access (see above).
- A directory ‘fielddata’, which contains both the fielddata archive files (subdirectories tide & wave) as well the Matlab routines that are used to access the information (subdirectory code).
- A directory ‘local’, which contains user-specific core routines. Do not remove any of the routines initially provided, since they are used at many occasions. Argus users are invited to add their own core routines to this directory.
- Two local settings files named ‘argusInit.m’ and ‘argusOpt.m’. These files are needed to initalize the ARE and to define the location/user/platform settings. During the installation of your ARE, both need manual editing (see below). The two files are located at the same level as all directories mentioned above. For reasons of uniformity, it is recommended to leave them there.

Besides the directories mentioned here, your ARE may contain a directory argusDB.msql. This directory is only relevant for use of the ARE at Oregon State University.

Setting your Argus Runtime Environment

The ARE is installed by copying the rootdirectory ‘argus’ to your local harddisk. Before being able to use your ARE, the Matlab files argusInit and argusOpt need to be edit manually. For both files, the lines that (may) need manual editing are located in a separate section, immediately at the beginning of the file.

In the file *argusInit.m*, the name of your local argusDB is defined (including the path). The following restrictions apply:

- Your local argusDB has to be a subdirectory of the directory argusDB.matlab, at the same level as the databases ‘delft’, ‘osu’, ‘plymouth’ etc.
- The name of your argusDB has to match the name of the directory (cf. ‘delft’, ‘osu’, etc).

In the file argusOpt.m, the following local settings are defined

- imageArchive.‘argusDB’. This variable represents a pathname which refers to the root of your local Argus image archive for a specific ‘argusDB’. Notice that ‘argusDB’ should exactly match with one of the databases that were defined in the routine *argusInit* (cf. ‘delft’, ‘osu’, etc.). At the level of this imageroot, we find the image archives of one or more Argus stations (hence directories named ‘argus02a’, ‘egmond’, ‘miyazaki’ etc), which obey a standard structure (cf. Section 3.2.1).
- fielddataRoot. Pathname to the directory which contain the fielddata archives (subdirectories ‘Tide’, ‘Wave’ and ‘Meteorological’) as well as the Matlab code for access.
- timeOffset & timeZoneString. Time offset in minutes with respect to GMT of the location where the ARE user lives (so Oregon = -480, Japan = +540). Also the associated timeZoneString (like MET = Middle European Time, JST = Japanese Standard Time, etc.) needs to be given here.
- username. Name of the ARE user. In this way you can be credited for perfect geometry solutions.

After assigning appropriate values to the variables mentioned here, no further modifications are needed and your ARE is ready for use.

<u>Initialization of your Argus Runtime Environment</u>

Only after running the initialization file argusInit.m, your ARE is ready for use. The function argusInit has to be called with a single input argument (character string), which represents the name of the argusDB that has to be initialized. For instance:

```
argusInit(‘delft’);
```

Only argusDB’s that are specified in the edit section of argusInit.m can be initialized. Besides, argusInit checks whether your ARE license is still valid. After running argusInit, the argusDB of interest is initialized (reported back on screen) and the directories which contain the various Argus core routines and Argus analysis tools are added to your Matlab path. Notice that

- argusInit can only be run if it is located at your Matlab path. Use the option File | Set path on Matlab command window menu or the standard Matlab command addpath to add directories to your Matlab path. Further details can be found in the Matlab manual.
- If you are a frequent ARE user, it is recommended to include the argusInit call in your startup.m file, so that your ARE is initialized for every Matlab session. Like argusInit.m, the startup.m file has to be located at your Matlab path.

<u>Matlab Requirements</u>

Using the 2002 release of the ARE requires the availability of Matlab 6.1, including the image processing toolbox and the statistics toolbox.

Appendix B: Inputfile Argus Merge Tool (AMT)

Apart from the opportunities offered by the AMT user interface, a series of images for merging can also be defined through an external input file. Each line of such an input file contains the time information that is needed to specify the merge of interest. Preceeding comment lines are ignored. Time information can be defined in 3 different ways:

1. yyyy mm dd GMThr
2. yyyy mm dd GMThr minute
3. Argus image filename

The example file listed below combines the 3 types of time definition.

```
% Input file for argusMergeTool AMT
%
% IMPORTANT NOTE: IF TIMES OF INTEREST ARE DEFINED THROUGH
% THE NAMES OF ARGUS IMAGES (OPTION 3), ONLY THE EPOCHTIME
% NUMBER IS USED! SO NO ACCOUNT IS TAKEN OF THE CAMERA NUMBER,
% IMAGE TYPE ETC. THESE ARE DETERMINED FROM THE SETTINGS OF
% THE UPPER PART OF THE GUI!!!
%

1999 05 31 10
1999 05 31 11
1999 05 31 12 20
1999 05 31 14
928138784.Mon.May.31_08_19_44.GMT.1999.jvspeijk.c1.timex.jpg
928142384.Mon.May.31_09_19_44.GMT.1999.jvspeijk.c1.timex.jpg
928145985.Mon.May.31_10_19_45.GMT.1999.jvspeijk.c1.timex.jpg
```

Appendix C: DEM on the basis of shorelines

This appendix provides an overview of the Matlab commands which allow for the set-up of a Digital Elevation Model (or ‘Mapping the intertidal beach bathymetry’) on the basis of video-derived shorelines. This type of post-processing is not a standard feature yet within the ARE.

Before running these commands, make sure that the shorelines of interest as well as the support file breakwater.mat (not crucial) are located in the Matlab’s working directory. Otherwise, the ‘dir’ command will fail.

Start of with nothing in mind & make sure the ARE is initialized.

```
clear all
close all

addpath('d:\argus');
argusInit('pari');


   Start date: 07-Mar-2002
   Now       : 21-Mar-2002 14:01:09
   End date  : 01-May-2005
   Date check: OK
   Owner     : PARI
   License OK.

   Argus runtime environment set up for argusDB pari
```

Get a listing of shorelines available for September 3rd, 2001. The result is a [10x1] structure array named fns, containing (amongst others) the filenames created by IBM.

```
fns = dir('*20010903*.mat')

fns =
10x1 struct array with fields:
    name
    date
    bytes
    isdir

fns(1)

ans =
     name: 'wl.miyazaki.20010903.gmt0000.c1.mat'
     date: '21-Mar-2002 13:45:38'
    bytes: 23456
    isdir: 0
```

Collect all xyz co-ordinates in one variable named xyz. Notice the inner ‘for j’ loop. This loop is needed to account for the second and further parts of a waterline, in the case that a waterline was derived from an image by multiple IBM runs (that is, the user applied multiple, neighbouring ROI’s).

```
xyz = [];
for i = 1:size(fns,1)
    load(fns(i).name);
    for j = 1:length(allWL)
        xyz = [xyz; allWL(j).xyz];
    end
end
```

Define the overall DEM grid. The function meshgrid is a standard Matlab routine to do so.

```
xi = 150:5:325;
yi = -750:-5:-2400;
[X Y] = meshgrid(xi,yi);
```

Interpolate the raw waterline locations xyz to the DEM grid. This yields an elevation matrix zi.

```
zi = griddata(xyz(:,1),xyz(:,2),xyz(:,3),X,Y,'cubic');
```

Now replace all elements of zi which are further than 10 m away from the nearest xyz point by NaN's. This is to remove interpolation-induced erroneous values from the elevation matrix. Notice that this operation may take a little time!

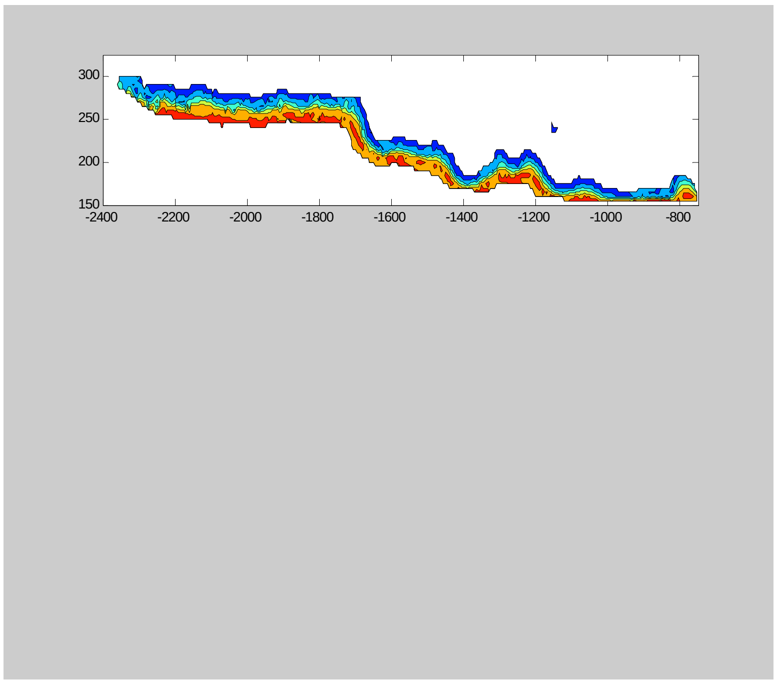
```
Nx = length(xi);
Ny = length(yi);
for x = 1:Nx
  for y = 1:Ny
   distance = sqrt((xyz(:,1)-xi(x)).^2 + (xyz(:,2)-yi(y)).^2);
   if distance > 10,
      zi(y,x) = NaN;
   end;
  end;
end;
```

Load the file breakwaters.mat (very miyazaki specific). This file contains the contours of the breakwaters in front of Miyazaki beach. Just used for reference purposes in the plot

```
load('breakwaters.mat');
```

Now create the plot with the help of the contourf command.

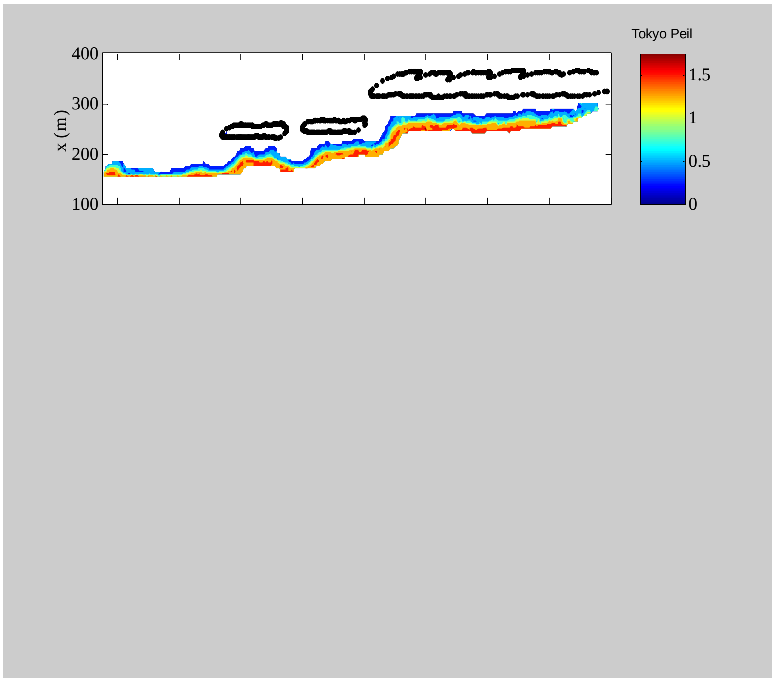
```
figure(1); subplot 311;
contourf(yi,xi,zi',[0:0.25:2]);
```



Shading flat makes the result look much better. Also set the axes, add labels and include a colorbar.

```
shading('flat');
set(gca,'xdir','reverse');
hold on
plot(allWL.xyz(:,2),allWL.xyz(:,1),'k.')
hold off;
axis([-2400 -750 100 400]);
set(gca,'fontname','times','fontsize',12);
h = colorbar;
set(h,'fontname','times','fontsize',12);
q = get(h,'Title');
set(q,'String','Tokyo Peil');
set(gca,'xticklabel',[]);
ylabel('x (m)','fontname','times','fontsize',12);
```

And here is the final result:


Tokyo Peil
400
300
200
100
x (m)
1.5
1
0.5
0

Appendix D: Manual Argus PIXel Toolbox