Chapter 1

Dual Surface Minimization in C++

1.1 Introduction

Dual Surface Minimization (DSM) [1] is a fully automatic method for extracting meaningful structures from three-dimensional image data. *dualsurfacemin* is a C++ implementation of the DSM algorithm. This chapter describes the usage of the *dualsurfacemin* tool in detail.

1.2 Usage

Parameters can be passed on the command line or via a configuration file ('config.dsm'). When the program is launched, parameters will be initialized to their hard-coded default values. If a valid configuration file is present, the values specified in the file will override the defaults. Arguments specified on the command-line override the default values and values specified in the configuration file.

1.2.1 Simple command-line usage

To get command-line help launch one of the following:

```
dualsurfacemin -help
dualsurfacemin -usage
dualsurfacemin -h
dualsurfacemin -?
```

To get a more detailed help screen, use the –verbose argument:

```
dualsurfacemin -help -verbose
```

To perform outer surface minimization with initial outer mesh 'outer.wrl', image 'hoffmann' and store results as 'hoffsurf.wrl', launch:

```
dualsurfacemin -outer_mesh outer.wrl -image hoffmann -output_mesh
hoffsurf.wrl
```

The most commonly used parameters have short command-line aliases. To achieve the same results as in the example above, one could launch:

```
dualsurfacemin -M outer.wrl -i hoffmann -o hoffsurf.wrl
```

Note that when only one input mesh is defined (as in the example above) *dualsurfacemin* will use a single surface version of the algorithm (DSM-OS or DSM-IS). When both input meshes are defined, the standard dual surface algorithm will be used.

See parameter summary for supported command-line arguments and their aliases.

1.2.2 Input files

1.2.2.1 Mesh files

Currently supported mesh file formats are VRML 1.0 [2] and OFF [3]. There are certain constraints for the meshes:

- Input meshes must be simplex (each vertex must be connected to exactly three neighbouring vertices by an edge);
- Faces must be properly oriented (vertex indices must be listed couter-clockwise);
- Only a certain subset of VRML 1.0 is supported, see sample VRML files for details.

Input meshes are assumed to be in surface-centered coordinates. If the input mesh is in OFF format, the reference point in image coordinates must be supplied with `-outer_ref` or `-inner_ref`. If the input mesh is in VRML, the reference point will be read from an enclosing *Translation* node in the VRML file.

1.2.2.2 Image files

Analyze 7.5 If image filename is supplied without extra arguments, the file is assumed to be in Analyze 7.5 format [4]. The *filename* argument to the `-image` parameter should be the Analyze file name without extension.

Raw voxel data If the input image is raw voxel data, `-image_dimensions` and `-image_data` parameters must be supplied.

1.2.3 Program output

1.2.3.1 Output mesh

By default the output mesh will be in VRML 1.0 format, simplex, and in surface-centered coordinates. Output format can be changed to .OFF with `-output_off`. Triangulated output can be produced with `-output_triangulated`. Output can be transformed to image coordinates with `-output_image_coords`.

1.2.3.2 Intermediate results

dualsurfacemin can produce intermediate result meshes while the algorithm is running. For example to produce intermediate meshes every 5 rounds of the algorithm and have these meshes stored in subdirectory `./Intermed`, one could launch the program with the following arguments:

```
-imed_interval 5 -imed_dir ./Intermed -imed_pattern foobar
```

This would produce files `foobar0001.wrl`, `foobar0002.wrl` etc. in directory `./Intermed`. Note that `-imed_dir` and `-imed_pattern` are not mandatory; If these arguments are not supplied, current

directory and pattern `result` will be used. Note also that the directory defined with `-imed_dir` must exist; *dualsurfacemin* will not attempt to create one automatically.

1.2.3.3 Console output

By default some basic console output will be produced (such as 'input image loaded ok blah blah...'). If the program is invoked with the `-verbose` parameter, detailed minimization statistics will be produced every round of the algorithm. If the program is invoked with the `-silent` parameter, all console output is suppressed.

1.2.4 Configuration files

When *dualsurfacemin* is launched, it will look for a file called 'config.dsm' (unless `-config` command-line argument was used; see parameter summary). A configuration file is a plain text file which contains key/value pairs separated by newline characters. Comments are allowed and are prefixed by hash (#) characters. Below is an example of a valid configuration file:

```
# My DSM config file
# input:
inner_mesh = SmallBall.wrl
outer_mesh = BigBall.wrl
image = energy.img
image_dimensions = 128 128 35
image_normalize = true
image_swap_bytes = true
image_data = float
lambda = 0.4
max_rounds = 1500
# output:
output_mesh = BrainHertz.wrl
```

Configuration file keys are generally syntactically identical to the corresponding command-line arguments, but there are some arguments that are not meaningful in the config file context (such as `-help`). Note also that short command-line aliases are not valid config file keys. See parameter summary for a table of supported config file keys.

Note: As command-line arguments override config file entries, specifying a single input mesh on the command line will always result in the single surface algorithm being used, even if input meshes are defined in the config file. For example if the config file contains an entry for `inner_mesh`, and the user specifies the `-outer_mesh` parameter on command-line, *dualsurfacemin* will use the single surface algorithm DSM-OS instead of the standard dual surface algorithm.

1.2.5 Parameter summary

Parameters are passed as key/value pairs. Command-line usage is:

`dualsurfacemin -key_name [value]` or

`dualsurfacemin -a [value]`, where *a* is a short command-line alias for the key.

Config file usage is:

`key_name = value`

On the command-line, for keys which take boolean values, specifying `-key_name` is equivalent to `-key_name true`. In the configuration file the boolean value (`true` or `false`) must be written out explicitly.

All currently supported parameter keys and their aliases are described in the following tables.

Category	Key	Alias	Value	Description	Mandatory	Default value
Input	outer_mesh	M	*filename*	Use initial outer mesh from file *filename*	yes[a]	
	outer_scale	S	*ratio*	Scale initial outer mesh by *ratio*	no	1.0
	outer_ref	R	*x y z*	Set reference point for the initial outer mesh	[b]	
	inner_mesh	m	*filename*	Use initial inner mesh from file *filename*	yes[a]	
	inner_scale	s	*ratio*	Scale initial inner mesh by *ratio*	no	1.0
	inner_ref	r	*x y z*	Set reference point for the initial inner mesh	[b]	
	image	i	*filename*	Use energy image from file *filename*	yes	
	image_-dimensions	d	*x y z*	Set input image resolution	[c]	
	image_data	a	float | int | uint	Set input image data type to floating point, integer or unsigned integer	[c]	
	image_swap_-bytes	b	true | false	Swap input image byte order	no	false
	image_-normalize	z	true | false	Normalize input image data	no	true

[a] At least one input mesh must be supplied.

[b] A reference point must be supplied if it is not present in the input VRML file.

[c] Mandatory for raw images. When this parameter is specified the input image will be treated as raw.

Category	Key	Alias	Value	Description	Mandatory	Default value
Output	output_mesh	o	*filename*	Store resulting mesh in file *filename*	no	result.wrl
	output_off		true | false	Store resulting mesh in OFF format	no	false
	output_-triangulated		true | false	Triangulate the resulting mesh	no	false
	output_-image_coords		true | false	Translate the resulting mesh to image coordinates	no	false
	imed_interval	y	#	Store intermediate results every # rounds	no	0 (Intermediate results are not stored by default)
	imed_pattern	w	*pattern*	Use *pattern* to name intermediate result files. For example if imed_pattern is set to 'foo', intermediate result files will be named 'foo0001.wrl', 'foo0002.wrl' etc.	no	'result'
	imed_dir	x	*dirname*	Store intermediate results to directory *dirname*. If the specified directory does not exist, the program will abort	no	./

Category	Key	Alias	Value	Description	Mandatory	Default value
Minimization	max_rounds	N	#	Set maximum rounds to #	no	50
	tolerance	t	*value*	Set tolerance (in voxels) when comparing mesh volumes[d]	no	0.5
	volume_ratio		*value*	Set target volume ratio for single surface method[e]	no	0.6
	lambda	l	*value*	Set regularization parameter to *value*	no	0.3
	gamma	g	*value*	Set penalty coefficient to *value*	no	0.03
	search_j	J	#	Set search space parameter J to #	no	8
	search_k	K	#	Set search space parameter K to #	no	1
	search_l	L	*value*	Set search space parameter L to *value*	no	0.5
	search_d	D	*value*	Set search space parameter D to *value*	no	0.15

[d] This parameter is meaningful only for the dual surface method.

[e] This parameter is meaningful only for the single surface method.

Category	Key	Alias	Value	Description	Mandatory	Default value
Console output	help, usage	h, ?		Output a help screen to the console. This key is not allowed in the configuration file.	no	
	verbose	v	true | false	Enable/disable verbose mode. Extra console output is produced in verbose mode.	no	false
	silent, quiet	q	true | false	Enable/disable silent mode. No console output whatsoever is produced in silent mode.	no	false

Category	Key	Alias	Value	Description	Mandatory	Default value
Miscellaneous	config	c	*filename*	Use configuration file *filename*	no	config.dsm

1.3 References

1. J. Tohka, J. Mykkänen, "Deformable mesh for automated surface extraction from noisy images", International Journal of Image and Graphics, Vol. 4, No.3 (2004) 405-432
2. Web 3D consortium <http://www.web3d.org/>
3. Geomview <http://www.geomview.org/>
4. Mayo Biomedical Imaging Resource <http://www.mayo.edu/bir/>