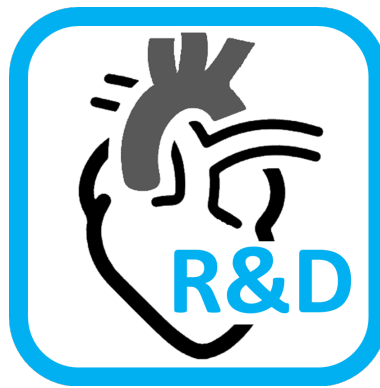


Segment - Technical Manual



2023-06-15

Software version v4.0 R12067

www.medviso.com



info@medviso.com



MEDVISO AB
Griffelvägen 3
SE-224 67 Lund
Sweden
support@medviso.com
<https://www.medviso.com>
P: +46-76-183 6442



<https://medviso.com/documents/segment/manual.pdf>

© Copyright 2009-2023 Medviso AB. All rights reserved.

Contents

1	License terms	1
2	Acknowledgements	2
3	Conventions and Abbreviations	3
3.1	Typographic conventions	3
3.2	Abbreviations	3
4	Document purpose	5
5	Coding Conventions	6
5.1	Naming functions	6
5.2	Naming variables	6
5.3	Name graphical handles	7
5.4	Object oriented programming	7
5.5	Packaging	7
5.6	General coding conventions	7
5.7	Units	8
5.8	Graphical user interfaces	8
5.9	Indentation and spacing	8
5.10	Documenting code	9
6	Coordinate Systems	10
6.1	Introduction	10
7	Running Segment from Matlab	11
8	Overview of Segment	12
9	DATA Object	13
10	SET Variable	22
11	Implementation Details	41
11.1	Version handling	41
11.2	Numeric representations	41

11.3	Loading data and interpretation of DICOM tags	41
11.4	Volume calculations	42
11.5	Mass calculations	43
11.6	Calculation of BSA	43
11.7	Peak ejection/filling rate	43
11.8	Wall thickness	43
11.9	Calculation of regurgitant volumes and shunts	44
11.10	Infarct size, extent and transmuralty	44
11.11	Number of SD from remote for Scar	44
11.12	MR relaxometry calculations	45
11.13	Pulse wave velocity	45
11.14	Torsion	45
	11.14.1 Least squares circle fit	45
	11.14.2 Angular discontinuity detection	47
11.15	Longaxis volumes	47
12	How to Create Own Plug-ins	49
13	Software support	50
	13.1 How to make a support request	50
	13.2 Additional support	50
	Bibliography	51

1 License terms

The general license terms of the usage of the software is provided on Medviso website <https://medviso.com/documents/segment/segmentresearchagreement.pdf>.

Though parts of Segment is released in source code form, this does not imply that standard open source rules do apply. Segment is a commercial product and is provided free of charge to the research community as a service and without any associated rights. Medviso AB does not give you any rights to do commercial derivative works of it. It does not give you the right to compile it to a distributable standalone form. See discussion on license terms in [1].

2 Acknowledgements

Even if this project started as a one man project, it has grown and it would never been possible without the help of many many people.

Financial support has been received from the Swedish Heart-Lung foundation, Swedish Research Council, local founds from Östergötland County, and Region of Scania.

I would like to acknowledge all the people that have put in feed back on usability and desired functionality, algorithm etc. Among others: Andreas Otto, Andreas Sigfridsson, Erik Bergvall, Erik Hedström, Henrik Haraldsson, Henrik Engblom, Håkan Arheden, Jan Engvall, Lars Wigström, Lisa Hård af Segerstad, Karin Markenroth Bloch, Marcus Carlsson, Martin Ugander, Mikael Kanski. Finally thank to you all Segment users in the research community that has inspired and contributed to the development.




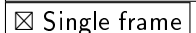
Special thanks to code providers Erik Bergvall (core routines of strain analysis), Helen Soneson (strain analysis module, SPECT module, Image fusion module), Shruti Agarwal (refactory of strain analysis module), Jonatan Wulcan (Sectra Plugin module and general improvements), Johannes Töger (3D flow and volume tracking), Måten Larsson (3D flow and kinetic energy).

Commercial development has been done by Jane Sjögren (improvements to general object segmentation, implementation of prototype based segmentation, CT functionality, and graphical seriesselector). General debugging and implementation of the new interpolated contours has been done by Johan Ugander and Erik Södervall. Report Module and general debugging have been performed by Nils Lundahl.

3 Conventions and Abbreviations

This chapter describes the typographic conventions and used abbreviations in this manual and in the program.

3.1 Typographic conventions

A	Key A at the keyboard.
Ctrl-A	Control key. Hold down Ctrl key and A simultaneously.
	Icon in toolbar.
*.mat	Filename extension.
C:/Program	Folder.
File	Menu, e.g. File menu.
File→Save As	Sub menu, e.g. under the File menu the item Save As is found.
	Push/Toggle button in the graphical user interface.
	Radiobutton in the graphical user interface.
	Checkbox in the graphical user interface.

3.2 Abbreviations

2CH	Two chamber view
3CH	Three chamber view
4CH	Four chamber view
3D	Three Dimensional
3D+T	Time Resolved Three Dimensional
AA	Ascending Aorta
ASW	Anterior Septal Wall Thickness
ARD	Aortic Root Diameter
BPM	Beats per minute
BSA	Body Surface Area
CMR	Cardiac Magnetic Resonance
CO	Cardiac Output
CT	Computed Tomography
DA	Descending Aorta
DE-MRI	Delayed Enhancement MRI
ED	End diastole
EDD	End Diastolic Dimension
EDL	End Diastolic Length
EDV	End Diastolic Volume
EF	Ejection Fraction
ES	End systole
ESD	End Systolic Dimension

ESL	End Systolic Length
ESV	End Systolic Volume
FWHM	Full Width Half Maximum
GUI	Graphical User Interface
HR	Heart Rate
LGE	Late Gadolinium Enhancement
LV	Left Ventricle
LVM	Left Ventricle Mass
MaR	Myocardium at Risk
MO	Microvascular obstruction
MB	Mega Byte
MIP	Maximum Intensity Projection
MPR	Multiplanar Reconstruction
MR	Magnetic Resonance
MRI	Magnetic Resonance Imaging
PET	Photon Emission Tomography
PER	Peak Ejection Rate
PDW	Proton Density Weighted
PFR	Peak Filling Rate
PLW	Posterior Lateral Wall Thickness
PWV	Pulse Wave Velocity
ROI	Region Of Interest
RV	Right Ventricle
RVmaj	Right Ventricle Major Axis
RVmin	Right Ventricle Minor Axis
SPECT	Single Photon Emission Computed Tomography
SSFP	Steady State Free Precision
SV	Stroke volume
TOF	Time of Flight
VENC	Velocity Encoding limit

4 Document purpose

The purpose of this document is three-fold;

1. Work as a technical documentation of Segment for Medviso AB
2. Work as a technical documentation for researcher who wish to use the plug-in functionality of Segment as described in [1].
3. Work as a implementation documentation of Segment.

Therefore, all details described in this technical manual may not be applicable to all developers since parts of the source code is only accessible by Medviso AB.

5 Coding Conventions

This chapter describes the coding conventions that has been used when coding Segment. Consistent naming rules has been sought to ensure that variables are given a logical name. However, there are occasions where historical reasons have lead to exceptions to the rules given below.

In general Segment originally used the global variable `NO` indicating the current image stack quite extensively. The use of `NO` is however not encouraged, and it is highly recommendable to in the function call indicate the current image stack.

5.1 Naming functions

Functions are always named in lower case letters only. The underscore letter is not used with the exception of:

- Callbacks are named as `functionname_Callback`.
- Functions for mouse motion callbacks are given names ending with `_Motion`.
- Some functions have helper functions that are only called from other functions with the same name. These functions are named `_helper`.
- Functions for mouse button up/down callbacks are given names ending with `_Buttonup` or `_ButtonDown`.

Note that functions should not be ended with an `end` (Matlabsupports both syntaxes). This is with the exception with objected oriented programming, see below.

5.2 Naming variables

The following rules apply:

- Global variables have all capital letters, i.e `DATA`.
- Field names of structures (and structure arrays) are given names starting with a capital letter. When two or more words are concatenated then the first letter of the second word also have a capital letter. For instance `DATA.CurrentTool`. Abbreviations are also given capital letters. For instance `SET(NO).TSize` where T stands for time.
- Field names with all capital letters corresponds to matrices. For instance `SET(NO).IM`, or `DATA.BALLOON`.
- The use of plural `s` is very limited and only used where there are limited numbers of values the field and it may be necessary to point out that there are more values than just a scalar. For instance `DATA.ViewPanels` is just instead of `DATA.ViewPanel` to point out there are (or may be) more than one view panel.

5.3 Name graphical handles

The graphical handles are named with all lower case letters and no underscore letters. Generally the names are rather long and exploratory. The following general naming conventions apply:

- Text edit boxes ends with `edit`
- Static text ends with `text`
- Axes objects ends with `axes`
- Pushbuttons ends with `pushbutton`
- Radiobuttons ends with `radiobutton`
- Checkboxes ends with `checkbox`
- Togglebuttons ends with `checkbox`

5.4 Object oriented programming

For new objected oriented functions the new syntax introduced in Matlab R2007b should be used. The old usage where the class definition is stored in a separate folder is obsoleted and existing code will be rewritten to new standard. Note that this new coding methods using `classdef` requires that an `end` statement is inserted in the end of the method code.

5.5 Packaging

To improve the file structure, Matlab's packaging feature is used. Files that are part of the same module are placed in a directory named with an initial '+' sign. Calling a function file `fcn_name.m` from a package `+package_name` is done by entering `package_name.fcn_name` (without the '+'). When adding packaged files to the compilation process, the full directory name must be included in the string. In the example above, this becomes `['+package_name' filesep 'fcn_name.m']`.

5.6 General coding conventions

The use of boolean variables in function calls should be avoided since this significantly decreases code readability. This is especially true in API-close functions. Instead it is suggested to write two functions with different names that performs the different tasks. This boolean coding convention is historically not followed, but should be followed for new code, and old code should successively be rewritten.

Temporarily variables should be avoided, and if they are used they should not have a scope that exceeds three lines of code. Also here this is something that historically is not always followed. This rule should be followed for new code and old code should successively be rewritten.

5.7 Units

The units of variables should be in SI-units, and the exact unit should also be documented in a comment on the same line as where the variable is initialized or calculated. For instance:

```
speed = length/time; %[m/s]
```

5.8 Graphical user interfaces

All new graphical user interfaces should be based on the class `mygui`. When doing so Segment automatically will remember the position of the user interface, and coding is considerably simplified so that one does not need to use persistent variables to get application data. Please consult the documentation of `mygui` for further details. The following function should be used as default to close a `mygui` user interface:

```
-----  
function close_Callback  
-----  
%Close the GUI.  
global DATA  
  
try  
close(DATA.GUI.GUIname);  
catch  
close(gcf);  
end  
DATA.GUI.GUIname = [];
```

5.9 Indentation and spacing

Matlab standard indentation system is to be used, but with using the tab size to two spaces instead of the default four. Spacing and use of parenthesis is to be used to enhance readability. For instance

- `A = 12;` instead of `A=12;`.
- `if (a>b) || (d<e)` instead of `if a>b||d<e`

MLINT syntax guidelines should be followed and ideally when syntax hints are overridden, they should be motivated. For instance when MLINT reports; Warning data seems to grow inside a loop and that you should consider to preallocate for speed. Then before inserting a pragma to remove the warning you should make a note on why you did not preallocate (which usually is that the routine is not time critical, but is should generally be documented). Another example is when MLINT warns that the variable is unassigned, but it is assigned

by loading a `.mat` file, then this should be commented.

The use of logical short cutting operands `||` and `&&` is strongly recommended.

5.10 Documenting code

The source code should be well documented so that new programmers easily can understand the code. Note, especially that the part of the code that was trickiest to write also deserves the most comments. Functions should be written as:

```
%-----  
%function out = foo(bar)  
%-----  
%Explanatory help text of the function.  
%Help text may span multiple lines.
```

Your code begins here

It is very important to follow this coding standard since part of this documentation is automatically generated from the source code.

6 Coordinate Systems

This chapter specifies the coordinate systems used in Segment.

6.1 Introduction

DICOM specifies coordinates in the RL (right/left), AP (Anterior/Posterior) and FH (Feet/Head) coordinate system. Internally Segment uses a coordinate system (x,y,z) that is relative to the respective image stack. In other words for each image stack the relations between (RL,AP,FH) and (x,y,z) is different.

Throughout the program, the x dimension refers to the first dimension along the arrays (i.e the **rows** in Matlab). The upper left pixel in the images displayed on the screen have the coordinate (1,1). The unit used is in pixels. In the normal image coordinate system this means what is usually meant as the y coordinate if one thinks of coordinate systems learned in elementary school. For instance plotting commands therefore generally takes the form:

```
plot(SET(NO).EndoY(:,3,1),SET(NO).EndoX(:,3,1));
```

This will plot the contour of the endocardium of the third time frame and the first slice of the current image stack. The upper left most slice in the montage view is slice one. Furthermore it is assumed that the slices should be ordered so that going from the first slice to the last would be going from the base to the apex of the heart.

The relations between the (x,y,z) and (RL,AP,FH) coordinate systems is given by the fields `ImagePosition` and `ImageOrientation`. Below are two examples given.

The (RL,AP,FH) coordinates (center) of the voxel(s) `SET(1).IM(1,1,:,1)` are given by `SET(1).ImagePosition`.

The (RL,AP,FH) coordinates (center) of the voxel `SET(1).IM(2,3,4)` are given by

```
zdir = cross(SET(1).ImageOrientation(1:3),SET(1).ImageOrientation(4:6))';  
pos = SET(1).ImagePosition(:)'+...  
      (2-1)*SET(1).ResolutionX*SET(1).ImageOrientation(4:6)'+...  
      (3-1)*SET(1).ResolutionY*SET(1).ImageOrientation(1:3)'+...  
      (4-1)*(SET(1).SliceThickness+SET(1).SliceGap)*zdir;
```

Coordinated conversions can be performed using the functions `xyz2rlapfh`, and `rlapfh2xyz` in `calcfunctions.m`.

7 Running Segment from Matlab

You need to have Matlab R2022a to run Segment. Running Segment from Matlab prompt is just as easy as running it as a stand-alone application. Note that necessary mex files have been compiled for Linux (64 bit), and Windows (64 bit). When Matlab has started, simply type:

```
>> segment
```

To get access to the internal variable direct from the Matlab prompt, simply type:

```
>> global DATA SET NO
```

To get the x-coordinates of the endocardial segmentation in time frame 3 and slice 4, simply type:

```
>> x = SET(NO).EndoX(:,3,4)
```

To plot the segmentation in the current slice in another window, type:

```
>> figure(22);  
>> plot(SET(NO).EndoY(:,SET(NO).CurrentTimeFrame,SET(NO).CurrentSlice), ...  
SET(NO).EndoX(:,SET(NO).CurrentTimeFrame,SET(NO).CurrentSlice),'r-');  
>> axis image off;
```

8 Overview of Segment

Segment is written in Matlab, and is a fairly large software project. Currently it contains of around 130 000 lines of `m-code` distributed more than 200 files, and 2815 subfunctions, and 69 `gui`'s. There are 72 files `c-code` with about 12 000 lines. There are 22 supporting `.mat` files, and 32 auxillary files. In addition there are 12 000 lines in around 40 files and 300 subfunctions of internally support lines of code keeping track on distribution, compilation process, packaging, documentation generation etc.

This Technical Manual aims to describe how to write own plug-ins that interface with the software and give some implementation details for the interested reader. A good technical overview of the Segment project is also given in [1].

Before reading this user manual the reader should be well acquainted with Segment and programming in Matlab. This manual is **not** a manual how to use Segment, it's intention is only to give technical details on how things are implemented in Segment. Some implementational details are also given in the user manual.

9 DATA Object

This global object is an instance to the class `segmentgui`. It contains methods and properties that are used to store properties and access the graphical handles, preferences, etc. It is an instance of one of the subclasses of the `maingui` class and contains the following properties:

<code>LogFile</code>	Location of the log file keeping track of the current session.
<code>ProgramVersion</code>	A string containing the version number. This string is initiated at startup.
<code>ProgramName</code>	Name of the program in use, e.g. <code>Segment</code> .
<code>fig</code>	Handle to the main GUI figure.
<code>BlockingFigs</code>	A list of open figures that contains GUI's that are sensitive to changing the current image stack NO.
<code>imagefig</code>	Handle to the figure where the images are plotted. Currently this is same as <code>fig</code> , but are reserved for future use.
<code>DataLoaded</code>	True if image data is present.
<code>Silent</code>	True if no graphical output should occur. This is usefull when writing plugins that perform batch processes.
<code>Handles</code>	Struct with all graphical handles in the main GUI.
<code>InteractionLock</code>	When set to true, new calculation initiated from callbacks are prohibited. This may be obsoleted in the future and replaced with Matlab's queueing properties.
<code>Undo</code>	Struct containing undo information.
<code>UndoN</code>	Scalar that contains a pointer in the undo history list.
<code>LastSaved</code>	Time stamp when the image stack(s) last were saved. This is used for the auto-save functionality.

<code>Volrend</code>	Reserved for future use with the future volume rendering module.
<code>ViewPanels</code>	A vector pointing to image stacks. If 2 x 2 panels are shown on the screen, then the length of <code>ViewPanels</code> is 4. For panels that does not contain any image stack the corresponding pointer is set to zero.
<code>ViewPanelsType</code>	<p>A cell array where each element contains the view mode for each image panel. Allowed view modes are:</p> <ul style="list-style-type: none"> • mode <code>one</code> shows the image stack as one slice. • mode <code>montage</code> shows all slices of the image stack arranged in a matrix style. • mode <code>mmodetemporal</code> shows the temporal part of a <code>mmode</code> image display. • mode <code>mmodespacial</code> shows the spatial part of a <code>mmode</code> image display. <p>The best way of changing viewing mode is to call the function <code>viewimage_Callback</code>.</p>
<code>ViewPanelsMatrix</code>	A cell array where each element <i>i</i> a two element vector containing the number of rows and columns for a montage view in the corresponding panel.
<code>ViewIM</code>	A cell array with the length same as <code>ViewPanels</code> . Each element contains a <code>uint8</code> array of size $N \times M \times T$ where T is the number of time frames, N and M are arbitrary numbers that depends on <code>ViewPanelsType</code> . This is used for graphical output and contains mapped intensities.
<code>ViewMatrix</code>	A two element vector with the size of the image panels ($n \times m$).
<code>LastView</code>	Save view settings for the last panel view. Used upon double-clicking to go from the current panel view to the last panel view.

<code>CurrentPanel</code>	Points to the current image panel. This number needs to be in the range $[1..N]$, where N is the number of valid panels.
<code>CurrentTool</code>	String containing the current tool. Valid tools are: <ul style="list-style-type: none"> • <code>point</code> • <code>measure</code> • <code>select</code> • <code>dragendo</code> • <code>dragepi</code> • <code>drawendo</code> • <code>drawepi</code> • <code>drawrvendo</code> • <code>drawrvepi</code> • <code>drawroi</code> • <code>drawscar</code> • <code>drawrubber</code> • <code>drawrubberpen</code> • <code>endopin</code> • <code>epipin</code> • <code>contrast</code> • <code>crop</code>
<code>CurrentTheme</code>	Indicates which set of segmentation tools is currently on display to the user.
<code>Tools</code>	Structure of handles to tools available in the toolbar.
<code>SegmentFolder</code>	Location of the folder from which Segmentis loaded.
<code>imagedescriptionfile</code>	Text document for setting image description parameters.
<code>manualfile</code>	Location of the Reference manual.

<code>GUI</code>	Structure of <code>mygui</code> objects for windows that use this class.
<code>GUISettings</code>	GUI settings
<code>GUIPositions</code>	Struct array to store the positions of GUI's that have been in use.
<code>AllowInt16</code>	Option whether or not to allow <code>int16</code> data.
<code>Pref</code>	Structure with preferences.
<code>PrefHandles</code>	Handles for Preferences window.
<code>PrefHandlesAdvanced</code>	Handles for Advanced Preferences window.
<code>PrefHandlesPacs</code>	Handles for PACS Preferences window.
<code>Preview</code>	Struct storing preview data. This struct is intensely used by the file open GUI. Generally the data is store in this struct and thereafter copied to the SET variable upon completion of the loading process. Therefore this struct also contains information on the last loaded/viewed image stack.
<code>StartTime</code>	Start time for video playback.
<code>Run</code>	True if a movie is playing.
<code>Record</code>	True if recording is active in the movie recorder GUI.
<code>LastPointer</code>	Type of the pointer displayed.
<code>LastPointerShapeCDATA</code>	data of shape of the pointer displayed.
<code>VisibleThumbnails</code>	Array of indices of thumbnails visible in main GUI.
<code>Icons</code>	Structure of images used as icons in toolbars and icon bars.

ImageTypes

A cell array contains the different precode image types in Segment. They are:

- General
- Perfusion Rest
- Perfusion Stress
- Strain FFE
- Strain TFE
- Late enhancement
- Cine
- Scout
- Qflow
- T2Stir
- T1BB

The image types are used among others to find what image stack is what for automated batch processing of files. This list is subject to future changes.

`ImageViewPlanes` A cell array contains the different precode image view planes in `Segment`. They are:

- 2CH
- 3CH
- 4CH
- Sagittal
- Coronal
- Frontal
- Transversal
- Short-axis
- RVOT
- Aorta
- Pulmonalis
- Vena cava inferior
- Unspecified

The image types are used among others to find what image stack is what for automated batch processing of files. This list is subject to future changes.

`ImagingTechniques` Short names of the parameter files (`*.par`). This variable is created a startup. Therefore, `Segment` needs to be restarted before new parameter files can be used from the menu. However, the files themselves are not cached so the content in the files can be changed without restarting `Segment`.

`ImagingTechniquesFileNames` / titles of the parameter files. See above for details.

`ThisFrameOnly` True if changes are applied only to the current timeframe. Example of functionality is segmentation, clearing segmentation etc. See details in the Reference Manual.

`StartFrame` First frame played in a movie. Initiated by `SET(NO).CurrentTimeFrame`, but needed for calculation what frame to display when playing a movie. See also `DATA.StartTime`.

<code>SegIntersection</code>	Seg intersection
<code>IntersectionIM</code>	Intersection image
<code>ShowIntersectionMask</code>	Show intersection mask
<code>BalloonLevel</code>	Used to determine if the field <code>DATA.BALLOON</code> needs to be recalculated. Stores the parameter that was used last time to calculate <code>DATA.BALLOON</code> .
<code>BALLOON</code>	A matrix of the same size as <code>SET(NO).IM</code> containing the radial balloon force. For further details, see [2, 3].
<code>EndoBalloonForce</code>	Used in the automatic LV segmentation for MR images.
<code>EpiBalloonForce</code>	Used in the automatic LV segmentation for MR images.
<code>LevelSet</code>	Helper structure to <code>SET(NO).LevelSet</code> . <code>DATA.LevelSet</code> stores parameters that are pertinent to the graphical user interface that does not need to be stored in <code>SET(NO).LevelSet</code> .
<code>DATASETPREVIEW</code>	A $64 \times 64 * N$ array containing the preview thumbnails, where N is the number of image stacks.
<code>EndoEDGE0</code>	Edge image of the endocardium in the first direction. For further details, see [2, 3].
<code>ENDOEDGE1</code>	See above.
<code>ENDOEDGE2</code>	See above.
<code>ENDOEDGE3</code>	See above.
<code>EpiEDGE0</code>	Edge image of the endocardium in the first direction. For further details, see [2, 3].
<code>EpiEDGE1</code>	See above.
<code>EpiEDGE2</code>	See above.

<code>EpiEDGE3</code>	See above.
<code>EndoEdgeDetected</code>	True if the edge images of the endocardium have been calculated.
<code>EpiEdgeDetected</code>	True if the edge images of the epicardium have been calculated.
<code>MovieFrame</code>	Temporary variable used to store one frame of a movie when recording a movie using the movie recorder in Segment.
<code>NumPoints</code>	Number of points along the endocardium, epicardium and ROI's. This default set to 80. This variable should be possible to change to get more points along a contour, but this have not been verified or tested. Note that changing this variable will most probably cause version incompatibilities with <code>.seg</code> and <code>.mat</code> files.
<code>BpInt</code>	Intensity in the blood pool of the image stack <code>SET(NO)</code> . This is used to avoid unnecessary recalculations of <code>DATA.BALLOON</code> .
<code>MInt</code>	Intensity of the myocardium of the current image stack. See above.
<code>Pin</code>	Keeps track of pins.
<code>Measure...</code>	Contains measurement data such as offset and coordinates of points.
<code>Cursor...</code>	Contains cursor data such as location and offsets.
<code>NeedToSave</code>	True if there are changes that need to be saved.
<code>Testing</code>	Used by <code>maketest</code> to tell a program when it is being tested.
<code>RecordMacro</code>	Used by <code>macro_helper</code> to tell when a macro is being recorded.
<code>Macro</code>	Used by <code>macro_helper</code> to keep track of the macros in a list.

<code>MacroN</code>	Used by <code>macro_helper</code> to keep track of the position in the Macro list.
<code>Buffer</code>	Contains pending actions used for macros and testing.
<code>CineTimer</code>	Timer used in <code>cinewindow</code> .
<code>contourp</code>	Used in the automatic LV segmentation for MR images.
<code>contourbimage</code>	Used in the automatic LV segmentation for MR images.
<code>DynamicPACS</code>	Stores PACS details specified through API by an external program.

10 SET Variable

This global variable is probably the most important variable/structure since it contains all image data and all measurements.

The variable is a struct array. As an example, `SET(2)` refers to the second image stack. `SET(2).CurrentTimeFrame` refers to the field that contains the current time frame of the second image stack. The function `loadfieldhelper` is used to control backwards compatibility in the `SET` struct. When adding new fields it is essential to also update this function.

Below a list of all fields are given.

<code>AccessionNumber</code>	DICOM tag Accession Number.
<code>AcquisitionTime</code>	DICOM tag AcquisitionTime
<code>AutoLongaxis</code>	True if the amount of long-axis compensation should be automatically calculated. See Reference Manual for details.
<code>BeatTime</code>	Controls how fast one heart beat is played when playing images as a movie. Originally set to <code>60/HeartRate</code> . May be adjusted with the <code>slower/faster</code> icons.
<code>CenterX</code>	Position of the center mark (+).
<code>CenterY</code>	Position of the center mark (+).
<code>Children</code>	List of numbers of image stacks that are children of (derived from) this stack.
<code>Colormap</code>	This field contains a colormap (256 x 3) vector. When the image is grayscale it is an empty field.
<code>ColormapOriginal</code>	This field contains a three-dimensional colormap retrieved from a DICOM file. When no information about the colormap is found in the DICOM, then this field is left empty.
<code>CurrentSlice</code>	This field contains the current slice of the image volume.

<code>CurrentTimeFrame</code>	This field contains the current time frame.
<code>Cyclic</code>	True for image stacks that are cyclic, i.e covers a complete heart cycle. This flag is used by the segmentation engine when delineating the left ventricle.
<code>DICOMImageType</code>	This field contains the <code>ImageType</code> from the DICOM files.
<code>EDT</code>	Left ventricle end diastolic time.
<code>EDV</code>	Left ventricle end diastolic volume.
<code>EF</code>	Left ventricle ejection fraction.
<code>EPV</code>	Volume inside the epicardial volume of the left ventricle.
<code>EST</code>	Left ventricle end systolic time.
<code>ESV</code>	Left ventricle end systolic volume.
<code>EchoTime</code>	Echo time in MRI pulse sequence. Read from DICOM tags if presented otherwise set to zero. Given in milliseconds.
<code>EndAnalysis</code>	End time of analysis, used in flow quantification. See also <code>StartAnalysis</code> . For further details, see Reference Manual for details. Default value is same as <code>TSize</code> .
<code>Endslice</code>	This field contains the last slice in the set of selected slices. The selected slices is then given by: <code>ind = SET(NO).StartSlice:SET(NO).EndSlice;</code> . The field may be an empty array when no slices are selected.
<code>EndoCenter</code>	True if center of endocardium is used as center of the left ventricle when doing regional wall motion analysis. If false then the center of the epicardial volume is used. See Reference Manual for further details.

EndoDraged	A logical array containing a 1 where the left ventricle endocardial contour have been dragged.
EndoInterpX	Interpolation points for the endocardium.
EndoInterpXView	Interpolation points for the endocardium.
EndoPinX	X -coordinates of endocardial pins. Stored as a cell-array of vectors of doubles with the size $T \times Z$, where T is the number of time frames, and Z is the number of slices. For more details about pins, see the Reference Manual.
EndoPinY	Y -coordinates of endocardial pins.
EndoPinXView	Compact representation for endocardial pins. Stored as a cell vector with the T elements containing vectors.
EndoPinYView	See above.
EndoX	X -coordinates of the left ventricle endocardium. Applied values are either [], or an array with size $N \times T \times Z$, where N is the number of points along the contour (80, but technically <code>DATA.NumPoints</code> , T is the number of timeframes, and Z is the number of slices.
EndoY	Y -coordinates of the endocardium.
EndoXView	A compact representation used for drawing the endocardial contour in montage view. Stored as a double array with the size $((N+1)*Z) \times T$, where N is the number of points along the contour (<code>DATA.NumPoints</code>), T is the number of timeframes, and Z is the number of slices. It is the same as <code>EndoX</code> but each slice offseted and packed separated with a <code>NaN</code> so that drawing of the endocardial contours can be done with a single command.
EndoYView	The correspondence for the Y -coordinates to <code>EndoXView</code> .
EpiDraged	Same as above, but for the epicardial contour.

EpiInterpX	Epicardial interpolation points.
EpiInterpXView	Epicardial interpolation points.
EpiPinX	Corresponding to EndoPinX.
EpiPinXView	Corresponding to EndoPinXView.
EpiPinY	Corresponding to EndoPinY.
EpiPinYView	Corresponding to EndoPinYView.
EpiX	Epicardial contour, otherwise same as EndoX.
EpiXView	Corresponding to EndoXView.
EpiY	See above.
EpiYView	Corresponding to EpiXView.
FileName	Filename of the file storing this image stack.
FlipAngle	Flip Angle in MRI pulse sequence. Read from DICOM tags if presented otherwise set to zero. Given in degrees.

Flow

This struct contains information regarding how different image stacks are related to store flow information. Generally all information that are common for all the coupled image stacks are only stored in the magnitude image stack to avoid data redundancy. One example of this is storage of ROI's that are only performed in the magnitude image stack. The `Flow` struct contains the following fields:

- `Angio` contains a reference to which image stack that contains the `angio` image. This `angio` image is essentially the absolute value of the velocity times the image magnitude and could be useful for vessel identification. Normally this field is set to an empty vector as the `angio` image is seldom constructed.
- `MagnitudeNo` contains reference to which image stack contains the magnitude information. For the image stack containing the magnitude information this points to itself.
- `PhaseNo` contains reference to the image stack containing the through plane flow information. Naming is somewhat inconsequent, by kept for legacy reasons. A better name would be `PhaseZ`.
- `PhaseX` contains reference to the image stack containing the flow in the X direction (Segment coordinate system).
- `PhaseY` contains reference to the image stack containing the flow in the Y direction (Segment coordinate system).

Then there are also a few fields that are optional and only available when eddy current compensation has been performed. These fields are:

- **PhaseCorr** contains the phase offset for the current direction (i.e if the current image stack is velocity/phase in the X direction), then **PhaseCorr** contains phase offset in the X direction, and so fourth. This field is always present, and if not applicable only an empty matrix is stored.
- **PhaseCorrPercentiles**, level of which percentiles to include in the detection of the static tissue. See documentation on the eddy current compensation for further details.
- **PhaseCorrMethod** contains the selected method for the eddy current compensation.
- **PhaseCorrTimeResolved** true if the eddy current compensation is time resolved. Default value is false. If time resolved, then **PhaseCorr** has the same image dimensions as **IM** otherwise the third image dimension is one.
- **PhaseCorrStaticTissueRois** true if regions where static tissue should be taken from drawn ROIs instead of estimated from the image. Default value is false.
- **VelMag** contains a reference to the image stack that contains a velocity magnitude stack (absolute value of the velocity). Normally this field is set to an empty vector as the velocity magnitude is seldom constructed.

Fusion	Struct reserved for Fusion Module.
Gadgetron	Field to store Gadgetron analysis data, such as MBF values. Such data are retrieved from private DICOM tags.
GEVENCSCALE	Special tag read from DICOM for GE scanners.

HeartRate	Hear rate of the image stack. Note that different image stacks may have different heart rates. Unit is beats per minute. HeartRate is used to calculate cardiac output.
IM	This field contains the image data stored as a 4D <i>single</i> array. The order of the dimensions are $X \times Y \times T \times Z$. In case DICOM image has the tag Photometric Interpretation set to RGB, then the image for each channel (R/G/B) is stored in the 6th dimension, resulting in $X \times Y \times T \times Z \times 1 \times 3$. For more details on coordinate system conventions, see Section 6. The image data should lie between 0..1. For more details on image scaling see the fields IntensityScaling and IntensityOffset .
ImageOrientation	Same as DICOM tag ImageOrientation . If not presented then set to 1 0 0 0 1 0. The three first numbers contains the normalized vector of Segments Y-direction given in the patient/table coordinate system. The last three numbers contains the direction of the X-direction. For further details on coordinate systems see Section 6.
ImagePosition	Same as DICOM tag ImagePosition . If not presented then set to 0 0 0. Contains position in mm in 3D of the upper left pixel in the top slice in the image stack. For further details on coordinate systems see Section 6.

ImageType

Type of images that the image stack depicts. The field is used when identifying what image stacks to use for different analysis. If not set at loading then Segmenttries to figure these details out by looking at what operations have been performed on what image stacks. Currently applied values are:

- 'General' (Generic Image type if not specified/identified).
- 'Perfusion Rest'
- 'Perfusion Stress'
- 'Strain FFE' (Strain Fast Field Echo).
- 'Strain TFE' (Strain Turbo Field Echo).
- 'Late enhancement' (Viability).
- 'Cine'
- 'Scout'
- 'Qflow'
- 'T2Stir'
- 'T1BB'

ImageViewPlane

View plane of images that the image stack depicts. The field is used when identifying what image stacks to use for different analysis. If not set at loading then Segmenttries to figure these details out by looking at what operations have been performed on what image stacks. Currently applied values are:

- 'Unspecified' (Generic Image view plane if not specified/identified).
- '2CH' (Long axis 2 chamber view).
- '3CH' (Long axis 3 chamber view).
- '4CH' (Long axis 4 chamber view).
- 'Sagittal'
- 'Coronal'
- 'Frontal'
- 'Transversal'
- 'Short-axis'
- 'RVOT'
- 'Aorta'
- 'Pulmonalis'
- 'Vena cava inferior'

ImagingTechnique

String of capital letters identifying type acquisition used when acquiring the image stack. Possible values depends are given by the .par files. Each such files corresponds to one image type and contains information of how image should be mapped before segmentation. The two first letters should correspond to imaging modality (i.e MR, CT, PT, US, CR...). Shipped .par files are:

- CTheart.par (Segmentation of LV).
- MRBB.par (MR black-blood sequence).
- MRDE.par (MR delayed contrast enhancement).
- MRGE.par (MR gradient echo images).
- MRPDW.par (MR proton density weighted images).
- MRSSFP.par (MR steady state free precision, or fiesta).
- MRSTIR.par (MR STIR pulse sequence, edema sequence).
- MRTOF.par (MR Time of flight sequence for vessels).
- NMBPSPECT.par (Blood pool SPECT images).
- OT.par (Myocardial probability, obsoleted).
- PT.par (Generic for PET, not for segmentation use).
- US.par (Generic for ultrasound. Not for segmentation use).

IntensityMapping A struct containing information how the voxel values of the image stack is shown on the screen. The struct has the following fields:

- **Brightness**
- **Contrast**
- **Compression** (Reserved for future use)

The fields **Brightness** and **Contrast** translates into intensity according to the equation:

$$I = cx_i + b - 0.5 \quad (1)$$

where c is contrast, b is brightness, x_i is the input intensity, and I is the output remapped intensity. The intensity I is then clipped to the range $[0..1]$. The field **Compression** is reserved for future use and will be used to implement image mappings that are sigmoid shaped.

IntensityScaling The fields **IntensityScaling** and **IntensityOffset** are used to convert the image data **IM** to the true data as presented in the DICOM images. The true data is calculated as:

$$z = I\alpha + \beta \quad (2)$$

where z is the true image data, I is the data stored in the field **IM**, α is **IntensityScaling**, and **IntensityOffset**.

IntensityOffset See above.

IntensityScaling See above.

InversionTime Inversion time in MRI pulse sequence. Read from DICOM tags if presented otherwise set to zero. Given in milliseconds.

LastUserInfo Struct containing information about the user who last modified the dataset (e.g. username, computer name, date)

LVM The left ventricle myocardial volume is calculated as EPV-LVV-PV. Unit is ml.

<code>LVV</code>	The volume within the left ventricular endocardium. Unit is ml.
<code>LevelSet</code>	Struct that contains general object segmentation module.
<code>Linked</code>	List of numbers of image stacks that are linked to this stack in <code>Parent/Children</code> relations.
<code>Longaxis</code>	Long-axis motion of the left ventricle, see Reference Manual for details on how the volume is compensated for long-axis motion. To convert to mm subtract with 1.
<code>MaR</code>	Struct that contains automatic and/or manual segmentation of the ischemic myocardium at risk from either MRI or SPECT.
<code>Measure</code>	Struct that contains data of measurements.
<code>Mmodex</code>	X-coordinate for the center of the mmode line.
<code>Mmodey</code>	X-coordinate for the center of the mmode line.
<code>Mmodelx</code>	Direction coefficient of the mmode line.
<code>Mmodey</code>	Direction coefficient of the mmode line.
<code>Mmodem1</code>	Distance from the mmode line center to the first measurement point. Usually a positive value.
<code>Mmodem2</code>	Distance from the mmode line center to the second measurement point. Usually a negative value.
<code>Modality</code>	Same as the corresponding DICOM tag.
<code>MontageRowZoomState</code>	Four element vector describing the current zoom state for the image stack in row montage view mode. This representation is potentially subject to future changes.
<code>MontageZoomState</code>	Four element vector describing the current zoom state for the image stack in montage view mode. This representation is potentially subject to future changes.

<code>NormalZoomState</code>	Four element vector describing the current zoom state for the image stack in normal mode. This representation is potentially subject to future changes.
<code>NumberOfAverages</code>	Number of averages acquired in MRI pulse sequence. Read from DICOM tags if presented otherwise set to zero.
<code>OrgXSize</code>	This refers to the original X-size of the images in the DICOM files. This is used to load <code>.seg</code> files to uncropped image data. The fields <code>OrgYSize</code> , <code>OrgTSize</code> , and <code>OrgZSize</code> have the same function in the other coordinate directions.
<code>OrigFileName</code>	Original filename. Note that this information is removed when anonymizing an image stack.
<code>Overlay</code>	Number of an image stack that is currently used as a color overlay of this stack.
<code>PER</code>	Left ventricular peak ejection rate. Taken as the largest negative derivative of <code>LVV</code> .
<code>PERT</code>	Time of peak ejection. Given in time frames.
<code>PFR</code>	Left ventricle peak filling rate. Calculated using circular convolution with a central difference (three elements) and taken as the largest derivative of <code>LVV</code> .
<code>PFRT</code>	Time of peak filling. Given in time frames.
<code>PV</code>	Volume of the papillary muscles. See Reference Manual for further details how that is calculated.
<code>PapillaryIM</code>	Visualisation of papillaries to overlay on image.
<code>Parent</code>	Number of the image stack from which this stack was derived.
<code>PathName</code>	Path to where the image is saved.

PatientInfo

- **Name** is the name of the patient.
- **ID** is the PatientID.
- **BirthDate** in the date format 'YYYYMMDD'.
- **Sex**. Applied values are '', 'M', 'F', '-'.
- **Age**. Applied values are [], '', numeric, '78Y'.
- **HeartRate**. Same as SET(NO).HeartRate, retained for backwards compability.
- **AcquisitionDate**. Applied values, '', [], and 'YYYYMMDD'.
- **BSA**. Applied values 0 or numeric value. Manually entered or automatically calculated from Weight and Length. See Reference manual for details and equations used.
- **Weight**. Measured in kilograms. Applied values are 0 or numeric value.
- **Length**. Measured in centimeters. Applied values are 0 or numeric value.

Perfusion	This is a struct containing information for perfusion analysis.
Point	Struct that contains data of annotation points.
ProgramVersion	Describes the version of which the set struct was created. Used for backwards compability issues, and also to detect potential forward compability issues.
RVEDV	Right ventricle end diastolic volume, see also EDV.
RVEF	Right ventricle ejection fraction, see also EF.
RVEPV	Right ventricle epicardial volume, see also EPV.
RVESV	Right ventricle end systolic volume, see also ESV.
RVEndoInterpX	RV interpolation points.

RVEndoInterpXView	RV interpolation points.
RVEndoInterpY	RV interpolation points.
RVEndoInterpYView	RV interpolation points.
RVEndoX	Endocardial contour of the right ventricle. See also EndoX .
RVEndoXView	Same as EndoXView , but for the right ventricle endocardium.
RVEndoY	See above.
RVEndoYView	See above.
RVEpiInterpX	RV epicardial interpolation points.
RVEpiInterpXView	RV epicardial interpolation points.
RVEpiInterpY	RV epicardial interpolation points.
RVEpiInterpYView	RV epicardial interpolation points.
RVEpiPinX	RV epicardial interpolation points.
RVEpiPinXView	RV epicardial interpolation points.
RVEpiPinY	RV epicardial interpolation points.
RVEpiPinYView	RV epicardial interpolation points.
RVEpiX	Epicardial contour of the right ventricle. See also EpiX .
RVEpiXView	Same as EpiXView , but for the right ventricle endocardium.
RVEpiY	See above.
RVEpiYView	See above.
RVM	Right ventricle mass, see also LVM .
RVSV	Right ventricle stroke volume, see also SV .

<code>RVV</code>	Right ventricle volume. See <code>LVV</code> .
<code>RepetitionTime</code>	Repetition time in MRI pulse sequence. Read from DICOM tags if presented otherwise set to zero. Given in milliseconds.
<code>Report</code>	Field used to contain information for the patient report sheet generator functionality. This field is only set for the first image stack (i.e <code>SET(1)</code>). The exact content of this struct is subject to change.
<code>ResolutionX</code>	Contains pixelspace in mm in X-direction. For discussion on coordinate systems, see Section 6. <code>ResolutionY</code> gives pixelsize in mm in Y-direction.
<code>Roi</code>	a struct that contains information of store region of interest (ROI).
<code>RoiCurrent</code>	number of current Roi
<code>RoiN</code>	number of Rois in total in the image
<code>Rotated</code>	True for image stacks that are rotated around a common rotation axis. This is currently implemented as an add-on and only affects the volume calculation. Currently 3D visualization does not take this flag into account. This will be addressed in future versions.
<code>RotationCenter</code>	Position of the rotation center in the image stack (y-coordinate).
<code>SV</code>	Left ventricle stroke volume.

Scanner	Identified scanner identified by parsing the DICOM tag Manufacturer . Presented values are: <ul style="list-style-type: none"> • ADAC • Bruker • GE • Philips • Siemens • Suinsa • Toshiba
Scar	This is a struct containing information for viability analysis.
SectorRotation	Rotation of the Sector compartmentization of the left ventricle used for regional myocardial analysis. Unit is degrees. This is can be manually adjusted from the bulls-eye GUI.
SequenceName	Name of the sequence, taken from DICOM tags.
SeriesDescription	Series description of the image serie, taken from DICOM tags.
SeriesNumber	Series Number taken from DICOM tags.
SliceGap	Gap between slices in millimeters (if present). Read from DICOM file by looking at the tag SpacingBetweenSlices if present. If not present, then the DICOM tags ImageOrientation and ImagePosition plus SliceThickness from above are used.
SliceThickness	Thickness of slices in millimeters. Read from DICOM file by looking at the tag SliceThickness . If not present, then the tags ImageOrientation and ImagePosition are used.
SpectSpecialTag	Read from DICOM file for determination of the image tag ImageType in SPECT images. Valid inputs are 'Rest', 'Rest Prone', 'Stress' or 'Stress Prone'.

<code>StartAnalysis</code>	Time frame for start of analysis, used in flow quantification. See Reference Manual for details. Default value is 1.
<code>StartSlice</code>	This field contains the first slice in the set of selected slices. For more details, see below.
<code>Strain</code>	This is a struct containing information for strain analysis from velocity encoded MR images.
<code>StrainTagging</code>	This is a struct containing information for strain analysis from tagged MR images.
<code>Stress</code>	Reserved for future usage.
<code>StudyID</code>	StudyID taken from DICOM tags.
<code>StudyUID</code>	StudyUID taken from DICOM tags.
<code>TDelay</code>	Trigger delay (i.e starting of image acquisition after trig pulse). Read from DICOM tags if presented otherwise set to zero. Given in seconds. Currently unused.
<code>TIincr</code>	Time increment in seconds between time frames. Must be non zero when image stack is time resolved. If increment between time frames is not uniform, the value contained here is the mean time increment.
<code>TimeVector</code>	Vector containing time position of each timeframe.
<code>TSize</code>	Size of image stack in temporal direction. The following two expressions are equivalent <code>SET(NO).TSize</code> and <code>size(SET(NO).IM,3)</code> .
<code>VENC</code>	Velocity encoding range in MRI pulse sequence. Read from DICOM tags if presented otherwise set to zero. Given in centimeters per second.
<code>View</code>	Struct that stores information of the current view. Used to retain the same view after saving an image stack. Only <code>SET(1).View</code> is filled.

XMin	When loading images it is possible to crop the images. XMin contains the number of pixels that the contours are translated due to cropping compared to the original image size. When no cropping is performed XMin is 1.
XSize	Size of image stack in X-direction. The following two expressions are equivalent SET(NO).XSize and size(SET(NO).IM,1) .
YMin	Same as XMin except relates to the Y-direction.
YSize	Size of image stack in Y-direction. The following two expressions are equivalent SET(NO).YSize and size(SET(NO).IM,2) .
ZSize	Size of image stack in Z-direction (number of slices). The following two expressions are equivalent SET(NO).ZSize and size(SET(NO).IM,4) .

11 Implementation Details

11.1 Version handling

A proper version handling is employed when developing Segment. A detailed version history of Segment is found in the revision log of Segment SVN.

11.2 Numeric representations

All numbers are stored and used internally as double precision floating points with the following exceptions:

- Images are stored as single floats (normalized) or as integers (uint8), and then as they are stored in the DICOM files. Most functions in Segment will automatically convert the data to floats.
- Edge detection results are stored as integers (16 bits, 'normalized')
- Character strings are stored in 8bit ASCII format
- Infarct maps are stored as int8 (manual interaction), and uint8 (result).
- General segmentation tool store objects as levelset function with an uint8 representation where the zero levelset resides at 128.

Internally the image stack is normalized upon loading by a global maximum intensity such that all values are $[0..1]$. Offset and scaling is also calculated so that the image stack can be reconverted back to original signal intensities.

11.3 Loading data and interpretation of DICOM tags

This section describes how Segment interprets DICOM information to calculate important parameters such as geometric properties of the images.

- Number of slices. This is calculated from the presence of different slices based on the DICOM tags `ImagePosition` and `ImageOrientation`.
- Number of timeframes. This is based on dividing the total number of images with the number of slices.
- Time increment in ms between each timeframe. If uniform, this is based on the difference between the number of timeframes divided by largest and the smallest value of the DICOM tag `TriggerTime`. If the DICOM tag `TriggerTime` is not present then the DICOM tag `TR` is used as time increment. Note that this might depend on your k-space acquisition scheme so for scanners that do not report `TriggerTime` you really need to double check the estimated value of time increment. For perfusion and other image stacks with non-uniform time increment, this is calculated using differences in `AcquisitionTime`.

- Heart rate. The heart rate is taken from the DICOM tag `HeartRate` if present. Note that many vendors (including Siemens) does not specify this. As a fall back Segment tries to calculate the heart rate assuming full R-R intervall coverage by using of trigger time (i.e it does not working for prospective imaging series). For long image acquisitions where one image is taken approximately for each heart beat then the heart rate is taken as the time between start of image acquisition and end of image acquisition adjusted for the number of frames. Note that in many cases this heart rate calculation will fail. Heart rate can be adjusted under patient details. Note also that heart rate may vary between image stacks therefore do not press Apply for all when manually changing heart rate. Heart rate is not used in any calculaion, instead time increment between image frames is used in all calculations.
- Slice thickness in mm. The slice thickness is taken from the DICOM tag `SliceThickness`. If this tag is not present then the information is taken from same DICOM tags as number of slices, and assuming slice gap to be 0.
- Gap between slices in mm. This is taken from the DICOM tag `Spacing BetweenSlices`.
- Pixelspacing in X-direction in mm (vertical direction in Segment). This is taken from the DICOM tag `PixelSpacing`.
- Pixelspacing in X-direction in mm (horisontal direction in Segment). This is taken from the DICOM tag `PixelSpacing`.
- Velocity encoding (VENC) in cm/s. For non velocity encoded images this should be 0. How this is interpreted involvs proprietry information of different scanner vender information.
- Rotated image stack. This should by default be false. If your image stack is rotated, then change this to true. Currently this parameter is not taken from information in the DICOM tags and the user needs to manually change this when loading rotated image stacks.
- Cyclic image. If the image stack is cyclic, i.e covers the whole heart cycle this should be true (default). For prospectively gated image series this should be false. This affects mainly the automated segmentatin algorithm. Currently this information is not read from the DICOM information.

11.4 Volume calculations

The volume calculations are done by a summing the area in each slice. The main reason for not using a more advanced volume integration method is that no one else is using that and therefore it might be difficult to compare the results. Segmentation (i.e. delineation of endocardium and epicardium) is stored on a sub-pixel accuracy and subsequent calculations are on a sub-pixel basis. For viability the classification into viable or scar is done on a pixel-wise basis and there the volume calculations are done by summing the number of pixels.

For the rotated image stacks the volume is given by a integration method. The volume contribution of each outline is given by :

$$\delta V = \frac{\pi}{2 * Z} \int y(s)^2 \text{sign}(y(s)) \frac{dx}{ds} ds \quad (3)$$

where the curve is given on a parametric representation $(x(s), y(s))$, Z is the number of slices in the rotated image stack. No long-axis compensation is performed for the rotated image stacks.

11.5 Mass calculations

When converting volume to mass the density is assumed to be 1.05 g/ml. Note that this number differs in the literature between 1.04 to 1.05. Furthermore, note that these numbers are valid for healthy myocardium ex-vivo, what happens in for instance infarcted regions is not shown in the literature. Therefore usually it is better to report volume instead of mass.

11.6 Calculation of BSA

The formula used is based on Mosteller.

$$BSA = \sqrt{\frac{w * h}{3600}} \quad (4)$$

where w is the body weight in kg, and h is height in cm.

11.7 Peak ejection/filling rate

When calculating peak ejection and peak filling rate the volume curve is differentiated using forward difference approximation. For cyclic datasets cyclic convolution is used for the calculation.

11.8 Wall thickness

Currently wall thickness is defined as the thickness along a radial spike from the endocardial or the epicardial center (depending on setting in the preferences. In the future I plan to also include the modified center line method. Note that the centers are calculated for each timeframe separately.

Wall thickening is defined as the wall thickness in end-systole minus the wall thickness in end-diastole. Note that it is possible to manually or automatically select what timeframes that are diastole or systole respectively.

Fractional wall thickening is defined as:

$$WT_f = \frac{WT - WT_{ED}}{WT_{ED}} \quad (5)$$

Where WT_f is fractional wall thickness and WT is wall thickness and WT_{ED} is wall thickness in end-diastole. In the bulls eye plot then fractional wall thickening is showed in end-systole.

11.9 Calculation of regurgitant volumes and shunts

The regurgitant fraction for the aortic valve and the pulmonary values are calculated as:

$$r = 100 \frac{v_{back}}{v_{forward}} \quad (6)$$

where r is regurgitant fraction, v_{back} is backward volume, and $v_{forward}$ is forward volumes. Backward volumes is taken as timeframes where the net flow is negative and integrated over the entire cardiac cycle.

The regurgitant fraction for the tricuspid and mitral valve are calculated as:

$$r = 100 \frac{SV - v_{forward}}{SV} \quad (7)$$

where r is regurgitant fraction, and SV is stroke volume for left or right ventricle, respectively. $v_{forward}$ is forward volume.

The Q_p/Q_s ratio is defined as

$$Q_p Q_s = \frac{Q_p}{Q_p} \quad (8)$$

where Q_p is the stroke volume of the pulmonary artery and Q_s is the stroke volume of the aortic artery.

11.10 Infarct size, extent and transmuralty

Calculations of infarct sizes etc are based on 'counting' pixels, i.e. each pixel has a binary classification. There are two methods for regional analysis available, one are based where the percentage of the pixels that are inside the sector. The other method is based on radial spikes from the center (endo- or epicardial depending on setting in the preferences). The line between endocardium and epicardium is resampled in 50 steps and the percentage of infarcted pixels are counted.

Infarct extent is defined as the projected infarcted area on the endocardial surface [4].

$$I_{ext} = \sum_i \frac{T_i R_i}{R_i} \quad (9)$$

where I_{ext} is the infarct extent, T_i is the transmuralty of sector i and R_i is the mean endocardial radius of sector i .

11.11 Number of SD from remote for Scar

The number of SD from remote for an existing scar segmentation is calculated by the function found in the main menu in Segment under MR menu Viability menu and then the menu option Get SD from Remote. The presented value is calculated by first calculate the mean and sd in

the remote area ($Mean_{remote}$ and SD_{remote}). If there exist ROIs named **Remote ROI**, these regions define the remote area. Otherwise the whole myocardium except for the scar region defines the remote area. The presented SD from remote value is then calculated by

$$SD_{fromRemote} = \frac{T_{optim} - Mean_{remote}}{SD_{remote}} \quad (10)$$

The optimal threshold value (T_{optim}) represent the optimal threshold for seperating the remote and the scar regions based on the existing scar segmentation. This value is defined by an exhaustive search where the threshold is set to all intensities represented in the image stack. For each threshold, the number of missclassified pixels are counted (total of both missclassified remote pixels and missclassified scar pixels). The optimal threshold value is then defined as the threshold corresponding to the minimal number of missclassified pixels.

11.12 MR relaxometry calculations

The MR relaxometry calculation for T1/T2 mapping is given in the paper [5]. For clarification, the implementation of Look Locker correction uses the standard formula as is used in previous literature: $T1 = T1 * (B/A - 1)$. The formula in [5] gives the same result as the standard formula. Implementation of the ADAPTS T2* mapping is given in the paper [6].

11.13 Pulse wave velocity

The implementation of the pulse wave velocity unit is described in the paper [7].

11.14 Torsion

In short axis cardiac images the heart muscle wall of the left chamber is well approximated by a circle. The method finds the axis of rotation, AoR, for the left chamber as the center of a circle fit to the tracking points generated by the segment strain module. For the circle fitting a least squares method is used.

11.14.1 Least squares circle fit

The circle is fitted by minimizing the global squared radial difference between all tracking points for all timeframes, (x_i, y_i) , $i = 1, \dots, N$ and a circle with radius $r = \sqrt{a}$ for each slice. For nicer calculations we make the tracking point cloud zero mean and define a new coordinate system

$$u = x - \frac{1}{N} \sum_i x_i, \quad v = y - \frac{1}{N} \sum_i y_i \quad (11)$$

The properties of the circle determining the fit is the radius r and center (u_c, v_c) The circle equation we are going to work with is

$$f(u, v) = (u - u_c)^2 + (v - v_c)^2 - a = 0 \quad (12)$$

which yields the least squares expression we want to minimize.

$$M(a, u_c, v_c) = \sum_i^N f^2(u_i, v_i) = \sum_i^N ((u_i - u_c)^2 + (v_i - v_c)^2 - a)^2 = 0 \quad (13)$$

The minima is found by solving,

$$\frac{dM}{da} = 0 \quad (14)$$

$$\frac{dM}{du_c} = 0 \quad (15)$$

$$\frac{dM}{dv_c} = 0 \quad (16)$$

for all parameters of M . From (14) we get that

$$\frac{dM}{da} = 2 \sum_i^N f(u_i, v_i) \frac{df(u_i, v_i)}{da} = -2 \sum_i^N f(u_i, v_i) = 0. \quad (17)$$

Resulting in

$$\frac{dM}{da} = 0 \iff \sum_i^N f(u_i, v_i) = 0. \quad (18)$$

Then consider (15). As (15) (16) only differ in notation, any result for (15) is applicable to 16.

$$\frac{dM}{du_c} = 2 \sum_i^N f(u_i, v_i) \frac{df(u_i, v_i)}{du_c} = 4 \sum_i^N (u_i - u_c) f(u_i, v_i) \quad (19)$$

Since 18,

$$\frac{dM}{du_c} = 0 \iff \sum_i^N u_i f(u_i, v_i) = 0. \quad (20)$$

and the same goes for 16.

$$\frac{dM}{dv_c} = 0 \iff \sum_i^N v_i f(u_i, v_i) = 0. \quad (21)$$

expanding equation (20) yields

$$\frac{dM}{du_c} = \sum_i^N u_i (u_i^2 - 2u_i u_c + u_c^2 + v_i^2 - 2v_i v_c + v_c^2 a) = 0 \quad (22)$$

Define $S_u = \sum_i^N u_i$ and $S_v = \sum_i^N v_i$ then

$$\frac{dM}{du_c} = S_u^3 - 2u_c S_u^2 + u_c^2 S_u + S_{uv^2} - 2v_c S_{uv} + v_c^2 S_u - a S_u = 0 \quad (23)$$

In making the coordinates zero mean $S_u = 0$ we get the equation

$$u_c S_{u^2} + v_c S_{uv} = \frac{1}{2}(S_{u^3} + S_{uv^2}) \quad (24)$$

After doing the same for (21) we obtain the system

$$\begin{cases} u_c S_{u^2} + v_c S_{uv} = \frac{1}{2}(S_{u^3} + S_{uv^2}) \\ u_c S_{uv} + v_c S_{v^2} = \frac{1}{2}(S_{v^3} + S_{vu^2}) \end{cases} \quad (25)$$

which can be converted into a matrix equation

$$\begin{bmatrix} S_{u^2} & S_{uv} \\ S_{uv} & S_{v^2} \end{bmatrix} \begin{bmatrix} u_c \\ v_c \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(S_{u^3} + S_{uv^2}) \\ \frac{1}{2}(S_{v^3} + S_{vu^2}) \end{bmatrix} \quad (26)$$

This gives us an easy way to get the least squares fitted circle center. For the center in the original (x, y) domain translate with the previously subtracted mean. Finally for the radius, expanding equation (18) and simplifying yields

$$a = u_c^2 + v_c^2 + \frac{S_{u^2} + S_{v^2}}{N}, \quad (27)$$

where

$$r = \sqrt{a}. \quad (28)$$

11.14.2 Angular discontinuity detection

After fitting a circle to each time frame with tracking points we can translate the points in each time frame so that the fitted circle center i.e the AoR is in origo. With this in place a polar coordinate change results in an approximate line like formation of the points, lets call it a worm. Who's movement along the θ axis is the rotation of the heart muscle. Here a problem arises. Since $\theta \in [-\pi, \pi]$, $\theta_t + \Delta\theta > |\pi|$ results in a sign change and the point appears at the lower limit if it passed the upper and vice versa. This needs to be mended if we are to measure angular distance from a starting point. This is done by examining

$$\Delta\theta_t = \theta_t - \theta_{t+1} \quad (29)$$

for each tracking point, adjusting the point with $\pm\pi$ (sign depends on border transition) if $|\Delta\theta_t| > 5$.

Torsion is then found as the difference between the rotation in a apical and a basal slice normalized with the distance along the long axis of the heart between the slices and the mean radius.

11.15 Longaxis volumes

Volumes can be calculated using segmentation from longaxis images. The algorithm begins with automatically locating images labeled 2CH, 3CH and 4CH that contain segmentation.

If the same kind of segmentation is found in two such images, the volume is calculated by rotating each segmentation area one full revolution around the axis of intersection and taking the mean of these volumes. If there are three images that contain the same segmentation, the volumes are calculated as described above for each pair of images, and the mean of these three values is used.

12 How to Create Own Plug-ins

The easiest method of how to learn to make own plug-ins is to study the example `plugin_template.m`. Writing own plug-ins is a great way of spreading your algorithms to users all over the world and also to contribute to the Segment project.

A plugin file must have a name beginning with `plugin_*.m`. Note that the plugin may of course call other functions that can reside anywhere on the Matlab path. When Segment is started the current folder is scanned for functions with this pattern. Matching functions are called with the argument `getname` and a string with the name that should appear in Segment menu is expected.

Currently there are two other plugins that are shipped with the standard Segment edition:

- `plugin_imageloader.m`. Plugin to load non DICOM images into Segment. This plugin can load for instance `.jpg`, `.bmp`, `.tif`, `.png` files into Segment. This plugin gives some elementary details on the internal data structure.
- `plugin_calibrate.m`. Plugin to calibrate image resolution. This plugin gives some hints on using own GUI's and also some details about the internal data structure in Segment.
- `plugin_template.m`. Template plugin, simple template for creating plugins.
- `plugin_summarize.m`. Plugin to summarize results from multiple `.mat` files. This function is useful to read when creating own export scripts.
- `plugin_phaseflow.m`. Plugin to enable to make flow measurement when there is only phase images available.

For further documentation of the two first plug-ins, please see the Segment User Manual.

13 Software support

On Medviso website you find solutions to the most frequently asked questions on the page <https://medviso.com/faq/>.

If you have any questions or support requests about our software products, please send us request to support@medviso.com.

We love to get feedback and are happy to hear from you about new software feature request and any potential question or software issue. We see our users as collaboration partners and always do our best to meet your requests. You can at any time upgrade to the latest software version found at our download page. Our unique software development platform allows us to have a quick turnaround time and provide updated software versions in a short time.

13.1 How to make a support request

If the support request is related to a data set, the easiest way to send us a support request is to use the **Support Request** feature under the **Help** menu in the software. Fill in the form and do not forget to attach the DICOM / .mat-file associated with the support request. You can also send your support questions to support@medviso.com. As applicable, please include the following:

- Issue description
- Error log
- DICOM data
- Analyzed files (.mat-files)

Even though the submitted files are encrypted your files should be pseudonymized. Pseudonymization of .mat files and DICOM files are available under the **Utilities** menu. If you have large data files to attach, you can send the data with Medvisos ftp account or file sharing services such as Dropbox, WeTransfer, etc. Email us to support@medviso.com for getting login details to our FTP server.

13.2 Additional support

As we have several hundreds of research group using the software daily, the time for supporting the free research only version of Segment is limited. However, we encourage all users to submit any possible questions or problems and we'll try our best to help out. Researchers can purchase an academic support contract for additional software support. Please contact sales@medviso.com for further details or to request a quote.

Bibliography

- [1] E. Heiberg, J. Sjogren, M. Ugander, M. Carlsson, H. Engblom, and H. Arheden. Design and validation of Segment—freely available software for cardiovascular image analysis. *BMC Med Imaging*, 10:1, 2010.
- [2] E. Heiberg. *Automated Feature Detection in Multidimensional Images*. PhD thesis, 91-85297-10-0. Linkoping universitet, Department of Biomedical Engineering, 2004.
- [3] E. Heiberg, L. Wigstrom, M. Carlsson, A. F. Bolger, and M. Karlsson. Time Resolved Three-dimensional Automated Segmentation of the Left Ventricle. In *IEEE Computers in Cardiology 2005*, volume 32, pages 599–602, Lyon, France, 2005.
- [4] H. Engblom, M. B. Carlsson, E. Hedstrom, E. Heiberg, M. Ugander, G. S. Wagner, and H. Arheden. The endocardial extent of reperfused first-time myocardial infarction is more predictive of pathologic Q waves than is infarct transmuralilty: a magnetic resonance imaging study. *Clin Physiol Funct Imaging*, 27(2):101–8, 2007.
- [5] S. Bidhult, G. Kantasis, A. H. Aletras, H. Arheden, E. Heiberg, and E. Hedstrom. Validation of T1 and T2 algorithms for quantitative MRI: performance by a vendor-independent software. *BMC Med Imaging*, 16(1):46, 2016. Bidhult, Sebastian Kantasis, George Aletras, Anthony H Arheden, Hakan Heiberg, Einar Hedstrom, Erik England BMC medical imaging BMC Med Imaging. 2016 Aug 8;16(1):46. doi: 10.1186/s12880-016-0148-6.
- [6] S. Bidhult, C. G. Xanthis, L. L. Liljekvist, G. Greil, E. Nagel, A. H. Aletras, E. Heiberg, and E. Hedstrom. Validation of a new t2* algorithm and its uncertainty value for cardiac and liver iron load determination from MRI magnitude images. *Magn Reson Med*, 75(4):1717–29, 2016.
- [7] K. Dorniak, E. Heiberg, M. Hellmann, D. Rawicz-Zegrzda, M. Wesierska, R. Galaska, A. Sabisz, E. Szurowska, M. Dudziak, and E. Hedstrom. Required temporal resolution for accurate thoracic aortic pulse wave velocity measurements by phase-contrast magnetic resonance imaging and comparison with clinical standard applanation tonometry. *BMC Cardiovasc Disord*, 16(1):110, 2016. Dorniak, Karolina Heiberg, Einar Hellmann, Marcin Rawicz-Zegrzda, Dorota Wesierska, Maria Galaska, Rafal Sabisz, Agnieszka Szurowska, Edyta Dudziak, Maria Hedstrom, Erik England BMC cardiovascular disorders BMC Cardiovasc Disord. 2016 May 26;16(1):110. doi: 10.1186/s12872-016-0292-5.