



Atracsys LLC

Route du Verney 20
1070 Puidoux
Switzerland

www.atracsys-measurement.com
Atracsys-Support@smith-nephew.com

Tel +41 (0)21 533 09 00

fusionTrack 250

User manual

File name	fTk250_InstructionsHowToUse.pdf
Document type	Manual
Version	v4.8.1
Revision	9601653
Pages	134
Date	9th May 2023



Contents

Glossary	5
Disclaimer of Warranties and Limitations of Liability	7
1 Important information	8
1.1 Warnings	8
1.2 Caution	10
1.3 Safety signs and symbols	10
1.4 Device label	11
1.5 Disclaimers	11
1.6 Contact information	11
1.7 Updates	11
2 Introduction	12
2.1 Device expected service life	12
2.2 Device shelf life	13
2.3 Device specifications	13
3 Description of the fusionTrack system	15
3.1 Hardware requirements	15
3.2 The fusionTrack device	16
3.2.1 Power input	16
3.2.2 Cables	19
3.2.3 Status LEDs	19
3.3 The fusionTrack software development kit	19
3.4 Authorised modifications	20
4 Shock sensor	21
4.1 Registers	21
4.1.1 Status register	21
4.1.2 Calibration register	22
4.1.3 Error register	22
5 Hardware installation	24
5.1 Operating environment requirements	24
5.2 Least favorable working conditions	25
5.3 Transportation and storage environmental conditions	25
5.4 Mounting the fusionTrack device	26
5.5 Network connection of the fusionTrack device	26
5.6 Connecting and powering the fusionTrack	27

5.6.1	The fusionTrack	27
5.6.2	fusionTrack in a ME system	29
5.6.3	Annex: Explanation of applied part	30
6	Software installation	31
6.1	Windows installation	31
6.2	Linux installation	31
7	How the fusionTrack device works	32
7.1	Communicating with the fusionTrack device	32
7.2	The fusionTrack device coordinates system and working volume	32
7.3	Information provided by the fusionTrack device	35
7.3.1	The fusionTrack events	37
7.4	Fiducial detection and marker tracking	37
7.4.1	Active fiducials	38
7.5	Temperature compensation	38
7.6	The fusionTrack device markers	39
7.6.1	Passive markers	41
7.6.2	Active markers	41
7.7	Marker geometry files	43
7.7.1	INI files version 0	43
7.7.2	INI files version 1	44
7.7.3	Binary files version 1	46
7.8	Marker tracking parameters	46
7.9	Phantom fiducials	47
7.10	Stray fiducials	48
8	Using the fusionTrack system	49
8.1	The fusionTrack system options	50
8.1.1	Library related options	50
8.1.2	Device related options	51
8.1.3	Detection stage related options	54
8.1.4	Wireless related options	56
8.1.5	Authentication related options	56
8.1.6	Environment and options	57
8.2	User interface software 'demo.exe'	57
8.2.1	Recalibration of a marker	62
8.2.2	Exporting the data	62
8.2.3	Dumping the data	63
8.3	Command line software samples	64
8.4	Latency measuring	67
8.4.1	Running the software	68
8.5	Compilation of provided samples	69
8.5.1	Windows compilation	69
8.5.2	Unices compilation	70
8.6	SDK configuration file	70
9	Language bindings	73
9.1	Python wrapper	73

9.2 Matlab wrapper	73
10 Command line utility tool	76
10.1 Establishing a connection with <code>atnetExecutable64</code>	76
10.2 Usefull commands	76
10.2.1 <code>info32</code>	76
10.2.2 <code>info_uC</code>	77
10.3 Updating the firmware of the fusionTrack	78
10.4 Updating the firmware of the fusionTrack shock detection unit	79
10.5 Updating the calibration of the wireless markers	79
10.6 Updating the firmware of the wireless markers	80
10.7 Obtaining a shock report	81
11 Control the operability of the fusionTrack system and instruments	82
11.1 Control fusionTrack operability	82
11.2 Control marker registration and epipolar errors	83
11.3 Control fusionTrack temperature	83
11.4 Control optical elements and IR illumination	84
11.5 Control environmental conditions	84
11.6 Integrator software	85
12 The zeroing method	87
12.1 General concept	87
12.2 Running the procedure	88
13 The accuracy verification tool (AVT)	89
13.1 General concept	89
13.2 Running the procedure	89
13.3 The accuracy verification tool from Atracsys	89
14 Electromagnetic compatibility	90
14.1 Essential performances	92
14.1.1 Quality of the measurement	92
14.1.2 Measurement rate	93
14.2 Cables	94
15 Communication parameters	95
15.1 Changing the IP	95
15.2 Changing the local UDP port	97
15.3 Changing the maximal MTU value	97
15.4 Connection to multiple fusionTrack devices	98
16 The fusionTrack device authentication	99
16.1 Implementation	99
16.2 Setting the key in the fusionTrack device	99
16.3 Checking the fusionTrack device	100
17 Maintenance	103
17.1 Cleaning the fusionTrack device	103
17.2 Sterilisation of the markers	103

17.2.1	Cleaning the fusionTrack device filters	104
17.3	Effects of multiple cleanings	105
17.4	Disposal of equipment	105
18	Troubleshooting	106
18.1	Common problems	106
18.1.1	Wrong or damaged cables	106
18.1.2	PoE+ problems	106
18.1.3	FTK_WAR_SHOCK_DETECTED warning	106
18.1.4	FTK_WAR_SHOCK_SENSOR_OFFLINE warning	107
18.1.5	FTK_WAR_SHOCK_SENSOR_AUTO_ENABLE warning	107
19	Licences	108
19.1	Atracsys fusionTrack SDK headers and library	108
19.2	Hashing code	108
19.3	About JsonCpp	108
19.4	About Qt	109
19.4.1	LGPL version 3	110
19.5	About QDarkStyle	112
19.5.1	The MIT License (MIT) - Code	112
19.5.2	Creative Commons Attribution International 4.0 - Images	113
19.5.3	Creative Commons Attribution 4.0 International Public License	113
19.6	Base 64	118
19.7	About FLTK	119
A	Mathematical considerations	120
A.1	Homogeneous coordinates	120
A.2	Marker pose	120
A.3	Inverting a marker pose	121
B	Warning summary	122
	References	134

Glossary

Accuracy Verification Tool

The Accuracy Verification Tool (AVT) is a product designed by Atracsys. It allows the user to check the trueness of an optical tracking device. It consists of an artefact, an acquisition software, as well as a processing software running on Atracsys servers, in case a detailed report is asked for.

API

Application Programming Interface. It is a set of routines, protocols, and tools for building software applications.

Fiducial

Simplest element (sphere covered with reflective material, sticker with reflective material, reflective glass sphere, IR-LED) detected by the optical tracking system, and reconstructed as a three-dimensional point (e.g. (x, y, z) coordinates).

GUI

Graphical User Interface, this is a type of interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation.

IGS

Image-Guided Surgery. This is the general term used for any surgical procedure where the surgeon employs tracked surgical instruments in conjunction with preoperative or intraoperative images in order to guide the procedure.

JSON

In computing, JavaScript Object Notation or [JSON](#) is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value).

Marker

Also often called rigid body. Mechanical structure with known geometry on which several fiducials are firmly attached. As soon as the tracking system detects and reconstructs at least three fiducials, the six degrees of freedom are computed by the system.

Navigation system

A navigation system is composed of at least a tracking system and a computer on which an IGS software is running.

PD

Powered Device, when talking about a device powered by a PoE. The fusionTrack is a PD.

PoE

Power Over Ethernet, providing both an Ethernet connection and power supply using only one Ethernet cable.

PSE

Power Source Equipment, when talking about a device powered by a PoE. The PSE provides power to the PD.

Raw data

Image of a fiducial on a camera, i.e. a set of contiguous white pixels. Also called blob.

SDK

Software Development Kit, which is typically a set of software development tools that allows the creation of applications for a certain software package, software framework, hardware platform.

Working volume

Portion of space in which the markers can move whilst being detected by the tracking system.

Troughout this document, references to glossary entries will appear in *slanted font*.

Disclaimer of Warranties and Limitations of Liability

© Copyright 2004-2023, Atracsys

All rights reserved. No part of this document may be reproduced, transcribed, transmitted, distributed, modified, merged, translated into any language or used in any form by any means - graphic, electronic, or mechanical, including but not limited to photocopying, recording, taping or information storage and retrieval systems - without the prior written consent of Atracsys. Certain copying of the software included herein is unlawful.

Atracsys has taken due care in preparing this document and the programs and data on the electronic media accompanying this document including research, development and testing. This document describes the state of Atracsys's knowledge respecting the subject matter herein at the time of its publication and may not reflect its state of knowledge at all times in the future. Atracsys has carefully reviewed this document for technical accuracy. If errors are suspected, the user should consult with Atracsys prior to proceeding. Atracsys makes no expressed or implied warranty of any kind with regard to this document or the programs and data on the electronic media accompanying this document.

Atracsys makes no representation, condition or warranty to the user or any other party with respect to the adequacy of this document or accompanying media for any particular purpose or with respect to its adequacy to produce a particular result. The user's right to recover damages caused by fault or negligence on the part of Atracsys shall be limited to the amount paid by the user to Atracsys for provision of this document. In no event shall Atracsys be liable for special, collateral, incidental, direct, indirect or consequential damages, losses, costs, charges, claims, demands or claim for lost profits, data, fees or expenses of any nature or kind.

Important notice:

The listed items in this user manual might follow different medical or non medical certifications/ standards.

Although most Atracsys devices are following medical standards, they cannot be - by their own - certified as medical devices.

For medical applications, appropriate approvals must be obtained by the integrator. Atracsys LLC disclaims all liability.

1 Important information

This documentation provides detailed information about the fusionTrack system.

Information from this chapter should be read before continuing with the rest of the document. This document contains user information for the physical and software installation of the fusionTrack system.

1.1 Warnings



In the whole documentation, warnings are indicated with this symbol and graphics (triangle with exclamation mark). Integrator should follow the accompanying paragraph and/or report necessary warnings in final product's instructions to avoid patient or operator injury.



Integrator should read carefully this user manual before operating the device.
Follow all the installation procedures step by step.



The integrator has to check that the device / position of the device is well suited for the application / surgery and that required regulations are respected.



Optical hazards according IEC-62471-1: Exempt group

The fusionTrack device uses powerful infrared LEDs for illumination and communication. This infrared light at approximately 850 nm is not visible by human eyes. According to the standard IEC-62471-1 (hazard values are reported at a fixed distance $d = 200$ mm) the fusionTrack is classified in the 'Exempt group'. To further reduce exposure, never watch closely and longly into the infrared LEDs around the cameras when the device is switched on. To further reduce exposure of the skin, never place any body part closer than 200 mm to the infrared LEDs.



The emission of near infrared light for measurement and communication (for example for active markers) is the intended behavior of the optical tracking system. Interference with other equipment using infrared light might be possible. As an example, interference with certain types and brands of pulse oximeters have been reported. In order to avoid the pulse oximeters to be disturbed by the optical tracking system, the pulse oximeters should be shielded either by draping or by using the designated ambient light shields.



Do not immerse the fusionTrack device in water or other fluids.



Do not spill water or other fluids on the fusionTrack device.



The fusionTrack device is composed of metallic parts and electronic components and, even if complying with medical norms, it may interfere or be affected when used in special environments like Magnetic Resonance Imaging or devices generating X-Rays. Integrator should contact Atracsys if they intend to operate the fusionTrack in such environments.



Any noisy environment such as high electromagnetic fields (e.g. MRI) might trigger a measurement error. The operability verification procedure (see Chapter 11) must be proceeded to guarantee that the fusionTrack is operational in the configured environment.



The fusionTrack device is a precision instrument and has to be handled with greatest care. If it receives a shock, falls on any surface or is transported without adequate packaging, it can easily be damaged or decalibrated.



Since the fusionTrack will be integrated in a medical system, integrator has to conduct the tests related to the electrical security and electromagnetic compatibility for the whole system.

1.2 Caution



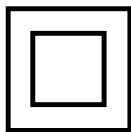
In the whole documentation, cautions are indicated with this symbol and graphics. The information from the accompanying paragraph must be followed to avoid damaging the equipment.

1.3 Safety signs and symbols

Various symbols are used in this manual, the user's manual and on the product itself to ensure correct usage, to prevent danger to the user and others, and to prevent property damage. The meanings of these symbols are described below. It is important that these descriptions are read thoroughly and the contents is fully understood.



The entire user manual must be read before operating the fusionTrack system.



The fusionTrack device is a Class II equipment. A Class II or double insulated electrical appliance is one which has been designed in such a way that it does not require a safety connection to electrical earth (ground).

IP20

The fusionTrack device degree of protection against dust and fluids (IEC standard 60529).

First digit: Solid particle protection

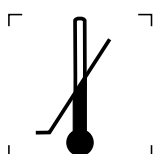
Level 2: Protected from touch by fingers and objects greater than 12 mm.

Second digit: Liquid ingress protection

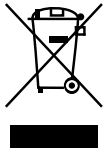
Level 0, Not protected from liquids.



A parcel containing a fusionTrack device must be handled with care.



A parcel containing a fusionTrack device must be transported and stored between the indicated temperatures. See Section 5.3.



The fusionTrack device must not be disposed of in the trash, see Section 17.4.

1.4 Device label

This section details the information printed on the label of the device. An example is provided on Figure 1.1. The label gives the serial number of the device, its version and the date of manufacture (format yyyy-mm-dd). The serial number is encoded in the [data matrix barcode](#) using ASCII character encoding, e.g. '0x000000000000000000'.

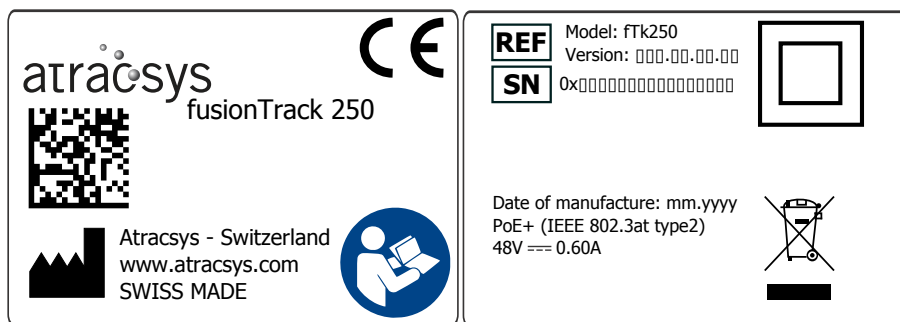


Figure 1.1: Blank example of a fTk 250 label.

1.5 Disclaimers

The entire user manual must be read before operating the fusionTrack system.

1.6 Contact information

For any question regarding the content of this guide or the operation of this product, please contact us:

Atracsys LLC

Route du Verney 20
1070 Puidoux
Switzerland

www.atracsys-measurement.com
Atracsys-Support@smith-nephew.com

Tel +41 (0)21 533 09 00

1.7 Updates

The latest version of this document can be downloaded from Atracsys website. The address, as well as the required username and password are provided by Atracsys, please refer to the 'Information for user' document.

2 Introduction

The fusionTrack systems are passive and active, real-time optical pose-tracking systems specially designed to detect and track reflective spheres, disks and IR-LEDs in real-time video streams. The fusionTrack device is composed of two cameras that observe reflective and/or active *fiducials* (IR LEDs) simultaneously, and it uses triangulation to calculate their locations with unrivalled precision and with an unparalleled non-interpolated measurement rate of 120 Hz. When several *fiducials* are affixed to a marker, the system can determine its pose (position and orientation) with 6 degrees of freedom ($x, y, z, \alpha, \beta, \gamma$).

The fusionTrack SDK enables access to data at different stages of processing, including raw images, individual 3D positions of *fiducials* (reflective spheres and disks / IR-LEDs) and on up to the pose of *markers*. The SDK also provides multi-level fault checking. It allows access to error information in real time at any processing stage: *fiducial* occlusion level, stereo de-calibration, *marker* registration error and more.

The fusionTrack can be customized to fit your requirements (e.g. precision level, acquisition speed, working volume, extensions). Moreover, the system is compatible with existing passive image-guided surgical (IGS) tools that are widely used in the medical field.

The fusionTrack system is one of the components of an IGS tool. The final (i.e. surgical) application will depend on the requirements of the integrator.

The fusionTrack measures distances and angles between *markers* fixed on the patient, as well as *markers* fixed on the surgical tools. Data generated by the fusionTrack are typically used by the IGS system to display information such as distances, angles, trajectories, position of surgical tools within preoperative images, etc. The entire IGS system (including the fusionTrack) helps surgeons to take decisions during the procedure.

The fusionTrack is not intended to make surgical decisions by itself or to replace surgical actions.

This document presents how to install and use the fusionTrack system. It is, as well as the current fusionTrack software, continuously improved and may differ from one version to another.

For any problem, question or suggestion do not hesitate to contact Atracsys either by email or by phone (see Section 1.6).

2.1 Device expected service life

The expected service life of the fusionTrack is eight years based on five hours of use per working day. However, the shock sensor battery should be recharged every 6 months, see Chapter 4 for more details.



The fusionTrack device must be powered off when not in use (e.g. using a switch before the *PoE+ PSE*). The powered-on time, be the fusionTrack device used or not, is cut away from the service life.

2.2 Device shelf life

The shelf life of the fusionTrack is four years based on five hours of use per working day. However, the shock sensor battery should be recharged every 6 months, see Chapter 4 for more details.

2.3 Device specifications

The specifications of the fusionTrack devices (model fTk250) are presented in Table 2.1.

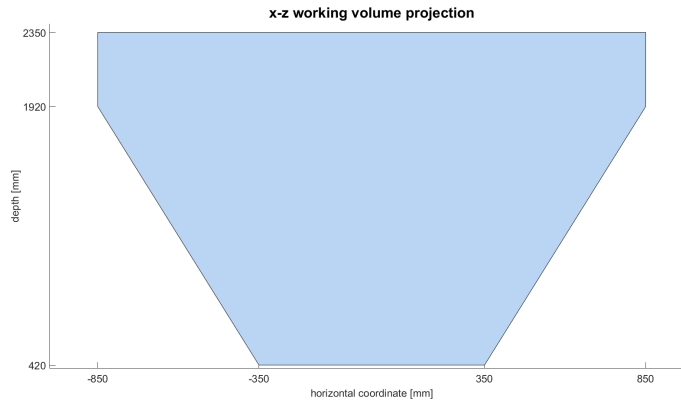


Figure 2.1: Accuracy Measurement volume, xz projection.

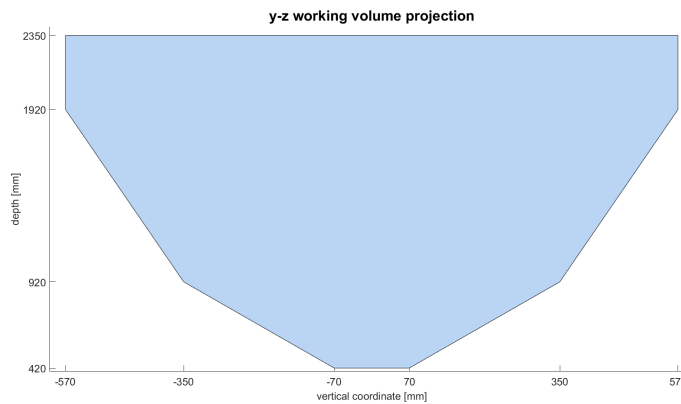


Figure 2.2: Accuracy Measurement volume, yz projection.

The volume presented on Figure 2.1 and Figure 2.2 is the *validated* volume, i.e. the volume in which the trueness of the fusionTrack device is checked. This volume is obtained from the smoothed union of all AVT boxes, as such, it is larger than the actual seen volume at low z . This is due to the fact the AVT boxes must be large enough to contain the AVT artefact with some additional tolerances, to accomodate for small field of view discrepancies.

Device	fusionTrack 250
Model	ftk250
Dimensions (L × W × H)	294 mm × 86 mm × 99 mm
Weight	1.28 kg
Unless otherwise specified, the fusionTrack 250 is factory released with a typical fiducial localisation error at calibration time ^d	Up to 1.4 m < 0.09 mm RMS / < 0.18 mm 95% CI ^a Up to 2.0 m < 0.20 mm RMS / < 0.40 mm 95% CI ^a Up to 2.4 m < 0.27 mm RMS / < 0.54 mm 95% CI ^a
Accuracy	Unless otherwise specified, the fusionTrack 250 is factory released with an <i>Accuracy Verification Tool (AVT)</i> distance RMS error smaller than 0.45 mm
Accuracy Measurement volume	Convex polyhedron with <i>xz</i> and <i>yz</i> projections shown in Figure 2.1 and Figure 2.2.
Working volume	Starts at 400 mm
Infrared illumination	~ 850 nm
Measurement rate	120 Hz ^b , native measurement rate (no upscaling)
Image exposition time	Active/Passive 0 µs to 250 µs
Latency	< 15 ms
Interface to PC	Gigabit Ethernet 1000BASE-T (IEEE 802.3ab) ^c
Max. simultaneous markers	16 markers
Max. fiducials per marker	6 spheres / discs / IR-LED per marker
Software Development Kit (SDK)	C/C++ (shared library)
Operating systems	Windows, Linux (see Section 3.1)
Power requirements	Power over Ethernet (PoE+ IEEE 802.3at-2009) 48 V 0.6 A
Pollution degree	2
Overvoltage category	I
Acoustic energy	< 80 dB
Standards	<ul style="list-style-type: none"> - Basic safety and essential performance: IEC 60601-1:2005+A1:2012, Ed 3.1 - Electromagnetic disturbances: IEC 60601-1-2:2014, Ed 4.0 - Photobiological safety of lamps: IEC 62471:2006, Ed 1.0 - Safety of laser products: IEC 60825-1:2014, Ed 3.0 - Medical device software: IEC 62304:2006+A1:2015, Ed 1.1 - Upon request, a CB-Report can be provided.

^a Confidence interval.

^b Depending on user's computer, network connection and numbers of blobs detected in the pictures.

^c Although the fusionTrack works with a 100Base-T network connection, the performance (framerate, delay) might be reduced.

^d Based on a single fiducial stepped uniformly throughout the measurement volume at 20 °C.

Table 2.1: fusionTrack specifications

3 Description of the fusionTrack system

The fusionTrack family of Optical Tracking Systems is a flexible, real-time tracking technology based on a stereo camera approach. The fusionTrack is based on measuring the 3D position of *fiducials* to provide real-time orientation and position of each *marker*.

For shock and temperature stability, the cameras inside the fusionTrack housing are floating (see Figure 3.1) and therefore the fusionTrack is not designed to measure absolute positions (positions and rotations in respect to the fusionTrack housing / the fixation base). The fusionTrack is designed to provide accurate relative measurements. 'Relative measurements' means that position and rotation from one respectively several markers to each other are computed.

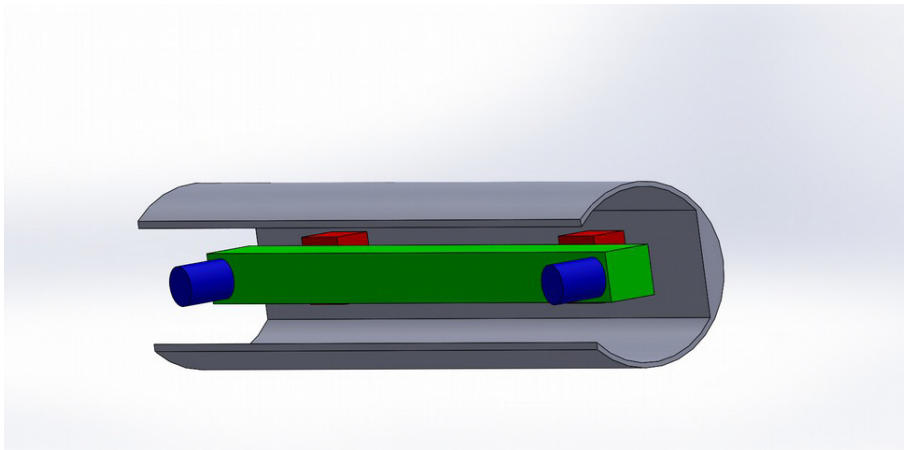


Figure 3.1: Sketch of the fusionTrack device. The cameras (in blue) are mounted on a rigid rod (in green). This rod is fixed to the 'shell' (grey) by rubber elements (red).

3.1 Hardware requirements

The minimum host PC requirement to use the fusionTrack system are:

- Intel(R) Core(TM) i3-6100U CPU @ 2.30GHz
- 4 GB DDR3 RAM:
- 50 MB (Windows) or 30 MB (Unix/Linux) disc space;
- Windows 8.1 (64 bits supported);
- Linux (64 bits supported), gcc 5.4 or clang 3.8.

3.2 The fusionTrack device

Here below is a simple description of its operation :

1. The fusionTrack illuminators emit infrared (IR) light, like the flash of a regular camera;
2. The IR light reflects on passive *markers* or triggers active *markers* which then emit IR light;
3. The fusionTrack detects the reflected (or emitted) IR light and transmits the information to the host computer along with status information;
4. The host computer extracts the position of each detected spot, computes the 3D position of each detected sources, then tries to match them to known *marker* geometries;
5. The information is forwarded to the user, with status information on each piece of data, allowing the user to do additional processing (display, filtering, collection, etc.).

The fusionTrack device can track two types of *markers*: passive and active ones. More detailed explanations about those two *marker* types are given in Section 7.6.

The fusionTrack camera is equipped with infrared interferometric bandpass filter. As the angle of incidence (AOI) has an impact on this type of filter (blue shift effect), for all AOI a bandpass interval of 840 nm to 900 nm is guaranteed. Light from a source in a near interval 740 nm to 1000 nm might be seen but the AOI will have an impact that should be evaluate by the user. Wavelengths far from this interval (outside the 740 nm to 1000 nm interval) will cut by the IR filer. The fusionTrack image sensor has a quantum efficiency response of $\sim 35\%$ at 840 nm and $\sim 21\%$ at 900 nm with an almost linear behavior between these two values.



The fusionTrack device must be powered off when not in use (e.g. using a switch before the *PoE+ PSE*). The powered-on time, be the fusionTrack device used or not, is cut away from the service life.

On Figure 3.2 is presented the front view of the fusionTrack. It should be noted that 'Left' and 'Right' always refer to the vision from the fusionTrack.

The back plane of the fusionTrack is presented on Figure 3.3. The Ethernet interface provides both communication and power. On older fusionTrack device versions, the device is switched on / off using the power switch (7). Finally the two LEDs indicate the status and the speed of the Ethernet communication.

The top view of the fusionTrack is presented on Figure 3.4.

3.2.1 Power input



The used power supply must be a limited power source. Any used power supply must not be able to deliver more than 100 W.



The integrator has to implement a mean to isolate the fusionTrack electrically from the supply mains.

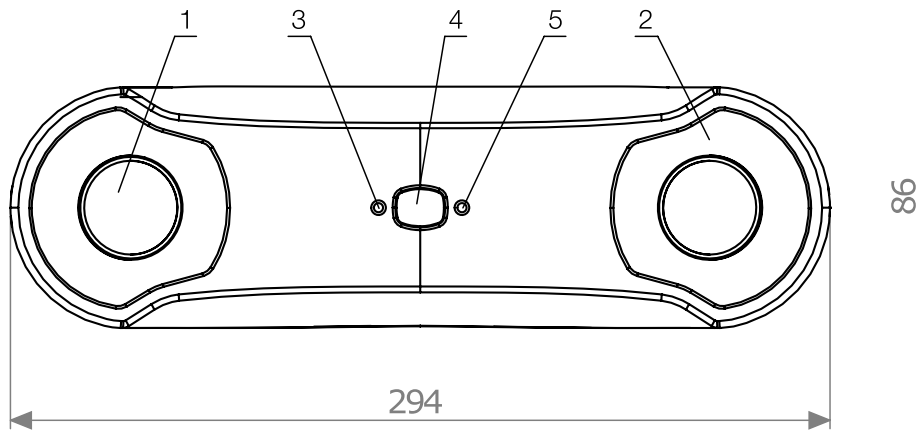


Figure 3.2: The fusionTrack front view, showing: (1) Camera sensor; (2) IR-LEDs ring; (3) User Status LED, refer to Section 3.2.3 for more information; (4) IR receiver; (5) Device Status LED, refer to Section 3.2.3 for more information.

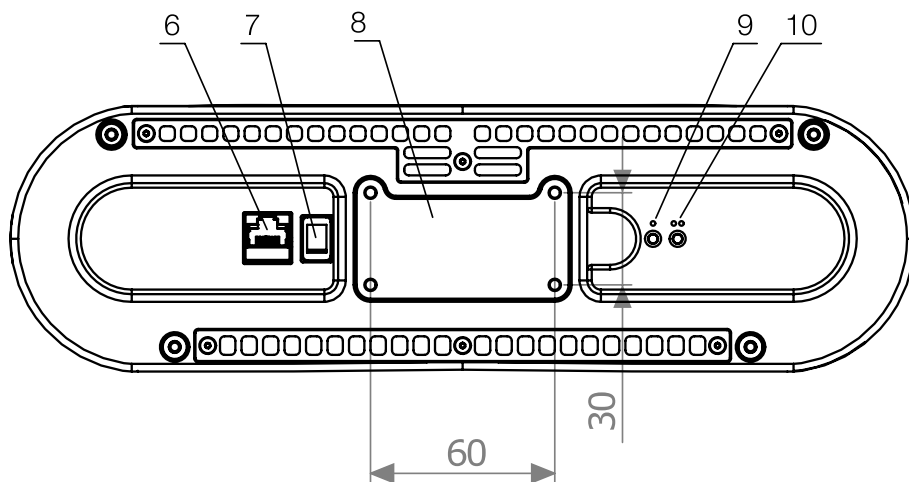


Figure 3.3: The fusionTrack backplane, showing: (6) RJ-45 connector, provides the power and the transmission of data between the fusionTrack and a computer; (7) Power switch, switching on and off the device^a; (8) Surface allowed for 4× M4 screws mount; (9) Ethernet RX LED (see Section 3.2.3); (10) Ethernet TX LED (see Section 3.2.3).

^aThe switch might be absent, depending on the fusionTrack device version.

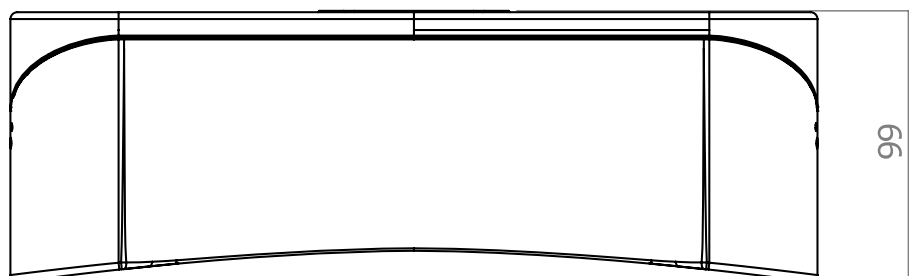


Figure 3.4: The fusionTrack top view

The fusionTrack is a Power over Ethernet powered device (*PD*) respecting IEEE 802.3at Type 2 for power consumption up to 25.50 W. The certification of the fusionTrack has been conducted excluding the Power over Ethernet power supply.

There are two ways of supplying power to the fusionTrack:

1. Using a IEEE 802.3at Type 2 compatible Power Source Equipment (*PSE*) for Gigabit Ethernet (1000Base-T) Networks. Gigabit network switches respecting IEEE 802.3at Type 2 are also available. The usage of network equipment specified for lower speeds than Gigabit is not recommended. Although the fusionTrack works with a 100Base-T network connection, the performance (framerate, delay) might be reduced. The specifications of Power Source Equipment (*PSE*) *PoE* compliant is given in Table 3.1. The list of *PoE+* power supplies successfully tested by Atracsys is found in Table 3.2. The list of *PoE+* power supplies with detected interoperability issues are listed in Table 3.3.
2. Using a passive Gigabit Midspan Injector and an appropriate DC power supply. As an example of such a passive Gigabit Midspan Injector the Power Over Ethernet Injector CAT6, L-com - BT-CAT6-P1-HP can be cited. Atracsys has also two versions (with horizontal and vertical RJ45 sockets) of a passive Gigabit Midspan Injectors under development. For more info, please contact Atracsys sales departement (See Section 1.6). The list of Gigabit Midspan injectors supplies with detected interoperability issues are listed in Table 3.3. Using an improper passive injector may damage the device.

Section 3.2.2 also provides valuable information regarding providing power to the fusionTrack device.

Property	802.3at Type 2
Also known as	<i>PoE+</i> , <i>PoE</i> plus
Power available at PD	25.50 W
Max power delivered by <i>PSE</i>	30.0 W
Voltage range (at <i>PSE</i>)	50.0 V to 57.0 V ^a
Voltage range (at PD)	42.5 V to 57.0 V ^b
Max current	600 mA ^c per mode
Max cable resistance	12.5 Ω (Cat 5e or better)
Power management	Four power class levels negotiated at initial connection or 0.1 W steps negotiated continuously
Derating of max cable ambient operating temperature	5 °C with one mode (two pairs) active
Supported cables	Cat 5e or better
Supported modes	Mode A (endspan), Mode B (midspan)

^a Source: IEEE 802.3at-2009 Table 33-11.

^b Source: IEEE 802.3at-2009 Table 33-18.

^c Source: IEEE 802.3at-2009 Table 33-1.

Table 3.1: *PoE+* parameters specifications.

Model	Remark
PHIHONG POE36U-1AT-R	Cannot be used with a shielded Ethernet cable on the PD side.
PHIHONG POE30U-560(G)	Cannot be used with two shielded Ethernet cables.
PHIHONG POE31U-1AT	–
PHIHONG POE29U-1AT	–
PHIHONG POE29U-1AT(PL)	–
PowerDsine 9001GR	–

Table 3.2: Tested *PoE+* devices.

Type	Model
Midspan injector	Enable-IT 360 PoE

Table 3.3: *PoE+* devices and midspan injectors with a known interoperability issue.

3.2.2 Cables

Atracsys does not supply cables with the fusionTrack system, as standard Cat 5e Ethernet cables can be used. The usage of shielded and unshielded cables are possible. In case a *PoE+* power supply is used, two Ethernet cables are needed (one from the host PC to the *PoE+* power supply and one from the *PoE+* power supply to the fusionTrack device). The *PoE+* power supply needs a main power cable.

Power over Ethernet is a quite complex standard that includes an initialisation phase where the *PD* (here fusionTrack device) negotiates with the *PSE* (here the *PoE+* power supply respectively the *PoE+* router) the amount of power needed. Depending on the *PSE*, the cables used (shielded or un-shielded) and even potential ground loops between the *PSE* and the computer, this negotiation phase might fail i.e. the fusionTrack device will not receive power.

This potential problem can be avoided when the following recommendations are respected:

- Using a passive Gigabit Midspan Injector and an appropriate DC power supply (see Section 3.2.1). In this case, no negotiation phase takes place and it is always guaranteed that the fusionTrack receives power. Both shielded and unshielded cables work perfectly;
- Using a *PoE+* *PSE*. In this case, the usage of unshielded cables (before and after the *PSE*) presents less problems. Ground loops between the *PSE* and the PC have to be avoided i.e. the main power connections of the *PSE* and the PC have to be connected together in the same wall plug respectively in the same multiple socket outlet.

3.2.3 Status LEDs

The status LEDs colour and behaviour is used to indicate the status of the fusionTrack device. When the fusionTrack device is switched on, the boot sequence starts with tests, during which the LEDs are switched on and off, with various colors. This lasts for about 20 s to 30 s, then the LED is used to indicate the status of the device. The various states are explained on Table 3.4.

3.3 The fusionTrack software development kit

The fusionTrack software development kit (*SDK*) is used to retrieve the measurements from the fusionTrack on the PC and to modify settings on the fusionTrack. It is completely documented in an annex document [1].

LED	Colour	Behaviour	Description
Device Status LED	Green	Flashing	Device is warming up, initialisation procedure.
	Green	Solid	Device is ready.
	Blue	Flashing	Communication with the firmware was detected.
	Red	Flashing	Indicates a fatal error, please contact Atracsys.
Ethernet RX LED	Red	Flashing ^{a,c}	Link 10BASE-T & PoE Type 2/Unknown
	Orange	Flashing ^{a,c}	Link 100BASE-T & PoE Type 2
	Yellow	Flashing ^{a,c}	Link 100BASE-T & PoE Unknown
	Blue	Flashing ^{a,c}	Link 1000BASE-T & PoE Unknown
	Green	Flashing ^{a,c}	Link 1000BASE-T & PoE Type 2
Ethernet TX LED	Red	Flashing ^{b,c}	Link 10BASE-T & PoE Type 2/Unknown
	Yellow	Flashing ^{b,c}	Link 100BASE-T & PoE Unknown
	Orange	Flashing ^{b,c}	Link 100BASE-T & PoE Type 2
	Blue	Flashing ^{b,c}	Link 1000BASE-T & PoE Unknown
	Green	Flashing ^{b,c}	Link 1000BASE-T & PoE Type 2
User Status LED	–	–	User Status LED is available for the user to indicate personal info, the colour and flashing frequency can be set through the SDK.

^a The LED is OFF when data is received, and ON otherwise.

^b The LED is OFF when data is sent, and ON otherwise.

^c The flashing might be unnoticeable.

Table 3.4: Description of the LEDs patterns

3.4 Authorised modifications

Strictly no modifications are allowed on the fusionTrack nor on the fusionTrack SDK. The fusionTrack device shall not be opened. If repairs or modifications are needed, please either contact or send the device back to Atracsys (Section 1.6).

4 Shock sensor

The fusionTrack device is equipped with a shock detection circuitry, hereafter called shock sensor. The shock sensor is a very low power, battery powered electronic unit designed by Atracsys based on a microcontroller (μC). The unit is composed of three accelerometers, a temperature sensor, a real time clock (RTC) and a non-volatile memory. Two of the three accelerometers are continuously monitoring shocks that might happen to the tracking system and the third one is providing the device orientation. Additionally, the shock sensor also records information about the use of the tracking system, such as the number of boot or the usage time.

The primary purpose of the shock sensor is to detect and record shocks encountered by the tracking system above a certain threshold, actually set to $100g$, where $g = 9.81 \text{ m s}^{-2}$ is the average Earth gravitational constant. Thanks to the built-in rechargeable battery, these shocks are recorded even when the tracking system is not powered. Each shock event is timestamped by the real-time clock. If such a shock event over the threshold has happened, the user is informed over the SDK if the corresponding option is activated.

The battery is automatically recharged each time the fusionTrack device is powered. Therefore, in a regular usage scenario, the battery is always (almost) fully charged. A completely empty battery is fully recharged in 24 h. The system runs autonomously for approximately 6 months without recharging the battery. In case the battery runs out of power, the shock sensor will be safely stopped and, thanks to the non-volatile memory, the recorded data stays saved. Shocks that might happen during the time where the shock sensor has an empty battery are of course not logged anymore. The user however is informed, if the battery runs out and therefore non-recorded shocks might have happened. In such an empty battery event, to validate the accuracy of the tracking system, Atracsys recommends to run an AVT analysis (see Chapter 13).



If not used, the fusionTrack device should be recharged for 24 h every 6 months.

4.1 Registers

An error registers and a status register are accessible using the `atnetExecutable64` software, using the `info_uC * command`.

4.1.1 Status register

Bit	7	6	5	4	3	2	1	0
	Reserved	Reserved	No bat- tery	Reserved	Reserved	Reserved	Reserved	Enabled

- Bit 5: no battery. This flag is set to 1 when the shock sensor battery is empty;
- Bit 0: enabled. This flag is set to 1 when shock sensor is enabled.

4.1.2 Calibration register

Bit	7	6	5	4	3	2	1	0
	Reserved	Reserved	Time calibrated	RTC configured	Thermometer calibrated	Acc2 configured	Acc1 configured	Acc0 configured

- Bit 5: Time calibrated. This flag is set when the RTC time is set.
- Bit 4: RTC configured. This flag is set when the internals registers of the RTC are correctly configured.
- Bit 3: Thermometer calibrated. This flag is set when the temperature sensor is calibrated.
- Bit 2: Acc2 configured. This flag is set when the internals registers of accelerometer 2 are correctly configured.
- Bit 1: Acc1 configured. This flag is set when the internals registers of accelerometer 1 are correctly configured.
- Bit 0: Acc0 configured. This flag is set when the internals registers of accelerometer 0 are correctly configured.

It should be noted the number of accelerometers depends on the fusionTrack version.

4.1.3 Error register

Bit	7	6	5	4	3	2	1	0
	μ C failed	Acc2 failed	Acc1 failed	Acc0 failed	RTC failed	EEPROM failed	No battery	Shock

- Bit 7: μ C failed. The flag is set to 1 when a failure on the microcontroller is detected. The failure is critical;
- Bit 6: Acc2 failed. The flag is set to 1 when a failure on accelerometer 2 is detected. The failure is critical;
- Bit 5: Acc2 failed. The flag is set to 1 when a failure on accelerometer 1 is detected. The failure is critical;
- Bit 4: Acc2 failed. The flag is set to 1 when a failure on accelerometer 0 is detected. The failure is critical;
- Bit 3: RTC failed. The flag is set to 1 when failure on the real-time clock is detected. The failure is critical;
- Bit 2: EEPROM failed. The flag is set to 1 when a failure on the non-volatile memory is detected. The failure is critical;
- Bit 1: No battery. The flag is set to 1 when the battery is empty. The failure is critical;

- Bit 0: Shock. The flag is set to 1 when the tracker recorded a shock above the threshold. This error is not critic. The flag can be cleared on a successful AVT run.

All critical failures lead to a stopped boot in the factory firmware with the status LED flashing red. The device need to be connected to the `atnetExecutable64` software, and the result of the error register needs to be sent to Atracsys support team Atracsys-Support@smith-nephew.com.

The shock report can be downloaded, this is explained in Section 10.7.

5 Hardware installation

In this chapter is presented how to properly set up a fusionTrack device.

When unpacking the received fusionTrack, it must be checked that no item is missing, and no item is damaged. The list of items is defined by the invoice or delivery notice. The Atracsys after sale service should be contacted in case of a missing or damaged item.

5.1 Operating environment requirements



The fusionTrack device must *not* be used covered: for instance it should not be operated under a sterile cover. This would change the operating temperature of the device and may change the optical path, both resulting in a change in the calibration parameters, causing a degradation of the device accuracy.



Since the fusionTrack can neither be covered with a sterile drape nor sterilised, the fusionTrack should not be located in the sterile area.



The fusionTrack must not be used if a hot air outlet is present between the fusionTrack and the *markers*. The refraction index changes of the hot and cold air interfaces could cause wrong measurements.



The fusionTrack has to be placed horizontal. An orientation other than horizontal (e.g. vertical) can deteriorate precision due to thermal gradients inside the device.

To allow the fusionTrack to deliver measurements, the *markers* must be in the fusionTrack working volume during the whole navigation process.

The fusionTrack cameras need a high contrast (in the IR spectrum, $\sim 850\text{ nm}$) to operate properly. Reflective surfaces and polluting lights (sunlight, lamps with a strong IR component around 850 nm , etc.) must be avoided. The SDK enables the reception of the images as seen by the cameras. In these images, polluting sources can be identified. During operation, potential sources of sunlight (or equivalent) must be eliminated. The usage of active fiducials (IR-LEDs) can help to reduce reflections.

The fusionTrack products are intended for indoor use in professional healthcare facility environments like physician offices, dental offices, clinics, limited care facilities, freestanding surgical centers, freestanding birthing centers, multiple treatment facilities, hospitals (emergency rooms, patient rooms, intensive care, surgery rooms) except for:

- near active HF (high frequency) SURGICAL EQUIPMENT,
- inside RF (radio frequency) shielded room of an ME SYSTEM for magnetic resonance imaging, where the intensity of EM DISTURBANCES is high
- in ambulances.

The calibration of the device is performed at 20 °C. The best accuracy and precision performances are obtained when the fusionTrack is operated in a room at 20 °C, once thermal equilibrium is reached. The fusionTrack should be used in the conditions listed in Table 5.1.

Environmental parameter	Range
Temperature	18 °C to 30 °C
Atmospheric pressure	70 kPa to 106 kPa
Relative humidity	20 % to 80 % non condensing
Altitude	–450 m to 3000 m

Table 5.1: Recommended environmental conditions.

5.2 Least favorable working conditions

The fusionTrack has only one functioning mode, therefore no *least favorable working conditions* exist.

5.3 Transportation and storage environmental conditions

The integrator of the fusionTrack system must take care of the mechanical stability of the fusionTrack at all times (when the fusionTrack is in use but also when it is not powered). For instance, the calibration of the device may be invalidated if the device suffered from shocks, drops and vibrations. The fusionTrack should therefore be transported and stored in an adequate transportation / storage package which minimises the probability of shocks, drops and vibrations and absorbs mechanical shocks and stress as good as possible. If the fusionTrack device was stored at extreme environmental conditions, a reasonable amount of time must be waited before using it, so that the operating environmental conditions are reached.

Environmental parameter	Range
Temperature	–30 °C to 60 °C
Atmospheric pressure	50 kPa to 106 kPa
Relative humidity	10 % to 90 % non-condensing

Table 5.2: Recommended short time (< 1 week) storage and transportation conditions.

The fusionTrack has been designed to be stored and transported in the conditions listed in Table 5.2 and Table 5.3. Note that during both transport and storage, special handling measures have to be taken:

Environmental parameter	Range
Temperature	–10 °C to 55 °C
Atmospheric pressure	50 kPa to 106 kPa
Relative humidity	10 % to 90 % non-condensing

Table 5.3: Recommended long-term (1 week – 6 months) storage and transportation conditions.

- keep away from direct sunlight;
- keep away from strong heat or cold;
- keep safe from shocks, drops and vibrations;
- keep away from fluids;
- a 'fragile' marking must be present on the package.

5.4 Mounting the fusionTrack device



The integrator has to perform tests related to instability hazards §9.4 of IEC 60601-1.

The integrator of the fusionTrack system must take care of the mechanical stability of the fusionTrack at all times (when the fusionTrack is in use but also when it is not powered). For instance, the calibration of the device may be invalidated if the device suffered from shocks, drop and strong vibrations. The fusionTrack must therefore be firmly fixed to avoid any fall and should be mounted on a structure which minimises the probability of shocks, drops and vibrations.

When installed on a tripod or on a cart, special care has to be taken to avoid tipping over. The shocks, drops and vibrations have also to be avoided when the equipment on which the fusionTrack is installed is moved like for example on a cart that passes through doors, is moved over a threshold or is put in the parking position.

Firm mounting can be achieved by using four (4) M4 screws (max depth 8 mm). More explanations are provided on Figure 5.1

5.5 Network connection of the fusionTrack device



Connection of the fusionTrack to an IT-NETWORK that includes other equipment could result in previously unidentified RISKS to PATIENTS, OPERATORS or third parties. A direct connection between the fusionTrack device and the host PC must be used. Otherwise the integrator shall apply chapter 14.13 of IEC 60601-1.

The fusionTrack uses a Gigabit Ethernet 1000BASE-T (IEEE 802.3ab) interface to communicate with the PC. The fusionTrack factory setting uses 172.17.1.7 as the static IP address. The fusionTrack must be properly powered (see Section 3.2.1) and connected to a PC.

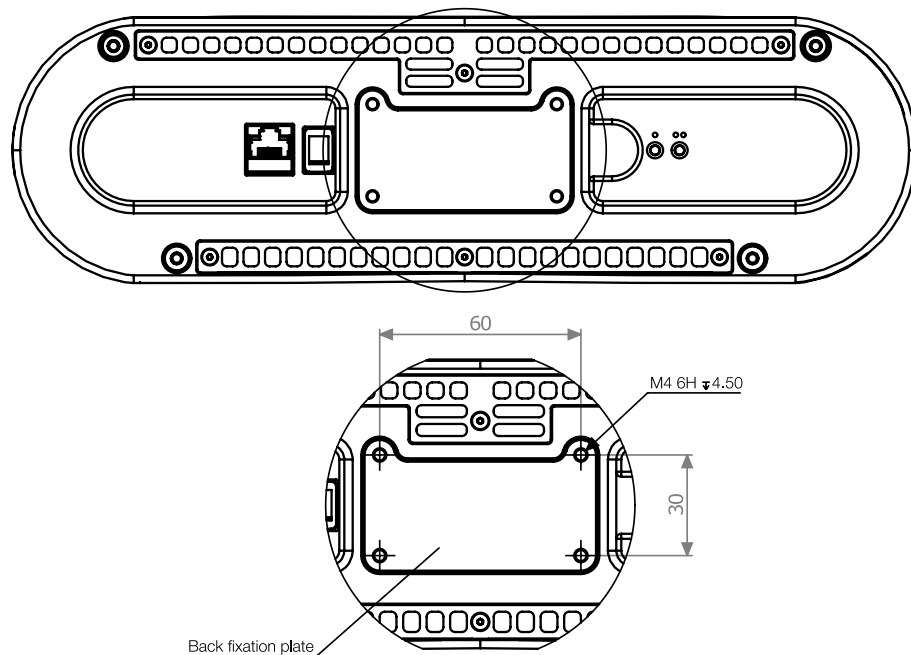


Figure 5.1: The fusionTrack screws fixation.

The Ethernet adapter of the host PC must be configured as indicated in Table 5.4. For some hardwares, not using jumbo frames may result in a much lower frame rate, see Table 5.5 for typical frame rates.

Address assignment	Static
IP address	172.17.1.100 ^a
Subnet mask	255.255.255.0
Jumbo frame	Enabled (at least 8500 bytes for maximal performance)

^a Actually, the last byte can be anything but 7, which corresponds to the device IP.

Table 5.4: Default network settings.

The fusionTrack device can be used with an Ethernet to USB adapter, however, the performances are not guaranteed. Tests performed by Atracsys show that the Ethernet to USB adapters listing in Table 5.6 allow a smooth usage of the GUI demo program.

5.6 Connecting and powering the fusionTrack

This section is an extract of the document 'fusionTrack electrical safety concept'. Upon request, a complete copy can be provided.

5.6.1 The fusionTrack

Even though no patient contact is necessary for the fusionTrack in normal operation, the fusionTrack has to be considered as Type B applied part since the fusionTrack can potentially be placed within the patient

Number of fiducials	Gigabit Ethernet 1000BASE-T, 1000 Mbit/s						100BASE-T, 100 Mbit/s					
	Ethernet MTU 1500 bytes (no Jumbo Frame)			Ethernet MTU 9014 bytes (Jumbo Frame)			Ethernet MTU 1500 bytes (no Jumbo Frame)			Ethernet MTU 9014 bytes (Jumbo Frame)		
	CPU load [%]	Network [Mb/s]	Frame Rate [Hz]	CPU load [%]	Network [Mb/s]	Frame Rate [Hz]	CPU load [%]	Network [Mb/s]	Frame Rate [Hz]	CPU load [%]	Network [Mb/s]	Frame Rate [Hz]
0	4	0.8	335	4	0.8	332	–	–	–	–	–	–
30	13	202	334	12	180	329	–	–	165	–	–	183
60	25	354	335	20	350	334	–	–	94	–	–	96
90	18	265	167	27	497	331	–	–	61	–	–	67
120	26	340	177	36	659	334	–	–	52	–	–	51

Table 5.5: Typical frame rates for fusionTrack 500 (Windows PC Intel i5-4690K). Green maximum performance, orange degraded performance. For the fusionTrack 250 there is two to three times less bandwidth and CPU necessary.

Model	Remark
Lenovo ThinkPad USB 3.0 Ethernet Adapter P/N: 4X90E51405	7 kbits Jumbo frames maximum.

Table 5.6: List of tested Ethernet to USB adapters.

environment (up to 1.5 m from the patient). Also indirect connection is possible if for example the surgeon touches the patient at the same time that they are touching the fusionTrack to adjust its position.

A Type B applied part according to the medical standard 60601-1 has to comply with:

- Requirements of the 60601-1 to provide **protection against electric shock**;
- Requirements of the 60601-1 regarding **touch current / enclosure leakage current**;
- Requirements of the 60601-1 regarding **patient leakage current**;
- Requirements of the 60601-1 regarding **patient auxiliary current**.

Patient leakage current and patient auxiliary current are not applicable since the fusionTrack has no patient connection (Type B applied part); it is only *considered* as Type B applied part.

Therefore only the two following requirements have to be respected:

- Requirements of the 60601-1 to provide **protection against electric shock**;
- Requirements of the 60601-1 regarding **touch current / enclosure leakage current**.

fusionTrack protection against electric shock

The fusionTrack is certified following the medical standard 60601-1. Following this standard, two Means of Patient Protection (2 MOPP) are necessary in order to guarantee the patient's safety. The fusionTrack systems provide 2 MOPP between the Ethernet connection (working voltage of up to 56VDC for PoE+) and the housing of the fusionTrack:

Description	Standard	Working Voltage	Withstanding Voltage	Creepage Distance	Air Clearance
fusionTrack	60601-1	57 V DC	3000 V rms	4.6 mm	2.5 mm

fusionTrack touch current / enclosure leakage current

The fusionTrack 250 system on a 57VDC source using the PHIHONG POE31U-1AT has the following maximum leakage currents measurements:

Condition	Standard leakage current measurement		Leakage current measurement non-frequency-weighted device	
	Maximum measurement	Allowable value	Maximum measurement	Allowable value
Normal condition	12 μ A	100 μ A	0.020 mA	10 mA
Single fault condition	28 μ A	500 μ A	0.022 mA	10 mA

5.6.2 fusionTrack in a ME system



The integrator has to discuss and verify the fusionTrack integration in a Medical Electrical system (ME system) with a certification body. This section of the manual is provided as information only.

If the fusionTrack is used as a component in a Medical Electrical system (ME system), the whole system has to respect the 60601-1 standard in order to guarantee the safety of the operator and the patient:

- Requirements of the 60601-1 to provide **protection against electric shock**;
- Requirements of the 60601-1 regarding **touch current / enclosure leakage current**.

The standard for medical electrical equipment medical covers ME systems that are built as a combination of – not necessarily medically certified – components. The safety of the whole ME system can therefore be ensured when respecting the following points:

- A 60601 component can be combined with components that respects relevant safety standards but not necessarily the medical 60601 standard
- The *total* touch current of the ME system has still to respect the 60601 standard (100 μ A in NORMAL CONDITION and 500 μ A in SINGLE FAULT CONDITION).

ME system protection against electric shock using 60650-1 components

The fusionTrack can safely be connected to a non-medical(PoE+) power supply, a non-medical PC or a nonmedical network. These non-medical elements have to respect the standard 60950-1 for information systems.

The combination of a fusionTrack system on a system following IEC 60950-1 is respecting 2 MOPP. This means that, as long as the computer or its peripherals (screen, mouse, keyboard) have not to be considered as applied parts Type B (i.e. they are outside of the patient environment of 1.5 m / there is no possible direct or indirect contact between the patient and this equipment), it is not mandatory to use a medical grade (following IEC 60601-1) computer system, power supply or network; a commercial grade computer system, power supply or network (following IEC 60950-1) is sufficient to guarantee the operator and patient's security regarding electric shock. It is however still necessary to also consider the touch current.

ME system touch current / enclosure leakage current

It is mandatory that the *total ME system* respects the limits for the touch current.

60601-1 states (non-exhaustive list):

- In NORMAL CONDITION, the TOUCH CURRENT from or between parts of the ME SYSTEM within the PATIENT ENVIRONMENT shall not exceed 100 μ A.
- In the event of the interruption of any non-PERMANENTLY INSTALLED PROTECTIVE EARTH CONDUCTOR, the TOUCH CURRENT from or between parts of an ME SYSTEM within the PATIENT ENVIRONMENT shall not exceed 500 μ A.

There is no means to calculate or estimate the total touch current; it has to be measured. Even if all individual components (medically certified and non-medically certified) respect the limits of 100 μ A in NC and 500 μ A in SFC, the combination of several components is most likely to exceed the limits. In this case, a medically certified separating transformer between the mains and the ME system will guarantee the total touch current to stay under the required limits. Additionally, the medically certified separating transformer also increases the means of protection regarding electric shock for the whole ME system and also for the individual components.

5.6.3 Annex: Explanation of applied part

IEC 60601-1 uses the term 'applied part' to refer to the part of the medical device which come into physical contact with the patient in order for the device to carry out its intended function.

Applied parts are classified as Type B, Type BF or Type CF according to the nature of the device and the type of contact. Each classification has differing requirements from the point of view of protection against electrical shock.

Type CF is the most stringent classification, being required for those applications where the applied part is in direct conductive contact with the heart or other applications as considered necessary.

Type BF is less stringent than CF, and is generally for devices that have conductive contact with the patient, or having medium or long term contact with the patient.

Type B is the least stringent classification, and is used for applied parts that are generally not conductive and can be immediately released from the patient.

Type B applied parts may be connected to earth, while Type BF and CF are 'floating' and must be separated from earth.

6 Software installation

This chapter presents how to install the *SDK* used to communicate with the device. The installation package is provided as an installation wizard for Windows and a compressed tarball for Unix/Linux, and contains:

- A Graphical User Interface (*GUI*) demo application;
- A cross-platform *GUI* application allowing to acquire data for assessment of the optical tracking device trueness (the *AVT* software);
- Headers and library files to integrate the driver in a project;
- Pre-compiled software examples, with the corresponding source code;
- Documentation of the Application Programming Interface (*API*);
- Some predefined *marker* files.

The latest installers are found on Atracsys website, using the username and password provided by Atracsys.

6.1 Windows installation

The installer for Windows is a wizard which allows an easy installation with possible customisation of the setup. Depending on the target environment, Visual Studio redistributables will be automatically installed as well.

6.2 Linux installation

The Linux package consists of a compressed tarball.

```
1 tar -xJvf fusionTrack_v_4_1_1-gcc-5.4_x64.tar.lzma
2 cd fusionTrack_v_4_1_1-gcc-5.4_x64
```

Listing 6.1: Example of setting up the SDK.

Setting the `ATRACSYS_SDK_HOME` environment variable to the root SDK directory, i.e. the `fusionTrack_v_x_y_z-gcc-5.4_x64` folder will allow the demo and precompiled software to find the geometry files located in the `data` 'installation' folder. Note that setting this variable is mandatory for the *AVT* to work.

Mono (open source implementation of the Common Language Infrastructure .NET) is required to run the *AVT*. Mono is used to send back the *AVT* data to Atracsys for analysis.

7 How the fusionTrack device works

This chapter describes the fusionTrack device functioning.

7.1 Communicating with the fusionTrack device

The fusionTrack communicates with the host PC via Ethernet using a proprietary protocol. The *SDK* shipped with the fusionTrack device implements the bidirectional communication: e.g. retrieving information from the fusionTrack device, setting options and reading the data.

7.2 The fusionTrack device coordinates system and working volume

The fusionTrack uses a Cartesian right-handed coordinates system. Its origin is the optical centre of the left camera. The x axis points along the sensor width in the direction of the right camera and the z axis is colinear to the optical axis. Unless specified otherwise, all distance are expressed in millimetres [mm] units. An alternate coordinate system can be used, which center lies at the middle of the fusionTrack device baseline, i.e. at the mid-point between the two cameras.

The fusionTrack contains an accelerometer, which system of coordinates is the same as for the tracking data (since SDK version 4.7.1).

Figure 7.1 and Figure 7.2 show the two coordinates systems from the front and the top respectively.

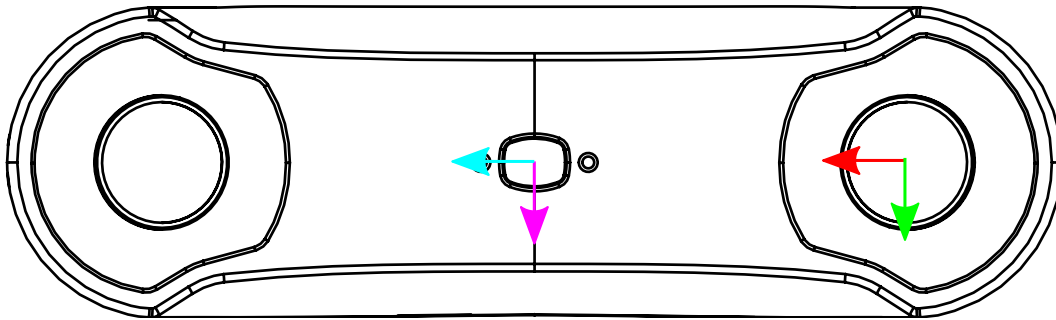


Figure 7.1: Coordinates systems, front view. The fusionTrack device x axis is in red, the fusionTrack device y axis is in green. The symmetrised x and y axes are in cyan and magenta respectively.

In order to be tracked by the fusionTrack device, the markers must lie inside the working volume described in Figure 7.3. It should be noted this volume is *not* the same as the validated volume, shown on Figure 2.1 and Figure 2.2.

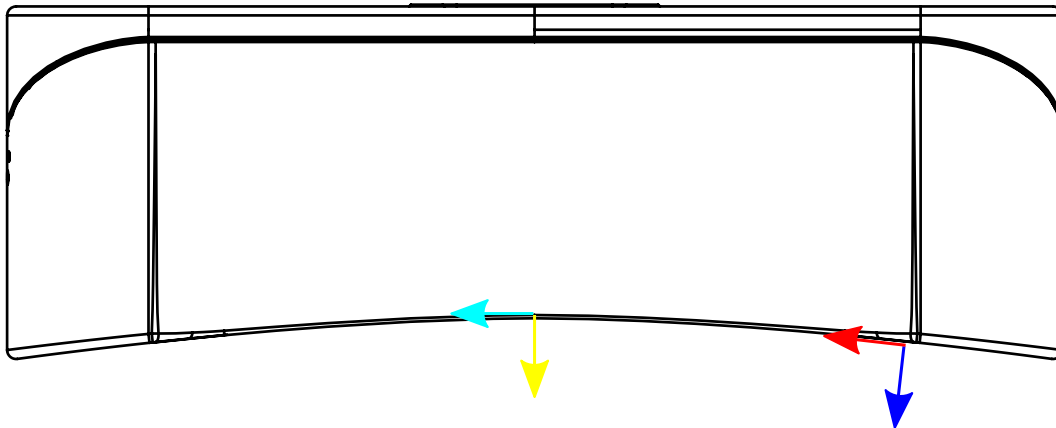


Figure 7.2: Coordinates systems, top view. The fusionTrack device x axis is in red, the fusionTrack device z axis is in blue. The symmetrised x and z axes are in cyan and yellow respectively.

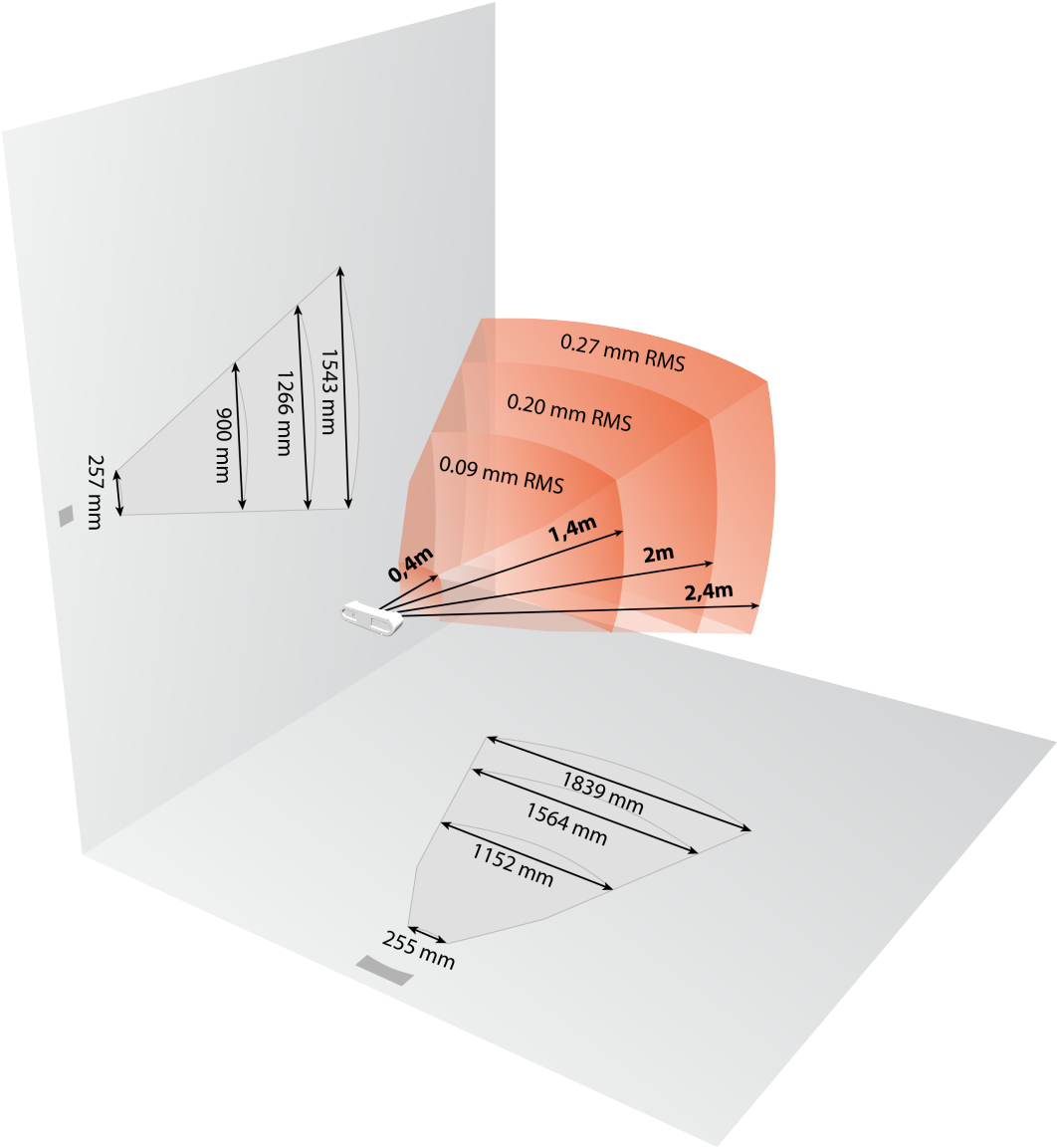


Figure 7.3: fusionTrack tracking volume.

7.3 Information provided by the fusionTrack device



The fusionTrack is designed to provide accurate *relative* measurements, and should *not* be used to perform *absolute* measurement. See Chapter 3 for more information.



Most of the functions of the *API* return a status code, which indicates different levels of warnings and errors. The returned code *must* be checked by the Integrator's software implementation. The documentation is available in [1].



The various information retrieved from the fusionTrack for each frame comes with a status flag, which must *always* be checked by the integrator's software implementation to ensure the integrity of the data.

The *API* is fully documented in [1]. In this section, the information which can be retrieved from the fusionTrack when *markers* are tracked is shortly presented:

- The *marker* information, consisting of
 - the *marker* container status, indicating if an error occurred when retrieving the data;
 - the number of reconstructed *marker*, indicating the size of the container;
 - *marker* 3D position in mm;
 - *marker* orientation (3x3 rotation matrix);
 - registration error in mm, corresponding to the result of the least-square minimisation between the *marker* geometry and the measured *fiducial* positions;
 - the geometrical id, corresponding to the 'id' field from the geometry file Section 7.7;
 - a tracking id, giving a unique number to identify a *marker* on a frame;
 - a mask indicating which *fiducials* have been used to reconstruct the *marker*, i.e. if bit #*i* has value 1, it means that *fiducial* #*i* was present;
 - the indices of the used *fiducials*, allowing to get the information about the used *fiducial*;
 - the *status* of the *marker*, which indicates whether the marker lies in the accuracy measurement volume or not, consisting of the union of the statuses of all its *fiducials*;
- the *fiducial* information, consisting of
 - the *fiducial* container status, indicating if an error occurred when retrieving the data;
 - the number of reconstructed *fiducials*, indicating the size of the container;
 - *fiducial* 3D position in mm;
 - triangulation error in mm, corresponding to the error coming out of the triangulation algorithm, i.e. the error on the *fiducial* position;
 - epipolar error in pixels;
 - *fiducial* probability, indicating if a *raw data* was used to build multiple *fiducials*;

- the indices of the left and right *raw data* used to reconstruct the *fiducial*, allowing to access the information about the used *raw data*;
- the *status* of the *fiducial*, which indicates whether the *fiducial* lies in the accuracy measurement volume or not, unioned with the statuses of the two *raw datas* of which it consists;
- the left / right *raw data* information, consisting of
 - the *raw data* container status, indicating if an error occurred when retrieving the data;
 - the number of detected *raw data*, indicating the size of the container;
 - *raw data* 2D position in pixels;
 - *raw data* surface in pixels;
 - the *raw data* probability, based on the width / height ratio of the detected *raw data*;
 - the *raw data* status, indicating if the *fiducial* is located at an edge of the sensor, or if it lies outside the theoretical field of view;
- the picture header information, consisting of
 - the header status, indicating if an error occurred when retrieving the data;
 - the picture dimensions;
 - the picture counter, a unique and consecutive generated number;
 - the picture timestamp, corresponding to the time in μ s since the device startup, the timestamp is taken at the start of the image exposure;
 - the picture image format;
- the left / right picture data, consisting of
 - the picture container status, indicating if an error occurred when retrieving the data;
 - the list of the picture pixels.

The position \vec{x} and orientation R of the *marker* are interpreted in the following way (see Section A.2 for a full example), using the homogeneous notation (see Section A.1):

$$\vec{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$R = \begin{pmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{pmatrix}$$

$$H = \begin{pmatrix} r_{00} & r_{01} & r_{02} & x \\ r_{10} & r_{11} & r_{12} & y \\ r_{20} & r_{21} & r_{22} & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where r_{ij} is the rotation matrix component at line i , row j .

The pictures are given as in [PGM format](#), an example of using the data is shown on Listing 7.1.

```
1 err = ftkGetLastFrame( lib, serialNbr, frame );
2 if ( err == FTK_OK && frame->imageLeftStat == QS_OK )
3 {
4     QImage* greyPicture( new QImage( frame->imageHeader.width, frame->imageHeader.height,
5                                     QImage::Format_Indexed8 ) );
6     for ( int row( 0 ); row < greyPicture->height(); ++row )
7     {
8         memcpy( greyPicture->scanLine( row ),
9                 frame->imageLeftPixels + row * frame->imageHeader.imageStrideInBytes,
10                greyPicture->bytesPerLine() );
11     }
12 }
```

Listing 7.1: Example of reading the picture using the [Qt framework](#).

7.3.1 The fusionTrack events

The application firmware is able to send ‘events’ to the host PC. Those events have a type (defined in [1]), and optional data. Depending on the event type, they are periodically updated and sent with each frame (the temperature-related events for instance), or sent on a specific condition on the device (when the temperature is too low or too high for example). The *SDK* provides a mechanism allowing the user to read those events.

7.4 Fiducial detection and marker tracking

Passive and active *fiducials* are detected by the fusionTrack using different methods.

The fusionTrack collects IR light for a period of time called integration time. This is an electronic shutter. The value of this integration time can be tuned via option.

For passive *markers*, during the whole integration time, IR illuminators are lit and the retro-reflecting coating of the passive *fiducials* reflects the IR light to the fusionTrack sensors.

For active *fiducials*, the fusionTrack emits an IR signal which triggers the LED emission on the connected *marker*.

In both cases, the IR light is detected by the fusionTrack sensors in the same way. The fusionTrack *SDK* then builds *fiducials* by combining pairs of *raw data*, wrong combinations are rejected using the epipolar error explained in Section 7.9. The reconstructed *fiducials* are finally combined into *markers*, using the provided geometry and applying the matching tolerance and a registration error cut to reject fake combinations.

The reconstructed *markers* are *tracked* from one frame to the next: this means a found *marker* on frame n will be looked for in frame $n + i$ around its last known position. This allows to save CPU cycles as the tracked *markers* do not undergo the whole reconstruction sequence.

The legacy *marker* reconstruction algorithm suffered from several flaws:

- a valid *marker* could be missed (i.e. not reconstructed) if, during the looking for *fiducial* i , an additional *fiducial* lay at about the same distance. The algorithm was able to see the *marker* could not be reconstructed due to the registration error but didn't look for another possibility;
- the ‘Matching Tolerance’ option (see Section 8.1.1) was not implemented correctly, resulting in the applied cut to be dependent on the compared distance in a non-trivial way. Due to the implementation, a simple fix was not possible;
- the *first* found candidate was chosen as the correct one. With large tolerances, this could lead to wrongly reconstructed *markers*.

For those reasons, a new *marker* reconstruction algorithm was developed. Starting from SDK version 4.7.1, the new algorithm is used as default, the old one is still available for backward compatibility reasons. It can be chosen using a dedicated option: 'New marker reco algorithm'.

7.4.1 Active fiducials

As long as the wavelength is within the range specified (in Table 2.1) and the emitted power is sufficient, the device can work with several active fiducials (i.e. LEDs). In order to minimise angular dependency, only LEDs with maximum opening angle and without embedded lenses should be used. For non-medical / non-autoclavable use, Atracsys recommends the OSRAM SFH4250-Z LED, driven at 100 mA.

7.5 Temperature compensation

The internal temperature of the fusionTrack device have a non-negligible rise time, which leads to a loss of trueness in both following situations:

1. as long as the device is not thermically stable;
2. if the steady temperature does not correspond to the temperature at which the calibration was performed.

In order to solve those issues, a temperature compensation algorithm is used. The compensation decreases the loss of trueness in the warm-up phase and if the reached temperature is different from the temperature during the calibration.



The value of the 'Calibration type' option must be checked by the integrator to correspond to the value of `CalibrationType::TemperatureCompensation` if the temperature compensation is to be used.

The temperature compensation consists of multiple calibrations, each of them obtained at a different temperature. The fusionTrack device has several temperature sensors, which values are combined to compute a 'synthetic temperature'. The 'temperature compensated calibration' is therefore stored as a set of calibration parameters indexed by a 'synthetic temperature' value. On each frame, the current 'synthetic temperature' value is computed, and a calibration is either interpolated or extrapolated from the ones contained in the calibration file. New events (see Section 7.3.1) have been added to allow the user to:

- get the current 'synthetic temperature' and the value of the 'synthetic temperature' when the device is stable in a room at 20 °C (called the reference temperature), only provided for user information¹;
- know if the current 'synthetic temperature' value is either below the lowest one or above the highest one: those events are `FtkEventType::fetLowTemp` and `FtkEventType::fetHighTemp` respectively.

The events are the only way to know if the SDK is interpolation or extrapolation mode, no specific status codes are issued. The presence of those events do not necessarily mean the data must be discarded: for instance, as soon as the synthetic temperature gets below the lowest temperature, the event is generated, not taking into account the temperature difference (i.e. the difference being 0.1 °C or 10.0 °C does not make any difference).

Depending on the fusionTrack device model, the temperature compensation might not be available.

¹Since the external temperature modifies the synthetic temperature value at thermal equilibrium, the latter cannot be used by the integrator in their regulation software.

7.6 The fusionTrack device markers



The *markers* used for surgery must be bio-compatible.



A *marker* shall be treated as an independent medical device. Use *markers* as specified by Atracsys. Other risks related to *markers* with reflecting *fiducials* may occur depending on the technology used by the manufacturer, these risks cannot be treated by Atracsys.



Any liquid on *markers* such as body fluids and water could lead to wrong measurements. The integrator has to specify in the end user manual instructions about not handling *markers* with dirty gloves, as well as instructions for cleaning / replacing *markers* and / or *fiducials* after getting in contact with liquids.



A *marker* in the camera view must have a specific, appropriately sized and distinguishable geometry. The integrator has to ensure that each used *marker* should be easily distinguishable by end users. A label on each *marker* has to be inserted. The integrator has to specify in the end user manual that operators have to choose *markers* with different geometries.



A *marker* is assumed to be calibrated and sterile. *Fiducials* are assumed to be correctly fixed on the *marker*. Instructions about fixing *fiducials* on *markers* must be clearly specified by the integrator in the end user manual.



Sterile *markers*, *fiducials* or battery packaging must be opened inside the Operating Room (OR). The integrator has to specify in the user manual to open sterile *markers*, *fiducials* or batteries packages only in the OR as well as the manner how to open it to guarantee sterility.



Sterile *markers*, *fiducials* or batteries in a damaged package cannot be used because it might have been deteriorated during transport which could lead to patient infection. Instructions about not using sterile *markers*, *fiducials* or batteries from a damaged package must be clearly specified by the integrator in the end user manual.



Sterile *markers*, *fiducials* or batteries in a damaged package cannot be used because it might have been deteriorated during transport which could lead to patient infection. The integrator has to provide sterile *markers*, *fiducials* or batteries with packaging presenting a certain robustness.



Markers or *fiducials* geometrically deformed due to shocks or manufacturing defects can create wrong measurements leading to patient injury. Instructions about not using geometrically deformed *markers* or *fiducials* must be clearly specified in the end user manual.



The integrator should define a zeroing method (see Chapter 12) or any equivalent method to verify marker integrity for all cases including corner cases of its application.



Markers or *fiducials* too close from lenses might lead to wrong measurement (see Table 2.1). The integrator has to mention in the end user manual that *markers* or *fiducials* should not be positioned too close from the lenses.



The integrator has to ensure generating an error message when the *markers* are placed on vibrating instruments and the frequency of vibrations could affect the measurements.



At all time, the integrator software should monitor the registration error of the reconstructed *markers*, to prevent badly reconstructed data to be used.



Instructions about using a calibrated *marker* and not geometrically deformed have to be clearly specified by the integrator to the user.

A *marker* is a rigid structure on which three or more *fiducials* are fixed so that the distance between them is constant.

The fusionTrack system is compatible with both passive and active wireless *markers*. The fusionTrack device identifies and tracks the *markers* using the geometry of the *fiducials* on the *marker*. The fusionTrack device needs a geometrical description of each tracked *marker* (this description is called the geometry file).

Several causes related to *markers* might cause wrong measurements. Even when the *marker* itself has not been geometrically deformed, any damages on *fiducials* might be an issue: when a fiducial is particularly damaged, the *marker* might not be detected and tracked. In presence of a damage, the device is not able to match the *marker* with a known geometry.

A significant value of error is critical for an accurate tracking. In some cases, the *marker* might even disappear suddenly because the geometry is not identified anymore. In some cases, the *marker* in the field of view of the fusionTrack could even not be detected. The integrator must indicate to the operator that the computed error is critical. It could be a warning or error message.

7.6.1 Passive markers

The passive *markers* use reflecting material in various shapes (spheres or discs). The used passive material has a specific coating which minimises the scattering of the IR light, allowing a large amount of it to be reflected back to its source. This causes the IR light from the illuminators to be reflected directly into the cameras. The fusionTrack system is designed to track the position and orientation of *markers* and the position of individual reflecting objects.



For single-use *markers*, the end user manual must clearly state that the *markers* should not be reprocessed. The *markers* packaging must be marked with a label indicating they cannot be reprocessed.



For single-use *fiducials*, instructions about replacing new *fiducials* after every (single) use must be clearly specified by the integrator in the end user manual. Packages of passive *fiducials* have to be marked with a label indicating that they cannot be reprocessed.



A passive fiducial could be damaged due to a shock / scratch of the reflecting material. Instructions about not using deteriorated *fiducials* and how to replace a deteriorated passive *fiducial* must be clearly specified by the integrator in the end user manual.



Integrator has to specify in the end user manual instructions how to screw / plug *fiducials* on *markers*.

7.6.2 Active markers

The active *markers* consist of IR-LEDs mounted in a rigid structure. The LEDs can either emit continuously or be synchronised with the fusionTrack cameras. In the latter case, the *marker* also contains an IR receiver. An active *marker* is powered by a battery, or from the equipment to which it is attached. Those *markers* can either be wired or wireless.

The fusionTrack device IR communication sends a pattern which is recognised by the IR receiver of the active *marker*. Once the IR pulse is detected by the *marker*, the *marker* emits IR radiations, which is detected by the fusionTrack device.



Active *marker* electronics has to be completely sealed in order to avoid liquid penetration due to cleaning. The integrator has to indicate in the end user manual instructions to clean active *markers*. The integrator has to specify in the end user manual that the operability verification procedure (see Chapter 12) must be done after each cleaning.



A geometrically deformed active *marker* due to a manufacturing defect could lead to wrong measurements causing patient injury. The integrator has to perform individual testing during *marker* calibration.



Active *markers* with batteries shall be handled with care. Operator who gets in contact with battery leakage incurs a risk of chemical burn. The integrator has to indicate in the end user manual that the battery of an active *marker* has to be removed after usage, especially in the case the active *marker* undergoes a sterilisation procedure.



The batteries used in active *markers* must comply with the regulation standards.



Battery for active *markers* must be sterile. The integrator has to indicate in the end user manual to use only sterilized batteries according to ISO 14937.



Any electric defects on active *markers* such as explosion, leakage, overheating might lead to operator or patient injury. The integrator has to use an adequate protection (e.g. fuse) that limits the current drawn from the battery. The integrator has to ensure that active *markers* contain a polarity marking and a polarity inversion circuit on the *marker*.



The measurement precision of an active *marker* can be deteriorated because of liquids on its fiducials or IR-receptor. Instructions about identifying and cleaning of active fiducial must be clearly specified by the integrator in the end user manual.



An active fiducial *marker* must be sterilised before each usage. Cleaning or sterilisation methods instructions must be clearly specified by the integrator in the end user manual.



An active fiducial *marker* is able to transmit its individual parameters and data to the system. It can also notify the fusionTrack with alert messages. An error message alerting that battery power is low is sent. That message must be handled by the integrator to alert the operator that the *marker* is not considered as operational due to a low battery and the *fiducials* must be turned off.



A weak optical signal from the IR-LEDs can lead to a wrong estimation of the position. The integrator should ensure the fiducials are turned off if the battery voltage goes under a given threshold.



Markers have to meet the IEC60601-1 requirement regarding 'Protection against excessive temperatures and other hazards'.



The integrator should use redundant *fiducials* (i.e. four or more *fiducials* instead of only three) and require the reconstructed *marker* to use all of them.



The battery must be removed from the *marker* after use, in order to prevent damage to the *marker* which may be caused by battery leakage.

7.7 Marker geometry files

In order to be able to track a new passive custom *marker*, a new *marker* geometry file must be registered in the SDK. This is performed using the `ftkSetRigidBody` function (or its legacy counterpart `ftkSetGeometry`), which documentation can be found in [1].

The demo application will automatically load the geometries from the installation directory and also from the 'user directory', as long as the file name is `geometryXXX.ini`. Active *markers* on the other side carries their geometries with them. The fusionTrack retrieves the geometries directly after pairing.

The geometry of a *marker* consists at least of the position of each fiducial composing the *marker*. The SDK supports two file types (INI files [2], with basic support) and binary, and two different versions (versions 0 and 1), which sum up to three types of files:

1. legacy version 0 INI files;
2. INI files version 1, with additional stored data,
3. binary format version 1, equivalent to INI file version 1, used in the wireless markers, as they don't have much available memory.

The SDK allows to load the geometry information from those three file types, so that the user doesn't have to code the parsing.

The used coordinate system is a Cartesian right-handed coordinate system, which origin is reported as the *marker* position. All distances and positions are expressed in millimetres.

7.7.1 INI files version 0

The INI geometry file version 0 is the legacy format used by Atracsys since the beginning of the fusionTrack SDK. The file contains:

- a **[geometry]** section, containing:
 - a **count** key, which indicates the number of *fiducials* (at least 3, at most 6) composing the *marker*;
 - an **id** key, indicating the unique id of the geometry;
- A **[fiducialX]** section for each *fiducial*, where *X* goes from 0 to $n-1$, n being the number of *fiducials* composing the *marker* (described by the **count** key). Each **[fiducialX]** section contains the coordinates of the *fiducial* (in mm, no requirement on the origin) as follows:
 - an **x** key indicating the *x* coordinate of the position of the *fiducial*;
 - a **y** key indicating the *y* coordinate of the position of the *fiducial*;
 - a **z** key indicating the *z* coordinate of the position of the *fiducial*;

A user-defined file may contain comments (indicated by a ';' as first character) or additional sections, key-values pairs. The SDK will simply not take the additional data into account. In Listing 7.2 is given an example of a geometry version 0 INI file for a *marker* with 4 *fiducials*, which id is 998.

```
1 ; Example of geometry file
2 [geometry]
3 count=4
4 id=998
5 [fiducial0]
6 x=-11.819
7 y=10.993
8 z=-0.006
9 [fiducial1]
10 x=-11.811
11 y=500.934
12 z=0.048
13 [fiducial2]
14 x=-11.77
15 y=970.879
16 z=0.266
17 [fiducial3]
18 x=-11.759
19 y=1165.952
20 z=-0.131
```

Listing 7.2: Example of geometry INI version 0 file.

7.7.2 INI files version 1

The INI geometry file version 1 extends the legacy format used by Atracsys since the beginning of the fusionTrack SDK, by allowing more information to be contained. The file contains:

- a **[geometry]** section, containing:
 - a **count** key, which indicates the number of *fiducials* (at least 3, at most 6) composing the *marker*;
 - an **id** key, indicating the unique id of the geometry;
 - a **version** key, containing the file version (i.e. 1);
 - a **divotCount** key, indicating the number of divots (from 0 to 6) present on the marker;

- A **[fiducialX]** section for each *fiducial*, where X goes from 0 to $n-1$, n being the number of *fiducials* composing the *marker* (described by the `count` key). Each **[fiducialX]** section contains the coordinates of the *fiducial* (in mm, no requirement on the origin) as follows:
 - a mandatory `x` key indicating the x coordinate of the position of the *fiducial*;
 - a mandatory `y` key indicating the y coordinate of the position of the *fiducial*;
 - a mandatory `z` key indicating the z coordinate of the position of the *fiducial*;
 - an optional `normalX` key indicating the x component of the normal of the *fiducial*;
 - an optional `normalY` key indicating the y component of the normal of the *fiducial*;
 - an optional `normalZ` key indicating the z component of the normal of the *fiducial*;
 - an optional `angleOfView`, containing the half aperture angle of the cone in which the *fiducial* can be seen;
 - an optional `fiducialType` key, indicating the type of the *fiducial* (this corresponds to the underlying `uint32` value of the `ftkFiducialType` `enum` identifier;
- Optional **[divotX]** sections for each *divot* Figure 12.1, where X goes from 0 to $n-1$, n being the number of *divots* present on the *marker* (described by the `divotCount` key). Each **[divotX]** section contains the coordinates of the *divot* (in mm, no requirement on the origin) as follows:
 - a mandatory `x` key indicating the x coordinate of the position of the *divot*;
 - a mandatory `y` key indicating the y coordinate of the position of the *divot*;
 - a mandatory `z` key indicating the z coordinate of the position of the *divot*;

For each **[fiducialX]** section, either no `normal` key must be present or all `normalX`, `normalY` and `normalZ`.

In Listing 7.3 is given an example of a fake (the coordinates of the *fiducials* and *divots* actually make no sense) geometry version 1 INI file for a *marker* with 4 *fiducials* and 2 *divots*, which id is 42.

```

1 [geometry]
2 version=1
3 id=42
4 count=4
5 divotCount=2
6 [fiducial0]
7 x=0
8 y=11
9 z=3
10 normalX=0
11 normalY=1
12 normalZ=0
13 angleOfView=1
14 fiducialType=16
15 [fiducial1]
16 x=-33.72
17 y=-38.63
18 z=3
19 normalX=0
20 normalY=1
21 normalZ=0
22 angleOfView=1
23 fiducialType=16
24 [fiducial2]

```

```
25 x=0
26 y=-75.55
27 z=3
28 normalX=0
29 normalY=1
30 normalZ=0
31 angleOfView=1
32 fiducialType=16
33 [fiducial3]
34 x=41.28
35 y=-39.21
36 z=3
37 normalX=0
38 normalY=1
39 normalZ=0
40 angleOfView=1
41 fiducialType=16
42 [divot0]
43 x=0
44 y=-13
45 z=3
46 [divot1]
47 x=5
48 y=-1
49 z=3
```

Listing 7.3: Example of geometry INI version 1 file.

7.7.3 Binary files version 1

Binary files are not meant to be read by the user, they are only meant to be stored on devices with limited memory (i.e. wireless markers). The information in the file is exactly the same as in INI files version 1, and the conversion from binary (version 1) to INI (version 1) is invertible.

7.8 Marker tracking parameters

The tracking of *markers* (and *fiducials*) is tuned by a small set of parameters:

- epipolar max distance;
- matching tolerance / distance matching tolerance;
- registration mean error;
- matching maximum missing points;
- tracking range;
- tracking time span.

In this section those parameters are explained.

Epipolar max distance The epipolar max distance affects the *raw data* pairing. The epipolar error tolerance expresses the distance (in pixels) between the centroid of the *raw data* from the right camera and its expected position deduced from the centroid position on the left camera. The bigger the tolerance, the higher the probability of finding a matching pair, however the probability of creating a 'phantom fiducial' (see Section 7.9) is also increased. Finally, the number of performed operations increases with the epipolar error tolerance, as more possible combinations have to be tested.

Matching tolerance / Distance matching tolerance The matching tolerance affects how *fiducials* are matched to a known geometry. The matching tolerance is the maximal difference between the length between two *fiducials* and the reference length, as deduced from the geometry file. The bigger the tolerance, the looser the matching criteria. The 'Matching tolerance' option changes the behaviour of the old marker reconstruction matching algorithm, whilst the 'Distance matching tolerance' modifies the behaviour of the new one.

Registration mean error The registration mean error sets a tolerance on the error returned by the *marker* registration algorithm. The computed error is a measurement of the deformation between the reconstructed *marker* and the theoretical geometry, linked to the accuracy of the positioning of the *marker*. The bigger the registration error, the looser the requirement on the system accuracy.

Matching maximum missing points The matching maximum missing points simply sets how many *fiducials* can be missed for a *marker* to be reconstructed by the system. When set to zero, every *marker* candidate for which a *fiducial* cannot be associated (due to the matching tolerance for instance) or is missing (e.g. occultation) won't be available in the frame data.

Tracking range The tracking range sets how far the *marker* tracking engine is allowed to look for a *raw data* when trying to find a *marker* from its last known position. This distance is measured in pixels, and must be adapted to the typical movement speed of the *marker* in real usage and the application frame rate.

Tracking time span The tracking time span sets over how many frames the tracking engine is allowed to look for a *raw data* when trying to find a *marker* from its last known position. As the *marker* are only reconstructed when a call to `ftkGetLastFrame` is performed, frames might be skipped if the application refresh rate is lower than the device acquisition frequency. The tracking time span can be seen as a tolerance on the number of skipped frame.

7.9 Phantom fiducials

A phantom fiducial is an artefact created by the triangulation algorithm, caused by pairing a wrong combination of *raw data*. This can happen when several *fiducials* are located on the same epipolar line. At the time of the triangulation, there is no way to know if a reconstructed fiducial is a real one (i.e. corresponding to a physical fiducial) or a phantom. *Fiducials* sharing a raw data have a probability smaller than 1 (one): if a *raw data* is used n times, the probability of the resulting *fiducials* is set to $\frac{1}{n}$. During the registration stage, if any of the 'clones' is used to reconstruct a *marker*, its siblings are removed from the data.

7.10 Stray fiducials

A stray fiducial is a fiducial which was not used to reconstruct a *marker*. The *SDK* allows to get the 3D position of such *fiducials*. The stray *fiducials* may include phantom *fiducials* (see definition in Section 7.9), and the application must take care of removing them, and checking the validity of the reported positions.

Using stray fiducial positions may lead to hazardous situations, such as:

- IR transmitter, incandescent light, reflections or other IR source may be reconstructed as a stray fiducial;
- There is no way to identify a stray fiducial, i.e. its id (number) may change from one frame to the next. The application must therefore perform checks based on the position to ensure an identification of the fiducial;
- There is no way to distinguish a real stray fiducial from a phantom fiducial, only the probability defined in Section 7.9 is available.

8 Using the fusionTrack system

In this chapter is explained how to use the fusionTrack. It is assumed that the fusionTrack is correctly setup as described in Chapter 5 and the software on the host PC is correctly installed as indicated in Chapter 6. Finally, the Ethernet card is configured as indicated in Section 5.4. The communication between the PC and the fusionTrack device may be blocked by the PC firewall. The fusionTrack device is started by setting the power switch on the 'ON' position (see Figure 3.3). The fusionTrack device is ready to be used once:

- the device beeps twice;
- the Device Status LED (see Section 3.2.3) appears in solid green.

If the Device Status LED blinks, is red, if the fusionTrack beeps continuously or the situation is not corresponding to the described working state, please refer to Chapter 18. Switching off the device is simply done by setting the power switch on the 'OFF' position (see Figure 3.3).



The default parameters have been chosen to ensure optimal tracking performances. Changing them might impact the performances of the fusionTrack system.



The fusionTrack device must be cleaned as described in Chapter 17.

To help the integrator to understand how to use the fusionTrack and how to configure it, Atracsys provides:

- **Example sample software:** The samples are lightweight applications using the *SDK* libraries. They are provided as compiled applications and as source code. They showcase the main features of the *SDK*, and can be used as a base to create a custom application. They can be found in the relative path `fusionTrack SDK/bin`. See the Section 8.3 for more details.
- **The "demo" GUI:** The demo GUI displays in real-time the data from the fusionTrack, and allows to configure the system graphically. It is an intuitive way to discover the features of the device and it may help in a debug process. The binary can be found in the relative path `fusionTrack SDK/bin`. See the Section 8.2 for more details.
- **Options:** The option interface system is a feature of the *SDK*. The options are used to configure the *SDK* and the fusionTrack device. They provide a unified interface for a broad range of use cases: they range from the configuration of the user LED in the device, to the configuration of the parameters of the triangulation algorithms in the *SDK*. See the Section 8.1 for more details.

8.1 The fusionTrack system options

This section describes the options available for the fusionTrack. It must be noted that some options are not available depending on the exact fusionTrack model and firmware version. The options are split into five categories:

1. Library related options (e.g. Library version);
2. Device related options (e.g. integration time);
3. Detection stage related options (e.g. min / max size of the blobs);
4. 3D reconstruction related options (e.g. epipolar tolerance);
5. Wireless related options (e.g. marker button status streaming).

Each option has a type (`int32`, `float` or `ftkBuffer`), a read / write status and an optional 'global' flag indicating whether the option is global, in which case it is handled at the library level and must be accessed by specifying `0uLL` as serial number.

Unless explicitly flagged as 'permanent' in the option description, the set values are reset if the connection to the fusionTrack device is closed (i.e. after a call to `ftkClose`). In any case, a restart of the fusionTrack device resets all options to their default values.

Options names and IDs All options are identified by a unique ID of type `uint32`. The mapping between the option and its unique ID is not necessarily preserved between releases, hence the option enumerating mechanism (i.e. the `ftkEnumerateOptions` function) which allows to get the unique ID of each option at runtime (see also the 'ListOptions' software described in Section 8.3). Therefore, it is strongly recommended to get the option list and the associated name-ID mapping for each option at runtime with `ftkEnumerateOptions`, and then, across the code when getting/setting an option, to call the options with their string names instead of their IDs.

Option configuration delay When setting an option value, the effect of the option may not be instantaneous, and there is especially no guarantee that the option will take effect in the next frame provided by the device.



When the application requires that an option is set, the integrator must not assume that the configuration is done when exiting the setting function, but must ensure that the option is applied before using the data.

8.1.1 Library related options

This subsection presents the library related options, which are options not directly linked to a specific device.

Name	Type	Description
Library Version	<code>ftkBuffer</code> , read-only, global	Allows to read the SDK full version string.

Name	Type	Description
Device library version	<code>ftkBuffer</code> , read-only, global	Allows to read the device layer library full version string.
Data directory	<code>ftkBuffer</code> , read/write, global	Allows to access the path from where the marker calibration files and AVT software parameters files are retrieved.
Frame processing wall-time	<code>int32</code> , read / write	Allows to set a walltime on the frame processing. The default value (-1) set no walltime (which is the legacy behaviour). Any positive number n is interpreted as follows: the walltime will be $n \cdot 100 \mu s$. If a walltime is set, the <code>ftkGetLastFrame</code> function can return the <code>ftkError_FTK_ERR_ALGORITHMIC_WALLTIME</code> if the set value is exceeded, and the gotten frame will contain no data but the picture header.

8.1.2 Device related options

This subsection presents the device related options, which tune the behaviour of a specific device. Some of those are permanent, i.e. are only reset after a reboot of the device.

Name	Type	Description
Picture rejection threshold	<code>int32</code> , read / write, permanent	Allows to set a maximum picture size, above which pictures won't be sent by the fusionTrack device, helping to keep up with the desired latency.
Image Compression Threshold	<code>int32</code> , read / write, permanent	Allows to access the picture compression threshold, i.e. minimum pixel intensity for a pixel to be considered white. Increasing the value decreases the compressed picture size.
Image Integration Time	<code>int32</code> , read / write, permanent	Allows to access the sensor integration time, i.e. the exposure time for the pictures. If the strobing is enabled, the strobe time of IR-LED is automatically synchronised.
Alternate image integration time	<code>int32</code> , read / write, permanent	Allows to access the <i>alternate</i> sensor integration time, i.e. the exposure time for the pictures. This alternate integration time is only applied when 'Strobe mode' is set to 2, and used for the frame with unsynchronised strobe and exposure.
Temperature sensor index	<code>int32</code> , read / write	Access the index of the temperature sensor to be accessed with the 'Temperature sensor location', 'Temperature sensor value' and 'Temperature sensor reference' options. Reading a value before setting a valid index will result in an error.

Name	Type	Description
Temperature sensor location	<code>ftkBuffer</code> , read-only	Allows to read the location of the temperature sensor for the index set with the 'Temperature sensor index' option. Reading a value before setting a valid index will result in an error.
Temperature sensor value	<code>float32</code> , read-only	Allows to read the current temperature sensor reading for the index set with the 'Temperature sensor index' option. Reading a value before setting a valid index will result in an error.
Temperature sensor reference	<code>float32</code> , read-only	Allows to read the median temperature sensor value for the index set with the 'Temperature sensor index' option, <i>during the calibration</i> . Reading a value before setting a valid index will result in an error.
Calibration type	<code>int32</code> , read-only	Allows to read the type of the calibration file stored in the device. The value corresponds to the <code>CalibrationType</code> enumeration.
Calibration processing datetime	<code>int32</code> , read-only	Allows to read the date and time of the calibration processing (given in UTC).
Calibration export	<code>int32</code> , read-write	Allows to set the export of the calibration parameters in the frame, so that they can be read using the <code>ftkExtractFrameInfo</code> function.
Enables IR strobe	<code>int32</code> , read / write, permanent	Allows to toggle on/off the IR strobe on each frame. This option is only available until firmware version 0x1.1.6.d2.
Strobe mode	<code>int32</code> , read / write, permanent	Allows to switch between the different strobing modes: 0: strobe and picture acquisition are synchronised; 1: strobe and picture acquisition are desynchronised; 2: strobe and picture acquisition are synchronised on every second acquired frame. This option is only available with firmware version 0x1.1.8.3 and newer.
Data sending	<code>int32</code> , read / write	Allows to toggle on/off the data sending by the fusionTrack device.
User-LED red/green/blue component	<code>int32</code> , read / write, permanent	Allow to access the red, green or blue component of the user LED. It has no effect as long as 'User-LED enable' is set to 0. ¹

¹Only available for the spyTrack and fusionTrack version 00.10.00 and more recent.

Name	Type	Description
User-LED frequency	int32, read / write, permanent	Allows to access the blinking 'frequency' of the user LED. A value of 0 disables the blinking, otherwise the higher the value the faster the LED blinks. It has no effect as long as 'User-LED enable' is set to 0. ²
Enables the user-LED	int32, read / write, permanent	Allows to toggle on/off the user LED. ³
Acquisition frequency	int32, read-only	Allows to read the fusionTrack device acquisition frequency, in Hz.
Shock monitoring period	int32, read-write, permanent	Allows to access the interval (in seconds) between two checks of the shock sensors. When set to default value (0), there will be no reporting of shocks. For any positive value, the fusionTrack device will check at the set interval if a shock occurred, and the <code>ftkGetLastFrame</code> function will report the <code>FTK_WAR_SHOCK_DETECTED</code> warning.
Shock sensor threshold	int32, read-only, permanent	Allows to access the shock sensor threshold value. The value is expressed in terms of the Earth gravitational constant (9.81 m s^{-2}).
Device Settings	int32, write-only	Allows to save and restore the device current option values. Writing 0 saves the current options in a file <code>SN_settings.ini</code> located at <code>install_path/data</code> , whilst writing one tries to restore from the <code>SN_settings.ini</code> file. This option is deprecated, use 'Save environment' and 'Load environment' instead.
Save environment	ftkBuffer, write-only	This option allows to save the value of <i>all</i> read-/write options to a <i>JavaScript Object Notation (JSON)</i> file, which can then be loaded to restore all settings at once. The <code>ftkBuffer</code> contains the path to the <i>JSON</i> file to create (see Section 8.1.6).
Load environment	ftkBuffer, write-only	This option allows to restore the value of <i>all</i> read-/write options from a <i>JSON</i> file. The <code>ftkBuffer</code> contains the path to the <i>JSON</i> file to load (see Section 8.1.6).
Buzzer period	int32, read / write, permanent	Allow to access the device buzzer sound period (i.e. inverse of frequency) register. It has no effect as long as 'Buzzer repetition' is not set to at least 1.
Buzzer duration	int32, read / write, permanent	Allow to access the device buzzer sound duration register. It has no effect as long as 'Buzzer repetition' is not set to at least 1.

²See note 1

³See note 1

Name	Type	Description
Buzzer repetition	int32, write	Start the buzzer with the given repetition. The buzzer period and duration are controlled with 'Buzzer period' and 'Buzzer duration' options.
MTU size	int32, read-only	Allows to read the MTU size used by the fusionTrack device for data sending. ⁴
Local UDP port	int32, write-only, global	Allows to force the local used UDP port for the communication with the fusionTrack device. If the set port is not available, enumerating devices will result in no devices found.
Counter of lost frames	int32, read-only	Allows to read how many frames were either not sent by the fusionTrack device or not collected by the device layer library. Reading this option before setting 'Copies lost counters' will result in an error.
Counter of corrupted frames	int32, read-only	Allows to read how many frames were rejected by the device layer library, for instance because one UDP packet was dropped. Reading this option before setting 'Copies lost counters' will result in an error.
Copies lost counters	int32, write-only	Copies the internal values accessed by the 'Counter of lost frames' and 'Counter of corrupted frames' options.
Resets lost counters	int32, write-only	Resets the internal values accessed by the 'Counter of lost frames' and 'Counter of corrupted frames' options to zero.

8.1.3 Detection stage related options

This section presents the options which are related to the 3D reconstruction of points and markers.

Name	Type	Description
Blob Minimum Aspect Ratio	float32, read / write	Defines the the minimal aspect ratio to consider a <i>raw data</i> during the raw detection phase. For example, an aspect ratio of 1 means that the bounding box of the <i>raw data</i> is a square. An aspect ratio of 0.5 defines a rectangle with one edge of length x and the other of length $2 \cdot x$.
Blob Minimum Surface	int32, read / write	Defines the minimal surface in pixels to consider a <i>raw data</i> during the raw detection phase. This threshold allows to reject small polluting regions in the picture.

⁴See note 1

Name	Type	Description
Blob Maximum Surface	int32, read / write	Defines the maximal surface in pixels to consider a <i>raw data</i> during the raw detection phase. This threshold allows to reject large polluting regions in the picture
Symmetrise coordinates	int32, read / write	Allows to toggle on/off the symmetrised coordinate system. When on, the origin of the coordinate system is computed as the point between the left and the right cameras.
Epipolar Maximum Distance	float32, read / write	Defines the maximum distance (in pixels) between the right <i>raw data</i> and the right epipolar line defined by the left candidate during the match search.
Matching Tolerance	float32, read / write	Defines the tolerance in mm between the measured and reference lengths defined by two <i>fiducials</i> of a <i>marker</i> , during the <i>marker</i> matching phase. The behaviour of this option is not corresponding to its expected behaviour, i.e a value of 5 mm does actually <i>not</i> correspond to a tolerance of 5 mm on the matched distance. If this option is changed when the new algorithm is used, a warning is issued.
Registration Mean Error	float32, read / write	Defines the tolerance in mm after registration of the <i>marker</i> geometry with the measured one.
Matching Maximum Missing Points	int32, read / write	Defines the maximum numbers of missing points when reconstructing a <i>marker</i> candidate valid during the <i>marker</i> matching phase (see the 'Matching Maximum Missing Points' paragraph of Section 7.8 for more details).
Tracking range	float32, read / write	Defines the range (in pixels) where to look for a <i>raw data</i> associated to a <i>marker</i> according to its position in the previous frame(s) (See 'Tracking time span').
Tracking time span	int32, read / write	Defines how many frames in the past the algorithm will inspect in order to identify a <i>raw data</i> associated to a <i>marker</i> .
New marker reco algorithm	int32, read / write	Toggles between the old marker reconstruction algorithm (0) and the new one (1).
Distance matching tolerance	float32, read / write	Defines the tolerance in mm between the measured and reference lengths defined by two <i>fiducials</i> of a <i>marker</i> , during the <i>marker</i> matching phase, when using the <i>new</i> marker reconstruction algorithm. If this option is changed when the old algorithm is used, a warning is issued.

8.1.4 Wireless related options

This section presents the options related to the wireless markers.

Name	Type	Description
Active Wireless Pairing Enable	int32, read-write	Toggles on / off the wireless marker search by the fusionTrack device. This option is not compatible with the 'Marker button streaming' and 'Marker battery streaming' ones.
Active Wireless Markers reset ID	int32, write-only	Resets a wireless marker using its short ID.
Active Wireless button statuses streaming	int32, read / write	Toggles the periodic scanning of the wireless button status and its streaming in the <code>ftkFrameQuery::events</code> member. If the 'Enable pairing' option is enabled, the streaming is disabled, whatever the value of this option.
Active Wireless battery state streaming	int32, read / write	Toggle the periodic scanning of the wireless battery level and its streaming in the <code>ftkFrameQuery::events</code> member. If the 'Enable pairing' option is enabled, the streaming is disabled, whatever the value of this option.
Active Wireless Markers enable mask	int32, read / write	Set which wireless markers are fired on each frame. The options is a bitfield, meaning that if bit i is set to 1, marker with short ID i will be on. Due to the maximum number of simultaneously tracked wireless markers (which is 16), only the 16 LSB are used.
Active Wireless Markers info	ftkBuffer, read-only	Reads the information on paired markers, in CSV format. The file contains a header explaining the content. The field separator is a comma ','.

8.1.5 Authentication related options

This section presents the options related to the authentication mechanism, described in Chapter 16.

Name	Type	Description
Set cypher key	ftkBuffer, write-only	Allows to set the 128 bits encryption key used in the device authentication mechanism.
Request challenge	ftkBuffer, write-only	Allows to set the 64 bits challenge used in the device authentication mechanism.
Challenge result	int32, read-only	Allows to get the result of the previously set challenge in the authentication mechanism.

8.1.6 Environment and options

In SDK version 4.3.1 was introduced the concept of *environment*, which represents a snapshot of the settings of a fusionTrack device. The environment is defined by all read *and* write options, which can then either be saved to or loaded from a *JSON* file. The usage of the environment is triggered by the two options *Save environment* and *Load environment*, which are write-only options and take the path of the *JSON* file to read / write. In addition to the option values, the file also contains the device firmware version and the SDK version used for generation. This allows to ensure the saved data are correctly applied when loaded.

8.2 User interface software 'demo.exe'

The Atracsys *SDK* (documented in [1]) is shipped with a precompiled (*GUI*) program. This program is run by executing `demo64.exe`. This program consists of a main window, which is composed of several panels, and a secondary window showing a 3D representation of the scene (Windows only).



The GUI demo software is not part of the fusionTrack software system. It is only meant to provide a showcase of what can be achieved using the SDK. The GUI demo software does not belong to the fusionTrack product.

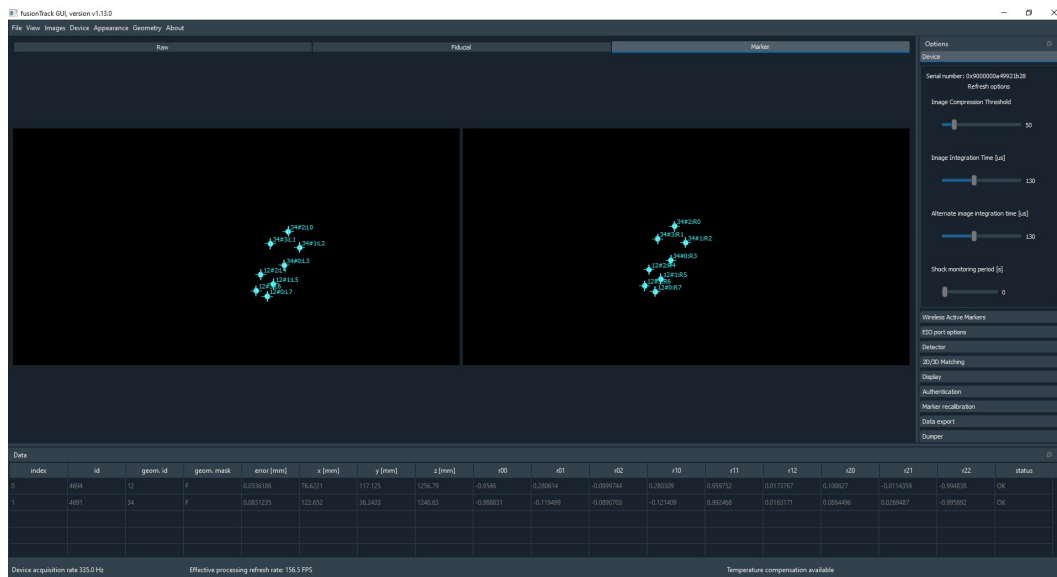


Figure 8.1: Running *GUI* program.

The menus allow to perform the following actions:

- 'File → Quit' closes the *GUI* program;
- 'View → Options' (re)opens the option panel;
- 'View → Data' (re)opens the data panel;
- 'View → 3D Viewer' (re)opens the 3D visualisation window;
- 'View → Log' (re)opens the log window;

- 'View → Reset Zoom' resets the zoom on the two pictures, this action can be achieved per picture by pressing the central mouse button over the desired picture;
- 'View → Save layout' saves the current layout (positions and sizes of the different panels and windows);
- 'View → Restore layout' restores the layout (position and sizes of the different panels and windows) from a previously saved state;
- 'Images → Save' allows to save pair(s) of pictures in the wanted directory, the picture file names are Left_*X*.png and Right_*X*.png, where *X* is an integer on 3 digits, automatically incremented;
- 'Geometry → Geometry *X*' (un)loads geometry with ID *X*. Unless specified differently during installation, geometry files are located at C:\Program Files\Atracsys\fusionTrack SDK\data and user-defined geometries are loaded from %APPDATA%\Atracsys\PassiveTrackingSDK A greyed-out file indicates the file could be seen but not read (due to a syntax error for instance).
- 'About → Help' shows the GUI-specific help document.

Central panel The main window central panel shows the two pictures captured by the cameras. Just above the two pictures is located a tab selector, allowing to choose the type of information to show, corresponding to the data at different processing stages:

- *Raw data*, which are the white areas (aka blobs) detected on each picture;
- 3D, which consist of a 3D point reconstructed from a pair of *raw data*;
- *Marker*, consisting of a 3D position and a rotation matrix (this is what is shown on Figure 8.1). The demo allows to open the two storage locations in order to see the files.

Lower panel Below the two pictures is located a dockable table, which shows data information. The shown data depends on the chosen type (i.e. *Raw*, *Fiducial* or *marker*). Part of the data (mainly position and indices) is replicated on the pictures, as superimposed information, whose colour depends on the data type: yellow is used for *raw data*, green for *fiducial* and cyan for *marker*.

Status bar The status bar is divided into several areas, which present (from left to right):

1. the device acquisition frequency in Hz;
2. the application refresh rate in Hz;
3. the coordinate of the cursor in the picture space⁵;
4. the value of the hovered pixel⁶;
5. the availability of the temperature compensation.

⁵Only active if the ALT/meta key is pressed and the cursor is hovering the left or right picture.

⁶See note 5

Right panel On the right-hand side of the window is a dockable panel, allowing to change the values of the available options, perform *marker* recalibration, and export data.

The options are classed in several groups:

1. Device;
2. Wireless Active Markers;
3. Detector;
4. 2D/3D matching;
5. Display;
6. Authentication.

The Device-related option tab contains the following options (which are defined in Section 8.1):

- **Serial Number** which is the serial number of the tracking device. This option is read-only;
- **Image Compression Threshold;**
- **Image Integration Time;**
- **Alternate image integration time;**
- **Shock monitoring period;**
- **Picture rejection threshold;**
- **Enable data sending;**
- **Enable IR strobe**⁷;
- **Strobe mode**⁸;
- **Buzzer period;**
- **Buzzer duration;**
- **User LED red component;**
- **User LED green component;**
- **User LED blue component;**
- **User LED frequency;**
- **Enables the user LED;**
- **Frame processing walltime;**
- **Calibration export;**
- **Temperatures** shows the temperature of the various temperature sensors in the device. The temperatures are presented in small frames;

⁷This option is only available until firmware version 0x1.1.6.d2.

⁸This option is only available with firmware version 0x1.1.8.3 and newer.

The Wireless Active Markers option tab contains the following options:

- **Paired markers** contains a table (See Figure 8.2) where paired *markers* are listed. The statuses of the buttons of the paired *markers* as well as the battery state are displayed. When a field is in RED, it means that the value has not been updated recently (2 second for the battery, 50ms for the buttons). The cause is most probably the *marker* being out of reach of the infrared transmitter and receptor of the fusionTrack. Letting the mouse cursor on one of the battery field in the table will display a tooltip indicating the maximum and minimum voltages for the battery. The table also contains a column called "Enable" with checkboxes. When checked, the fusionTrack will activate the LEDs of a given marker at each cycle. This allows the tracking of the marker. When unchecked, the marker LEDs will never be lighted on. The tracking is therefore impossible. By default, the tracking is enabled after a pairing.

When a row is selected in the table, additional information about the selected *markers* is displayed:

- **Serial:** The serial number of the *marker*. This number should also be available on a sticker on the back of the *marker*.
- **Firmware Version:** The version of the firmware used in the *marker*.
- **Firmware Date:** The timestamp of the firmware used in the *marker*.
- **Model:** This value identifies the type of the *marker*. Currently three types are available: Wireless Boomerang (23), Wireless Pointer (73) and Wireless Development Kit (98).
- **PCB Version:** This value identifies the version of the electronic (the PCB) in the *marker*.
- **PCB Assembly:** This value identifies the version of the electronic parts assembled on the PCB.
- **Mechanical version:** This value identifies the version of the mechanicals parts of the *marker*.
- **Geometry:** The identifier of the geometry used for the tracking. The same value can be found in the data panel under the column "geom.id".

A button named "Reset selected" under the table allows to unpair the selected *marker*.

- **Active Wireless Pairing Enable** allows to enable or disable the pairing of active *markers*. When enabled, any *marker* put in front of the fusionTrack will be paired. While enabled, the performances are slightly degraded. Therefore, it should be disabled once all *markers* are paired.
- **Active Wireless button statuses streaming** allows to enable or disable the streaming of button statuses of each paired *markers*. When enabled, the fusionTrack will ask every *markers* the statuses of their buttons (pressed or not) and stream it to the application at every frame.
- **Active Wireless battery state streaming** allows to enable or disable the streaming of the battery state of each paired *markers*. When enabled, the fusionTrack will ask every *markers* the statuses of their battery and stream it to the application every second.

The Detector-related option tab contains the following options:

- **Blob Minimum Aspect Ratio;**
- **Blob Minimum Surface;**
- **Blob Maximum Surface;**
- **Symmetrise coordinates.**

The 2D/3D matching-related option tab contains the following options (which are defined in Section 8.1):

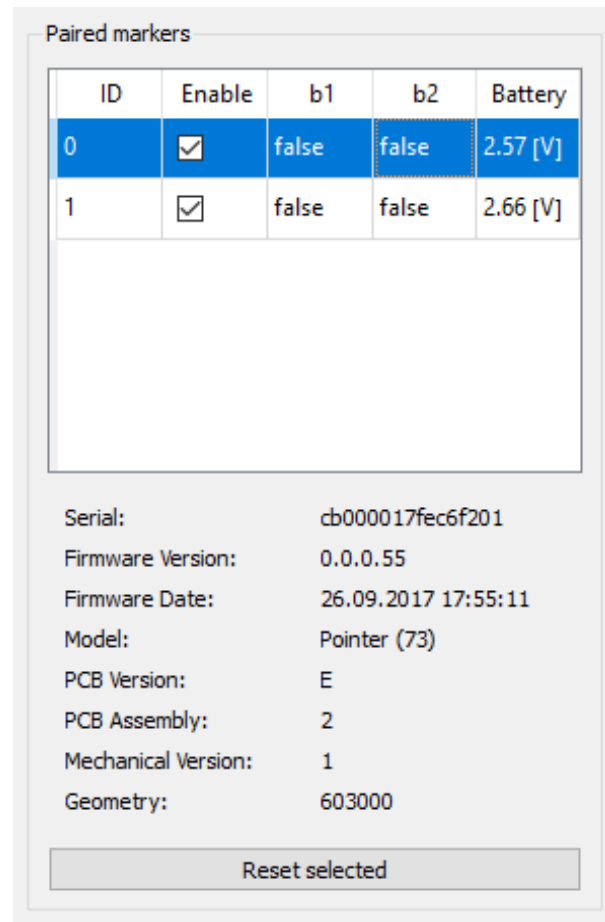


Figure 8.2: The Paired markers widget lists all the paired wireless markers and display their statuses.

- **Epipolar Maximum Distance;**
- **Ignore hard working volume cuts;**
- **Distance matching tolerance;**
- **Matching Tolerance;**
- **Registration Mean Error;**
- **Matching Maximum Missing Points;**
- **Tracking range;**
- **Tracking time span;**
- **New marker reco algorithm.**

The Display-related option tab gathers options which are specific to the *GUI*, i.e. they do not correspond to any option from the *SDK*. They are:

- **Display images on UI** enables picture display. This can have an impact on the *GUI* framerate on low-end PCs:

- **Display error info in images** enables showing additional information on the picture, i.e.:
 - aspect ratio for the *raw data*;
 - epipolar and triangulation errors for *fiducials*;
 - registration error for *markers*;
- **Update Timer Timeout** allows to limit the *GUI* framerate, the value of the minimal interval between two calls to `ftkGetLastFrame` in ms.

The Authentication-related option tab allows to check if the fusionTrack device configuration was tampered with. The authentication requires to upload a 128 bits key on the device. This widget asks for the key, and allows to enter a 64 bits challenge, which is sent to the fusionTrack device and the result is checked against the locally computed number. The following options are available:

- **Set cypher key** as 'Set key';
- **Request challenge** as 'Send challenge';
- **Challenge result** as 'Get result'.

8.2.1 Recalibration of a marker

In order to get even more accurate measurements, the *markers* can be calibrated each time the reflecting *fiducials* are mounted / dismounted. The demo application allows the user to perform this operation. This operation requires that only the *marker* to recalibrate is seen by the fusionTrack (and the corresponding geometry must be activated), it also requires the temperature of the device to be stable, i.e. the device must be running for some time. The procedure records the *fiducial* positions for a maximum of 10,000 frames and creates a new geometry file, which is saved automatically and can be opened directly from the demo application. It also will be automatically added to the list of known geometries and read at the next start of the demo application. The old geometry is automatically disabled and the new one enabled. The produced file is stored in a user-specific location (%APPDATA%\Atracsys\PassiveTrackingSDK on Windows).

The recalibration algorithm can reassign fiducial IDs, i.e. fiducial 0 from the old geometry might become fiducial 2 in the new one. This is done such that similar geometries have similar geometry files. The choice is based on distances to choose fiducials 0, 1 and 2. The first three fiducials are always coplanar.

8.2.2 Exporting the data

The demo program allows to export the received data in CSV format⁹. The dedicated panel 'Data export' allows to chose the exported data and the number of saved frames. The export consists of up to four files, respectively containing:

1. the *raw data*;
2. the 3D *fiducial* data;
3. the *marker* data;
4. the temperature data.

⁹A comma-separated values (CSV) file stores tabular data (numbers and text) in plain text. Plain text means that the file is interpreted as a sequence of characters, so that it is human-readable with a standard text editor. Each line of the file is a data record. Each record consists of one or more fields, separated by commas.

It must be noted that the 'demo.exe' software acquires data from the fusionTrack at the display update rate. This means that exported values won't be recorded at the maximum speed of the fusionTrack. The user can choose which data to export (the default setting is to export everything, i.e. to create three files). The number of frames to record can be set, in which case the saving will automatically stop when this number is reached, or let free. In the latter case, the user will need to stop the recording manually. In order to ensure compatibility, a version of the export can be set, which guarantees that a reading application will be able to read the data created by a newer version of the application. The separator used in the CSV files is the semi-colon (';'), to comply with systems using a comma as decimal mark. The saved information is the same as contained in the `ftkRawData`, `ftk3DFiducial` and `ftkMarker` structures. The written information for *raw data* is the timestamp (in hexadecimal), 'left' or 'right' as an indication of the camera, the index of the *raw data*, the 2D coordinates of the *raw data*, its surface, probability and status, as presented in Listing 8.1.

```
1 timestamp;left/right;index;x;y;surface;probability;status
```

Listing 8.1: Structure of a line of the raw data export CSV file.

The written information for *fiducials* is the timestamp (in hexadecimal), the index of the *fiducial*, its 3D position, the values of the epipolar and triangulation errors, its probability and the indices of the used left and right *raw datas*, as presented on Listing 8.2.

```
1 timestamp;index;x;y;z;epipolar_err;triangulation_err;probability;left_index;right_index
```

Listing 8.2: Structure of a line of the 3D *fiducial* export CSV file.

Finally, the written information for *markers*, shown in Listing 8.3, is the timestamp, followed by the index, the tracking id, the 3D position, the rotation (row-wise), the registration error, the geometry id, the *fiducial* presence mask and the corresponding *fiducial* index.

```
1 timestamp;index;tracking_id;x;y;z;rot[1-9];registration_err;geometry_id;presence_mask;
  fiducials[1-4]
```

Listing 8.3: Structure of a line of the *marker* data export CSV file.

The synchronisation of the data between the various files is achieved by a manual matching of the frame timestamp, present in all files.

8.2.3 Dumping the data

The demo software allows to dump all data of a frame, for diagnostics purpose. The used file format is [XML version 1.0](#), and the produced files contain all the needed information to reprocess the data. The user can enter:

- the number of frames to record (the value `-1` is equivalent to `inf`, i.e. the user must stop the record);
- a file name prefix;
- a 'period' for picture saving, i.e. a picture pair will be saved every n frames (a value of 0 results in *no* saved pictures).

The 'Record' button triggers the choice of the output directory. Then the record starts until the user hits 'Stop recording' or the number of frames to record is reached. The file name consists of the prefix, followed by the device serial number (in hexadecimal) and the current timestamp, e.g. `Dumper_0xbb000005c21ffa28_2017-11-07_14h39m53.xml`.

8.3 Command line software samples

The Atracsys *SDK* is also shipped with precompiled command line software, which show C++ code examples of how to use the *SDK*. From the install directory they can be found in the relative path `fusionTrack SDK/bin`. Each sample demonstrates one operation. In order to run them, the fusionTrack must first be switched on. Once the fusionTrack boot sequence has successfully finished, any of the command line samples can be run. Default firewall settings may block the communication between the computer and the fusionTrack device and should be checked. The documentation of the various samples is present in [1], a short description is given here below.

Each software can take at least an additional configuration file as argument, using the `-c/--config path_to_file`. The configuration files are described in Section 8.6. This configuration file allows to set option values, including an alternate IP for the device (see Chapter 15). Other arguments are described when calling the software with the `-h/--help` switch.

ftk1_ListOptions exposes how to open/close the driver, enumerates the connected devices and lists the options for a device.

ftk2_AcquisitionBasic shows how to load a geometry from a file and how to get the *marker*'s position and orientation. This program only produces output if the geometry of the presented *marker* matches the one in the file given in arguments, and shows the position and error for any found *marker* of that kind (units are mm).

ftk3_AcquisitionAdvanced explains how to get the *marker*'s position and orientation, the *fiducials* positions and the raw data. Again this program requires the use of a *marker* and that its geometry corresponds to the geometry file given in argument. For each found *marker*, the registration error and the position of the fiducials from which the marker was reconstructed are indicated.

ftk4_AcquisitionRawData presents how to display raw data from the frame. This program does not require any *marker*, single or multiple *fiducials* can be used. The raw data for both cameras are printed, then the reconstructed *fiducials* are printed. As the program also tries to save a pair of pictures, it may require admin privileges to be fully functional on Windows.

ftk5_AcquisitionRelative shows how to use one *marker* as reference for the transformation of a second one. This is the usual way to use the data in a *navigation system*. The two original transformations are printed, and the relative position and rotation of the 'slave' *marker* with respect to the 'master' one is computed.

ftk6_ControlLED presents how to control the user LED, by setting the LED colour (the red, green and blue components can be set individually) and its blinking frequency.

ftk7_AcquisitionExtended demonstrates how to get additional information on the current frame. The sample contain an example of how to get the real number of reconstructed 3D *fiducials*, even if no storage has been reserved for the 3D *fiducials*.

ftk8_GetTemperatures shows how to get the temperature sensor names and temperature values. The temperature sensors are identified by a number, and Atracsys guarantees that a sensor with a given ID will always be located at the same place, and that there always will be *at least* one sensor in each of the following regions:

- the left part of the device;
- the right part of the device;
- the front part of the device;
- the bottom part of the device.

But there is no guarantee that a given ID (e.g. 1) will persist during the whole lifetime of the product (i.e. on all past, current or future versions of the product). The current positions for the sensors are illustrated on Figure 8.3, Figure 8.4 and Figure 8.5.

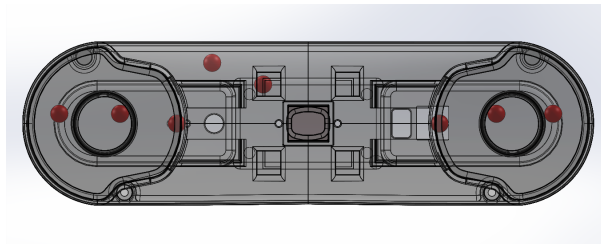


Figure 8.3: Temperature sensors location, front view.

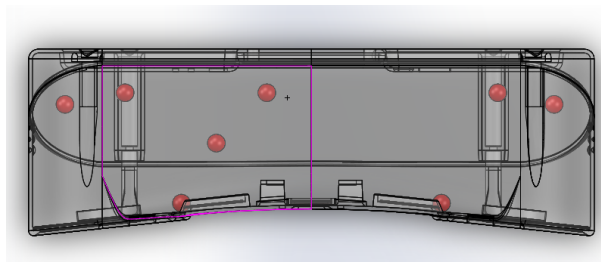


Figure 8.4: Temperature sensors location, top view.

ftk9_GetAcceleration presents how to get the current acceleration values from the accelerometers.

ftk10_GetTimestamp demonstrates how to extract the device timestamp from its Real-Time Clock (RTC) device. This RTC allows to put a timestamp on recorded entries (e.g. shocks, updates, etc.) even if the device is not powered. This RTC timestamp is not to be confused with the μ s-timestamp in the images.

ftk11_WirelessMarker shows how to activate wireless markers and how to use them. The process consists of 4 steps:

1. the wireless mode must be activated;
2. for each detected device, its geometry must be retrieved;
3. the desired wireless markers must be enabled;
4. the data taking must be enabled.

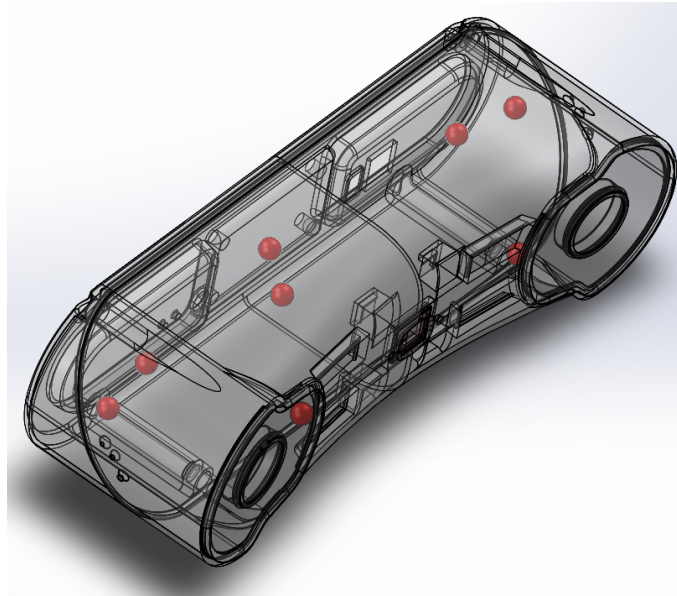


Figure 8.5: Temperature sensors location, isometric view.

ftk18_DumpDeviceInfo shows how to get a dump of the device information. In case of a problem, this dump can be sent to Atracsys for diagnostics. The software creates an XML file, which contains the following information (non-exhaustive list):

- date and time;
- device serial number and type;
- SDK version;
- values of the various options;
- values of the fusionTrack device internal registers.

ftk19_OpticalCommunication presents how to get a wireless marker battery and button status, using events.

ftk20_Latency provides a way to study the relation between latency and the 'Picture rejection threshold', so that the user can determine the value of this option depending on their environment and requirements.

ftk21_MultipleAcquisition shows an example of the C++ API, and allows to connect to several fusionTrack devices.

ftk22_EnvironmentHandling shows how to save and load environment using the C++ API.

ftk24_CheckDeviceAuthentication allows to set a key, request a challenge and check the challenge result. It provides an example on how to implement the authentication mechanism.

ftk26_StrobeMode allows to set the wanted strobe mode, gets 10 frames and displays the strobe status for each of the retrieved frames.

ftk29_ExtractCalibrationParameters demonstrates how to perform the extraction of the calibration parameters from the gotten `ftkFrameQuery` instance. Please note that the distortion parameters order does follow the [Matlab calibration toolbox](#) convention. It must be pointed out to openCV users that this parameter order is different from the openCV convention.

ftk32_InterpolateFrames allows to get an interpolated frame intercalated between the two input ones. The current implementation only interpolates the `ftkRawData`, `ftk3DFiducial` and `ftkMarker` instances corresponding to the `ftkMarker` instances found in *both* input frames, all other data will simply be discarded from the interpolated frame.

ftk33_Buzzer provides an example on how to use the fusionTrack internal buzzer by playing an ascending major scale.

ftk34_ReprocessFrame demonstrates how the user can plug their own reprocessing code and get the SDK to recompute 3D fiducials and markers from the altered data.

ftk35_PTPBasic shows how to use the PTP module. It first swithes ON the module, then enables the synchronisation and shows how to read the sent events related to PTP.

ftk36_TaggerModes provides an example of how to use the EIO tagging mode.

8.4 Latency measuring

The fusionTrack latency depends on several factors, among which

- the picture size,
- the Ethernet adapter performance,

play a key role. When the picture size is too high, the transmission time impacts the latency and the frame rate. This is the reason why the 'Picture rejection threshold' option has been introduced in the 3.0.1 version of the SDK. Atracsys now provides a sample which allows to tune the value of the 'Picture rejection threshold' option according to the target environment and running platform: `ftk20_Latency`, available both as a pre-compiled binary and source code.

The software acquires data at the maximal available rate. It performs several loops during which a certain number of frames are retrieved. The following quantities are computed:

- the average time between two pictures acquisitions at the hardware level (i.e. in the fusionTrack firmware) is computed (and compared with the hardware-set value);
- the average and maximum time between two pictures acquisitions at the host PC level (i.e. the time between two successive calls to `ftkGetLastFrame`;
- the average and maximum difference between the hardware acquisition time (i.e. time between two picture acquisition at the hardware level) and the software acquisition time (i.e. time between two picture acquisition at the host PC level).

8.4.1 Running the software

The user can modify the behaviour of the software via the following options:

- number of recorded frames in each loop, set via the `-f/--nbframes M` switch;
- number of loops, set via the `-n/--nbloops N` switch;
- tolerance on the maximum time between two acquisitions at the host PC level, set via the `-u/--maxUpdateTime T` switch;
- tolerance on the maximum difference between the hardware acquisition time and software acquisition time, set via the `-l/--maxLatencyTime T` switch;
- maximum picture size in bytes, set via the `-s/--pictureSize S` switch.

The software is meant to be run with the fusionTrack device sending busy pictures corresponding to the maximal picture size expected in the user application. In the case of maximal size pictures (blank), these can be easily generated by covering both cameras with a white sheet of paper. The picture size cutoff can then be empirically determined once the maximum allowed latency is determined by the user. An example of output with only 10 loops is given in Listing 8.4.

```

1 ./ftk20_Latency64.exe -n 10
2 This is the Atracsys latency test and update rate program.
3 Copyright (c) Atracsys LLC 2018
4 Detected one with serial number 0x400000091e566128
5 1/10: 1000 received frames among which 1000 have been ignored (size cutoff),    0 dropped
        frames,    0 processed frames    0 missed frames
6 2/10: 1000 received frames among which 1000 have been ignored (size cutoff),    0 dropped
        frames,    0 processed frames    1 missed frames
7 3/10: 1000 received frames among which 1000 have been ignored (size cutoff),    0 dropped
        frames,    0 processed frames    1 missed frames
8 4/10: 1000 received frames among which 1000 have been ignored (size cutoff),    0 dropped
        frames,    0 processed frames    2 missed frames
9 5/10: 1000 received frames among which 1000 have been ignored (size cutoff),    0 dropped
        frames,    0 processed frames    0 missed frames
10 6/10: 1000 received frames among which 1000 have been ignored (size cutoff),    0 dropped
        frames,    0 processed frames    0 missed frames
11 7/10: 1000 received frames among which 1000 have been ignored (size cutoff),    0 dropped
        frames,    0 processed frames    0 missed frames
12 8/10: 1000 received frames among which 1000 have been ignored (size cutoff),    0 dropped
        frames,    0 processed frames    0 missed frames
13 9/10: 1000 received frames among which 1000 have been ignored (size cutoff),    0 dropped
        frames,    0 processed frames    0 missed frames
14 10/10: 1000 received frames among which 1000 have been ignored (size cutoff),    0 dropped
        frames,    0 processed frames    0 missed frames
15 Closing device connection...
16     Theoretical acquisition period: 2985 us.
17     Device raw acquisition period: mean = 2985.08 us, STD = 0.271293 us,  min = 2985 us,
        max = 2986 us, median = 2985 us.
18     Device sending period: mean = 2986.27 us, STD = 73.1067 us,  min = 2985 us,  max =
        8955 us, median = 2985 us.
19
20 UPDATE TIME TEST:
21     Host User PC time between updates: mean = 2986.37 us, STD = 161.047 us,  min =
        11.093 us,  max = 9297.64 us, median = 2986.1 us.
22 Acceptance criteria: Update time maximum < 30000 us. Test result: PASSED
23
24 MAX downstream latency TEST:

```

```
25         Host Latency time added downstream from the device for measurement frames : mean =  
           0.0921108 us, max = 2870.29 us, median = 0.816 us.  
26 Acceptance criteria: Update time maximum < 20000 us. Test result: PASSED
```

Listing 8.4: Example of output.

8.5 Compilation of provided samples

The command-line samples discussed in Section 8.3 are shipped with their source code, in order to present how to use the SDK functions. The makefiles are built using CMake. The cmake configuration file to create the makefile is located in the `fusionTrack SDK/samples` directory, lying in the installation folder. On windows, creating the makefiles in the default installation directory may require administrator rights.

Compiling the samples requires CMake version 3.14 or newer, and a C++11 compliant compiler (e.g. gcc 5.4 or newer, clang 3.8 or newer, Microsoft Visual Studio 2015 or newer).

The SDK comes with helper cmake files, which allows to simply use either the C or the C++ SDK with minimal efforts in a custom-made CMakeListst.txt file, as demonstrated in Listing 8.5.

```
1 /set(Atracsys_DIR "path_to_your_install")  
2 find_package(Atracsys REQUIRED COMPONENTS SDK AdvancedAPI)  
3  
4 # Using the C API  
5 add_executable(test_sample_c test_sample.cpp)  
6 target_link_libraries(test_sample_c Atracsys::SDK)  
7  
8 if (NOT (MSVC AND MSVC_VERSION LESS 1900))  
9     # Using the C++ API  
10    add_executable(test_sample_c++ test_sample_advanced.cpp)  
11    target_link_libraries(test_sample_c++ Atracsys::AdvancedAPI)  
12 endif ()
```

Listing 8.5: Example of linking against the C and C++ APIs.

The CMakeLists.txt file will be looking for the shared libraries in the SDK installation folder. In order to deploy an application built using the SDK, the device64 and fusionTrack64 shared libraries must be distributed as well.

8.5.1 Windows compilation

The cmake command from the cygwin environment won't work, as it will use Unix-like pathes, which are not understandable by Visual Studio.

The procedure using Visual Studio is the following:

1. Create an empty build directory (in which file can be written) and open a terminal in this directory (either cmd or PowerShell);
2. From the aforementioned terminal, run the command below:

```
cmake -G <generator_string> [-A x64] <path_to_the_sample_directory>
```

Valid examples are

```
cmake -G "Visual Studio 15 2017 Win64" "C:\Program Files\Atracsys\fusionTrack SDK\
samples"
cmake -G "Visual Studio 16 2019" -A x64 "C:\Program Files\Atracsys\fusionTrack SDK\
samples"
```

3. Open the `samples.sln` solution and compile. Or Run

```
cmake --build . --config=Release
```

8.5.2 Unices compilation

A build system (e.g. GNU make, Ninja) is also needed on top of cmake and a compatible compiler.

The procedure is the following:

1. Untar the SDK tarball:

```
% tar xJvf fusionTrack_SDK-[...].tar.xz
```

2. Create a build directory and enter it:

```
% mkdir build
% cd build
```

3. Run cmake and compile, using the GNU Make build system:

```
% cmake -G "Unix Makefiles" -DCMAKE_BUILD_TYPE=Release ../fusionTrack_SDK-[...]/samples
% make -j <number_of_cpu>
```

or with the Ninja build system:

```
% cmake -G "Ninja" -DCMAKE_BUILD_TYPE=Release ../fusionTrack_SDK-[...]/samples
% ninja
```

the explicit call to `make` or `ninja` from the two aforementioned examples can be replaced by

```
% cmake --build .
```

8.6 SDK configuration file

Since version 4.2.1 of the SDK, the `ftkInitExt` function allows to specify a *JSON* configuration file. This file currently allows to tune the communication parameters (see Chapter 15) and set custom value of options for a device (as obtained from the 'Save environment' option, see Section 8.1.6). This configuration file consists of optional `"network"` and `"environment"` objects.

Two versions of `"network"` are currently supported: version 2 (see Listing 8.6) (since SDK version 4.7.1) and version 1 (see Listing 8.7). The `"environment"` object currently only supports version 1.

```
1 {
2   "network": // Optional "network" object
3   {
4     "version": 2,
5     "interfaces":
6     [
7       // Mandatory array of at least one adapter object
8       {
9         "adapter": "172.17.1.42", /* Mandatory address (or subnet) of the adapter
10                                for communication with the device */
11         "mtu": 9000,           // Optional maximal value for the MTU
12         "localPort": -1       // Optional, default is -1
13         "devices":
14         [
15           // Mandatory array of at least one device
16           {
17             "address": "172.17.1.9", // Mandatory device address
18             "port": 3509             // Mandatory UDP port used by the fusionTrack
19           },
20           {
21             "address": "172.17.1.3",
22             "port": 3509
23           }
24         ]
25       }
26     ],
27   },
28   "environment": // Optional "environment" object
29   {
30     "devices":
31     {
32       "0x110A1C1B1ADE5028":
33       {
34         "options": // Mandatory array of options
35         [
36           // ...
37         ],
38         "sdkVersion": "v4.5.1" // Mandatory SDK version
39       }
40     }
41   }
42 }
```

Listing 8.6: Configuration file with "network" section version 2.


```
1 {
2   "network" : // Optional "network" object
3   {
4     "version": 1,
5     "interfaces": [
6       // Mandatory array of at least one object giving address and port
7       {
8         "address": "172.17.10.7", // Mandatory address of the fusionTrack
9         "port": 3509,             // Mandatory UDP port used by the fusionTrack
10        "localPort": 13485        // Optional, default is -1
11      }
12    ]
13  },
14  "environment": // Optional "environment" object
15  {
16    // ...
17  }
18 }
```

Listing 8.7: Configuration file with "network" section version 1.

9 Language bindings

In this chapter are described the different bindings Atracsys makes available to the users.



The reference implementation is the distributed C / C++ API. The other language bindings are not subject to thorough testing, as the C / C++ do. Integrators are responsible for their software validation using other language bindings.

9.1 Python wrapper

A Python™ wrapper is distributed, which has been tested with Python 3.7.2 (64 bits) on Windows and on Linux. The C++ / Python interface framework used is [pybind11](#).

The wrapper allows to use *almost* all possibilities allowed by the fusionTrack C++ API, meaning that the classes of the C++ API have just been wrapped in Python.

The Python wrapper is distributed through a compressed archive in the python folder of the SDK. The wrapper needs Python (64 bits), git, CMake 3.10 or newer, and a C++11-compliant compiler (i.e. Microsoft Visual Studio 2015 or newer, gcc 5.0 or newer, clang 3.8 or newer).

To install the wrapper, just run `pip install archiveName.tar.gz` from the command line in the python directory of the SDK.

9.2 Matlab wrapper

A Matlab® wrapper is distributed, which has been tested with Matlab R2016a and Microsoft Visual Studio 2015 on Windows. It consists of a set of C++ source files, which are then compiled via Matlab using Visual Studio, and Matlab scripts files. The Matlab main script, provided as an example, uses the `FusionTrack` class, which is fully documented, as well as the underlying `FusionTrack` class written in C++.

The wrapper allows to:

- detected connected device(s);
- get and set `int32` and `float32` options;
- get `ftkBuffer` options;
- register / clear geometries;
- get the latest frame, with
 - *marker* data;
 - *fiducial* data;

- timestamp, counter;
- image size.

The Matlab wrapper is distributed as a source package containing a CMakeLists.txt file. CMake is responsible to get the Matlab installation and libraries, so that the correct flags are passed to the compiler. The wrapper needs Matlab (64 bits), a C++11-compliant compiler (i.e. Microsoft Visual Studio 2015 or newer, gcc 5.0 or newer, clang 3.8 or newer) compatible with the installed Matlab version.

In order to compile the wrapper on Windows, the cmake generation must be set in 64 bits (See Listing 9.1), and the build type must be specified on Unices (See Listing 9.2).

```
1 REM In a temporary folder do:
2 cmake -DCMAKE_GENERATOR_PLATFORM=x64 "C:\Program Files\Atracsys\fusionTrack SDK x64\matlab"
3 cmake --build . --config Release
```

Listing 9.1: Compiling the Matlab wrapper in windows.

```
1 # In a temporary folder do:
2 cmake -DCMAKE_BUILD_TYPE=Release $ATRACSYS_SDK_HOME/matlab
3 cmake --build .
```

Listing 9.2: Compiling the Matlab wrapper on unices.

On Listing 9.3 is presented how to initialise the wrapper, enumerate connected devices and get the SDK version. A geometry is then registered and finally, a frame is retrieved, showing 4 fiducials and a marker.

```
1 >> trSystem = FusionTrack()
2
3 trSystem =
4
5 FusionTrack with properties:
6
7     FTK_OPT_DRIVER_VER: 4
8     FTK_MIN_VAL: 0
9     FTK_MAX_VAL: 1
10    FTK_DEF_VAL: 2
11    FTK_VALUE: 3
12    version: 1
13
14 >> serials = trSystem.devices()
15
16 serials =
17
18     11889503043542755368
19
20 >> trSystem.getData( serials( 1 ), trSystem.FTK_OPT_DRIVER_VER )
21
22 ans =
23
24 v4.3.1 (1548418737) Windows-6.3.9600
25
26 >> geom = loadGeometry( 'C:\Program Files\Atracsys\fusionTrack SDK x64\data\geometry004.ini'
27 );
28 >> trSystem.setGeometry( serials( 1 ), geom )
29 >> frame = trSystem.getlastFrame( serials( 1 ) )
30
31 frame =
32
33     threeDFiducials: [4x1 struct]
34     markers: [1x1 struct]
35     imageHeader: [1x1 struct]
```

Listing 9.3: Using the Matlab wrapper.

10 Command line utility tool

The fusionTrack SDK is shipped with a tool allowing to perform updates of the fusionTrack device and wireless markers firmware. This tool is a command-line software, called `atnetExecutable64`, available both on Windows and Linux, and a shared library `atnetLib64`. Prior to SDK version 4.8.1, only an executable (`atnet64`) was available (on both Windows and Linux). The `atnetExecutable64` software provides the same functionality as `atnet64` provided. The `atnetExecutable64` library allows a better integration.

The `atnetLib64` library is provided as a shared library and a set of header files, and the API is documented in [1].

10.1 Establishing a connection with `atnetExecutable64`

In order to connect the `atnetExecutable64` software with the fusionTrack device, it is assumed the fusionTrack is correctly setup as described in Chapter 5 and the software on the host PC is correctly installed as indicated in Chapter 6. The procedure is the following:

1. Launch the `atnetExecutable64` (usually located in the `INSTALL_PATH/bin` folder on Linux and in `C:\Program Files\Atracsys\fusionTrack SDK x64\bin` on Windows) with the 0 argument;

```
1 $INSTALL_PATH/bin % ./atnetExecutable64 0
```

2. (Re)boot the fusionTrack;
3. Wait for the connection between `atnetExecutable64` and the fusionTrack to be correctly established. It is established when the `scan` command is completed and the `list` command prints the serial number in the console. A successful connection is achieved when the device status LED is blinking in blue.

If the fusionTrack beeps twice in a row and the status LED is solid green, then the procedure must be started again from step 2.

10.2 Useful commands

The `atnetExecutable64` software allows to get information about the fusionTrack device, for instance when its boot sequence encounters an error and stops, or the used firmware versions must be checked. The `atnetExecutable64` software provides the `info32` and `info_uC` commands, the former allowing to get information about the device and the latter about the shock sensor.

10.2.1 `info32`

The `info32` command provides the information presented in Table 10.1, and the status meaning is explained in Table 10.2. Table 10.2 reads the following way: if bit i is 1, it means the corresponding operation has

been performed, *without* any indication about the success or failure of the operation. Exceptions are bits 1, 8 and 9, which are not tests.

What	Information
appFpgaHard	Application firmware software version ^{a,b}
appFpgaSoft	Application firmware hardware version ^{a,c}
factFpgaHard	Factory firmware software version ^{a,b}
factFpgaSoft	Factory firmware hardware version ^{a,c}
ip	IP address of the device
prot	Version of the communication protocol
status	Status of the POST tests (see Table 10.2)

^a The version in parentheses is the hexadecimal representation of the version number.

^b Version of the software running on the NIOS II processor.

^c Version of the VHDL compilation.

Table 10.1: `info32` output

Bit(s)	Meaning
0	Initialisation tests performed
1	Self-diagnosed error occurred
2	Status LED tests performed
3	Ethernet loopback tests performed
4	Memory tests performed
5	One-wire tests performed
6	Accelerometer tests performed
7	Buzzer tests performed
8	Internal usage
9	Internal usage
10	Reserved
11	Camera tests performed
12	Real Time Clock tests performed
13	Micro-controller (i.e. shock sensor) tests performed
14	EEPROM tests performed
15	IR board tests performed
30...16	Unused
31	All tests performed

Table 10.2: `info32` status explanation

10.2.2 `info_uC`

The `info_uC` command provides the information presented in Table 10.3. The meaning of 'Calibration', 'Error' and 'Status' words is presented in Section 4.1.

10.4 Updating the firmware of the fusionTrack shock detection unit



When performing step 7 of the fusionTrack shock detection unit firmware update procedure, the fusionTrack *must* be at room temperature and switched off for at least 4 h.

The update of the fusionTrack shock detection unit firmware is performed through the command line software `atnetExecutable64`. The connection between the fusionTrack and `atnetExecutable64` is assumed to be established as presented in Section 10.1. The update procedure is the following:

1. Use the command 'update_shockSensor' to upload the shock detection unit firmware in the fusionTrack (see Figure 10.2). For example:

```
1 | update_shockSensor * "C:\Users\john.doe\Download\fTk_accMega_0.0.1.09.bin"
```

Listing 10.2: Uploading the new shock sensor detection unit firmware to the fusionTrack.

2. The device will beep twice;
3. Wait for --> OK to be displayed (See Figure 10.2).
4. Boot the device in application by running the 'reset * 1' command;
5. Exit atnetExecutable64 with the 'exit' command;
6. Reboot the fusionTrack and connect with atnetExecutable64;
7. Run the 'init_uC *' command;
8. Run the 'info_uC *' command, which should output --> Calibration=94 Error=0 Status=3 uC_Firmware=..., followed by --> OK.

```

johannes.burkhardt@ATRONIC210: ~/Windows/research/accMega/Arduino/Firmware/accMega/accMega.ino % ./atnet64.exe 0
Atracsys boot loader command-line utility
version v1.5.0-4-gbf54b3b (1539245313) Windows-10.0.17134
type help to get command list
=====

Searching for devices
--> device=0xf50000097f6ccb28
--> OK

update_shockSensor * "\\192.168.2.13\measurement\FTk firmwares\accMega\30\accMega.bin"
.....
--> size=9212

--> Calibration=23 Error=0 Firmware=0.0.1.38 (0x0.0.1.26) Status=2
--> OK

```

Figure 10.2: Example of updating the shock sensor detection unit firmware.

10.5 Updating the calibration of the wireless markers

The update of the wireless active *markers* with a fusionTrack is done through the command line software `atnetExecutable64`. The connection between the fusionTrack and `atnetExecutable64` is assumed to be established as presented in Section 10.1. The update procedure is the following:

1. Place the active marker in front of the fusionTrack. Make sure this is the only active *marker* switched on in the room. Plug out and in again the battery to reboot the marker.
2. As soon as the led on the marker is in purple, use the command 'set_wMarker_calibration' to upload the wireless marker firmware. For example:

```
1 set_wMarker_calibration * /home/john.doe/Tracking/geometry_file_wMarker.ini
```

Listing 10.3: Uploading the active *marker* calibration to the *marker*.

3. Wait for the fusionTrack to emit a double beep.
4. Boot the fusionTrack with the `reset` command.

```
1 reset * 1
```

Listing 10.4: Starting the fusionTrack.

5. Verify that the marker did boot correctly by checking that its user led is in green.

10.6 Updating the firmware of the wireless markers

The update of the wireless active *markers* with a fusionTrack is done through the command line software `atnetExecutable64`. The connection between the fusionTrack and `atnetExecutable64` is assumed to be established as presented in Section 10.1. The update procedure is the following:

1. Place the active marker in front of the fusionTrack. Make sure this is the only active *marker* switched on in the room. Plug out and in again the battery to reboot the marker.
2. As soon as the led on the marker is in purple, use the command 'update_wMarker' to upload the wireless marker firmware (see Figure 10.3. For example:

```
1 update_wMarker * /home/john.doe/Downloads/wTm_a_00000037.bin
```

Listing 10.5: Uploading the active *marker* firmware to the *marker*.

The firmware is different depending on the type of the marker:


Marker type	Firmware name
Wireless Boomerang Marker	wBo_e_xxxxxxx.bin
Wireless Pointer Regular	wTp_e_xxxxxxx.bin
Wireless Marker Development Kit	wTm_a_xxxxxxx.bin
Marker Transceiver	aTt_a_xxxxxxx.bin

3. Wait for the fusionTrack to emit a double beep.
4. Boot the fusionTrack with the `reset` command.

```
1 reset * 1
```

Listing 10.6: Starting the fusionTrack.

5. Verify that the marker did boot correctly by checking that its user led is in green.



```
atracsys@atraclinux ~/mnt/secure/marker:soff/saFt...marker-20mm...soff...file/device/1e6cfd28 % ./atnet64 0
=====
Atracsys boot loader command-line utility
version v1.4.1-20-g788581a (1535102307) Linux-4.14.65-desktop-1.mga6
type help to get command list
=====

Searching for devices
--> device=0xb50000091e6cfd28
--> OK

update_wMarker * wTp_e_00000060.bin
.....
--> size=4154

--> OK
```

Figure 10.3: Example of an atnet session to upgrade active *markers*.

10.7 Obtaining a shock report

The shock reports are obtained via the command line software `atnetExecutable64`. The connection of the fusionTrack and `atnetExecutable64` is assumed to be established in Section 10.1. The process is in two steps:

1. The data is downloaded from the device;
2. The data is sent to Atracsys' processing server and the pdf report is sent by email.

The procedure is the following:

1. Use the command '`dIEEPROM`' to download the data and send them to Atracsys. For instance:
`dIEEPROM * your_email_address`
2. Wait for `--> OK` to be displayed;
3. Boot the device in application firmware by switching off and after 5 s on again or by running the '`reset * 1`' command;

An acknowledge of the report request must be received in the next few minutes, and the report should be received in the next 30 min.

11 Control the operability of the fusionTrack system and instruments

In this chapter is explained how to guaranty the operability of the fusionTrack. Outputs and alert messages are generated to protect against hazardous behaviours. They can also be used to validate the fusionTrack operability.

It is assumed that the fusionTrack is correctly setup as described in Chapter 5 and the software on the host PC is correctly installed as indicated in Chapter 6.

The various checks, presented in the following sections, allow to ensure that the computed *marker* position is valid. Different outputs and messages could be used to indicate the status of those tests.

11.1 Control fusionTrack operability



The zeroing method must be performed before the navigation procedure is started, or during use, as soon as a marker is changed, cleaned, or if the fusionTrack device must be cleaned, whilst the accuracy assessment method is performed with a frequency determined by the integrator.

There are three ways to check the fusionTrack system is correctly working:

1. the monitoring of the registration error, see Section 11.2;
2. the 'zeroing method', described in Chapter 12;
3. the 'accuracy assessment method', described in Chapter 13.

The two last methods guarantee the fusionTrack is operational in the configured environment, by checking the calibration, integrator settings and environment configuration. The zeroing method must be performed before the navigation procedure is started, or during use, as soon as a *marker* is changed, cleaned, or if the fusionTrack device must be cleaned, whilst the accuracy assessment method is performed with a frequency determined by the integrator. The two methods allow to avoid (non-exhaustive list):

- using a decalibrated fusionTrack device, which can be due to shock during transport, storing or handling;
- using the fusionTrack device in noisy environment, such as an air perturbation between fusionTrack device and *marker*;
- any electromagnetic interference with internal electronics;

all of those might lead to wrong measurements.

11.2 Control marker registration and epipolar errors



At any time, the marker registration error and the marker epipolar error (defined in Section 7.8) computed by the *SDK* must be monitored. The integrator must define thresholds above which the errors can lead to unacceptable risk for the patient. If the errors go above the aforementioned thresholds, the integrator must prevent the user from accessing the data.



A large epipolar error affecting all fiducials can be caused by the fusionTrack operating outside its temperature range or being decalibrated.



A large registration error can be caused by the orientation of the *marker*: e.g. if the *marker* becomes close to parallel to the optical axis. The integrator shall generate an error message if the orientation goes above a defined angular threshold.



A large registration error can be caused by an object in the field of view which occludes partially the marker. The integrator shall state in their user manual that no objects in the field of view are allowed but the *markers*, pointers or surgical tools.



A large registration error can be caused by a scratched fiducial, liquids on fiducial, damaged fiducial, wrongly fixed fiducials on the marker, or a deformed marker. The integrator must generate an error message when the marker registration error is too high.

11.3 Control fusionTrack temperature



The integrator shall monitor temperature sensors. If the measured temperatures diverge from the reference temperatures, the tracking data shall not be given to the end-user.

The fusionTrack device is operational after warm up. If used too early, during thermal expansion, this could lead to wrong measurements. The most accurate measurements are obtained when the temperature of the fusionTrack corresponds to its calibration temperature. The *SDK* allows to access the fusionTrack device temperature sensors. This is explained in details in the `ftk8_GetTemperatures` sample, which code and binaries are available in the *SDK* installer / tarball. The calibration temperature is reached after approximately 20 min for a room temperature of 20 °C.



A too high temperature due to internal overheat, or due to the fusionTrack device being wrongly placed near a heat source, might lead to unexpected temperature measurements. The integrator has to trigger an error message indicating that the fusionTrack sensor has detected an abnormal temperature within the system.



Using the fusionTrack device out of humidity specifications might lead to wrong measurements. The integrator has to specify in the end user manual the fusionTrack humidity specifications as described in Table 5.1.

11.4 Control optical elements and IR illumination



Any degradation of optical elements due to the cleaning method or scratches during transport can lead to wrong measurements. Optical elements should be handled with care during transport and storage. The integrator has to mention in the end user manual the cleaning methods for optical elements as described in Chapter 17.

11.5 Control environmental conditions



Instability of the fusionTrack device can cause injury by falling on patient or operator. Attachment between feet and holding mechanism must be perfectly stable. Instructions indicating how to fix fusionTrack using the threads in the feet are provided in Section 5.4.



Polluting light in the field of view might lead to wrong measurements. The integrator can access the camera images to visualize the scene of view. A segmentation overflow mechanism and status flags in the frame structures can be used by the integrator to detect any perturbation. The integrator is suggested to periodically check the camera pictures.



The fusionTrack is not intended to be sterilised. A contamination on cameras could lead to patient contamination. The fusionTrack has to be placed outside of the sterile field and depending on the intervention, cleaned before every single use. Methods of cleaning are indicated in Chapter 17.



The integrator must check that the fusionTrack device and the position of the fusionTrack device is well-suited to the application / surgery, and that required regulations are respected.

11.6 Integrator software



At any time, the hardware (i.e. the fusionTrack device), the firmware and the software (i.e. the *SDK*) must be compatible. New releases of firmware, software and their respective compatibility will be notified to the integrator.



A software dysfunction due to software crash might cause wrong measurements causing patient injury. The integrator has to indicate in the end user manual that the fusionTrack device must be restarted when the software freezes or crashes.



A software dysfunction such as frozen pictures might cause wrong measurements causing patient injury. The integrator has to implement a software mechanism to detect such cases (e.g. check timestamps, etc.).



In the case of the fusionTrack device stops functioning during use / operation, it is the integrator responsibility to ensure that surgeon resume operation with the conventional method.



Any wrong settings of the fusionTrack device, misuse of the *SDK* or misuse of the OS functionalities might cause wrong measurements causing patient injury. The integrator should perform unit tests during the development process to validate configured settings.



The integrator must take care that the host PC meets the *SDK* and their software requirement, especially the needed memory and the nominal CPU usage must be paid attention to.



The integrator implementation may cause inconsistent latency. The integrator should evaluate the system's latency once implemented and integrated in his product.



The integrator must check that UDP packets are well-ordered, i.e. that packets sent sequentially are received in the same order. The UDP protocol does not provide any guarantee, therefore the integrator must check the UDP packet ordering on their platform. Unordered packets may affect the performances of the fusionTrack.



A software dysfunction due to unavailable data could lead to wrong measurements or the non performing of the surgery as planned. The integrator has to perform unit tests according to the end application of the fusionTrack device.



A non respect of the software functionality or unexpected software failure might cause wrong measurements. The integrator should perform unit tests during the development process to validate the software functionality.



On Unix/Linux, the integrator must ensure the software is functional with the used GNU C library, C++ Standard Library and POSIX thread versions.



The integrator must generate a 'noisy environment' error message when the environment is polluted with reflective areas or other light sources.

The version of the *SDK* can be retrieved using the `ftkVersion` function, documented in [1].

12 The zeroing method



The zeroing method described in Chapter 12 must be implemented by the integrator and be automatically run at the application startup and inform the user on how to proceed.

The zeroing method is a procedure which checks that the fusionTrack device and the used *markers* are correctly calibrated. A failure of this test does not allow to determine which element is faulty: it is a global test. In case of a failure of this test, the navigation process is not allowed to start.

12.1 General concept

The principle of the zeroing method is simple: a known distance is checked to be measured correctly. Most surgical applications use a pointer, the position of the pointer tip is therefore a good candidate to be monitored. This is achieved by using the pointer and an additional *marker*, the latter is required to have a cavity, named "divot", in which the tip of the pointer must fit whatever the pointer position and orientation (e.g. a cone or a cylinder), as illustrated in Figure 12.1. Once the pointer tip has been inserted in the *marker* cavity, the pointer should be rotated around its tip, which must remain in the *marker* cavity. The fusionTrack records the position and orientation of the marker and the pointer, from which the position of the tip of the pointer relative to the marker is extracted. This relative position should not vary within measurement uncertainties. During this operation, the marker must lie around the centre of the working volume, as shown in Figure 12.2.

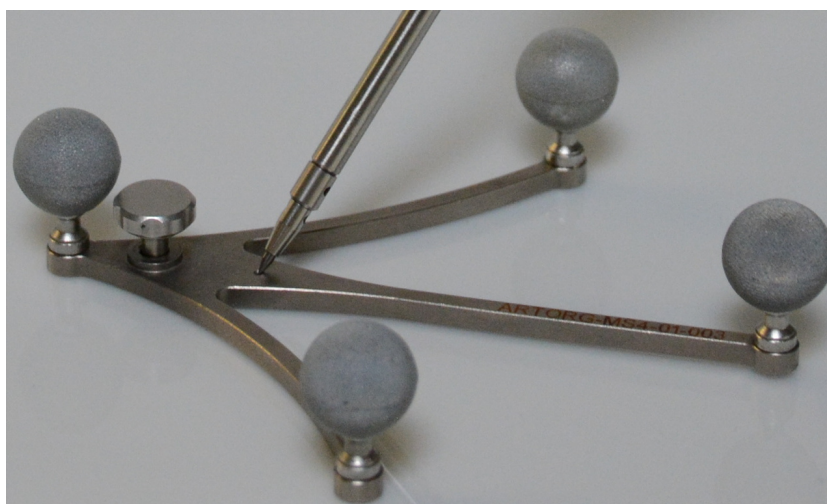


Figure 12.1: The tip of the pointer must lie in the cavity.



Figure 12.2: Running the zeroing procedure.

The integrator must specify a maximal value on the error of the position, above which the test is considered to fail. A failure of the test may be caused by (non-exhaustive list):

- a deformed pointer or *marker*;
- a decalibrated fusionTrack device;
- dirty optics on the fusionTrack device.

12.2 Running the procedure

The zeroing procedure must be integrated in the workflow at each startup of the application or as soon as the pointer (whole pointer or any of its *fiducials* in case of passive *markers*) is changed. The operator for this test is the surgeon. How to proceed must be clearly stated in the user manual.



Any liquid on lenses such as body fluids ejection or condensation could lead to wrong measurements. Lenses have to be cleaned as soon as they get in contact with liquids. The zeroing method has to be re-performed after each cleaning.



Some disposable passive *fiducials* cannot be cleaned, as the reflective coating would be damaged, and may require to be changed during the operation. The zeroing method has to be re-performed after changing any *fiducials*.

13 The accuracy verification tool (AVT)



Accuracy verification must be implemented by the integrator. They can optionally purchase the required software and hardware at Atracsys.



The operator performing accuracy verification must be correctly trained.

The accuracy verification method is a procedure which checks that the fusionTrack device is able to perform accurate measurements in the whole working volume. A failure of this test indicates that the fusionTrack device must not be used and must be returned to Atracsys.

13.1 General concept

The principle of this test is to verify that a given rigid body keeps its fixed geometry wherever the measurement is performed. For a *marker*, the distance between two *fiducials* is determined and cannot change, independently of the *marker*'s position in the working volume. The test consists of moving a *marker* in the whole working volume, the best way to ensure repeatability being to define 3D positions at which the *marker* should stay for a while (e.g. for 500 frames) and monitor the *marker*'s registration error as well as the distances between the *marker*'s *fiducials*. If the used *marker* is accurately calibrated, it may be enough to look at the width of the distribution of the registration error. The integrator must specify an upper bound on this width, above which the test is considered to fail. In general, monitoring the distance between two *fiducials* of the *marker* is a suitable way to perform this test: again the width of the distribution of the measured length must not be larger than a threshold specified by the integrator.

13.2 Running the procedure

This procedure is not part of the usual running of the *navigation system*: it must be run by a trained technician, on a regular basis. For instance the integrator can choose to run this procedure once every week or every 5 surgeries, whatever comes first.

13.3 The accuracy verification tool from Atracsys

Atracsys offers as an additional accessory a rigid body designed to perform accuracy verification on the fusionTrack. Along with this rigid body, Atracsys provides the required software to perform the capture of data and the accuracy analysis. Please contact the Atracsys after sale service for further information (See Section 1.6).

14 Electromagnetic compatibility

The fusionTrack is a medical electrical equipment and needs special precautions regarding EMC and needs to be installed and put into service according to EMC information provided in this document.

EMC tests have not included any AC-adapter with the product.

Applied standards	
EN 60601-1-2:2007 +AC:2010 IEC 60601-1-2:2007 (ed3.0)	Medical electrical equipment – Part 1-2: General requirements for basic safety and essential performance – Collateral standard: Electromagnetic compatibility – Requirements and tests
EN 60601-1-2:2015 IEC 60601-1-2:2014 (ed4.0)	Medical electrical equipment – Part 1-2: General requirements for basic safety and essential performance – Collateral standard: Electromagnetic compatibility – Requirements and tests

Electromagnetic emissions		
The fusionTrack is intended for use in the electromagnetic environment specified below. The integrator, the customer or the user should assure that it is used in such an environment.		
Emissions test	Compliance	Electromagnetic environment - guidance
RF emissions EN 55011:2016 CISPR 11:2015 (ed6.0)	Group 1	The fusionTrack uses RF energy only for its internal function. Therefore, its RF emissions are very low and are not likely to cause any interference in nearby electronic equipment. The fusionTrack is suitable for use in all establishments, including domestic establishments and those directly connected to the public low-voltage power supply network that supplies buildings used for domestic purposes.
RF emissions EN 55011:2016 CISPR 11:2015 (ed6.0)	Class B	
Harmonic emissions EN 61000-3-2:2014 IEC 61000-3-2:2014	Not applicable	
Voltage fluctuations / Flicker emissions EN 61000-3-3:2013 IEC 61000-3-3:2013	Not applicable	



Use of the fusionTrack adjacent to, or stacked with other equipment should be avoided because it could result in improper operation and can lead to wrong measurements to serious injury including death. If such use is necessary, the fusionTrack and the other equipment should be observed to verify that they are operating normally.



Use of accessories, transducers and cables other than those specified or provided by Atracsys could result in increased electromagnetic emissions or decreased electromagnetic immunity of this equipment and result in improper operation and/or loss of performance. Such failure can lead to wrong measurements, serious injury including death.

Electromagnetic immunity			
The fusionTrack is intended for use in the electromagnetic environment specified below. The integrator, the customer or the user should assure that it is used in such an environment.			
Immunity test	Test level	Compliance level	Electromagnetic environment - guidance
Electrostatic discharges EN 61000-4-2:2009 IEC 61000-4-2:2008	±8 kV contact ±15 kV air	±8 kV contact ±15 kV air	Floors should be wood, concrete or ceramic tile. If floors are covered with synthetic material, the relative humidity should be at least 30 %.
Electrical fast transient / burst EN 61000-4-4:2012 IEC 61000-4-4:2012	±2 kV for power supply lines ±1 kV for input/output lines	±2 kV for power supply lines ±1 kV for input/output lines	Mains power quality should be that of a typical commercial or hospital environment.
Surge EN 61000-4-5:2014 IEC 61000-4-5:2014	±1 kV differential mode ±2 kV common mode	Not applicable	
Voltage dips, short interruptions and voltage variations on power supply input lines EN 61000-4-11:2004 IEC 61000-4-11:2004	< 5 % U_t (> 95 % dip in U_t) for 0.5 cycle 40 % U_t (60 % dip in U_t) for 5 cycles 70 % U_t (30 % dip in U_t) for 25 cycles < 5 % U_t (> 95 % dip in U_t) for 5 s	Not applicable	Not applicable
Power frequency (50/60Hz) magnetic field EN 61000-4-8:2010 IEC 61000-4-8:2009	30 A m ⁻¹ 50 Hz and 60 Hz	30 A m ⁻¹ 50 Hz and 60 Hz	Power frequency magnetic fields should be at levels characteristic of a typical location in a typical commercial or hospital environment.
Conducted RF EN 61000-4-6:2014 IEC 61000-4-6:2013	3 V _{RMS} 150 kHz to 80 MHz 6 V _{RMS} ISM bands	3 V _{RMS} 150 kHz to 80 MHz 6 V _{RMS} ISM bands	Portable and mobile RF communications equipment should be used no closer to any part of the fusionTrack, including cables, than the recommended separation distance calculated from the equation applicable to the frequency of the transmitter. Recommended separation distance $d = 1.2 \sqrt{P}$ 150 kHz to 80 MHz $d = 0.35 \sqrt{P}$ 80 MHz to 800 MHz $d = 0.70 \sqrt{P}$ 800 MHz to 2.7 GHz Where (P) is the maximum output power rating of the transmitter in watts (W) according to the transmitter manufacturer and d is the recommended separation distance in metres (m). Field strengths from fixed RF transmitters, as determined by an electromagnetic site survey ^a , should be less than the compliance level in each frequency range. ^b Interference may occur in the vicinity of equipment marked with the following symbol:
Radiated RF EN 61000-4-3:2006 +A1 +A2 IEC 61000-4-3:2006 +A1 +A2	3 V m ⁻¹ 80 MHz to 2.7 GHz	10 V m ⁻¹ 80 MHz to 2.7 GHz	



^a Field strengths from fixed transmitters, such as base stations for radio (cellular/cordless) telephones and land mobile radios, amateur radio, AM and FM radio broadcast and TV broadcast cannot be predicted theoretically with accuracy. To assess the electromagnetic environment due to fixed RF transmitters, an electromagnetic site survey should be considered. If the measured field strength in the location in which the fusionTrack is used exceeds the applicable RF compliance level above, the fusionTrack should be observed to verify normal operation. If abnormal performance is observed, additional measures may be necessary, such as reorienting or relocating the fusionTrack.

^b Over the frequency range 150 kHz to 80 MHz, field strengths should be less than 3 V m⁻¹.

Table 14.1: Electromagnetic immunity test.

Notes:

- U_t is the AC mains voltage prior to application of the test level.
- At 80 MHz and 800 MHz, the higher frequency range applies.
- These guidelines may not apply in all situations. Electromagnetic propagation is affected by absorption and reflection from structures, objects and people.



Portable radio frequency communications equipment (including peripherals such as antenna cables and external antennas) should be used no closer than 30 cm (12 inches) to any part of the fusionTrack, including cables specified by Atracsys. Otherwise, degradation of the performance of this equipment could result, which can lead to wrong measurements, serious injury including death.

Recommended separation distances between portable and mobile RF communications equipment and the fusionTrack			
The fusionTrack is intended for use in an environment in which radiated RF disturbances are controlled. The integrator, the customer or the user of the device can help prevent electromagnetic interference by maintaining a minimum distance between portable and mobile RF communications equipment (transmitters) and the fusionTrack as recommended below, according to the maximum output power of the communications equipment.			
Rated maximum output power of transmitter (W)	Separation distance according to frequency of transmitter (m)		
	150 kHz to 80 MHz $d = 1.2 \sqrt{P}$	80 MHz to 800 MHz $d = 0.35 \sqrt{P}$	800 MHz to 2.7 GHz $d = 0.70 \sqrt{P}$
0.01	0.120	0.035	0.070
0.1	0.38	0.11	0.22
1	1.20	0.35	0.70
10	3.8	1.1	2.2
100	12.0	3.5	7.0

For transmitters rated at a maximum output power not listed above, the recommended separation distance d in metres (m) can be determined using the equation applicable to the frequency of the transmitter, where P is the maximum output power rating of the transmitter in watts (W) according to the transmitter manufacturer.

Notes:

- At 80 MHz and 800 MHz, the separation distance for the higher frequency range applies.
- These guidelines may not apply in all situations. Electromagnetic propagation is affected by absorption and reflection from structures, objects and people.

14.1 Essential performances

As the fusionTrack device is only a component for a medical device, the essential performances of the overall medical device have to be defined and validated by the legal manufacture of the medical device.

14.1.1 Quality of the measurement

The precision, accuracy and trueness of the position measurement of the fusionTrack device depend on a multitude of factors such as fiducial type (LED, sphere, sticker), marker geometry, working distance, partial occlusion and many more. In the typical setup that was tested, the quality of the measurement was not affected by any of the electromagnetic immunity tests stated in Table 14.1. To supervise this immunity, the parameters shown in Table 14.2 have been evaluated during the electromagnetic immunity tests.

Evaluation parameter	Value
Blob number	Constant
Number of fiducials used to build a marker	Constant
Marker registration error	Constant ^a
Registration error max - min	Constant ^a
Fiducial interdistance	Constant ^a
Fiducial interdistance max - min	Constant ^a

^a With presence of a jitter, which depends on: *fiducial* type (LED, sphere, sticker), *marker* geometry, working distance.

Table 14.2

14.1.2 Measurement rate

The rate of which the measurements are delivered by the fusionTrack device on the host PC depends on a multitude of factors such as number and size of blobs (white zones) in the images, Ethernet link speed, host PC, OS, blocked optical path and many more. In the typical setup that was tested, the measurement rate was not affected by any of the electromagnetic immunity tests stated in the Table 14.1 except for the following cases:

Immunity test	Note
ESD discharges	Communication status LED might blink shortly on discharges. Acquisition rate might temporarily drop down to 100 Hz ^a .
Electrical fast transient / burst	Acquisition rate might temporarily drop down to 110 Hz ^a .

^a Those values may depend on the test environment.

For the supervision of the measurement rate, the following parameters have been evaluated during the electromagnetic immunity tests.

Evaluation parameter	Value
Acquisition rate	Constant > 118 Hz ^a
Dropped frames rate	Constant ^a
Processed frames rate	Constant ^a

^a Those values may depend on the test environment and/or host PC.

14.2 Cables

Access	Length	Type	Screened
Ethernet RJ-45 port 100/1000 Mbps <i>PoE+</i>	max. 100 m ^a	Cat 5e or better	(un)shielded

^a Total length, i.e. before and after a potential *PoE+* injector. Not intended to connect directly to outdoor cables.

15 Communication parameters

The fusionTrack device uses by default the 172.17.1.7 IPv4 address, and the local UDP port is automatically chosen. These two parameters can be changed. SDK 4.7.1 and newer versions provide a new functionality, allowing to limit the the maximal value of the tested MTU. This may allow a quicker detection of the device on systems with a low MTU value.

15.1 Changing the IP

The IPv4 address of the device can be changed to fit the user's desiderata. To change this address, the `atnetExecutable64` software must be used. In this section it is assumed the `atnetExecutable64` was connected to the fusionTrack device as explained in Section 10.1.

The fusionTrack device only supports private IPv4 addresses. The valid ranges are:

- 10.0.0.0 – 10.255.255.255;
- 172.16.0.0 – 172.31.255.255;
- 192.168.0.0 – 192.168.255.255.



When changing the IP, the user must carefully note the new IP, otherwise it may be difficult to connect to the device again.

1. Use the 'set_ip' to set the new IP, for instance

```
1 set_ip * "172.16.2.4"
```

2. Wait for '--> New IPv4 address set. Please reset device' to be printed;
3. Either reboot manually the device or use the 'reset * 0' command.
4. Quit `atnetExecutable64` with the `exit` command.

Once a new IP has been set, all the programs connecting to the device should be modified, so that they are aware of the new IP. This is done by using the `ftkInitExt` function to initialise the library. Its first argument is the path to a *JSON* file, which describes the interface that must be used (see Listing 15.1 for version 2 and Listing 15.2 for version 1).

Please note the UDP port used by the fusionTrack (the 3509 value) cannot currently be changed.


```
1 {  
2   "network":  
3   {  
4     "version": 2,  
5     "interfaces": [  
6       {  
7         "adapter": "172.16.2.42",  
8         "devices":  
9         [  
10          {  
11            "address": "172.16.2.4",  
12            "port": 3509  
13          }  
14        ]  
15      }  
16    ]  
17  }  
18 }
```

Listing 15.1: *JSON* file setting the IP to 172.16.2.4 using a version 2 "network" section.

```
1 {  
2   "network":  
3   {  
4     "version": 1,  
5     "interfaces": [  
6       {  
7         "address": "172.16.2.4",  
8         "port": 3509  
9       }  
10    ]  
11  }  
12 }
```

Listing 15.2: *JSON* file setting the IP to 172.16.2.4 using a version 1 "network" section.

15.2 Changing the local UDP port

The local UDP port (i.e. the random port on the machine used to communicate with the UDP server on the fusionTrack device) can be forced to a value. To be valid, the value should neither be currently in use nor reserved by the operating system. The configuration *JSON* file must be changed, and a new attribute `"localPort": port_number` added to the interface to be modified (see example on Listing 15.4). If the port cannot be used for any reason, the device will not be detected, resulting in a 'No device detected' error.

```
1 {
2   "network":
3   {
4     "version": 1,
5     "interfaces": [
6       {
7         "address": "172.17.1.7",
8         "port": 3509, // fTk port, cannot currently be changed
9         "localPort": 13485 // Local port information
10      }
11    ]
12  }
13 }
```

Listing 15.3: *JSON* file trying to force the local UDP port using a version 1 "network" section.

15.3 Changing the maximal MTU value

On systems with a low maximal value of the MTU, the detection sequence of a fusionTrack device can be long as packets with size of about 8, 7, 6, 4 and 2 KiB are tested. For instance, on a system with only 1500 B MTU, the tests lasts several seconds. SDK 4.7.1 introduced an upper limit for the tests MTU candidates. This can be set either with a configuration file with a `"network"` version 2 section.

```
1 {
2   "network":
3   {
4     "version": 2,
5     "interfaces":
6     [
7       {
8         "adapter": "172.17.1.42",
9         "mtu": 4500, // Maximal value for the tested MTU
10        "devices":
11        [
12          {
13            "address": "172.17.1.9",
14            "port": 3509
15          }
16        ]
17      }
18    ]
19  }
20 }
```

Listing 15.4: *JSON* file setting a maximal value of 4500 B for the MTU.

15.4 Connection to multiple fusionTrack devices

The new `ftkInitExt` function also allows to use the same library handle to connect to more than one fusionTrack device. The recommended way to do it is to use one dedicated Ethernet adapter per fusionTrack device, in which case, all devices must have a different IP address, if possible each on a different subnet. Then the host PC must have one direct connection of each fusionTrack. The configuration file to set one device with IP 172.23.42.3 and another one with IP 172.18.42.4 is presented on Listing 15.5. It is also possible to use a GigaBit switch, allowing several fusionTrack device having *different* IP addresses in the *same* subnet.

```
1 {
2   "network":
3   {
4     "version": 2,
5     "interfaces":
6     [
7       {
8         "adapter": "172.23.0.0",
9         "mtu": 7500,
10        "devices":
11        [
12          {
13            "address": "172.23.42.3",
14            "port": 3509
15          }
16        ]
17      },
18      {
19        "adapter": "172.18.0.0",
20        "mtu": 7500,
21        "devices":
22        [
23          {
24            "address": "172.18.42.4",
25            "port": 3509
26          }
27        ]
28      }
29    ]
30  }
31 }
```

Listing 15.5: Example of configuration allowing the connection to 2 fusionTracks on two dedicated subnets.

16 The fusionTrack device authentication

This chapter discusses the device authentication: a mechanism is provided to ensure the following situations do not occur:

- A malicious attacker replaces a fusionTrack by one of its own system and send incorrect data on the network cable linking the tracker to the end-user application;
- A malicious attacker replaces the firmware of a fusionTrack device by a malicious one.

The following scenarii are *not* covered:

- A malicious attacker installs a 'man in the middle' device between the fusionTrack device and the end-user application. The device modifies data on the fly;
- A malicious attacker listens to the network traffic between the fusionTrack device and the end-user application to gather information.

16.1 Implementation

A key is programmed in the fusionTrack device and in the end-user application by the integrator.

The Atracsys SDK let the integrator send 'challenges' to the tracker whenever the need arise. For example, this can be done at startup and during surgical workflow transitions.

The challenge consist into sending data to the tracker and receiving back the encrypted version of the data. The encryption is done using the pre-programmed keys. The encrypted challenge received from the fusionTrack device is compared to the challenge encrypted by the SDK.

The chosen algorithm is [XTEA](#)[3] (eXtended TEA), with a 128 bits key and 64 bits challenges. This algorithm was chosen for its simplicity of implementation, execution speed and relative robustness. In order to mitigate brute force attacks, the rate of challenge testing is limited in the firmware 1 Hz. The integrator is responsible for the key(s) handling, Atracsys only provides the storage in the fusionTrack device and computation in both the SDK and the fusionTrack device firmware.

16.2 Setting the key in the fusionTrack device

A 128 bits key must first be uploaded in the fusionTrack device, this requires to use the `atnetExecutable64` software. First, the communication between the fusionTrack device and `atnetExecutable64` must be established, as explained in Section 10.1. A binary file containing the 128 bits key (i.e. 16 bytes) must be prepared beforehand, and the upload procedure is the following:

1. Use the command '`set_cypherKey`' to register the key in the device, for example:

```
1 set_cypherKey * ~/key_file
```

Listing 16.1: Setting the key into the fusionTrack.

2. Wait for '--> OK' to be displayed; Boot the device in application firmware by switching off and after 5 s on again or by running the 'reset * 1' command;

The key file can be generated from an hexadecimal number on a Linux command line with the following:

```
1 perl -e 'print pack "H*", "12345678901234567890123456789012"' > key_file
```

Listing 16.2: Generating the key file.



Any write action performed by the `atnetExecutable64` software will destroy the stored key.

16.3 Checking the fusionTrack device

The Atracsys SDK allows to check if the connected fusionTrack device is correctly configured. The key and challenge setting and the result getting are handled via options (see Section 8.1). The key must first be set in the SDK using the 'Set cypher key' option: it is a binary option, which content must be 16 bytes long. Then, challenges can be requesting using the 'Request challenge' binary option, which content must be 8 bytes long. It will fail if no key has previously been set or if the last request was performed less than 1 s ago. Finally, the integer 'Challenge result' option allows to check whether the two encrypted challenges (i.e. the one computed by the fusionTrack device and the SDK) are the same, in which case a 1 is returned, or not, in which case a 0 is returned. A full example is shown on Listing 16.3.

```
1 /* The uploaded file contains "This is my key!!", which translates into
2  * 0x54 0x68 0x69 0x73 0x20 0x69 0x73 0x20 0x6d 0x79 0x20 0x6b 0x65 0x79 0x21 0x21 in ASCII.
3  * lib is a valid library handle, sn the serial number of an enumerated device.
4  * setKeyId, requestChallengeId and checkChallengeId are the option IDs of the
5  * 'Set cypher key', 'Request challenge' and 'Challenge result' options
6  * respectively.
7  */
8
9 ftkBuffer buffer;
10 buffer.reset();
11 buffer.data[ 0x0u ] = 0x54;
12 buffer.data[ 0x1u ] = 0x68;
13 buffer.data[ 0x2u ] = 0x69;
14 buffer.data[ 0x3u ] = 0x73;
15 buffer.data[ 0x4u ] = 0x20;
16 buffer.data[ 0x5u ] = 0x69;
17 buffer.data[ 0x6u ] = 0x73;
18 buffer.data[ 0x7u ] = 0x20;
19 buffer.data[ 0x8u ] = 0x6d;
20 buffer.data[ 0x9u ] = 0x79;
21 buffer.data[ 0xau ] = 0x20;
22 buffer.data[ 0xbu ] = 0x6b;
23 buffer.data[ 0xcu ] = 0x65;
24 buffer.data[ 0xdu ] = 0x79;
25 buffer.data[ 0xeu ] = 0x21;
```

```
26 buffer.data[ 0xfu ] = 0x21;
27 buffer.size = 16u;
28
29 if ( ftkSetData( lib, sn, setKeyId, &buffer ) != ftkError::FTK_OK )
30 {
31     // handle error
32 }
33
34 std::default_random_engine generator;
35 std::uniform_int_distribution<uint64_t> distribution();
36
37
38 buffer.reset();
39 uint64_t value( distribution( generator ) );
40 memcpy( buffer.data, &value, sizeof( value ) );
41 buffer.size = 8u;
42
43 if ( ftkSetData( lib, sn, requestChallengeId, &buffer ) != ftkError::FTK_OK )
44 {
45     // handle error
46 }
47
48 int32_t result( 0 );
49
50 if ( ftkGetInt32( lib, sn, checkChallengeId, &result,
51                 ftkOptionGetter::FTK_VALUE ) != ftkError::FTK_OK )
52 {
53     // handle error
54 }
55
56 if ( value == 1 )
57 {
58     cout << "Device successfully authenticated" << endl;
59 }
60 else
61 {
62     cerr << "Could not authenticate device" << endl;
63 }
```

Listing 16.3: Full example of key setting, challenge requesting and checking.

The GUI demo application also allows to check if the connected fusionTrack device is correctly configured (see Figure 16.1).

Detector	
2D/3D Matching	
Display	
Authentication	
Authentication of the device	
Key:	<input type="text" value="12345678901234567890123456789012"/> <input type="button" value="Set key"/>
Challenge:	<input type="text" value="1111111111111111"/> <input type="button" value="Send challenge"/>
	<input type="button" value="Get result"/>
<div>Successfully set key Successfully set challenge Device successfully authenticated</div>	
<input type="button" value="Clear logs"/>	
Marker recalibration	
Data export	

Figure 16.1: The Authentication tab in the GUI demo allows to check a key by requesting a challenge to the connected fusionTrack

17 Maintenance

In this chapter are found the instructions for maintaining basic safety and performances for the expected service life, in particular it is explained how to clean the fusionTrack device and its accessories.

17.1 Cleaning the fusionTrack device



The fusionTrack was not designed to be autoclavable. The end user manual instruction shall provide a validated cleaning and disinfection method to the end-user.

The fusionTrack can be disinfected with regular hospital disinfectant detergent applied using a clean cloth. The optical parts must be cleaned only using camera lenses cleaning solutions. Eyeglass lens tissues must not be used as it may scratch the lens. Other surfaces must be dried using a clean and dry cloth.



The device must be turned off and disconnected prior to cleaning. The disinfecting detergent must not be spilled directly onto the device or any of its components.



It must be ensured that no fluids enter the fusionTrack. This may damage the fusionTrack by causing short-circuits and corrode the material. Do not immerse the fusionTrack in liquid.



Only non-corrosive and non-acidic detergents, whose interactions with material are known, may be used.

17.2 Sterilisation of the markers



The integrator must ensure markers are not deformed after sterilisation.



The battery of an active marker must be properly removed and disposed of before sterilisation.

17.2.1 Cleaning the fusionTrack device filters

The filters placed in the back of the fusionTrack device have to be cleaned every 1250 working hours. They have to be removed by un-screwing the fixation screws and have to be gently cleaned using a small vacuum cleaner. If the filters are too dirty or damaged they have to be replaced by original Atracsys spare parts. The usage of pressurized air is not allowed as it might damage the filters.

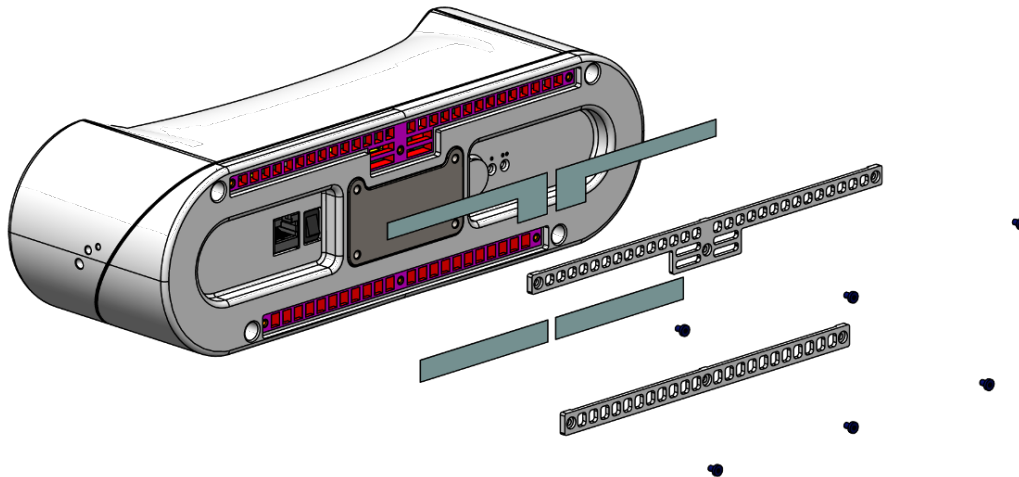


Figure 17.1: Scattered view of the fusionTrack 250 for filter replacement.



The filters have to be inspected regularly against dust and also controlled for any damages like holes.



Do not clean the filters while still attached to the device.



Do not leave or run the system without the filters as dust might enter.



Do not use a vacuum cleaner or pressurized air on the system.



The integrator must indicate in their user manual the frequency of filter replacement.

17.3 Effects of multiple cleanings

The fusionTrack optics may be scratched or the window transparency may be altered, leading to poor quality picture and therefore to poor tracking accuracy. The loss of tracking accuracy can be detected using the procedure described in Chapter 13. In the case of a device not passing the test, it should not be used anymore and sent back to Atracsys for repair.

17.4 Disposal of equipment

To ensure environmentally responsible disposal decommissioned equipment, please contact Atracsys, see Section 1.6.

18 Troubleshooting

This chapter addresses the main problems a user can experience when using the fusionTrack system.

18.1 Common problems

Problem	Suggested solution
All LEDs are off when switching on.	Check the power supply or use other <i>PoE+</i> and Ethernet cable combinations (see Section 3.2.2).
The device cannot be detected by the software.	<ul style="list-style-type: none">- Check the Ethernet cables;- Check the Ethernet card settings;- Check the fusionTrack device IP address is the expected one (see Chapter 15)
The fusionTrack device is beeping and the status LED is blinking in red.	Launch <code>atnetExecutable64</code> and extract the error statuses (see Section 10.2). Contact Atracsys (see Section 1.6).

18.1.1 Wrong or damaged cables

Poorer performances (degraded effective acquisition rate) are common if using 100BASE-T cables instead of 1000BASE-T ones. The user should check that the used cables are intact: a 1000BASE-T cable with a broken wire pair might be detected as a 100BASE-T cable, leading to the same rate restrictions. Both problems can be detected by looking at the RX / TX LEDs at the back of the device (see Section 3.2.3).

18.1.2 PoE+ problems

If the fusionTrack device seems to have problem receiving power from the power injector, refer to Section 3.2.2 for troubleshooting.

18.1.3 FTK_WAR_SHOCK_DETECTED warning

If the `ftkGetLastFrame` function returns the `ftkError::FTK_WAR_SHOCK_DETECTED` warning, it means that the fusionTrack has received a shock above the threshold. The fusionTrack is potentially decalibrated.

Depending on the fusionTrack device version, the shock sensor might be not available. Please contact Atracsys for more information.

In order to clear this warning, an accuracy verification must be performed (See Chapter 13). It can be done either on site using the Atracsys accuracy verification tool or the device can be sent back to Atracsys (See Section 1.6). If done on site, the AVT user manual describes the procedure.

18.1.4 FTK_WAR_SHOCK_SENSOR_OFFLINE warning

If the `ftkGetLastFrame` function returns the `ftkError::FTK_WAR_SHOCK_SENSOR_OFFLINE` warning, it means that the fusionTrack shock sensor was offline. A shock might have happened during the downtime of the shock sensor.

Depending on the fusionTrack device version, the shock sensor might be not available. Please contact Atracsys for more information.

In order to determine whether a harming shock occurred during the shock sensor downtime, an accuracy verification must be performed (See Chapter 13).

18.1.5 FTK_WAR_SHOCK_SENSOR_AUTO_ENABLE warning

If the `ftkGetLastFrame` function returns the `ftkError::FTK_WAR_SHOCK_AUTO_ENABLE` warning, it means that the fusionTrack shock sensor was offline for a moment and automatically enabled back at boot. A shock might have happened during the downtime of the shock sensor.

Depending on the fusionTrack device version, the shock sensor might be not available. Please contact Atracsys for more information.

In order to clear this warning, the `enable_uC` command of the `atnetExecutable64` software is used. In order to determine whether a harming shock occurred during the shock sensor downtime, an accuracy verification must be performed (See Chapter 13).

19 Licences

This chapter enumerates the various software used by Atracsys fusionTrack SDK, the demo software and the AVT software.

19.1 Atracsys fusionTrack SDK headers and library

These files are part of the Atracsys fusionTrack library. You can freely use this SDK for your own applications.

A license to use binary files of this distribution is granted to owner of infiniTrack, spyTrack and/or fusionTrack devices.

It is not allowed to redistribute any source file of this distribution without written permission of Atracsys.

19.2 Hashing code

The fusionTrack SDK uses hashing code by Stephan Brumme.

Unless otherwise noted in a file's first 5 lines, all source code published on <http://create.stephan-brumme.com> and its sub-pages is licensed similar to the zlib license:

This software is provided 'as-is', without any express or implied warranty. In no event will the author be held liable for any damages arising from the use of this software. Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software.
2. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
3. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

19.3 About JsonCpp

The fusionTrack SDK uses JsonCpp library to parse JSON files.

The JsonCpp library's source code, including accompanying documentation, tests and demonstration applications, are licensed under the following conditions...

Baptiste Lepilleur and The JsonCpp Authors explicitly disclaim copyright in all jurisdictions which recognize such a disclaimer. In such jurisdictions, this software is released into the Public Domain.

In jurisdictions which do not recognize Public Domain property (e.g. Germany as of 2010), this software is Copyright © 2007-2010 by Baptiste Lepilleur and The JsonCpp Authors, and is released under the terms of the MIT License (see below).

In jurisdictions which recognize Public Domain property, the user of this software may choose to accept it either as 1) Public Domain, 2) under the conditions of the MIT License (see below), or 3) under the terms of dual Public Domain/MIT License conditions described here, as they choose.

The MIT License is about as close to Public Domain as a license can get, and is described in clear, concise terms at:

http://en.wikipedia.org/wiki/MIT_License

The full text of the MIT License follows:

```
=====
Copyright ©2007-2010 Baptiste Lepilleur and The JsonCpp Authors

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated
documentation files (the "Software"), to deal in the Software without restriction, including without limitation
the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software,
and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions
of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PAR-
TICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF
CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFT-
WARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

=====
(END LICENSE TEXT)
```

The MIT license is compatible with both the GPL and commercial software, affording one all of the rights of Public Domain with the minor nuisance of being required to keep the above copyright notice and license text in the source code. Note also that by accepting the Public Domain "license" you can re-license your copy using whatever license you like.

19.4 About Qt

The demo software uses Qt 5.15.

Qt is a C++ toolkit for cross-platform application development.

Qt provides single-source portability across MS Windows, Mac OS X, Linux, and all major commercial Unix variants. Qt is also available for embedded devices as Qt for Embedded Linux and Qt for Windows CE.

Qt is available under different licensing options designed to accommodate the needs of our various users:

- Qt licensed under commercial licenses is appropriate for development of proprietary/commercial software where you do not want to share any source code with third parties or otherwise cannot comply with the terms of the GNU LGPL version 3.

- Qt licensed under the [GNU Lesser General Public License \(LGPL\) version 3](#) is appropriate for the development of Qt applications provided you can comply with the terms and conditions of the GNU LGPL version 3 (or GNU GPL version 3).
- Qt components licensed under the [Qt Marketplace License Agreement](#) are appropriate for the development of Qt applications commonly with Qt software components licensed under the commercial or GNU LGPL version 3 (or GNU GPL version 3) terms and conditions.

Qt contains also [third-party](#) code that is licensed under specific open-source licenses from the original authors.

Qt documentation is available under commercial licenses from The Qt Company, and under the terms of the [GNU Free Documentation License \(FDL\)](#) version 1.3, as published by the Free Software Foundation.

Qt examples are available under commercial licenses from The Qt Company, and under a [BSD-3-clause](#) license.

See <http://qt.io/licensing/> for an overview of Qt licensing.

The Qt Toolkit is Copyright © 2018 The Qt Company Ltd. and other contributors.

19.4.1 LGPL version 3

GNU LESSER GENERAL PUBLIC LICENSE Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <<http://fsf.org/>> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

0. Additional Definitions.

As used herein, “this License” refers to version 3 of the GNU Lesser General Public License, and the “GNU GPL” refers to version 3 of the GNU General Public License.

“The Library” refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An “Application” is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A “Combined Work” is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the “Linked Version”.

The “Minimal Corresponding Source” for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The “Corresponding Application Code” for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

- a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or
- b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

- a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

- a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the Combined Work with a copy of the GNU GPL and this license document.
- c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.
- d) Do one of the following:
 - 0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.
 - 1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.

- e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

- a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.
- b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy’s public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.

19.5 About QDarkStyle

The demo software uses [QDarkStyle](#).

19.5.1 The MIT License (MIT) - Code

Copyright © 2013-2019 Colin Duquesnoy

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

19.5.2 Creative Commons Attribution International 4.0 - Images

QDarkStyle © 2013-2019 Colin Duquesnoy

QDarkStyle © 2019-2019 Daniel Cosmo Pizetta

Creative Commons Corporation ("Creative Commons") is not a law firm and does not provide legal services or legal advice. Distribution of Creative Commons public licenses does not create a lawyer-client or other relationship. Creative Commons makes its licenses and related information available on an "as-is" basis. Creative Commons gives no warranties regarding its licenses, any material licensed under their terms and conditions, or any related information. Creative Commons disclaims all liability for damages resulting from their use to the fullest extent possible.

Using Creative Commons Public Licenses

Creative Commons public licenses provide a standard set of terms and conditions that creators and other rights holders may use to share original works of authorship and other material subject to copyright and certain other rights specified in the public license below. The following considerations are for informational purposes only, are not exhaustive, and do not form part of our licenses.

Considerations for licensors: Our public licenses are intended for use by those authorized to give the public permission to use material in ways otherwise restricted by copyright and certain other rights. Our licenses are irrevocable. Licensors should read and understand the terms and conditions of the license they choose before applying it. Licensors should also secure all rights necessary before applying our licenses so that the public can reuse the material as expected. Licensors should clearly mark any material not subject to the license. This includes other CC-licensed material, or material used under an exception or limitation to copyright. [More considerations for licensors.](#)

Considerations for the public: By using one of our public licenses, a licensor grants the public permission to use the licensed material under specified terms and conditions. If the licensor's permission is not necessary for any reason—for example, because of any applicable exception or limitation to copyright—then that use is not regulated by the license. Our licenses grant only permissions under copyright and certain other rights that a licensor has authority to grant. Use of the licensed material may still be restricted for other reasons, including because others have copyright or other rights in the material. A licensor may make special requests, such as asking that all changes be marked or described. Although not required by our licenses, you are encouraged to respect those requests where reasonable. [More considerations for the public.](#)

19.5.3 Creative Commons Attribution 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights

in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 - Definitions

- a. **Adapted Material** means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.
- b. **Adapter's License** means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.
- c. **Copyright and Similar Rights** means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.
- d. **Effective Technological Measures** means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.
- e. **Exceptions and Limitations** means fair use, fair dealing, and/or any other exception or limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.
- f. **Licensed Material** means the artistic or literary work, database, or other material to which the Licensor applied this Public License.
- g. **Licensed Rights** means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.
- h. **Licensor** means the individual(s) or entity(ies) granting rights under this Public License.
- i. **Share** means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.
- j. **Sui Generis Database Rights** means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.
- k. **You** means the individual or entity exercising the Licensed Rights under this Public License. **Your** has a corresponding meaning.

Section 2 – Scope

a: **License grant.**

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:
 - A. reproduce and Share the Licensed Material, in whole or in part; and
 - B. produce, reproduce, and Share Adapted Material.
2. **Exceptions and Limitations.** For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.
3. **Term.** The term of this Public License is specified in Section 6(a).
4. **Media and formats; technical modifications allowed.** The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures. For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.
5. **Downstream recipients.**
 - A. **Offer from the Licensor – Licensed Material.** Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.
 - B. **No downstream restrictions.** You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.
6. **No endorsement.** Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b: **Other rights.**

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.
2. Patent and trademark rights are not licensed under this Public License.
3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties.

Section 3 – License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. Attribution.

1. If You Share the Licensed Material (including in modified form), You must:
 - A. retain the following if it is supplied by the Licensor with the Licensed Material:
 - i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);
 - ii. a copyright notice;
 - iii. a notice that refers to this Public License;
 - iv. a notice that refers to the disclaimer of warranties;
 - v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;
 - B. indicate if You modified the Licensed Material and retain an indication of any previous modifications; and
 - C. indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.
2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.
3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.
4. If You Share Adapted Material You produce, the Adapter's License You apply must not prevent recipients of the Adapted Material from complying with this Public License.

Section 4 – Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

- a. for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database;
- b. if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material; and
- c. You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

Section 5 – Disclaimer of Warranties and Limitation of Liability.

- a. **Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors, whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.**
- b. **To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.**
- c. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Section 6 – Term and Termination.

- a. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.
- b. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:
 1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or
 2. upon express reinstatement by the Licensor.

For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.

- c. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.
- d. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7 – Other Terms and Conditions.

- a. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.
- b. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

Section 8 – Interpretation.

- a. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully be made without permission under this Public License.
- b. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.
- c. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.
- d. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

Creative Commons is not a party to its public licenses. Notwithstanding, Creative Commons may elect to apply one of its public licenses to material it publishes and in those instances will be considered the “Licensor.” Except for the limited purpose of indicating that material is shared under a Creative Commons public license or as otherwise permitted by the Creative Commons policies published at creativecommons.org/policies, Creative Commons does not authorize the use of the trademark “Creative Commons” or any other trademark or logo of Creative Commons without its prior written consent including, without limitation, in connection with any unauthorized modifications to any of its public licenses or any other arrangements, understandings, or agreements concerning use of licensed material. For the avoidance of doubt, this paragraph does not form part of the public licenses.

Creative Commons may be contacted at creativecommons.org.

19.6 Base 64

The demo software uses base 64 encoding by René Nyffenegger.

Copyright © 2004-2008 René Nyffenegger.

This source code is provided 'as-is', without any express or implied warranty. In no event will the author be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not claim that you wrote the original source code. If you use this source code in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original source code.
3. This notice may not be removed or altered from any source distribution.

René Nyffenegger rene.nyffenegger@adp-gmbh.ch

19.7 About FLTK

The AVT software is based in part on the work of the FLTK project (<http://www.fltk.org>) version 1.3.3.

December 11, 2001

The FLTK library and included programs are provided under the terms of the GNU Library General Public License (LGPL) Version 2, June 1991 with the following exceptions:

1. Modifications to the FLTK configure script, config header file, and makefiles by themselves to support a specific platform do not constitute a modified or derivative work.
2. The authors do request that such modifications be contributed to the FLTK project - send all contributions through the "Software Trouble Report" on the following page: <http://www.fltk.org/str.php>
3. Widgets that are subclassed from FLTK widgets do not constitute a derivative work.
4. Static linking of applications and widgets to the FLTK library does not constitute a derivative work and does not require the author to provide source code for the application or widget, use the shared FLTK libraries, or link their applications or widgets against a user-supplied version of FLTK.
5. If you link the application or widget to a modified version of FLTK, then the changes to FLTK must be provided under the terms of the LGPL in sections 1, 2, and 4.
6. You do not have to provide a copy of the FLTK license with programs that are linked to the FLTK library, nor do you have to identify the FLTK license in your program or documentation as required by section 6 of the LGPL.

However, programs must still identify their use of FLTK. The following example statement can be included in user documentation to satisfy this requirement:

[*program/widget*] is based in part on the work of the FLTK project (<http://www.fltk.org>).

A Mathematical considerations

A.1 Homogeneous coordinates

In mathematics, homogeneous coordinates[4] for a 3-dimensional space uses 4-dimensional vectors to represent a position:

$$\vec{x} = (x_0 \ x_1 \ x_2)^T \iff X = (x_0 \ x_1 \ x_2 \ 1)^T. \quad (\text{A.1})$$

This notation allows a much easier handling of transformations composed of a rotation and a translation, as the transformation matrix is written as:

$$T = \begin{pmatrix} R & \vec{t} \\ 0 & 1 \end{pmatrix} \quad (\text{A.2})$$

with

$$R = \begin{pmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{pmatrix}$$

and

$$\vec{t} = \begin{pmatrix} t_0 \\ t_1 \\ t_2 \end{pmatrix}$$

From those definitions result the combination of two transformations T_1 and T_2 result in a transformation $T_{tot} = T_2 \cdot T_1$.

A.2 Marker pose

Atracsys gives access to the marker orientation, either via the `ftkMarker::rotation` or the `atracsys::MarkerData::_Rotation` fields. The `ftkMarker::rotation` matrix is stored such that it is accessed like this: `ftkMarker::rotation[rowId][columnId]`, and the information is simply copied into the `atracsys::MarkerData::_Rotation` field (i.e. indices are used in the same way). Consider the following example, taken from the simulator data, in which a marker with geometry ID 2 is reconstructed with a registration error of 344 μm . The given rotation and translation are:

$$R = \begin{pmatrix} 0.5880 & -0.8023 & 0.1029 \\ -0.7987 & -0.5960 & -0.0828 \\ 0.1278 & -0.0336 & -0.9912 \end{pmatrix} \quad \vec{t} = \begin{pmatrix} 199.319 \\ -103.765 \\ 876.320 \end{pmatrix}$$

The rotation components are retrieved from the SDK as presented on Listing A.1. Geometry ID 2 defines the following four points:

$$\begin{aligned} \vec{p}_0 &= (0.000 \ 11.000 \ 3.000)^T & \vec{p}_1 &= (24.090 \ -15.750 \ 3.000)^T \\ \vec{p}_2 &= (0.000 \ -47.690 \ 3.000)^T & \vec{p}_3 &= (23.630 \ -10.580 \ 3.000)^T \end{aligned}$$

```

1 // C API example: framePtr is a ftkFrameQuery pointer
2 r12 = framePtr->markers[ 0u ].rotation[ 1u ][ 2u ]; // this is -0.0828
3 // C++ API example: frame is a atracsys::FrameData instance
4 r21 = frame.markers().front().rotation().at( 2u ).at( 1u ); // this is -0.0336

```

Listing A.1: Access to the rotation components, from both C and C++ API, same R and \vec{t} as above.

The corresponding reconstructed 3D points are:

$$\begin{aligned}\vec{x}_0 &= (191.153 \quad -110.391 \quad 873.010)^T & \vec{x}_1 &= (198.294 \quad -75.553 \quad 870.821)^T \\ \vec{x}_2 &= (237.517 \quad -75.823 \quad 874.934)^T & \vec{x}_3 &= (221.837 \quad -116.360 \quad 876.675)^T\end{aligned}$$

Now, using homogeneous coordinates, it can be checked easily that:

$$\begin{pmatrix} R & \vec{t} \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \vec{p}_i \\ 1 \end{pmatrix} \simeq \begin{pmatrix} \vec{x}_i \\ 1 \end{pmatrix} \quad \forall i \in [0; 3]$$

$$\begin{aligned}\begin{pmatrix} R & \vec{t} \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \vec{p}_0 \\ 1 \end{pmatrix} &= \begin{pmatrix} 190.802 \\ -110.569 \\ 872.977 \\ 1 \end{pmatrix} \simeq \begin{pmatrix} \vec{x}_0 \\ 1 \end{pmatrix} & \begin{pmatrix} R & \vec{t} \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \vec{p}_1 \\ 1 \end{pmatrix} &= \begin{pmatrix} 198.099 \\ -75.386 \\ 870.775 \\ 1 \end{pmatrix} \simeq \begin{pmatrix} \vec{x}_1 \\ 1 \end{pmatrix} \\ \begin{pmatrix} R & \vec{t} \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \vec{p}_2 \\ 1 \end{pmatrix} &= \begin{pmatrix} 237.889 \\ -75.902 \\ 874.949 \\ 1 \end{pmatrix} \simeq \begin{pmatrix} \vec{x}_2 \\ 1 \end{pmatrix} & \begin{pmatrix} R & \vec{t} \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \vec{p}_3 \\ 1 \end{pmatrix} &= \begin{pmatrix} 222.010 \\ -116.581 \\ 876.743 \\ 1 \end{pmatrix} \simeq \begin{pmatrix} \vec{x}_3 \\ 1 \end{pmatrix}\end{aligned}$$

A.3 Inverting a marker pose

If the position and orientation of a marker is needed in the coordinate system of another marker, the transformation should be inverted. When homogeneous coordinates are used, this inversion is trivial:

$$\begin{aligned}X_{\text{tracker}} &= T \cdot X_{\text{marker}} \\ X_{\text{marker}} &= T^{-1} \cdot X_{\text{tracker}}\end{aligned}$$

If usual 3-dimensional coordinates are used, the inversion reads:

$$\begin{aligned}\vec{x}_{\text{tracker}} &= R \cdot \vec{x}_{\text{marker}} + \vec{t} \\ \vec{x}_{\text{marker}} &= R^{-1} \cdot \vec{x}_{\text{tracker}} - R^{-1} \cdot \vec{t}\end{aligned}$$

As R is a rotation matrix, R^{-1} can be trivially obtained using $R^{-1} = R^T$. A common mistake is to use $-\vec{t}$ instead of $-R^{-1} \cdot \vec{t}$.

B Warning summary

The warnings resulting from the risk analysis are compiled in the checklist below. This table can be used by the integrator as a checklist during their integration of the Atracsys *SDK*.

Warning	
In the whole documentation, warnings are indicated with this symbol and graphics (triangle with exclamation mark). Integrator should follow the accompanying paragraph and/or report necessary warnings in final product's instructions to avoid patient or operator injury.	<input type="checkbox"/>
Integrator should read carefully this user manual before operating the device. Follow all the installation procedures step by step.	<input type="checkbox"/>
The integrator has to check that the device / position of the device is well suited for the application / surgery and that required regulations are respected.	<input type="checkbox"/>
Optical hazards according IEC-62471-1: Exempt group The fusionTrack device uses powerful infrared LEDs for illumination and communication. This infrared light at approximately 850 nm is not visible by human eyes. According to the standard IEC-62471-1 (hazard values are reported at a fixed distance $d = 200$ mm) the fusionTrack is classified in the 'Exempt group'. To further reduce exposure, never watch closely and longly into the infrared LEDs around the cameras when the device is switched on. To further reduce exposure of the skin, never place any body part closer than 200 mm to the infrared LEDs.	<input type="checkbox"/>

The emission of near infrared light for measurement and communication (for example for active markers) is the intended behavior of the optical tracking system. Interference with other equipment using infrared light might be possible. As an example, interference with certain types and brands of pulse oximeters have been reported. In order to avoid the pulse oximeters to be disturbed by the optical tracking system, the pulse oximeters should be shielded either by draping or by using the designated ambient light shields.	<input type="checkbox"/>
Do not immerse the fusionTrack device in water or other fluids.	<input type="checkbox"/>
Do not spill water or other fluids on the fusionTrack device.	<input type="checkbox"/>
The fusionTrack device is composed of metallic parts and electronic components and, even if complying with medical norms, it may interfere or be affected when used in special environments like Magnetic Resonance Imaging or devices generating X-Rays. Integrator should contact Atracsys if they intend to operate the fusionTrack in such environments.	<input type="checkbox"/>
Any noisy environment such as high electromagnetic fields(e.g. MRI) might trigger a measurement error. The operability verification procedure (see Chapter 11) must be proceed to guarantee that the fusionTrack is operational in the configured environment.	<input type="checkbox"/>
The fusionTrack device is a precision instrument and has to be handled with greatest care. If it receives a shock, falls on any surface or is transported without adequate packaging, it can easily be damaged or decalibrated.	<input type="checkbox"/>
Since the fusionTrack will be integrated in a medical system, integrator has to conduct the tests related to the electrical security and electromagnetic compatibility for the whole system.	<input type="checkbox"/>
The used power supply must be a limited power source. Any used power supply must not be able to deliver more than 100W.	<input type="checkbox"/>
The integrator has to implement a mean to isolate the fusionTrack electrically from the supply mains.	<input type="checkbox"/>
If not used, the fusionTrack device should be recharged for 24 h every 6 months.	<input type="checkbox"/>

The fusionTrack device must <i>not</i> be used covered: for instance it should not be operated under a sterile cover. This would change the operating temperature of the device and may change the optical path, both resulting in a change in the calibration parameters, causing a degradation of the device accuracy.	<input type="checkbox"/>
Since the fusionTrack can neither be covered with a sterile drape nor sterilised, the fusionTrack should not be located in the sterile area.	<input type="checkbox"/>
The fusionTrack must not be used if a hot air outlet is present between the fusionTrack and the <i>markers</i> . The refraction index changes of the hot and cold air interfaces could cause wrong measurements.	<input type="checkbox"/>
The fusionTrack has to be placed horizontal. An orientation other than horizontal (e.g. vertical) can deteriorate precision due to thermal gradients inside the device.	<input type="checkbox"/>
The integrator has to perform tests related to instability hazards §9.4 of IEC 60601-1.	<input type="checkbox"/>
Connection of the fusionTrack to an IT-NETWORK that includes other equipment could result in previously unidentified RISKS to PATIENTS, OPERATORS or third parties. A direct connection between the fusionTrack device and the host PC must be used. Otherwise the integrator shall apply chapter 14.13 of IEC 60601-1.	<input type="checkbox"/>
The integrator has to discuss and verify the fusionTrack integration in a Medical Electrical system (ME system) with a certification body. This section of the manual is provided as information only.	<input type="checkbox"/>
The fusionTrack is designed to provide accurate <i>relative</i> measurements, and should <i>not</i> be used to perform <i>absolute</i> measurement. See Chapter 3 for more information.	<input type="checkbox"/>
Most of the functions of the <i>API</i> return a status code, which indicates different levels of warnings and errors. The returned code <i>must</i> be checked by the Integrator's software implementation. The documentation is available in [1].	<input type="checkbox"/>
The various information retrieved from the fusionTrack for each frame comes with a status flag, which must <i>always</i> be checked by the integrator's software implementation to ensure the integrity of the data.	<input type="checkbox"/>

The value of the 'Calibration type' option must be checked by the integrator to correspond to the value of <code>CalibrationType::TemperatureCompensation</code> if the temperature compensation is to be used.	<input type="checkbox"/>
The <i>markers</i> used for surgery must be bio-compatible.	<input type="checkbox"/>
A <i>marker</i> shall be treated as an independent medical device. Use <i>markers</i> as specified by Atracsys. Other risks related to <i>markers</i> with reflecting <i>fiducials</i> may occur depending on the technology used by the manufacturer, these risks cannot be treated by Atracsys.	<input type="checkbox"/>
Any liquid on <i>markers</i> such as body fluids and water could lead to wrong measurements. The integrator has to specify in the end user manual instructions about not handling <i>markers</i> with dirty gloves, as well as instructions for cleaning / replacing <i>markers</i> and / or <i>fiducials</i> after getting in contact with liquids.	<input type="checkbox"/>
A <i>marker</i> in the camera view must have a specific, appropriately sized and distinguishable geometry. The integrator has to ensure that each used <i>marker</i> should be easily distinguishable by end users. A label on each <i>marker</i> has to be inserted. The integrator has to specify in the end user manual that operators have to choose <i>markers</i> with different geometries.	<input type="checkbox"/>
A <i>marker</i> is assumed to be calibrated and sterile. <i>Fiducials</i> are assumed to be correctly fixed on the <i>marker</i> . Instructions about fixing <i>fiducials</i> on <i>markers</i> must be clearly specified by the integrator in the end user manual.	<input type="checkbox"/>
Sterile <i>markers</i> , <i>fiducials</i> or battery packaging must be opened inside the Operating Room (OR). The integrator has to specify in the user manual to open sterile <i>markers</i> , <i>fiducials</i> or batteries packages only in the OR as well as the manner how to open it to guarantee sterility.	<input type="checkbox"/>
Sterile <i>markers</i> , <i>fiducials</i> or batteries in a damaged package cannot be used because it might have been deteriorated during transport which could lead to patient infection. Instructions about not using sterile <i>markers</i> , <i>fiducials</i> or batteries from a damaged package must be clearly specified by the integrator in the end user manual.	<input type="checkbox"/>

Sterile <i>markers</i> , <i>fiducials</i> or batteries in a damaged package cannot be used because it might have been deteriorated during transport which could lead to patient infection. The integrator has to provide sterile <i>markers</i> , <i>fiducials</i> or batteries with packaging presenting a certain robustness.	<input type="checkbox"/>
<i>Markers</i> or <i>fiducials</i> geometrically deformed due to shocks or manufacturing defects can create wrong measurements leading to patient injury. Instructions about not using geometrically deformed <i>markers</i> or <i>fiducials</i> must be clearly specified in the end user manual.	<input type="checkbox"/>
The integrator should define a zeroing method (see Chapter 12) or any equivalent method to verify marker integrity for all cases including corner cases of its application.	<input type="checkbox"/>
<i>Markers</i> or <i>fiducials</i> too close from lenses might lead to wrong measurement (see Table 2.1). The integrator has to mention in the end user manual that <i>markers</i> or <i>fiducials</i> should not be positioned too close from the lenses.	<input type="checkbox"/>
The integrator has to ensure generating an error message when the <i>markers</i> are placed on vibrating instruments and the frequency of vibrations could affect the measurements.	<input type="checkbox"/>
At all time, the integrator software should monitor the registration error of the reconstructed <i>markers</i> , to prevent badly reconstructed data to be used.	<input type="checkbox"/>
Instructions about using a calibrated <i>marker</i> and not geometrically deformed have to be clearly specified by the integrator to the user.	<input type="checkbox"/>
For single-use <i>markers</i> , the end user manual must clearly state that the <i>markers</i> should not be reprocessed. The <i>markers</i> packaging must be marked with a label indicating they cannot be reprocessed.	<input type="checkbox"/>
For single-use <i>fiducials</i> , instructions about replacing new <i>fiducials</i> after every (single) use must be clearly specified by the integrator in the end user manual. Packages of passive <i>fiducials</i> have to be marked with a label indicating that they cannot be reprocessed.	<input type="checkbox"/>
A passive fiducial could be damaged due to a shock / scratch of the reflecting material. Instructions about not using deteriorated <i>fiducials</i> and how to replace a deteriorated passive <i>fiducial</i> must be clearly specified by the integrator in the end user manual.	<input type="checkbox"/>

Integrator has to specify in the end user manual instructions how to screw / plug <i>fiducials</i> on <i>markers</i> .	<input type="checkbox"/>
Active <i>marker</i> electronics has to be completely sealed in order to avoid liquid penetration due to cleaning. The integrator has to indicate in the end user manual instructions to clean active <i>markers</i> . The integrator has to specify in the end user manual that the operability verification procedure (see Chapter 12) must be done after each cleaning.	<input type="checkbox"/>
A geometrically deformed active <i>marker</i> due to a manufacturing defect could lead to wrong measurements causing patient injury. The integrator has to perform individual testing during <i>marker</i> calibration.	<input type="checkbox"/>
Active <i>markers</i> with batteries shall be handled with care. Operator who gets in contact with battery leakage incurs a risk of chemical burn. The integrator has to indicate in the end user manual that the battery of an active <i>marker</i> has to be removed after usage, especially in the case the active <i>marker</i> undergoes a sterilisation procedure.	<input type="checkbox"/>
The batteries used in active <i>markers</i> must comply with the regulation standards.	<input type="checkbox"/>
Battery for active <i>markers</i> must be sterile. The integrator has to indicate in the end user manual to use only sterilized batteries according to ISO 14937.	<input type="checkbox"/>
Any electric defects on active <i>markers</i> such as explosion, leakage, overheating might lead to operator or patient injury. The integrator has to use an adequate protection (e.g. fuse) that limits the current drawn from the battery. The integrator has to ensure that active <i>markers</i> contain a polarity marking and a polarity inversion circuit on the <i>marker</i> .	<input type="checkbox"/>
The measurement precision of an active <i>marker</i> can be deteriorated because of liquids on its fiducials or IR-receptor. Instructions about identifying and cleaning of active fiducial must be clearly specified by the integrator in the end user manual.	<input type="checkbox"/>
An active fiducial <i>marker</i> must be sterilised before each usage. Cleaning or sterilisation methods instructions must be clearly specified by the integrator in the end user manual.	<input type="checkbox"/>

An active fiducial <i>marker</i> is able to transmit its individual parameters and data to the system. It can also notify the fusionTrack with alert messages. An error message alerting that battery power is low is sent. That message must be handled by the integrator to alert the operator that the <i>marker</i> is not considered as operational due to a low battery and the <i>fiducials</i> must be turned off.	<input type="checkbox"/>
A weak optical signal from the IR-LEDs can lead to a wrong estimation of the position. The integrator should ensure the fiducials are turned off if the battery voltage goes under a given threshold.	<input type="checkbox"/>
<i>Markers</i> have to meet the IEC60601-1 requirement regarding 'Protection against excessive temperatures and other hazards'.	<input type="checkbox"/>
The integrator should use redundant <i>fiducials</i> (i.e. four or more <i>fiducials</i> instead of only three) and require the reconstructed <i>marker</i> to use all of them.	<input type="checkbox"/>
The default parameters have been chosen to ensure optimal tracking performances. Changing them might impact the performances of the fusionTrack system.	<input type="checkbox"/>
The fusionTrack device must be cleaned as described in Chapter 17.	<input type="checkbox"/>
When the application requires that an option is set, the integrator must not assume that the configuration is done when exiting the setting function, but must ensure that the option is applied before using the data.	<input type="checkbox"/>
The GUI demo software is not part of the fusionTrack software system. It is only meant to provide a showcase of what can be achieved using the SDK. The GUI demo software does not belong to the fusionTrack product.	<input type="checkbox"/>
The reference implementation is the distributed C / C++ API. The other language bindings are not subject to thorough testing, as the C / C++ do. Integrators are responsible for their software validation using other language bindings.	<input type="checkbox"/>
When performing step 7 of the fusionTrack shock detection unit firmware update procedure, the fusionTrack <i>must</i> be at room temperature and switched off for at least 4 h.	<input type="checkbox"/>

The zeroing method must be performed before the navigation procedure is started, or during use, as soon as a marker is changed, cleaned, or if the fusionTrack device must be cleaned, whilst the accuracy assessment method is performed with a frequency determined by the integrator.	<input type="checkbox"/>
At any time, the marker registration error and the marker epipolar error (defined in Section 7.8) computed by the <i>SDK</i> must be monitored. The integrator must define thresholds above which the errors can lead to unacceptable risk for the patient. If the errors go above the aforementioned thresholds, the integrator must prevent the user from accessing the data.	<input type="checkbox"/>
A large epipolar error affecting all fiducials can be caused by the fusionTrack operating outside its temperature range or being decalibrated.	<input type="checkbox"/>
A large registration error can be caused by the orientation of the <i>marker</i> : e.g. if the <i>marker</i> becomes close to parallel to the optical axis. The integrator shall generate an error message if the orientation goes above a defined angular threshold.	<input type="checkbox"/>
A large registration error can be caused by an object in the field of view which occludes partially the marker. The integrator shall state in their user manual that no objects in the field of view are allowed but the <i>markers</i> , pointers or surgical tools.	<input type="checkbox"/>
A large registration error can be caused by a scratched fiducial, liquids on fiducial, damaged fiducial, wrongly fixed fiducials on the marker, or a deformed marker. The integrator must generate an error message when the marker registration error is too high.	<input type="checkbox"/>
The integrator shall monitor temperature sensors. If the measured temperatures diverge from the reference temperatures, the tracking data shall not be given to the end-user.	<input type="checkbox"/>
A too high temperature due to internal overheating, or due to the fusionTrack device being wrongly placed near a heat source, might lead to unexpected temperature measurements. The integrator has to trigger an error message indicating that the fusionTrack sensor has detected an abnormal temperature within the system.	<input type="checkbox"/>

Using the fusionTrack device out of humidity specifications might lead to wrong measurements. The integrator has to specify in the end user manual the fusionTrack humidity specifications as described in Table 5.1.	<input type="checkbox"/>
Any degradation of optical elements due to the cleaning method or scratches during transport can lead to wrong measurements. Optical elements should be handled with care during transport and storage. The integrator has to mention in the end user manual the cleaning methods for optical elements as described in Chapter 17.	<input type="checkbox"/>
Instability of the fusionTrack device can cause injury by falling on patient or operator. Attachment between feet and holding mechanism must be perfectly stable. Instructions indicating how to fix fusionTrack using the threads in the feet are provided in Section 5.4.	<input type="checkbox"/>
Polluting light in the field of view might lead to wrong measurements. The integrator can access the camera images to visualize the scene of view. A segmentation overflow mechanism and status flags in the frame structures can be used by the integrator to detect any perturbation. The integrator is suggested to periodically check the camera pictures.	<input type="checkbox"/>
The fusionTrack is not intended to be sterilised. A contamination on cameras could lead to patient contamination. The fusionTrack has to be placed outside of the sterile field and depending on the intervention, cleaned before every single use. Methods of cleaning are indicated in Chapter 17.	<input type="checkbox"/>
The integrator must check that the fusionTrack device and the position of the fusionTrack device is well-suited to the application / surgery, and that required regulations are respected.	<input type="checkbox"/>
At any time, the hardware (i.e. the fusionTrack device), the firmware and the software (i.e. the <i>SDK</i>) must be compatible. New releases of firmware, software and their respective compatibility will be notified to the integrator.	<input type="checkbox"/>
A software dysfunction due to software crash might cause wrong measurements causing patient injury. The integrator has to indicate in the end user manual that the fusionTrack device must be restarted when the software freezes or crashes.	<input type="checkbox"/>

A software dysfunction such as frozen pictures might cause wrong measurements causing patient injury. The integrator has to implement a software mechanism to detect such cases (e.g. check timestamps, etc.).	<input type="checkbox"/>
In the case of the fusionTrack device stops functioning during use / operation, it is the integrator responsibility to ensure that surgeon resume operation with the conventional method.	<input type="checkbox"/>
Any wrong settings of the fusionTrack device, misuse of the SDK or misuse of the OS functionalities might cause wrong measurements causing patient injury. The integrator should perform unit tests during the development process to validate configured settings.	<input type="checkbox"/>
The integrator must take care that the host PC meets the SDK <i>and</i> their software requirement, especially the needed memory and the nominal CPU usage must be paid attention to.	<input type="checkbox"/>
The integrator implementation may cause inconsistent latency. The integrator should evaluate the system's latency once implemented and integrated in his product.	<input type="checkbox"/>
The integrator must check that UDP packets are well-ordered, i.e. that packets sent sequentially are received in the same order. The UDP protocol does not provide any guarantee, therefore the integrator must check the UDP packet ordering on their platform. Unordered packets may affect the performances of the fusionTrack.	<input type="checkbox"/>
A software dysfunction due to unavailable data could lead to wrong measurements or the non performing of the surgery as planned. The integrator has to perform unit tests according to the end application of the fusionTrack device.	<input type="checkbox"/>
A non respect of the software functionality or unexpected software failure might cause wrong measurements. The integrator should perform unit tests during the development process to validate the software functionality.	<input type="checkbox"/>
On Unix/Linux, the integrator must ensure the software is functional with the used GNU C library, C++ Standard Library and POSIX thread versions.	<input type="checkbox"/>
The integrator must generate a 'noisy environment' error message when the environment is poluted with reflective areas or other light sources.	<input type="checkbox"/>

The zeroing method described in Chapter 12 must be implemented by the integrator and be automatically run at the application startup and inform the user on how to proceed.	<input type="checkbox"/>
Any liquid on lenses such as body fluids ejection or condensation could lead to wrong measurements. Lenses have to be cleaned as soon as they get in contact with liquids. The zeroing method has to be re-performed after each cleaning.	<input type="checkbox"/>
Some disposable passive <i>fiducials</i> cannot be cleaned, as the reflective coating would be damaged, and may require to be changed during the operation. The zeroing method has to be re-performed after changing any <i>fiducials</i> .	<input type="checkbox"/>
Accuracy verification must be implemented by the integrator. They can optionnaly purchase the required software and hardware at Atracsys.	<input type="checkbox"/>
The operator performing accuracy verification must be correctly trained.	<input type="checkbox"/>
Use of the fusionTrack adjacent to, or stacked with other equipment should be avoided because it could result in improper operation and can lead to wrong measurements to serious injury including death. If such use is necessary, the fusionTrack and the other equipment should be observed to verify that they are operating normally.	<input type="checkbox"/>
Use of accessories, transducers and cables other than those specified or provided by Atracsys could result in increased electromagnetic emissions or decreased electromagnetic immunity of this equipment and result in improper operation and/or loss of performance. Such failure can lead to wrong measurements, serious injury including death.	<input type="checkbox"/>
Portable radio frequency communications equipment (including peripherals such as antenna cables and external antennas) should be used no closer than 30 cm (12 inches) to any part of the fusionTrack, including cables specified by Atracsys. Otherwise, degradation of the performance of this equipment could result, which can lead to wrong measurements, serious injury including death.	<input type="checkbox"/>
The fusionTrack was not designed to be autoclavable. The end user manual instruction shall provide a validated cleaning and disinfection method to the end-user.	<input type="checkbox"/>

It must be ensured that no fluids enter the fusionTrack. This may damage the fusionTrack by causing short-circuits and corrode the material. Do not immerse the fusionTrack in liquid.	<input type="checkbox"/>
The integrator must ensure markers are not deformed after sterilisation.	<input type="checkbox"/>
The filters have to be inspected regularly against dust and also controlled for any damages like holes.	<input type="checkbox"/>
Do not clean the filters while still attached to the device.	<input type="checkbox"/>
Do not leave or run the system without the filters as dust might enter.	<input type="checkbox"/>
Do not use a vacuum cleaner or pressurized air on the system.	<input type="checkbox"/>
The integrator must indicate in their user manual the frequency of filter replacement.	<input type="checkbox"/>
ensure markers are not deformed after sterilisation.	<input type="checkbox"/>

References

- [1] Atracsys LLC, *Atracsys Passive SDK API documentation (doxygen documentation)*.
- [2] http://en.wikipedia.org/wiki/INI_file, 2014. Online; accessed 2015-01-13.
- [3] Roger M. Needham, David J. Wheeler, “Tea extension,” tech. rep., Computer Laboratory, University of Cambridge, 1997.
- [4] A. F. Möbius, *Der barycentrische Calcül: ein neues Hilfsmittel zur analytischen Behandlung der Geometrie*. 1827.