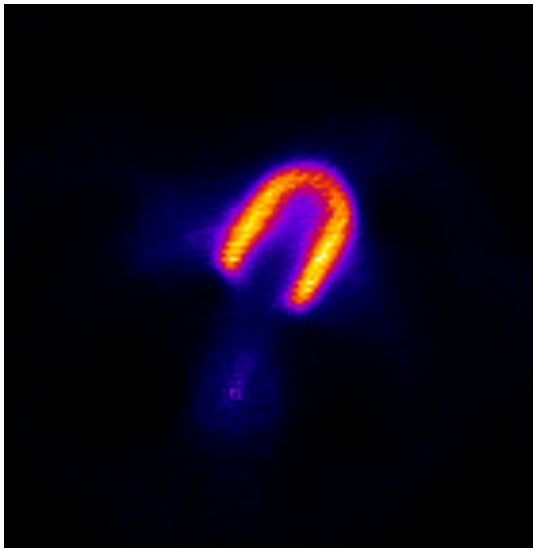
List-mode tracking and motion correction for PET imaging using low-activity fiducial markers



Marc Chamberland, MSc (PhD in Spring 2015!)
Robert deKemp, PhD
Tong Xu, PhD

STIR User's Meeting at IEEE MIC

Carleton UNIVERSITY

The Ottawa Medical Physics Institute

UNIVERSITY OF OTTAWA HEART INSTITUTE
INSTITUT DE CARDIOLOGIE DE L'UNIVERSITÉ D'OTTAWA

My claim to fame, from the STIR Wiki FAQ:

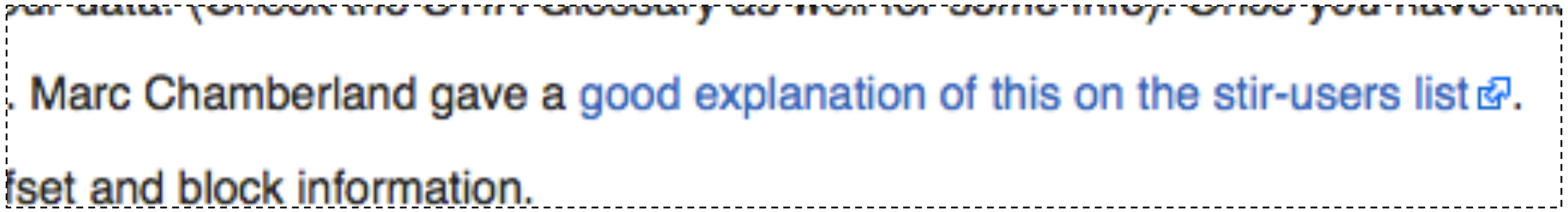
How do I add my own scanner to STIR?

You might not need to. If you specify the scanner geometry in your Interfile header, STIR will handle it ok.

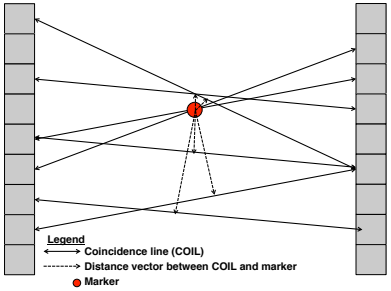
For instance, you could use create_projdata_template, pick a scanner that might be somewhat similar to yours, and then edit the generated Interfile header. The scanner part of the header takes the same information as Scanner:set_params() (take care of changes between mm and cm). Obviously, it contains more information such as the actual number of views, ring differences etc that is supposed to be in your data. (Check the STIR Glossary as well for some info). Once you have this template, you should be good to go.

Alternatively, you will have to modify the `Scanner` class. Marc Chamberland gave a good explanation of this on the stir-users list.

Note that STIR (at least 2.2 and earlier) ignores view_offset and block information.

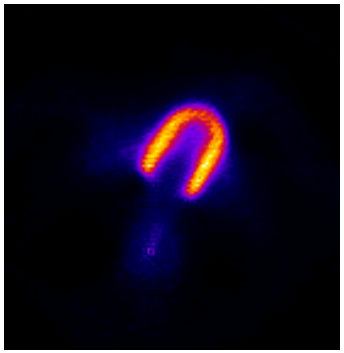


Outline



Motion tracking and correction (Poster M10-5)

Use of STIR



Results

Motion tracking and correction

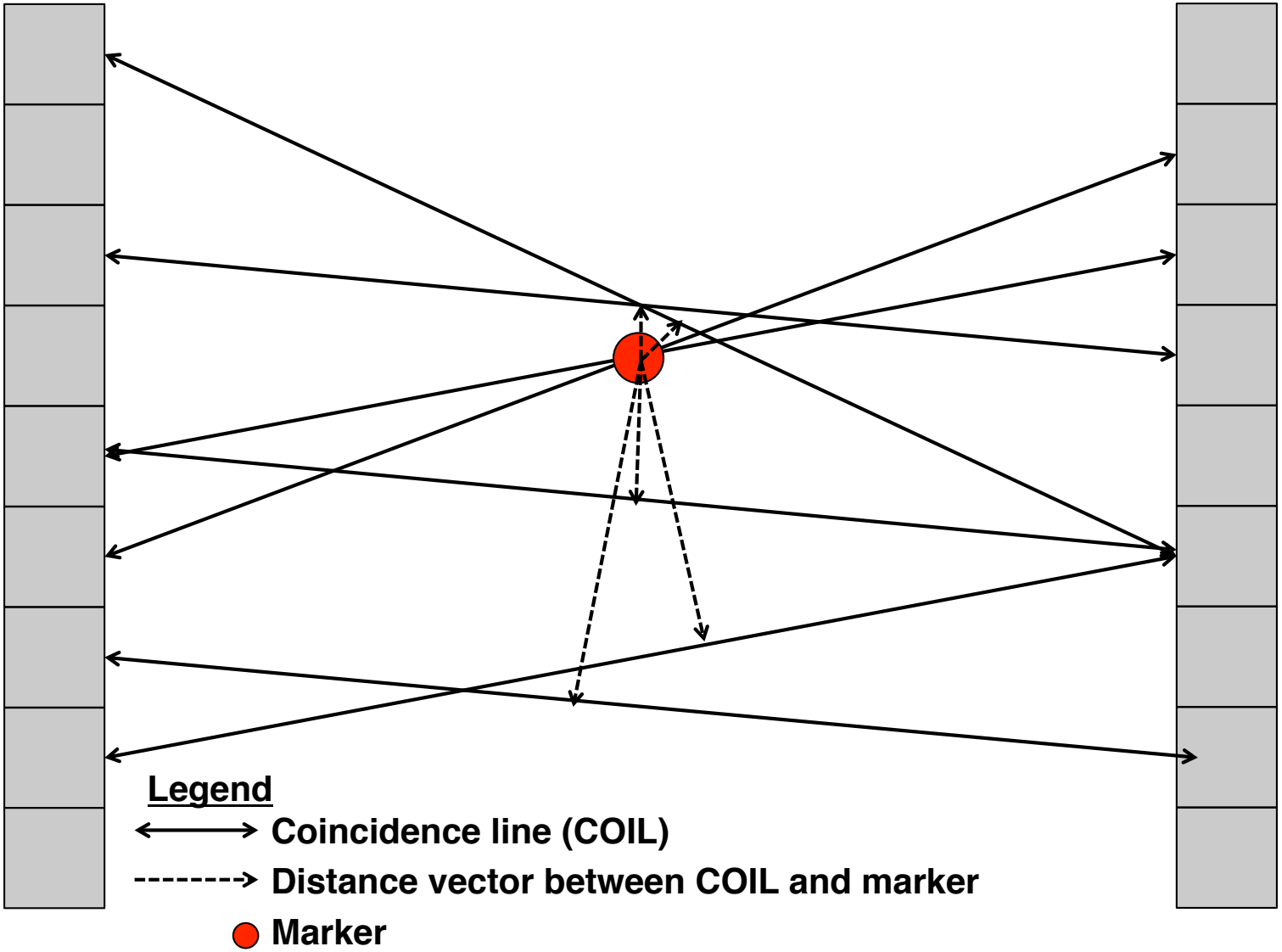
Positron emission tomography (PET) imaging is routinely used to assess myocardial perfusion...

Walls of the heart insert of a phantom, filled with rubidium-82 (^{82}Rb) tracer. (Coronal slice) Above: no motion.

…but respiratory, cardiac, and patient body motion create artefacts on the images.

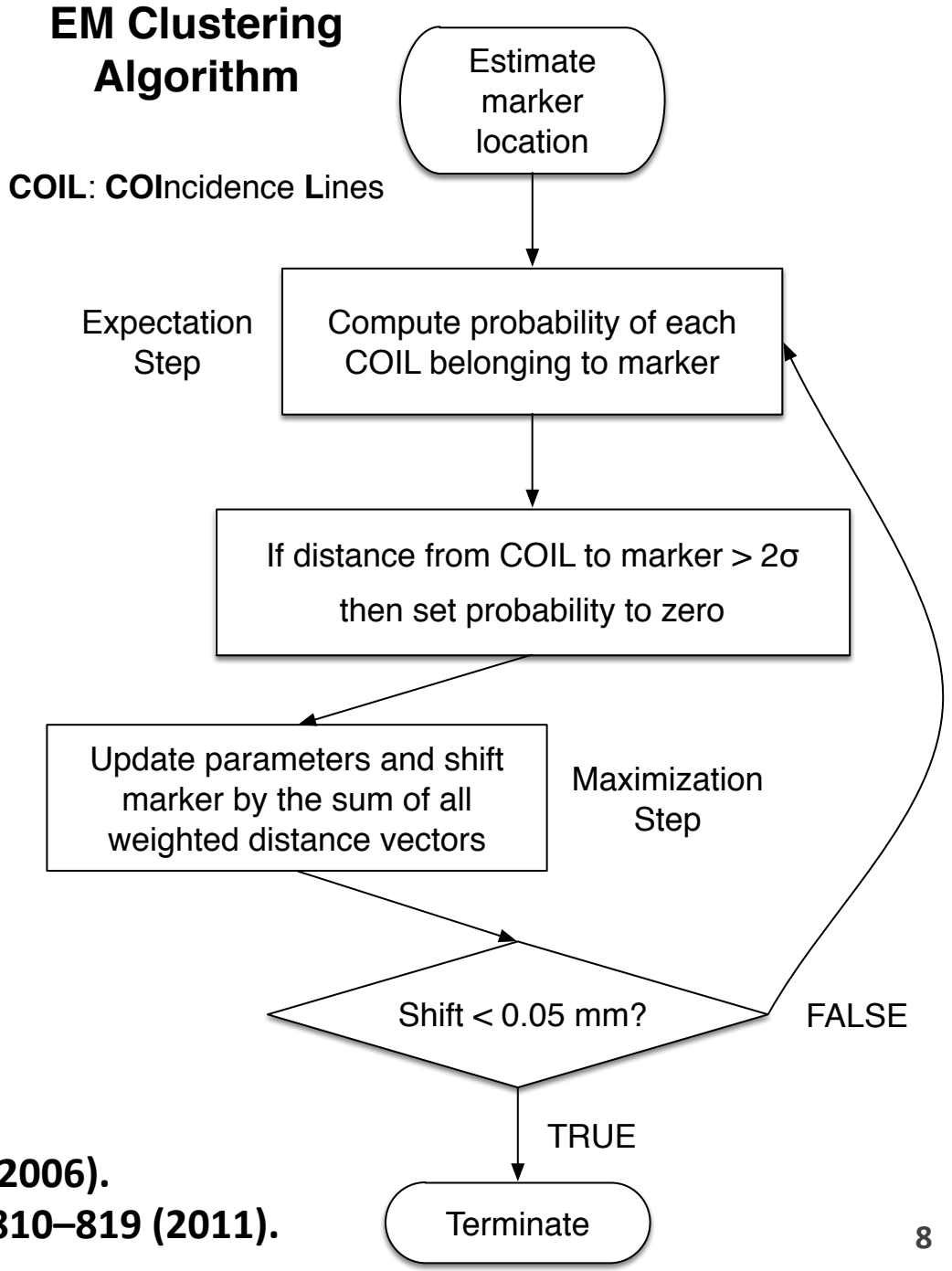
Walls of the heart insert of a phantom, filled with rubidium-82 (^{82}Rb) tracer. (Coronal slice) Above: no motion; *below: motion blurring.*

A fiducial point-like positron-emitting marker can be localized by finding the position in space which minimizes the root mean square distance to its coincidence lines.



We developed an iterative expectation-maximization (EM) algorithm to track the 3D location of fiducial positron-emitting markers.

Some background rejection is needed to deal with the tracer activity in the patient's body. (Not shown in figure.)

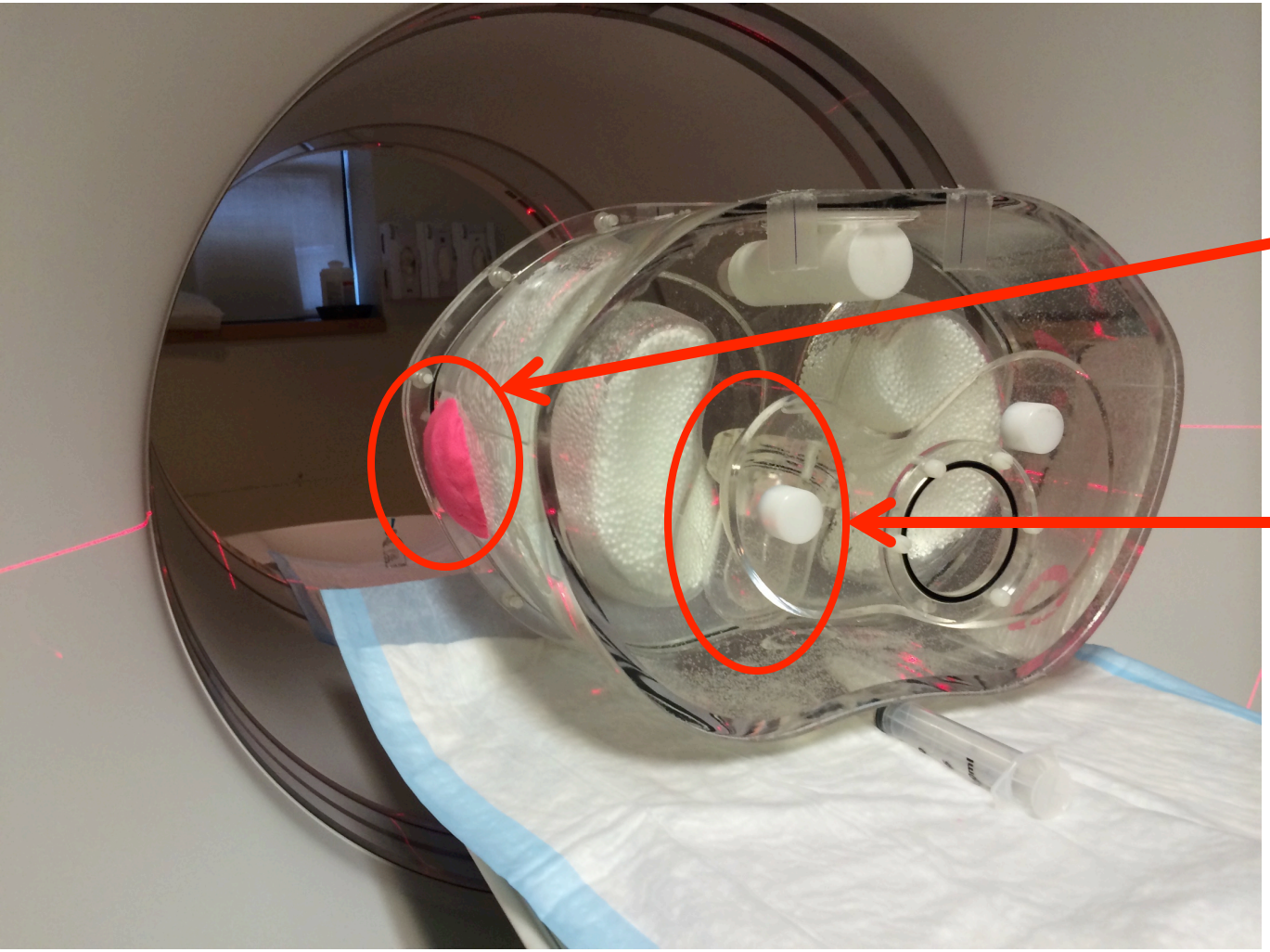


References

T. Xu et al., Med. Phys. 33(7), 2598–2609 (2006).

M. Chamberland et al., Med. Phys. 38(2), 810–819 (2011).

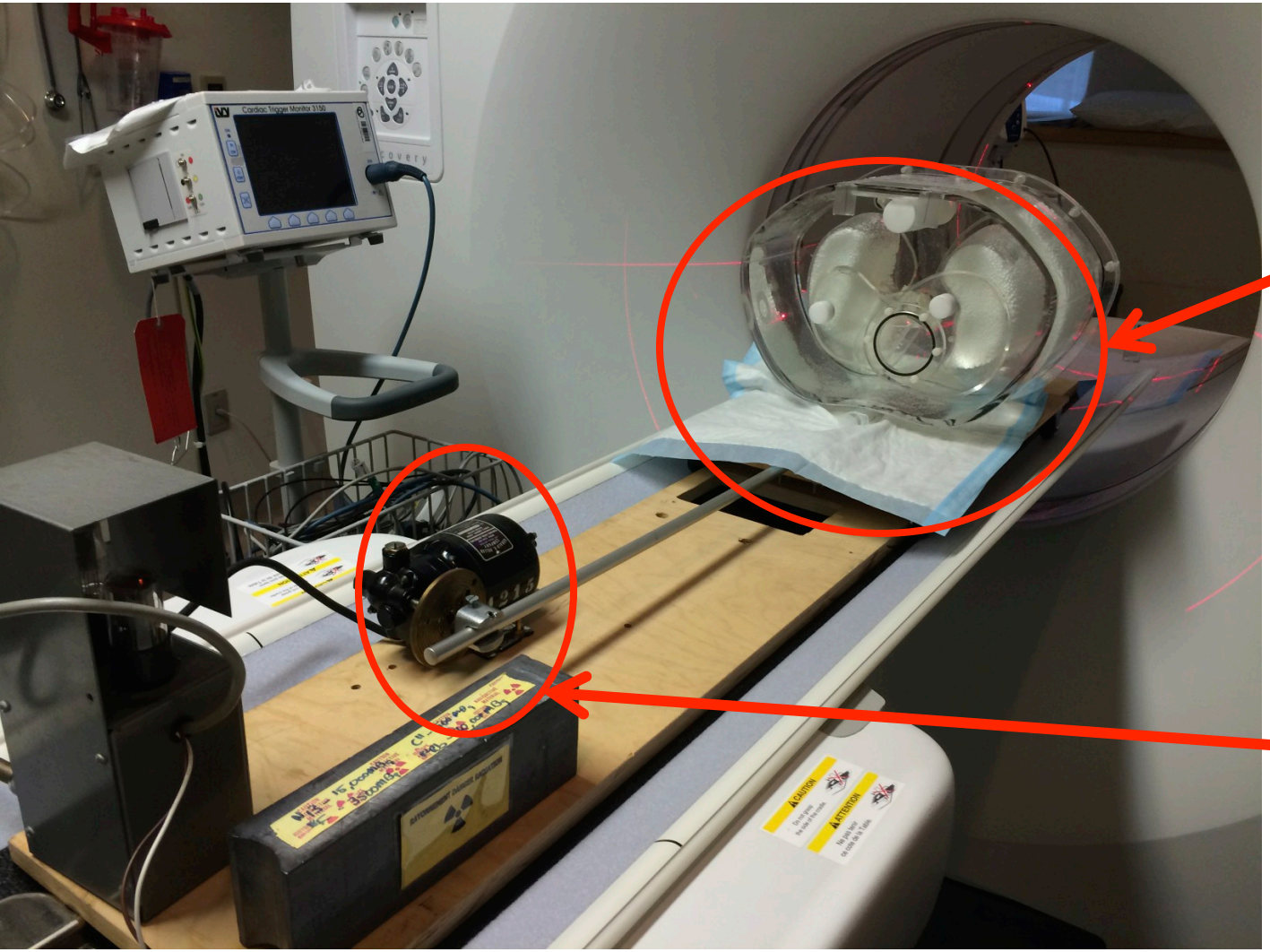
We conducted a phantom study with a thorax phantom.



^{22}Na (92.5 kBq)

Heart insert filled with ^{82}Rb (680 MBq initially)

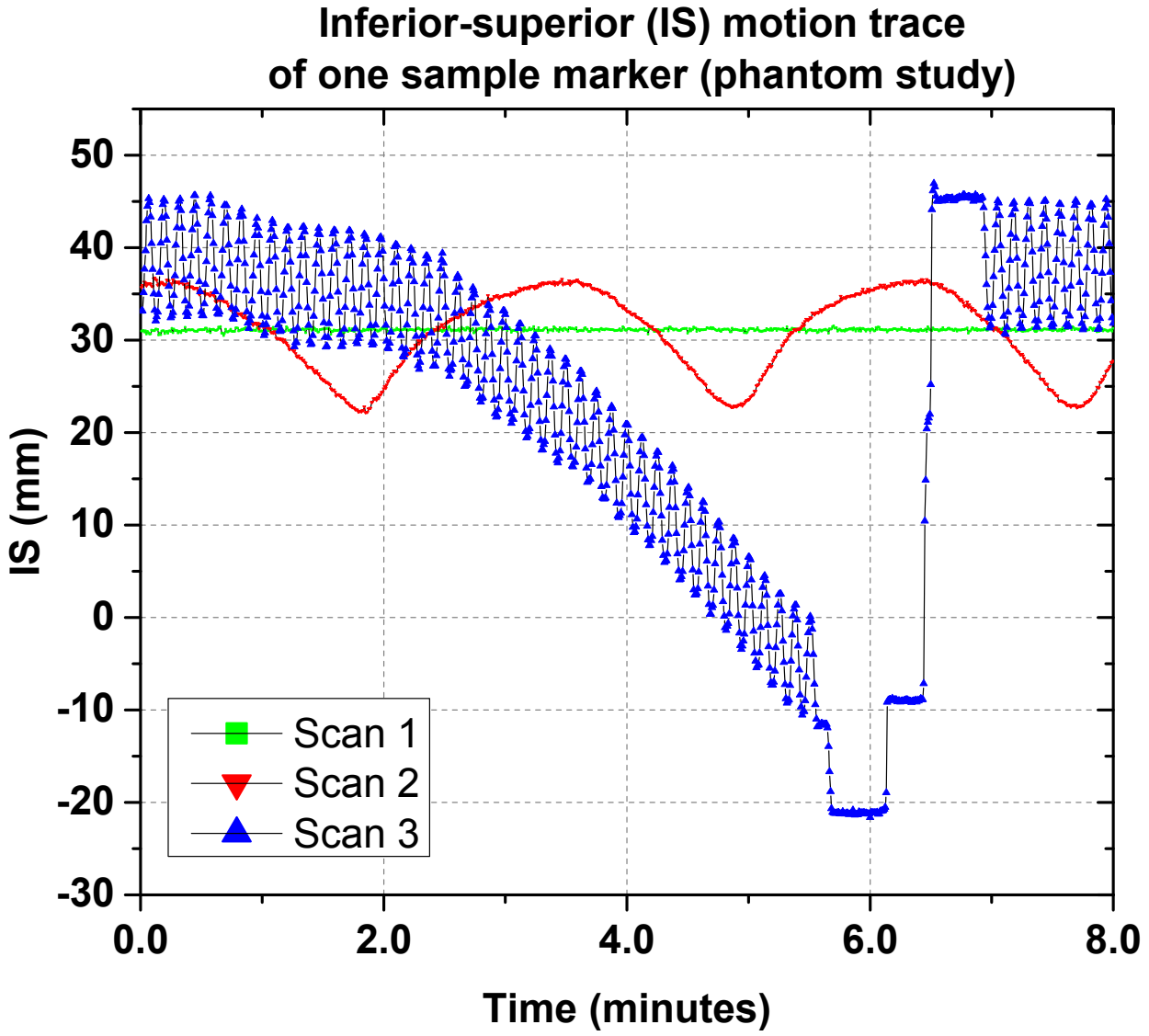
We transformed the phantom into a dynamic one!



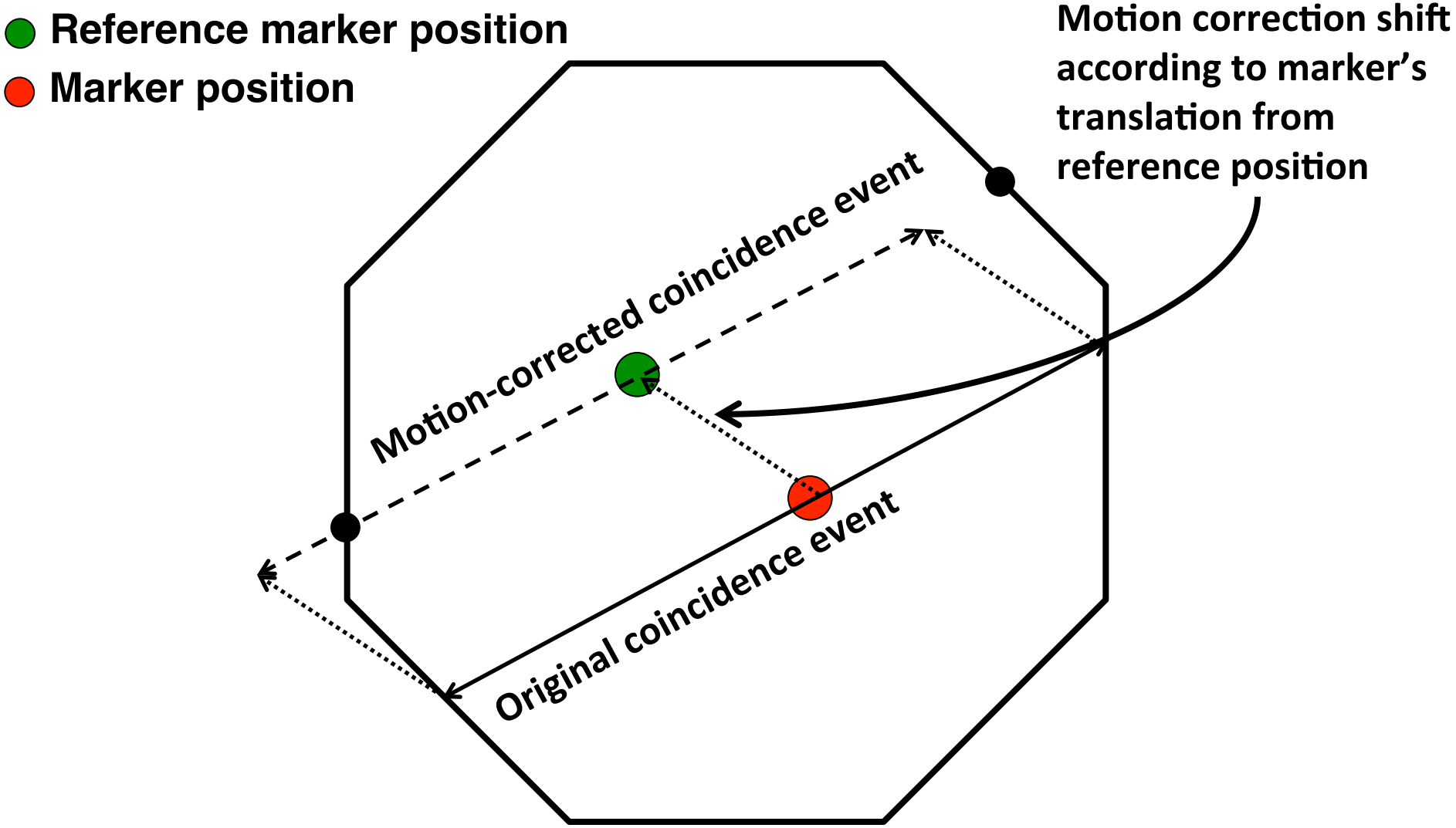
Phantom resting on wooden board with wheels

Motor with rotating plate

The marker was tracked using the raw list-mode data acquired during the PET scan.



The motion trace of the marker can be used to apply motion correction to the raw list-mode data.



Use of STIR

To check if the motion correction worked, I had two choices:

Use the GE Discovery 690's console to reconstruct the motion-corrected list-mode...

...or use STIR.

To reconstruct with STIR, I had to rely on projection data. Adding GE list-mode support was outside my C++ comfort zone.

make corresponding derived classes of CListModeData, CListRecord etc.
InputFileFormat class.

Notes for developers

If you want to add a new type of list mode data, you have to make corresponding derived classes of **CListModeData**, **CListRecord** etc. You also have to modify make sure that read_from_file<CListModeData> recognises your data. This normally involves creating a new **InputFileFormat** class.

Member Function Documentation

virtual std::string stir::CListModeData::get_name ( ) const pure virtual

Returns the name of the list mode data.

I added (non-validated) support for the GE Discovery 690 scanner in STIR by modifying Scanner.h…

Scanner.h

Scanner.h › class Scanner

```
92  class Scanner
93  {
94   public:
95
96     /************** static members*********************************/
97    static Scanner * ask_parameters();
98
99    //! get the scanner pointer from the name
100   static Scanner * get_scanner_from_name(const string& name);
101   //! get the list of all names for the particular scanner
102   static string list_all_names();
103
104   // E931 HAS to be first, Unknown_scanner HAS to be last
105   // also, the list HAS to be consecutive (so DO NOT assign numbers here)
106   // finally, test_Scanner assumes E966 is the last in the list of CTI scanners
107   // supported by ecat_model from the LLN matrix library
108   // 08-3-2004, zlong, add user defined scanner
109   //! enum for all predefined scanners
110   /* \a Userdefined_scanner can be used to set arbitrary scanner geometry info.
111      \a Unknown_scanner will be used when parsing (e.g. from an Interfile header)
112      to flag up an error and do some guess work in trying to recognise the scanner from
113      any given parameters.
114   */
115   enum Type {E931, E951, E953, E921, E925, E961, E962, E966, E1080, Siemens_mMR, RPT,HIDAC,
116          Advance, DiscoveryLS, DiscoveryST, DiscoverySTE, DiscoveryRX, Discovery600, Discovery690,
117          HZLR, RATPET, PANDA, HYPERimage, nanoPET, HRRT, Allegro, GeminiTF, User_defined_scanner,
118          Unknown_scanner};
```

…and Scanner.cxx using the published characteristics of the scanner (and some detective work).

Scanner.cxx

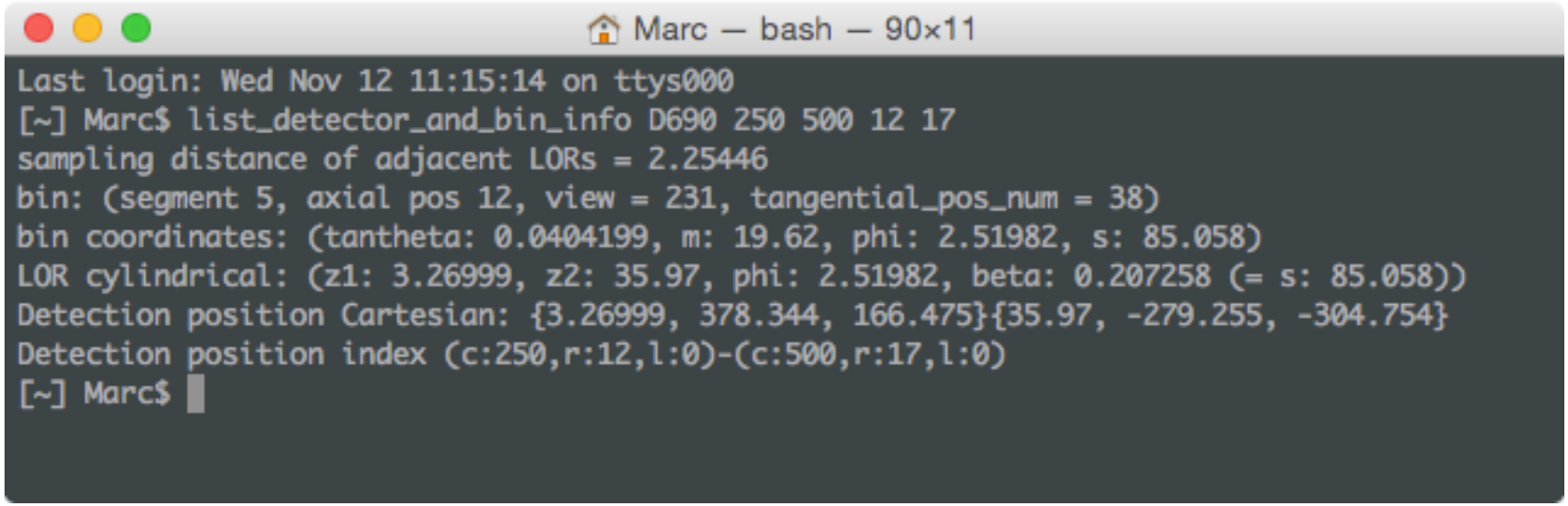
Scanner.cxx › Scanner::Scanner(Type scanner_type)

```
324
325 case Discovery690:
326
327     set_params(Discovery690, string_list("GE Discovery 690", "Discovery 690", "GE D690", "D690"),
328                24,
329                381,
330                331, // TODO
331                2 * 288,
332                826.70F/2.F - 8.4F,
333                8.4F,
334                6.54F,
335                2.25446F,
336                static_cast<float>(-4.9730*_PI/180),//sign? TODO value
337                4,
338                2,
339                6,
340                9,
341                1,
342                1,
343                1);
344     break;
345
346   case HZLR:
347
348     set_params(HZLR, string_list("Positron HZL/R"),
349                32, 256, 2 * 192,
350                780.0F, 7.0F, 5.1875F, 2.F, 0.0F,
351                0, 0, 0, 0, 0,0, 1);
352     // Default 7.0mm average interaction depth.
353     //  crystals per singles unit etc unknown
354     break;
355
```

Effective ring diameter and average depth of interaction.

Number of blocks per module, crystals per module, etc.

The list_detector_and_bin_info utility was extremely useful in wrapping my head around STIR and projection data conventions.



This allowed me to check if the detectors were where I thought they were in Cartesian coordinates.

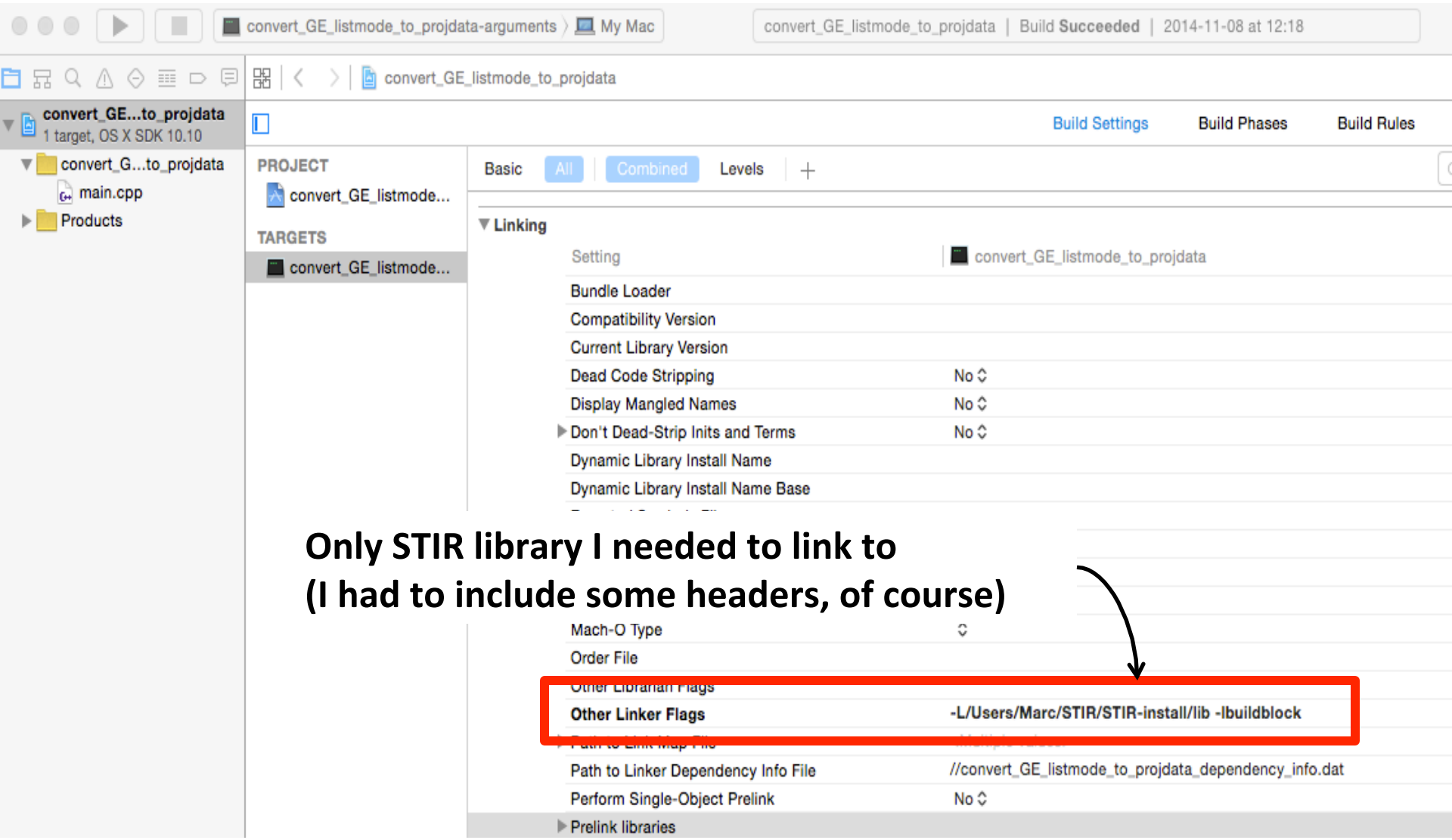
Fun discovery: everybody uses different conventions! (E.g. left-handed coordinate system instead of right-handed.)

In order to reconstruct the images, I wrote a utility that converts GE list-mode to projection data.

```
205         //assume it's a coincidence event
206         listmode.read((char*)&coincidenceEvent, sizeof(coincEvent));
207
208         //check if it's really a coincidence event
209         if (coincidenceEvent.coinc == 1) {
210             int X1 = coincidenceEvent.hiXtalRadialID;
211             int Z1 = coincidenceEvent.hiXtalAxialID;
212             int X2 = coincidenceEvent.loXtalRadialID;
213             int Z2 = coincidenceEvent.loXtalAxialID;
214
215             //in STIR, the first ring is at the back of the scanner;
216             //for GE, it is at the front
217             Z1 = (num_rings - 1) - Z1;
218             Z2 = (num_rings - 1) - Z2;
219
220             Bin bin;
221             proj_data_info_sptr->get_bin_for_det_pair(bin, X1, Z1, X2, Z2)
222
223             //for each coordinate, make sure it is positive
224             //i.e. segments and tangential bins can be negative in STIR
225             int segment = bin.segment_num() + (int)(num_segments/2);
226             int view = bin.view_num();
227             int z = bin.axial_pos_num();
228             int r = bin.tangential_pos_num() + (int)(num_tangential_pos/2);
229
230             //fill proj data array
231             ++projectionData4D[segment][view][z][r];
232         }//if (coinc event)
```

Member of ProjDataInfoCylindricalNoArcCorr class

In order to reconstruct the images, I wrote a utility that converts GE list-mode to projection data.



Only STIR library I needed to link to
(I had to include some headers, of course)

I also did some work on a utility to convert crystal and geometric efficiency files from the D690 to projection data.

create_normalisation_files_from_GE_D690_calibration_files.cxx

create_normalisation_files_from_GE_D690_calibration_files.cxx › main(int argc, char *argv[])

```
177     //now loop over projection data array and fill it with factors
178     for ( int segment = min_segment_num; segment <= max_segment_num; ++segment )
179     {
180       //each segment has a different number of axial coordinates
181       int num_axial_pos = proj_data_info_sptr->get_num_axial_poss(segment);
182       for ( int view = min_view_num; view <= max_view_num; ++view )
183       {
184         //a new array is needed every time because of the changing
185         //number of axial coordinates
186         array2D projectionData(boost::extents[num_axial_pos][num_tangential_pos]);
187         for ( int z = 0 ; z < num_axial_pos ; ++z)
188         {
189           for ( int r = min_tangential_pos_num ; r <= max_tangential_pos_num ; ++r)
190           {
191             Bin bin = Bin(segment, view, z, r);
192             int detector1, detector2;
193             int ring1, ring2;
194
195             proj_data_info_sptr->get_det_pair_for_bin(detector1, ring1, detector2, ring2, bin);
196
197             //in STIR, the first ring is at the back of the scanner; for GE, it is at the front
198             ring1 = (num_rings - 1) - ring1;
199             ring2 = (num_rings - 1) - ring2;
200
201             float detector1_efficiency = crystalEfficiency[ring1][detector1];
202             float detector2_efficiency = crystalEfficiency[ring2][detector2];
203
204             //in STIR, the tangential bin index can be negative, so I need to
205             //shift it and make it positive to store it in the right element
206             int r_index = r + (int)(num_tangential_pos/2);
207             projectionData[z][r_index] = detector1_efficiency * detector2_efficiency;
208
```

Member of ProjDataInfoCylindricalNoArcCorr class

Side note: I found neat stuff that is being worked on for future STIR releases.

file:///Users/Marc/STIR/STIR-doc/documentation/doxy/html/classstir_1_1LORAs2Poi

Public Member Functions | List of all members

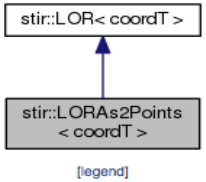
stir::LORAs2Points< coordT > Class Template Reference

Items related to Line Of Responses (preliminary)

A class for LORs. More...

`#include "stir/LORCoordinates.h"`

Inheritance diagram for stir::LORAs2Points< coordT >:



My tracking and correction code implements its own LOR class and methods…

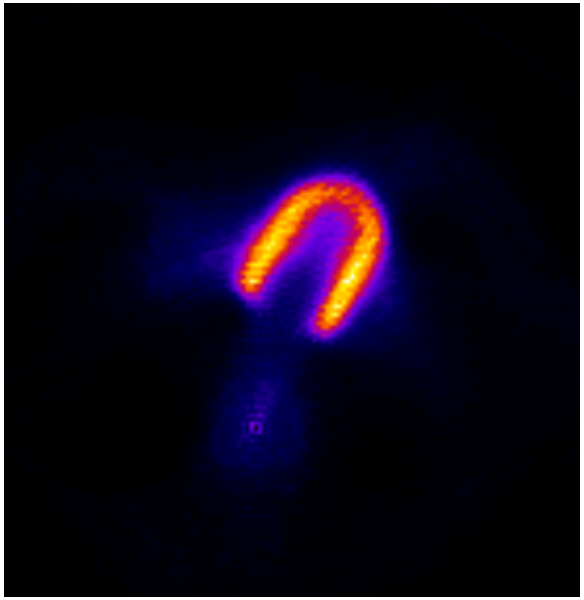
Public Member Functions

const CartesianCoordinate3D < coordT > &	**p1** () const
CartesianCoordinate3D< coordT > &	**p1** ()
const CartesianCoordinate3D < coordT > &	**p2** () const
CartesianCoordinate3D< coordT > &	**p2** ()
	LORAs2Points (const Ca
	LORAs2Points (const LO
	LORAs2Points (const LO
	LORAs2Points (const LORInAxialAndNoArcCorrSinogramCoordinates< coordT > &)
virtual self_type *	**clone** () const
virtual Succeeded	**change_representation** (LORInCylinderCoordinates< coordT > &, const double radius) const
virtual Succeeded	**change_representation** (LORInAxialAndNoArcCorrSinogramCoordinates< coordT > &, const double radius) const
virtual Succeeded	radius) const
virtual Succeeded	**get_intersections_with_cylinder** (LORAs2Points< coordT > &, const double radius) const

I implemented a function to get the intersections between an N-sided polygon and a LOR…

Detailed Description

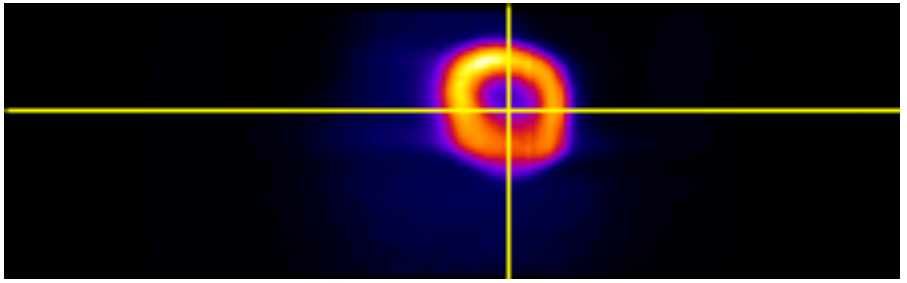
template<typename coordT>
class stir::LORAs2Points< coordT >



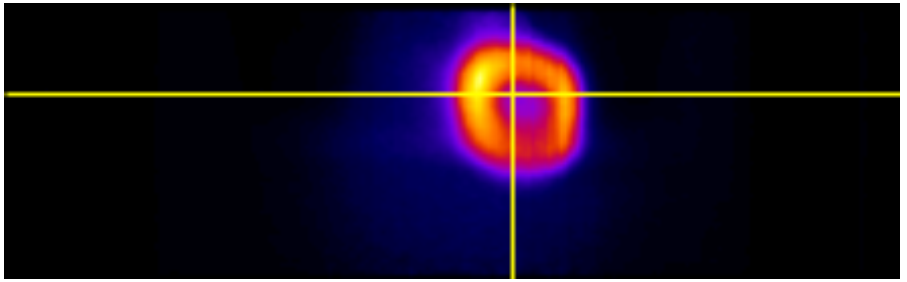
Results

Qualitatively, the images are (relatively) motion free. (Several corrections were not applied!)

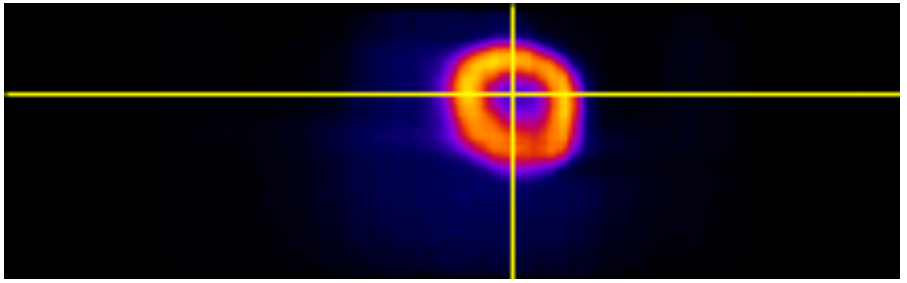
Static scan



Fast, irregular motion

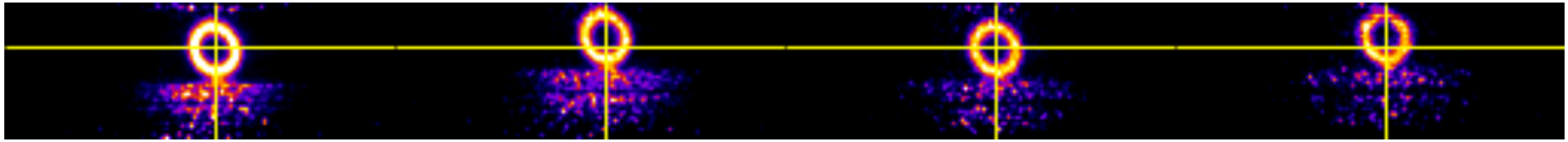


Fast, irregular motion; motion corrected

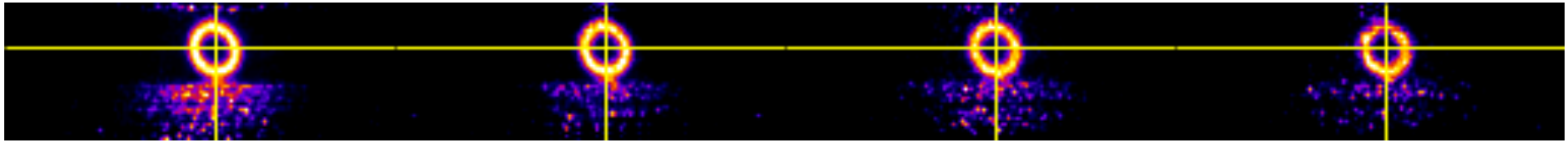


Qualitatively, the images are (relatively) motion free. (Several corrections were not applied!)

No motion correction



0-15 s | 105-120 s | 210-225 s | 285-300 s



With motion correction

NOTE: Those images were reconstructed from the GE console, not by STIR.

Take-away message:

1. **Motion tracking using low-activity fiducial positron-emitting markers and correcting the raw list-mode data from a GE D690 scanner.**

2. **Using STIR to convert the list-mode and calibration files to projection data, correct the data, and reconstruct for qualitative assessment of motion.**

3. **It works! (And I have patient data, but not quite ready to show.)**

Future work:

1. **Better corrections after applying motion correction.**

2. **LOR-as-2-points-based reconstruction?**

3. **Contribute something to STIR!**

Acknowledgements

1. Dr. Rob de Kemp (University of Ottawa Heart Institute)

2. All the PET staff involved (University of Ottawa Heart Institute)

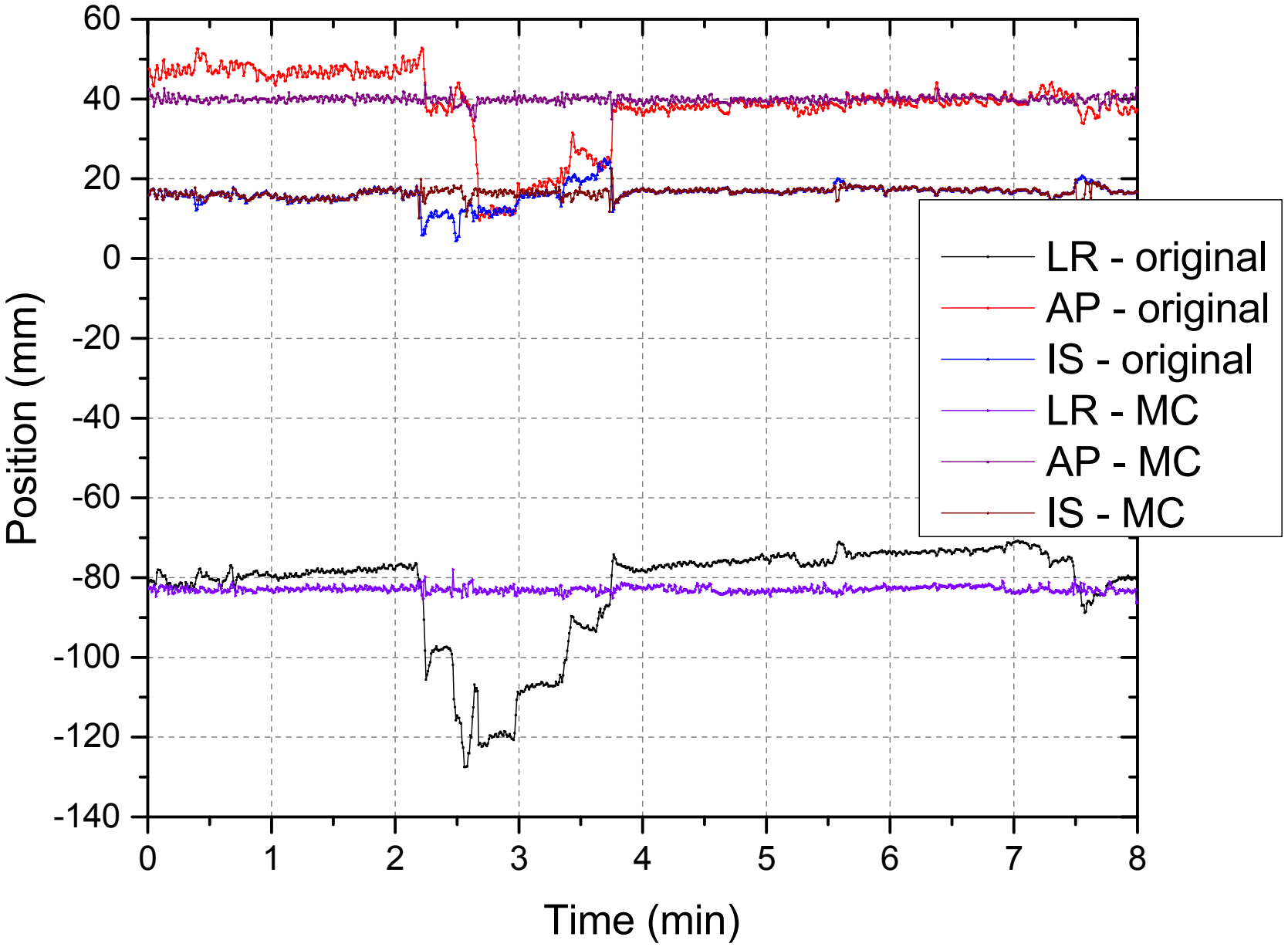
3. Dr. Charles Stearns for technical discussion of the D690 (GE Healthcare)

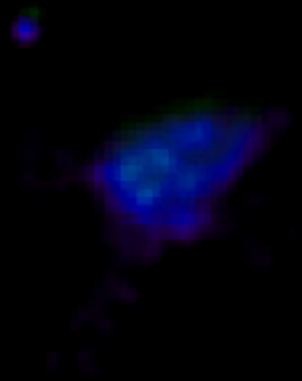
4. Dr. Tong Xu (Department of Physics, Carleton University)

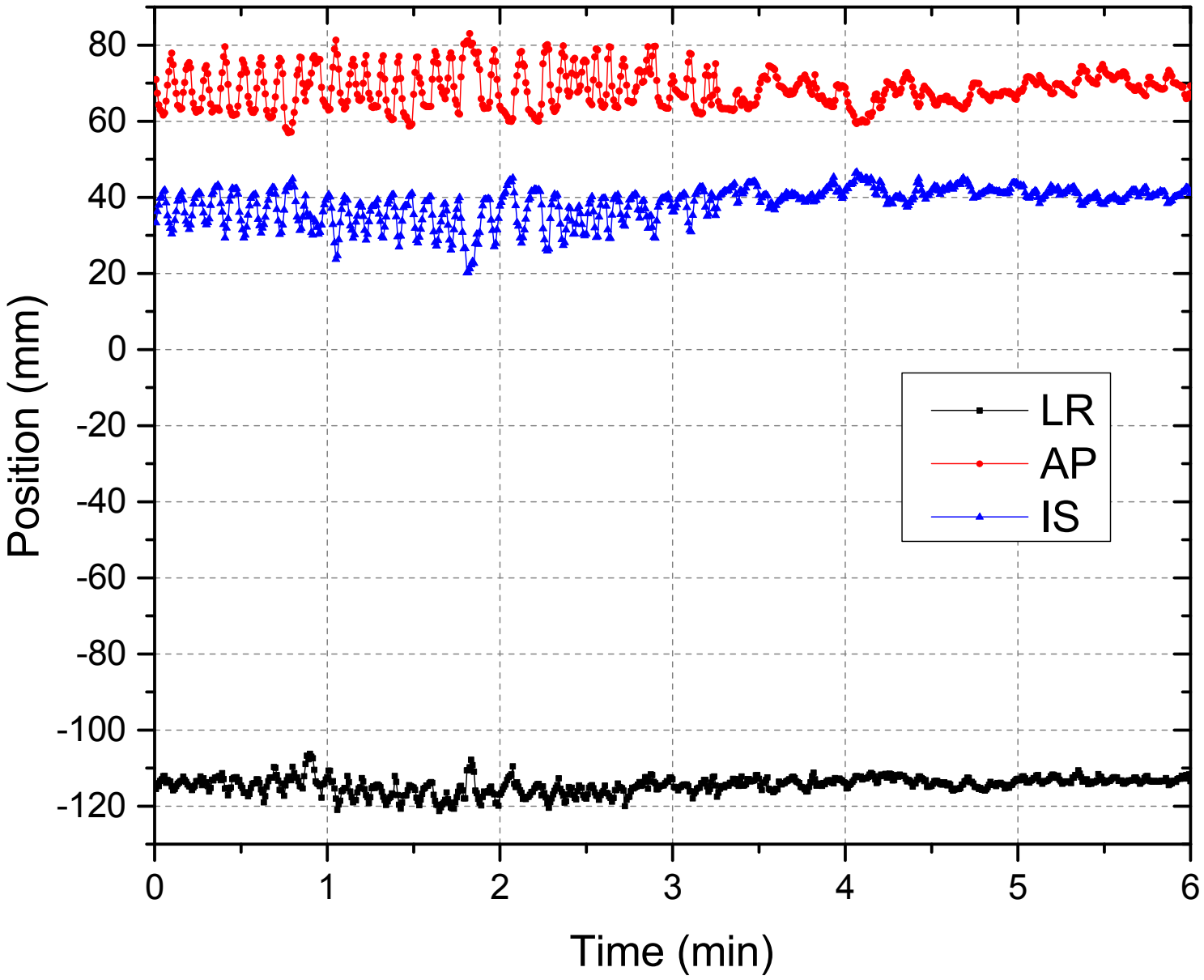
5. IEEE NSS/MIC Conference Trainee Grant for travel support

6. Dr. Thielemans and Dr. Tsoumpas for having answered some of my questions on the STIR mailing list over the years.

Thank you for your time!


60
40
20
0
-20
-40
-60
-80
-100
-120
-140
Position (mm)
0
1
2
3
4
5
6
7
8
Time (min)
LR - original
AP - original
IS - original
LR - MC
AP - MC
IS - MC




Position (mm)
80
60
40
20
0
-20
-40
-60
-80
-100
-120
0
1
2
3
4
5
6
Time (min)
LR
AP
IS