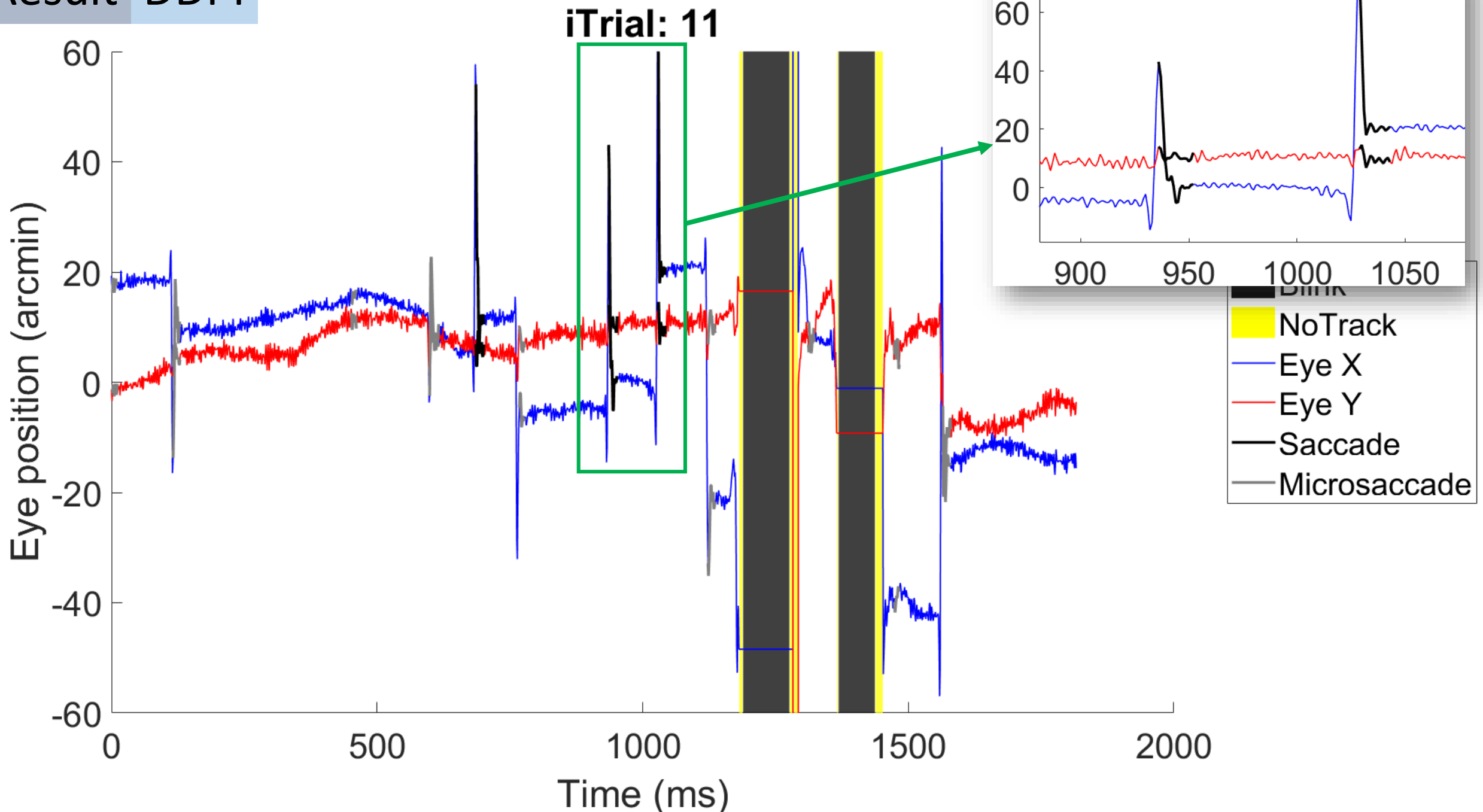
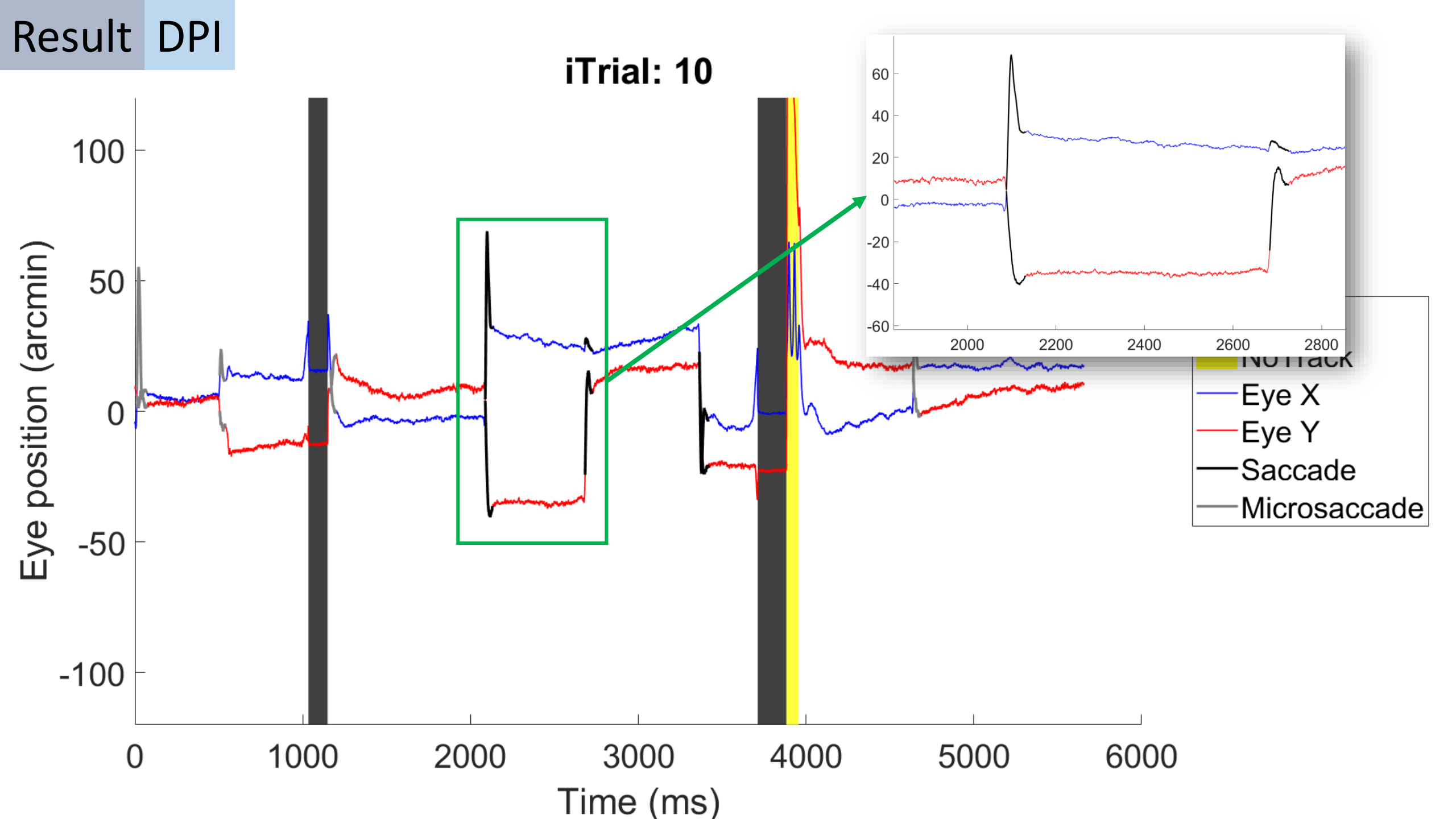
DDPI Online Detection of Eye Movements

Bin Yang
2018.10.16

Result DDPI

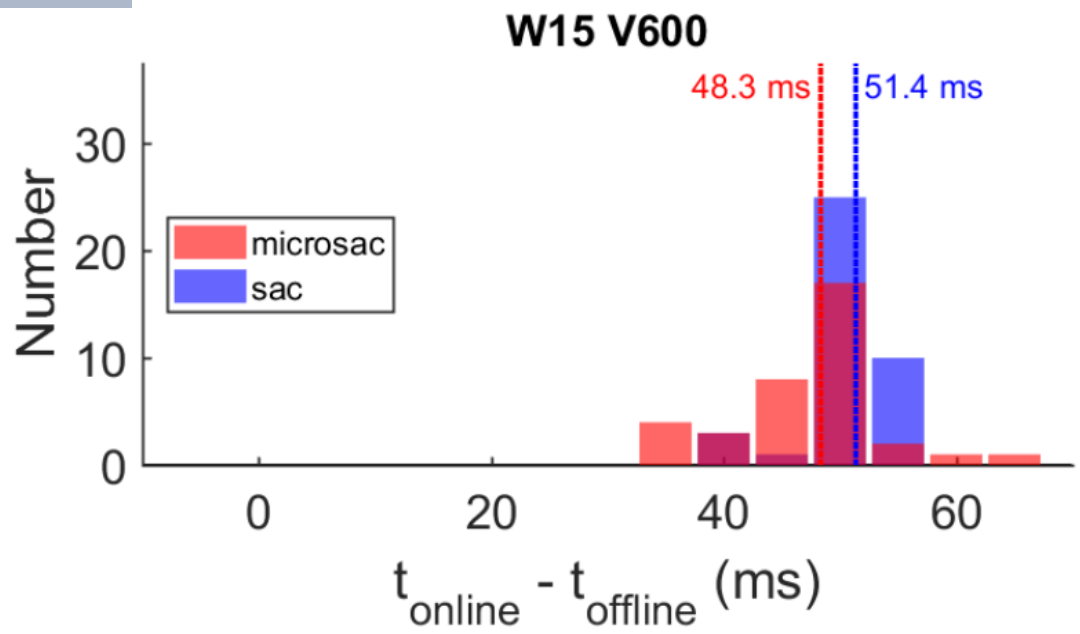

iTrial: 11
Eye position (arcmin)
Time (ms)
60
40
20
0
-20
-40
-60
0
500
1000
1500
2000
900
950
1000
1050
Blink
NoTrack
Eye X
Eye Y
Saccade
Microsaccade

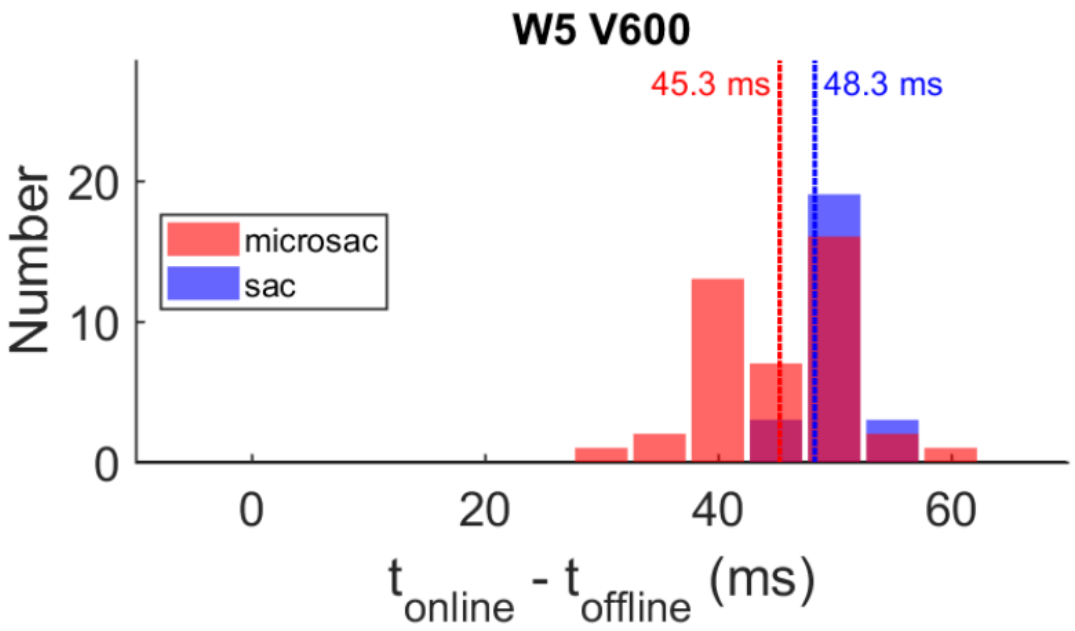
Result DPI

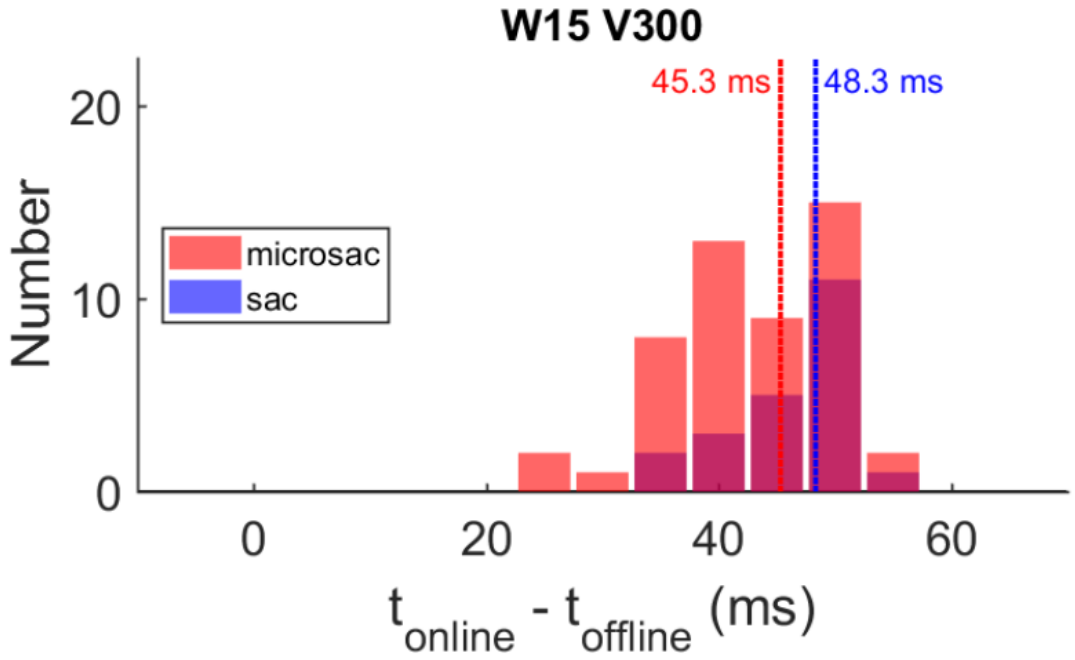

iTrial: 10
Eye position (arcmin)
Time (ms)
100
50
0
-50
-100
0
1000
2000
3000
4000
5000
6000
60
40
20
0
-20
-40
-60
2000
2200
2400
2600
2800
NoTrack
Eye X
Eye Y
Saccade
Microsaccade

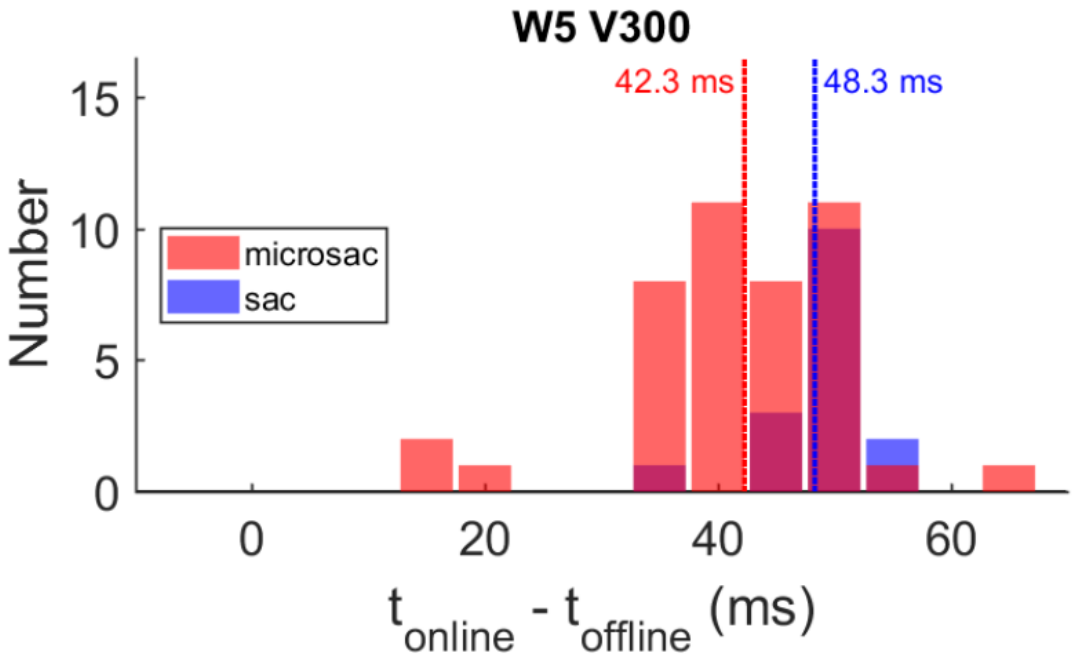
Result

DDPI

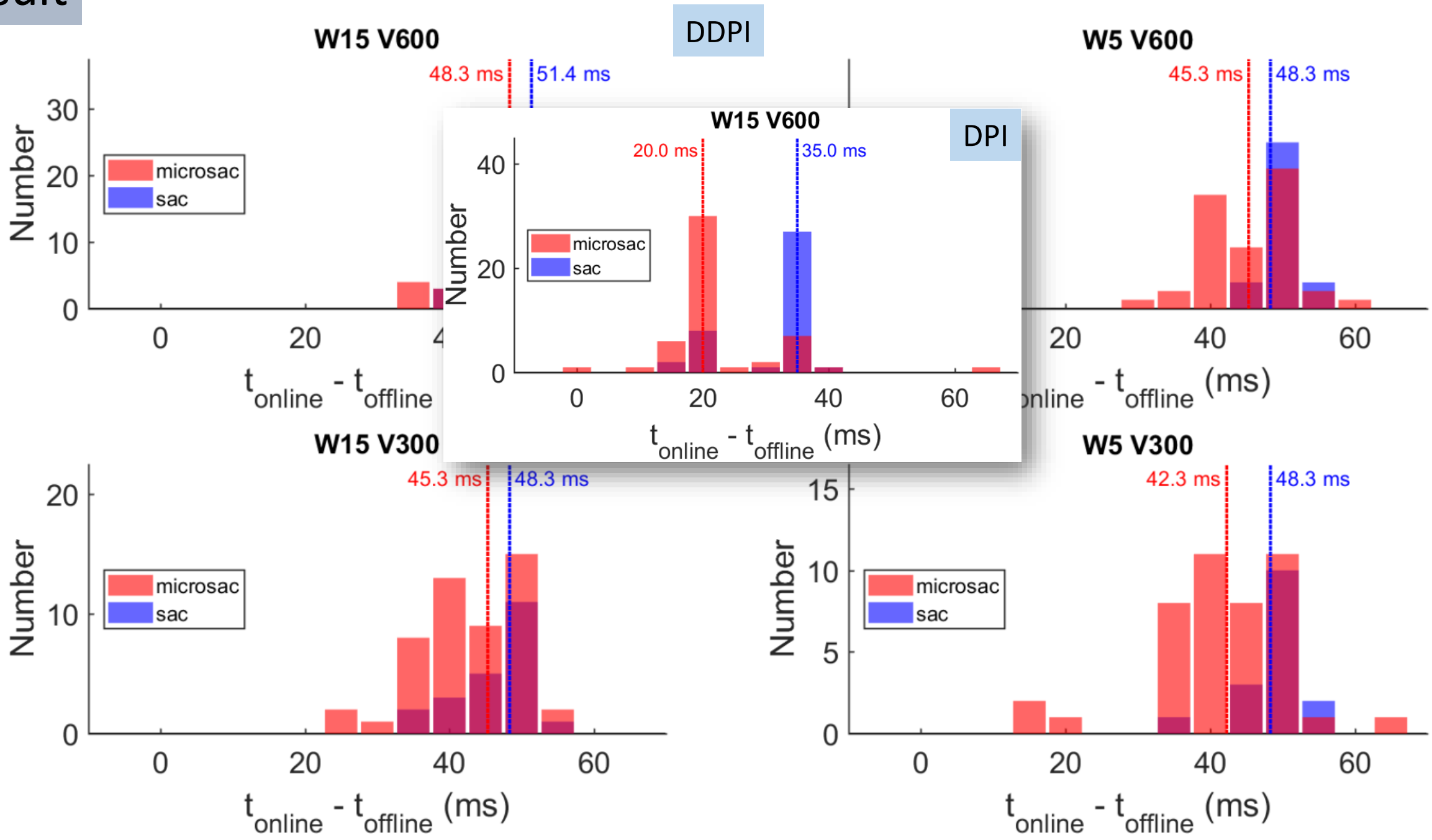

W15 V600
48.3 ms
51.4 ms
microsac
sac
Number
0
10
20
30
0
20
40
60
t online - t offline (ms)


W5 V600
45.3 ms
48.3 ms
microsac
sac
Number
0
10
20
0
20
40
60
t online - t offline (ms)

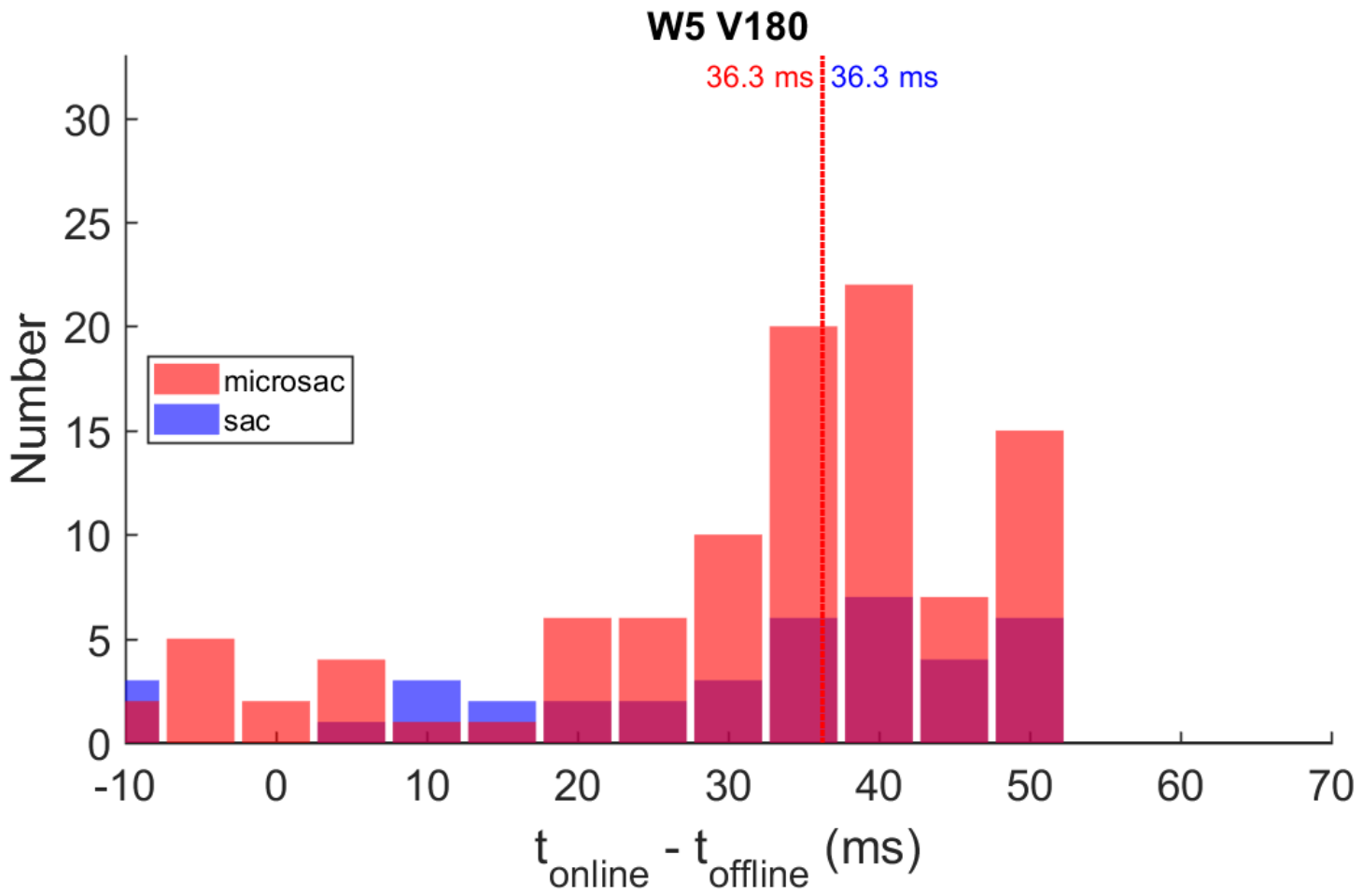

W15 V300
45.3 ms
48.3 ms
microsac
sac
Number
0
10
20
0
20
40
60
t online - t offline (ms)


W5 V300
42.3 ms
48.3 ms
microsac
sac
Number
0
5
10
15
0
20
40
60
t online - t offline (ms)

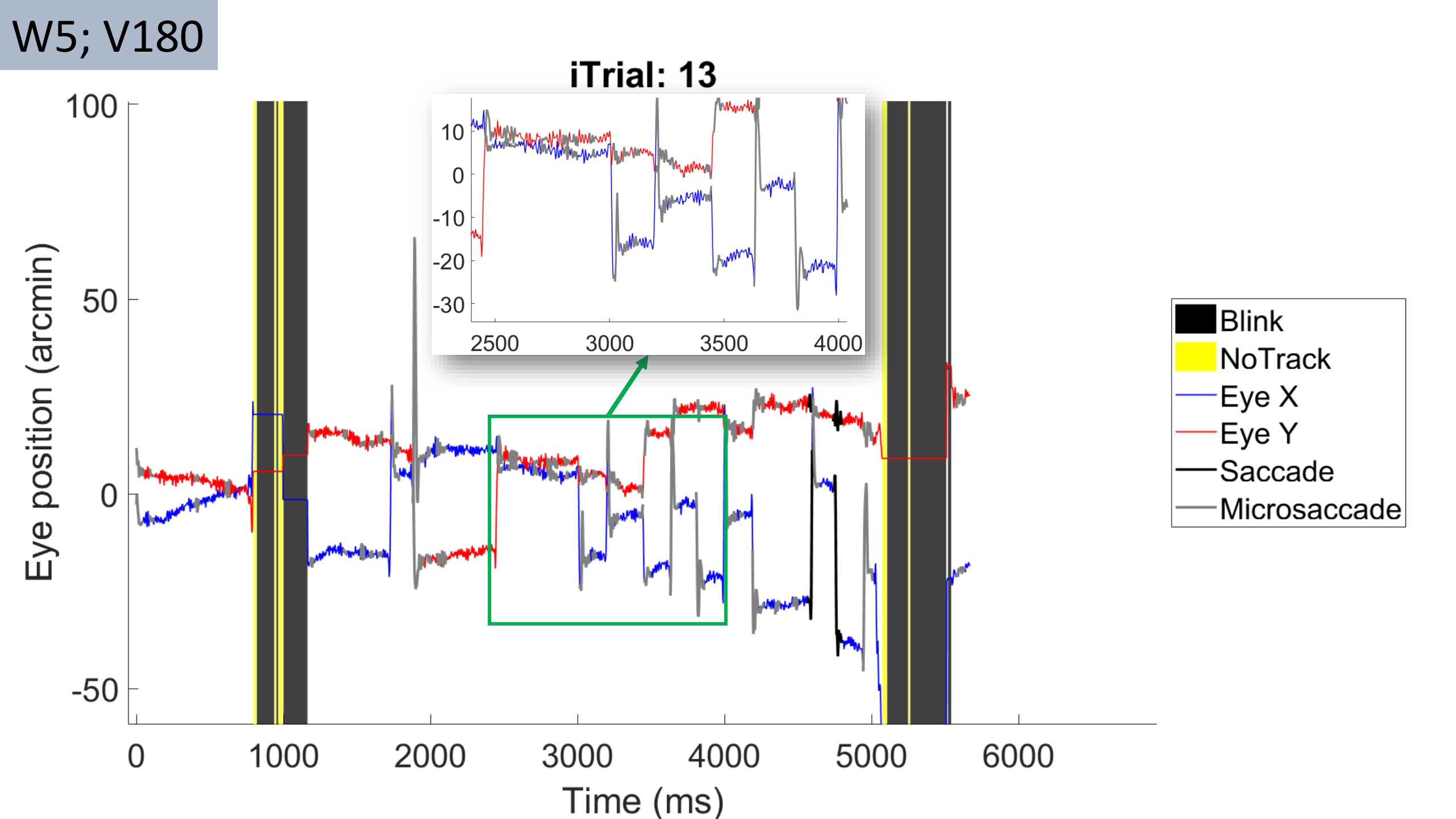
Result


DDPI
W15 V600
48.3 ms
51.4 ms
microsac
sac
Number
$t_{online} - t_{offline}$ (ms)
W5 V600
45.3 ms
48.3 ms
DPI
W15 V600
20.0 ms
35.0 ms
W15 V300
45.3 ms
48.3 ms
W5 V300
42.3 ms
48.3 ms

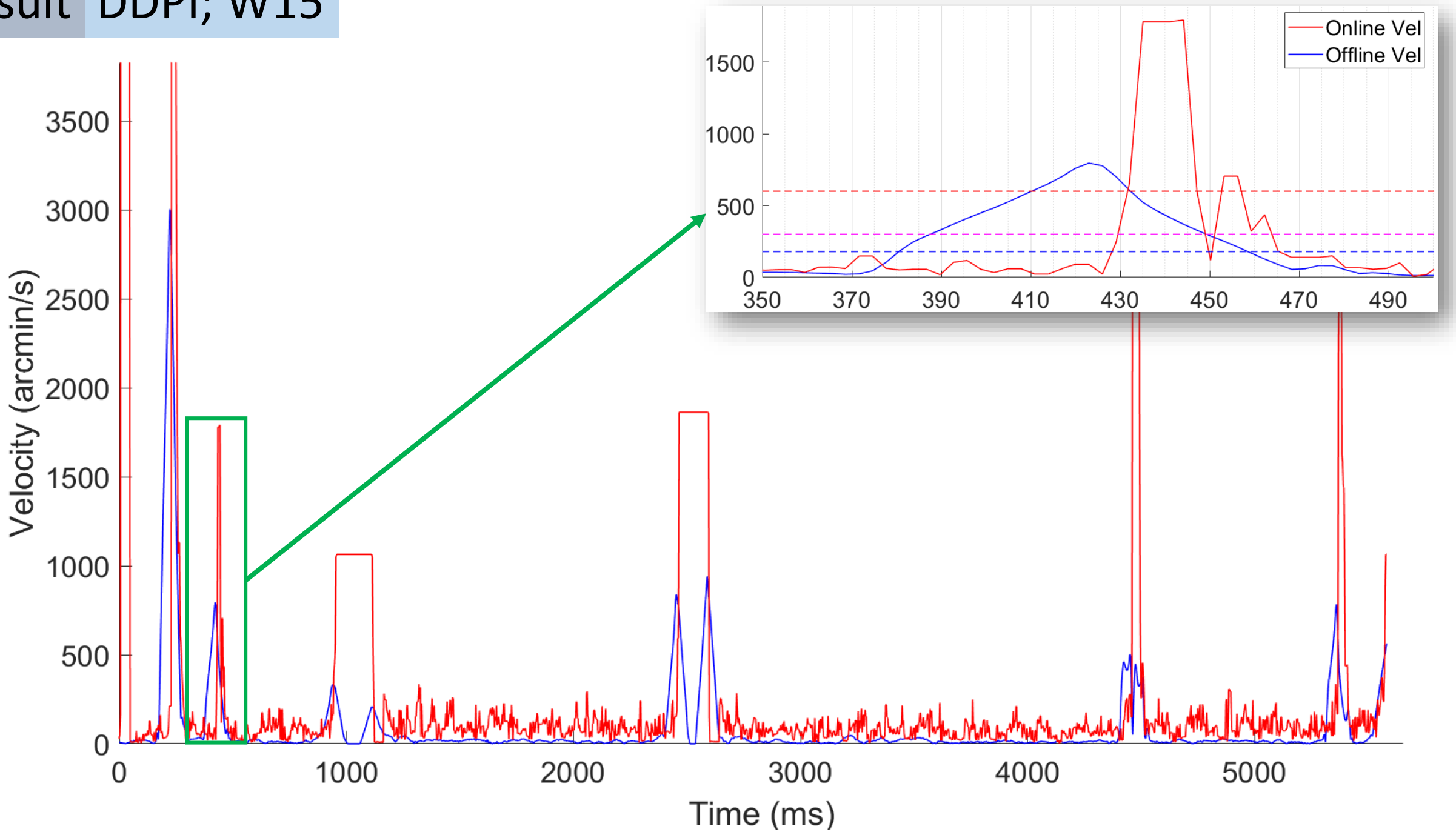
Result


W5 V180
36.3 ms
36.3 ms
Number
microsac
sac
0
5
10
15
20
25
30
-10
0
10
20
30
40
50
60
70
t online - t offline (ms)

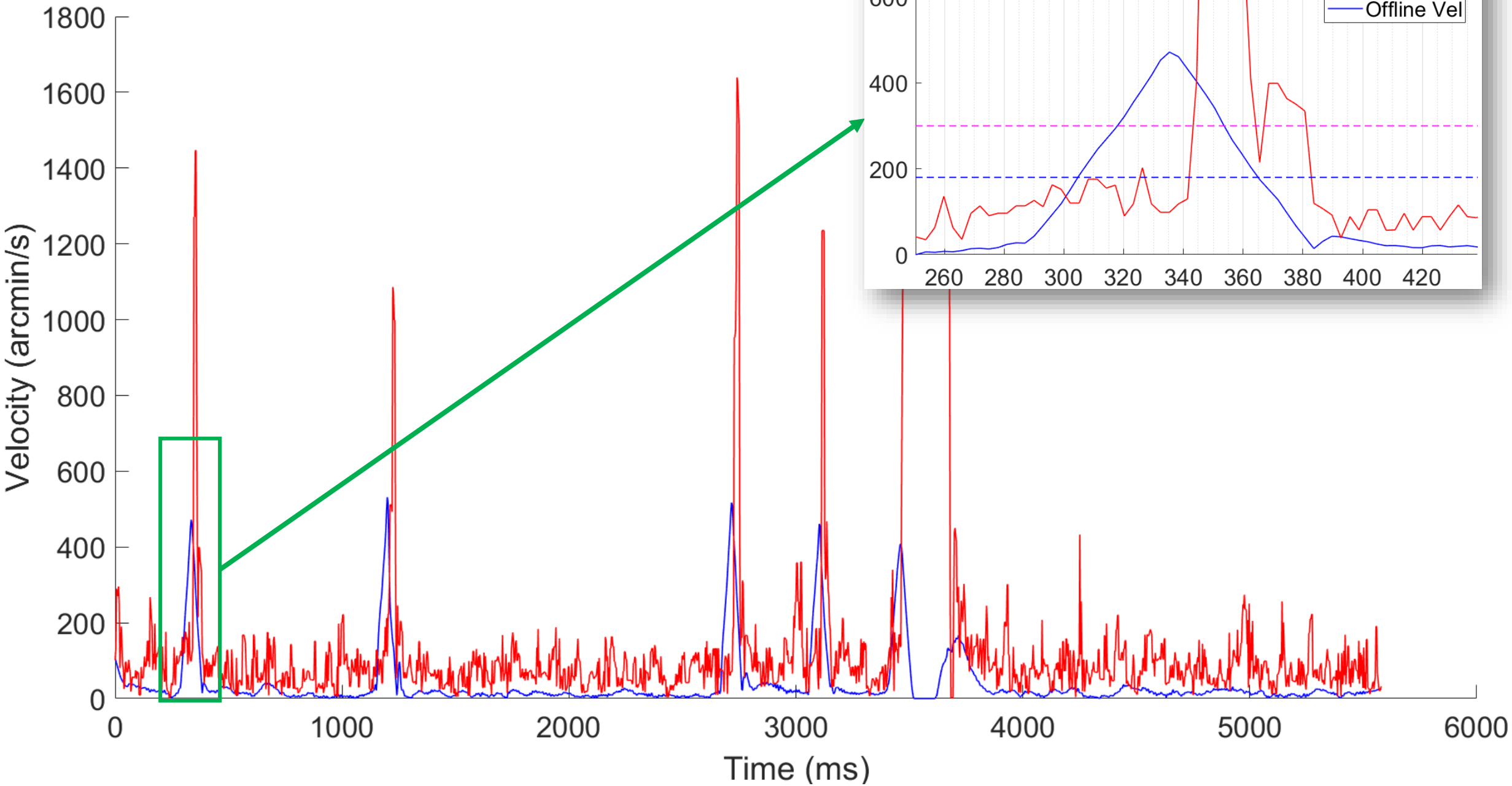
W5; V180


iTrial: 13
100
50
0
-50
Eye position (arcmin)
0
1000
2000
3000
4000
5000
6000
Time (ms)
10
0
-10
-20
-30
2500
3000
3500
4000
Blink
NoTrack
Eye X
Eye Y
Saccade
Microsaccade

Result DDPI; W15



Result DDPI; W5



Code

CAWDerivative.hpp | CAWDerivative.cpp

```
1  #pragma once
2
3  /// Adaptive Windowing Derivative
4  /** This class implements the adaptive windowing derivative algorithm introduced
5        in F. Janabi-Sharifi, V. Hayward, and C.J. Chen , "Discrete-Time Adaptive
6          Windowing for Velocity Estimation", IEEE Transactions on Control Systems Technology
7          8(6), 1003-1009, 2000. */
8
9  #include "CMedianFilter.hpp"
10
11 class CAWDerivative
12 {
13 public:
14     // Dimension of the estimation window (in samples)
15      enum {
16          MAX_WINDOW_LENGTH = 5
17      };
```

```
32     /// Set the sampling interval
33     void setT(double T)
34     {
35         m_T = T;
36     }
```

Code

CStage_EMTM2.hpp

```
1  #pragma once
2  #include "CStage.hpp"
3  #include "CAWDerivative.hpp"
4
5  /// Stage for the classification of eye movements (Eye Movements Tagging Module)
6  /** This class implements a real-time monocular tagging of eye movements.  Currently, the algorithm can
7  differentiate between saccades, microsaccades, and drifts. */
8
9  class CStage_EMTM2 : public CStage
```

CStage_EMTM2.cpp

```
72 ////////////////////////////////////////////////////////////////////////////////
73 // Process and return the output for the next pipeline stage
74 void CStage_EMTM2::process(Uint32 SampleCounter, DDPIData &dataStream)
75 {
```

```
78     // If the stage is disabled, just return
79   if (!_enabled) return;
```

```
 98     _vx = _xVelocity.estimate(SampleCounter, _x);
 99     _vy = _yVelocity.estimate(SampleCounter, _y);
100     _v = sqrt(_vx * _vx + _vy * _vy);
```

```
185         case STATE_EM:
186
187             // Check if the velocity of the eye movement is lower
188             // than the set threshold and that the event is
189             // longer than the minimum accepted length
190             if (fabsf(_v) < _eventOffThreshold &&
191                 (SampleCounter - _t0)/_samRate*1000 > _eventMinDuration) {        //  ******yb.2018.10
```

Code

CStage_EMTM2.hpp × CStage_EMTM2.cpp ×

```
232  // Perform a reset the pipeline stage
233  void CStage_EMTM2::setParameters(
234      float SamRate,  //  ******yb.2018.10******
235      Uint32 EventMinDuration, float EventOnThreshold, float EventOffThreshold, Uint32 EventIsteresis,
236      Uint32 BNTDroppedSamples, bool BNTWaitForV,
237      float SaccadeThreshold,
238      float LfixThreshold)
239  {
240      std::lock_guard<std::mutex> lk(_mtx);
241
242      _samRate = SamRate;      //  ******yb.2018.10******
243      _eventMinDuration = EventMinDuration;
244      _eventOnThreshold = EventOnThreshold;
245      _eventOffThreshold = EventOffThreshold;
```

Code

CStage_EMTM2.hpp × CStage_EMTM2.cpp ×

```
255 // Perform a reset the pipeline stage
256 void CStage_EMTM2::setVelocity(
257     int XNoiseThreshold,
258     float XT,
259     bool XFilterEnabled,
260     int YNoiseThreshold,
261     float YT,
262     bool YFilterEnabled)
263 {
264     std::lock_guard<std::mutex> lk(_mtx);
265 
266     _xVelocity.setNoiseThreshold(XNoiseThreshold);
267     _xVelocity.setT(XT);
268     _xVelocity.enableFilter(XFilterEnabled);
269 
270     _yVelocity.setNoiseThreshold(YNoiseThreshold);
271     _yVelocity.setT(YT);
272     _yVelocity.enableFilter(YFilterEnabled);
273 }
```

Code

CDriver_FrameGrabberBase.cpp

```
51  void CDriver_FrameGrabberBase::EMTM_setParameters(
52      float SamRate,  //  ******yb.2018.10******
53      Uint32 EventMinDuration, float EventOnThreshold, float EventOffThreshold, Uint32 EventIsteresis,
54      Uint32 BNTDroppedSamples, bool BNTWaitForV,
55      float SaccadeThreshold,
56      float LfixThreshold)
57  {
58      _emtm.setParameters(
59          SamRate,    //  ******yb.2018.10******
60          EventMinDuration, EventOnThreshold, EventOffThreshold, EventIsteresis,
61          BNTDroppedSamples, BNTWaitForV,
62          SaccadeThreshold,
63          LfixThreshold);
64  }
65  ////////////////////////////////////////////////////////////////////////////////
66  void CDriver_FrameGrabberBase::EMTM_setVelocity(
67      int XNoiseThreshold, float XT, bool XFilterEnabled,
68      int YNoiseThreshold, float YT, bool YFilterEnabled)
69  {
70      _emtm.setVelocity(
71          XNoiseThreshold, XT, XFilterEnabled,
72          YNoiseThreshold, YT, YFilterEnabled);
73  }
```

Code

CEOS.cpp

```
31  void CEOS::_configureEMTM()
32  {
33      float SamRate = CEnvVariables::Instance()->getInteger(CFG_N_SAMPLING_RATE);      //  ******yb.2018.
34      int EventMinDuration = CEnvVariables::Instance()->getInteger(CFG_N_EVENT_MINDURATION);
35      float EventOnThreshold = CEnvVariables::Instance()->getFloat(CFG_N_EVENT_ONTHRESHOLD);
36      float EventOffThreshold = CEnvVariables::Instance()->getFloat(CFG_N_EVENT_OFFTHRESHOLD);
37      int EventIsteresis = CEnvVariables::Instance()->getInteger(CFG_N_EVENT_ISTERESIS);
38      int BNTDroppedSamples = CEnvVariables::Instance()->getInteger(CFG_N_BNT_DROPPEDSAMPLES);
39      bool BNTWaitForV = CEnvVariables::Instance()->getBoolean(CFG_N_BNT_WAITFORV);
40      float SaccadeThreshold = CEnvVariables::Instance()->getFloat(CFG_N_SACCADE_THRESHOLD);
41      float LfixThreshold = CEnvVariables::Instance()->getFloat(CFG_N_LFIX_THRESHOLD);
42
43      EMTM_setParameters(
44          SamRate,    // ******yb.2018.10******
45          EventMinDuration,
46          EventOnThreshold, EventOffThreshold,
47          EventIsteresis,
48          BNTDroppedSamples, BNTWaitForV,
49          SaccadeThreshold, LfixThreshold);
50
51      int XNoiseThreshold = CEnvVariables::Instance()->getInteger(CFG_N_X_NOISE_THRESHOLD);
52      float XT = CEnvVariables::Instance()->getFloat(CFG_N_X_T);
53      bool XFilterEnabled = CEnvVariables::Instance()->getBoolean(CFG_N_X_ENABLE_FILTER);
54      int YNoiseThreshold = CEnvVariables::Instance()->getInteger(CFG_N_Y_NOISE_THRESHOLD);
55      float YT = CEnvVariables::Instance()->getFloat(CFG_N_Y_T);
56      bool YFilterEnabled = CEnvVariables::Instance()->getBoolean(CFG_N_Y_ENABLE_FILTER);
57
58      EMTM_setVelocity(XNoiseThreshold, XT, XFilterEnabled, YNoiseThreshold, YT, YFilterEnabled);
59  }
```

Code

CEOS.cpp

```
119 void CEOS::initialize()
120 {
121     if (CDriver_MIL::Instance()->init() && CDriver_MIL::Instance()->foundCamera())
122     {
123         m_linkMode = CL_MODE_ACTIVE;
124     }
125     else
126     {
127         m_linkMode = CL_MODE_INACTIVE;
128     }
129     _configureEMTM();       // To make parameters for online eye movements detection in emil-library.cfg
130                             //      Call CEOS::initialize() before CEnvVariables::Instance()->loadFile()
131                             //                          ******yb.2018.10******
132 }
```

Code

emil-library.cfg

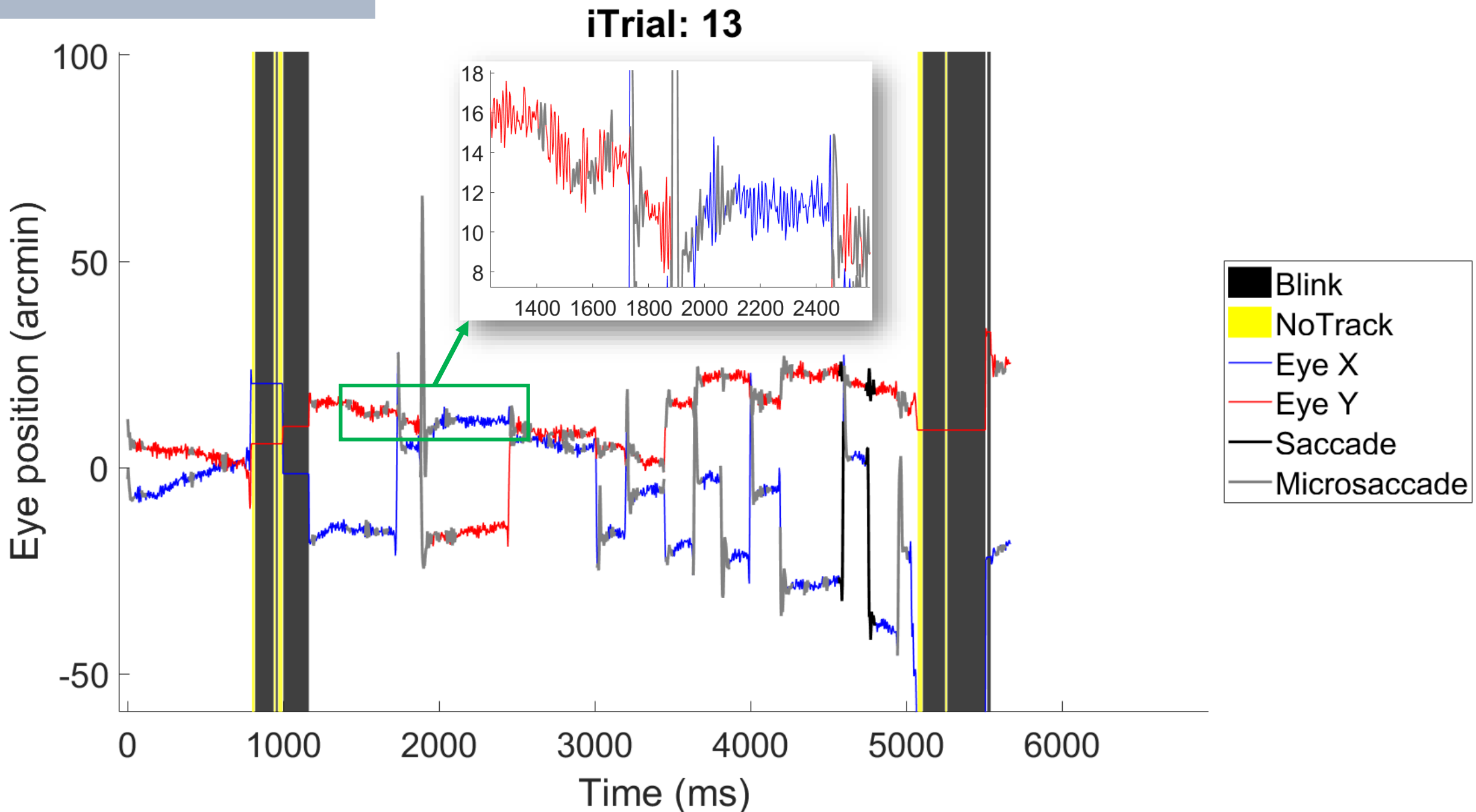
```
183  Sampling-Rate = 331
184
185  Event-MinDuration = 15
186  Event-OnThreshold = 600.0
187  Event-OffThreshold = 300.0
188  Event-Isteresis = 5
189  Bnt-DroppedSamples = 150
190  Bnt-WaitForV = true
191  SaccadeThreshold = 30.0
192  LFixThreshold = 10.0
193
194  ##########################################################
195  # Velocity estimation parameters
196  #
197  # WARNING: DO NOT attempt to change the parameters below unless
198  # you know exactly what you are doing.
199  #
200
201  # XNoiseThreshold = 4.0
202  # XT = 0.001 ms
203  # XEnableFilter = true
204  # YNoiseThreshold = 4.0
205  # YT = 0.001 ms
206  # YEnableFilter = true
207
208  XNoiseThreshold = 4.0
209  XT = 0.00302
210  XEnableFilter = true
211  YNoiseThreshold = 4.0
212  YT = 0.00302
213  YEnableFilter = true
```

Code

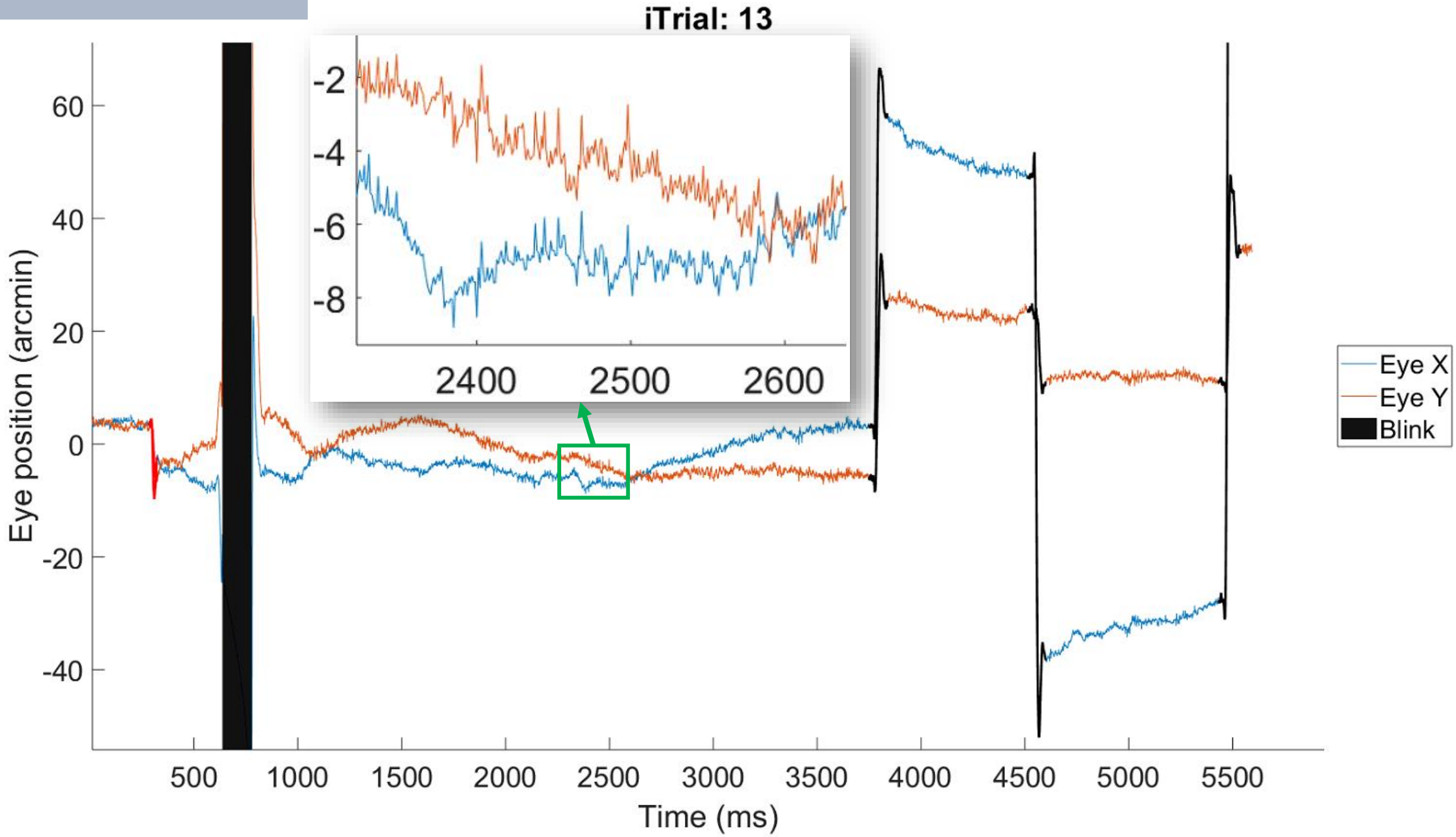
CDriver_FrameGrabberBase.cpp

```
391  void CDriver_FrameGrabberBase::switchMode(int mode)
392  {
393      switch (mode)
394      {
416      case MODE_IDLE_CAL_SAVE:
417          _emtm.enable(/*false*/true);    // activate emtm here.  ******yb.2018.10******
418          _vatm.enable(true);
419          _recording = true;
420          enableSaveImage(true);
421          _mode = mode;
422          break;
441      case MODE_REC_CAL:
442          // Activate/Deactivate stages
443          _emtm.enable(/*false*/true);    // activate emtm here.  ******yb.2018.10******
444          _vatm.enable(true);
445          _recording = true;
446          enableSaveImage(false);
447          // Prepare the class for a new recording
448          _mode = mode;
449          break;
```

DDPI: Noise Level



DDI: Noise Level



Noise Level: DPI VS DDPI