```


    typedef double XFLOAT
    typedef double OTA_FLOAT

using std::string


{

SegListTraversal::SegListTraversal(SQTimeAlignment   const *timeAlignmentInfo,
                                   SQTA_ResampResult const *resamplingInfo,
                                   SQSignal const *refSig, SQSignal const *degSig,
                                   int frameSize, int frameStep)
: mTAinfo               (NULL),
  mResampInfo           (NULL),
  mDetailedSegList      (NULL),
  mSegList              (NULL),
  mUnusedDegSegments    (NULL),
  mRef                  (NULL),
  mDeg                  (NULL),
  mSignalPosRef         (-1),
  mSignalPosDeg         (-1),
  mTASearch_bestFrmPos  (-1),
  mJumpedToNewSeg       (false),
  mSegType              (TA_SEG_PAUSE),
  mRefMissingInDeg      (false),
  mDegMissingInRef      (false)
{
    OPTTRY
    {
        if (timeAlignmentInfo == NULL || resamplingInfo == NULL || refSig == NULL ||
            degSig == NULL || frameSize < 2 || frameStep < 1 || frameStep > frameSize ||
            refSig->SamplingFreq() != degSig->SamplingFreq())
            OPTTHROW ((string("ERROR in SegListTraversal: Invalid constructor arguments.\n")))

        if (timeAlignmentInfo->MergedSegments() == NULL ||
            timeAlignmentInfo->UnusedDegSegments() == NULL ||
            timeAlignmentInfo->MergedSegments()->size() < 1)
            OPTTHROW ((string("ERROR in SegListTraversal: SQTimeAlignment object contains invalid
data.\n")))

        sqStorage = new SQStorage()

        //Init time alignment member vars
        mTAinfo            = timeAlignmentInfo
        mResampInfo        = resamplingInfo
        mDetailedSegList   = timeAlignmentInfo->Segments()
        mSegList           = new TA_SegList(*timeAlignmentInfo->MergedSegments())
        mUnusedDegSegments = new TA_SegList(*timeAlignmentInfo->UnusedDegSegments())

        //Init signal and frame member vars
        mRef = refSig
        mDeg = degSig
        mFrameSize = frameSize
        mFrameStep = frameStep

        //Determine the beginning and end positions for time alignment traversal

        mStartRefPos = 0
        mEndRefPos   = mRef->NrOfSamples()-1

        PreprocessSegList()
        FindStartingSegments()
    }
    OPTCATCH( (...))
    {
        delete mSegList
```

```
        delete mUnusedDegSegments
        sqStorage->ClearAllItemsWhoseIDStartsWith("SegListTraversal_")
        OPTTHROW ((string("ERROR in SegListTraversal constructor.\n")))
    }
}
SegListTraversal::~SegListTraversal()
{
    delete mSegList
    delete mUnusedDegSegments
    sqStorage->ClearAllItemsWhoseIDStartsWith("SegListTraversal_")
    delete sqStorage

    sqStorage = 0
    mSegList = mUnusedDegSegments = NULL
}

bool SegListTraversal::JumpedToNewSeg() const
{
    return mJumpedToNewSeg
}

int SegListTraversal::SignalPosRef() const
{
    return mFinalSignalPosRef
}

int SegListTraversal::SignalPosDeg() const
{
    return mFinalSignalPosDeg
}

XFLOAT SegListTraversal::SegReliability() const
{
    return mSegReliability
}

TA_SEG_TYPE SegListTraversal::SegType() const
{
    return mSegType
}

bool SegListTraversal::RefMissingInDeg() const
{
    return mRefMissingInDeg
}

bool SegListTraversal::DegMissingInRef() const
{
    return mDegMissingInRef
}

XFLOAT SegListTraversal::OverallMatchQuality() const
{
    return mTAinfo->MatchQuality()
}

XFLOAT SegListTraversal::ResamplingFac() const
{
    return mResampInfo->fMeanResamplingFac
}

void SegListTraversal::MoveToNextFramePair()
{
    if (!mRefMissingInDeg)
        mSignalPosDeg += mFrameStep

    else
        mSignalPosRef += mFrameStep
```

```
136 }
137
138 int SegListTraversal::FullTraversal(int &curRefPos,    int &curDegPos,
139                                     int &minSearchPos, int &maxSearchPos)
140 {
141     OPTTRY
142     {
143         int const minContFrmLen
144             = (int)(mFrameSize - mFrameStep/2)
145
146         //Have we reached the end of the current segment? -> find the next valid one.
147         while (mCurSegs[mActiveList] != mEndOfLists[mActiveList] &&
148
149                (SkipThisMissingSegment(mCurSegs[mActiveList], mActiveList) ||
150                 mSignalPosDeg + minContFrmLen >
151                  mCurSegs[mActiveList]->degPos + mCurSegs[mActiveList]->segLen ||
152                 (mCurSegs[mActiveList]->segLen < mFrameSize &&
153                  mCurSegs[mActiveList]->segType != TA_SEG_MISSING) ||
154                 (mRefMissingInDeg &&
155                  mSignalPosRef + minContFrmLen > mCurSegs[mActiveList]->refPos +
    mCurSegs[mActiveList]->segLen)))
156             mCurSegs[mActiveList]++
157
158         //Leave if both iterators are the list ends or beyond the active signal.
159         if ((mCurSegs[UNUSED_DEG_LIST] == mEndOfLists[UNUSED_DEG_LIST] ||
160              mSignalPosDeg + minContFrmLen >
161               mEndRefPos + mCurSegs[UNUSED_DEG_LIST]->degPos - mCurSegs[UNUSED_DEG_LIST]->refPos)
162              &&
163             (mCurSegs[SEGLIST] == mEndOfLists[SEGLIST] ||
164              mSignalPosDeg + minContFrmLen >
165               mEndRefPos + mCurSegs[SEGLIST]->degPos - mCurSegs[SEGLIST]->refPos))
166             return LEAVE_LOOP
167
168         //Select which segment of the two lists to use.
169         while (mCurSegs[SEGLIST] != mEndOfLists[SEGLIST] &&
170                mCurSegs[UNUSED_DEG_LIST] != mEndOfLists[UNUSED_DEG_LIST])
171         {
172             if (mCurSegs[SEGLIST]->degPos >= mCurSegs[UNUSED_DEG_LIST]->degPos)
173                 if (SkipThisMissingSegment(mCurSegs[UNUSED_DEG_LIST], UNUSED_DEG_LIST) ||
174                     mCurSegs[UNUSED_DEG_LIST]->degPos + mCurSegs[UNUSED_DEG_LIST]->segLen <=
    mSignalPosDeg)
175                     mCurSegs[UNUSED_DEG_LIST]++
176                 else
177                     { mActiveList = UNUSED_DEG_LIST  break  }
178             else
179                 if (SkipThisMissingSegment(mCurSegs[SEGLIST], SEGLIST) ||
180                     (mCurSegs[SEGLIST]->segLen  <  mFrameSize &&
181                      mCurSegs[SEGLIST]->segType != TA_SEG_MISSING) ||
182                     mCurSegs[SEGLIST]->degPos + mCurSegs[SEGLIST]->segLen <= mSignalPosDeg)
183                     mCurSegs[SEGLIST]++
184                 else
185                     { mActiveList = SEGLIST  break  }
186         }
187
188         if (mCurSegs[mActiveList] == mEndOfLists[mActiveList])
189         {
190             mActiveList = (mActiveList+1)%2
191             while (mCurSegs[mActiveList] != mEndOfLists[mActiveList] &&
192                    (SkipThisMissingSegment(mCurSegs[mActiveList], mActiveList) ||
193                     (mCurSegs[mActiveList]->segLen  <  mFrameSize &&
194                      mCurSegs[mActiveList]->segType != TA_SEG_MISSING)))
195                 mCurSegs[mActiveList]++
196             if (mCurSegs[mActiveList] == mEndOfLists[mActiveList])
197                 return LEAVE_LOOP
198         }
199
200         //Get current position in the deg signal, and associated segment info.
```

```
201         if (mSignalPosDeg < 0)
202             if (mActiveList == SEGLIST)
203                 mSignalPosDeg = mStartRefPos + mCurSegs[mActiveList]->degPos -
    mCurSegs[mActiveList]->refPos
204             else
205                 mSignalPosDeg = mCurSegs[mActiveList]->degPos
206
207         mSegType = mCurSegs[mActiveList]->segType
208         if ((mSegType == TA_SEG_MISSING && mActiveList == SEGLIST) &&
209             !mRefMissingInDeg)
210         {
211             mSignalPosRef = mCurSegs[mActiveList]->refPos
212             mTASearch_bestFrmPos = -100000
213         }
214         else if (!(mSegType == TA_SEG_MISSING && mActiveList == SEGLIST) &&
215                  mRefMissingInDeg)
216             mTASearch_bestFrmPos = -100000
217
218         mRefMissingInDeg = mSegType == TA_SEG_MISSING && mActiveList == SEGLIST
219         mDegMissingInRef = mActiveList == UNUSED_DEG_LIST
220
221         if (mSignalPosDeg < mCurSegs[mActiveList]->degPos)
222         {
223             mSignalPosDeg   = mCurSegs[mActiveList]->degPos
224             mJumpedToNewSeg = true
225         }
226         else
227             mJumpedToNewSeg = false
228
229         //Handle segments with imprecise matching of ref and deg signal,
230         //e.g. missing or additional utterances, or matching based on guess.
231         bool handledRefFrame = false, handledDegFrame = false
232         minSearchPos = maxSearchPos = -1
233         if (mSegType == TA_SEG_MISSING)
234             HandleMissingSegment(mTASearch_bestFrmPos,
235                                  minSearchPos, maxSearchPos,
236                                  handledRefFrame, handledDegFrame)
237         else if (mSegType != TA_SEG_MATCHED)
238             HandleGuessedSegment(mTASearch_bestFrmPos,
239                                  minSearchPos, maxSearchPos,
240                                  handledRefFrame, handledDegFrame)
241         else
242             mSegReliability = 1.0f
243
244         if (!handledRefFrame && !mRefMissingInDeg)
245             mSignalPosRef = mSignalPosDeg -
246                 (mCurSegs[mActiveList]->degPos - mCurSegs[mActiveList]->refPos)
247
248         //Now determine the remaining ref/deg frame position, depending
249         //on the value of the handledRefFrame/handledDegFrame flags.
250
251         //Reached last frame of current segment -> allow a smaller
252         //frame step than mFrameStep for the last frame.
253         if (mActiveList == SEGLIST && !handledRefFrame &&
254             mSignalPosRef + mFrameSize > mCurSegs[mActiveList]->refPos +
    mCurSegs[mActiveList]->segLen)
255         {
256             int lastFramePos = max(mCurSegs[mActiveList]->refPos + mCurSegs[mActiveList]->segLen
257                                    - mFrameSize, 0)
258
259             handledRefFrame = true
260             mFinalSignalPosRef = lastFramePos
261             if (!handledDegFrame)
262             {
263                 int curShift = limit(mCurSegs[mActiveList]->degPos - mCurSegs[mActiveList]->refPos,
264                                      0 - lastFramePos,
    (int)mDeg->NrOfSamples()-lastFramePos-mFrameSize)
265                 handledDegFrame = true
```

```
266                 mFinalSignalPosDeg = lastFramePos + curShift
267             }
268
269             mCurSegs[mActiveList]++
270         }
271         else if (mActiveList == UNUSED_DEG_LIST &&
272                  mSignalPosDeg + mFrameSize > mCurSegs[mActiveList]->degPos +
    mCurSegs[mActiveList]->segLen)
273         {
274             int lastFramePos = max(mCurSegs[mActiveList]->degPos + mCurSegs[mActiveList]->segLen
275                                    - mFrameSize, 0)
276
277             if (!handledDegFrame)
278             {
279                 handledDegFrame = true
280                 mFinalSignalPosDeg = lastFramePos
281             }
282
283             mCurSegs[mActiveList]++
284
285             if (!handledRefFrame)
286                 OPTTHROW(( string("HandleMissingSegment did not copy the ref segment!")))
287         }
288         else //'standard case'
289         {
290             if (mActiveList == SEGLIST)
291             {
292                 if (!handledRefFrame)
293                 {
294                     handledRefFrame = true
295                     mFinalSignalPosRef = mSignalPosRef
296                 }
297                 if (!handledDegFrame &&
298                     mSignalPosDeg + mFrameSize > mCurSegs[mActiveList]->degPos +
    mCurSegs[mActiveList]->segLen)
299                 {
300                     int lastFramePos = max(mCurSegs[mActiveList]->degPos +
    mCurSegs[mActiveList]->segLen
301                                            - mFrameSize, 0)
302                     handledDegFrame = true
303                     mFinalSignalPosDeg = lastFramePos
304                     mCurSegs[mActiveList]++
305                 }
306                 else if (!handledDegFrame)
307                 {
308                     handledDegFrame = true
309                     mFinalSignalPosDeg = mSignalPosDeg
310                 }
311
312             }
313             else if (!handledDegFrame)
314             {
315                 handledDegFrame = true
316                 mFinalSignalPosDeg = mSignalPosDeg
317             }
318         }
319
320         curRefPos = mFinalSignalPosRef
321         curDegPos = mFinalSignalPosDeg
322         if (minSearchPos < 0)
323         {
324             if (mSegType == TA_SEG_MISSING)
325                 OPTTHROW(( string("Min search position not set inside HandleMissingSegment(). This
    should never happen!")))
326             minSearchPos = mFinalSignalPosRef
327         }
328         if (maxSearchPos < 0)
329         {
```

```
330             if (mSegType == TA_SEG_MISSING)
331                 OPTTHROW(( string("Max search position not set insidTHROW((dleMissingSegment(). This
    should never happen!")))
332             maxSearchPos = mFinalSignalPosRef
333         }
334
335         return SQ_NO_ERRORS
336     }
337
338     OPTCATCH((string errorMsg))
339     {
340         OPTTHROW(( string("ERROR in FullTraversal: " + errorMsg + "\n")))
341     }
342 }
343
344 void SegListTraversal::PreprocessSegList()
345 {
346     if (mSegList == NULL || mSegList->size() == 0 || mFrameSize < 2 ||
347         mDeg == NULL || mDeg->NrOfSamples() < mFrameSize)
348         OPTTHROW(( string("ERROR in PreprocessSegList: Invalid segList, frame size or deg
    signal.\n")))
349
350     //Set the most probable deg frame starting position for all ref frames missing in the deg signal
351     for (int i = 0  i < (int)mSegList->size()  i++)
352     {
353         if ((*mSegList)[i].segType == TA_SEG_MISSING)
354         {
355             (*mSegList)[i].degPos += (*mSegList)[i].segLen/2 - mFrameSize/2
356             (*mSegList)[i].degPos =  limit((*mSegList)[i].degPos, 0,
    (int)mDeg->NrOfSamples()-mFrameSize)
357         }
358     }
359 }
360
361 void SegListTraversal::FindStartingSegments()
362 {
363     //Position ourselves at the beginning of the segments lists
364     mCurSegs   [SEGLIST]         = mSegList->begin()
365     mCurSegs   [UNUSED_DEG_LIST] = mUnusedDegSegments->begin()
366     mEndOfLists[SEGLIST]         = mSegList->end()
367     mEndOfLists[UNUSED_DEG_LIST] = mUnusedDegSegments->end()
368     mActiveList                  = SEGLIST
369
370     //Find starting segment
371     while(mCurSegs[SEGLIST] != mEndOfLists[SEGLIST] &&
372           ((mCurSegs[SEGLIST]->segType != TA_SEG_MISSING &&
373             mCurSegs[SEGLIST]->segLen   < mFrameSize) ||
374            mCurSegs[SEGLIST]->refPos + mCurSegs[SEGLIST]->segLen <= mStartRefPos))
375         mCurSegs[SEGLIST]++
376
377     //Find first valid segment
378     while(mCurSegs[UNUSED_DEG_LIST] != mEndOfLists[UNUSED_DEG_LIST] &&
379           mCurSegs[UNUSED_DEG_LIST]->refPos + mCurSegs[UNUSED_DEG_LIST]->segLen <= mStartRefPos)
380         mCurSegs[UNUSED_DEG_LIST]++
381 }
382
383 struct SkipThisMissingSegmentStruct:SQStorageSkeletonClass
384 {
385     int minSegLenRef
386     int minSegLenDeg
387     int firstSpeechActRef
388     int firstSpeechActDeg
389     TA_SegList::const_iterator prevSegRef
390     TA_SegList::const_iterator prevSegDeg
391
392     SkipThisMissingSegmentStruct(int argNum, va_list *argList)
393     {
394         if (argNum != 0)
```

```
395             OPTTHROW(( string("ERROR in SkipThisMissingSegmentStrTHROW((onstructor: Invalid
    arguments.\n")))
396
397         minSegLenRef = 0
398         minSegLenDeg = 0
399
400         firstSpeechActRef = -1
401         firstSpeechActDeg = -1
402     }
403 }
404 bool SegListTraversal::SkipThisMissingSegment(TA_SegList::iterator const seg, int listIdx)
405 {
406     int const STMS_SEGLEN_TOLERANCE =
407         min(round(0.005f * mRef->SamplingFreq()), mFrameSize)
408     int const STMS_MIN_PAUSE_LEN    =
409         round(0.3f*mRef->SamplingFreq())
410     int const STMS_MIN_SEGLEN       =
411         min(round(20e-3f*mRef->SamplingFreq()), mFrameSize)
412
413     SkipThisMissingSegmentStruct *varsToStore = NULL
414     sqStorage->GetOrStore("SegListTraversal_SkipThisMissingSegmentStruct", &varsToStore, 0)
415
416     if (varsToStore->firstSpeechActDeg < 0)
417     {
418         varsToStore->prevSegRef = mSegList->end()
419         varsToStore->prevSegDeg = mUnusedDegSegments->end()
420
421         if (mCurSegs[mActiveList]->segType == TA_SEG_GUESSED ||
422             mCurSegs[mActiveList]->segType == TA_SEG_MATCHED)
423         {
424             varsToStore->firstSpeechActRef = mCurSegs[mActiveList]->refPos
425             varsToStore->firstSpeechActDeg = mCurSegs[mActiveList]->degPos
426         }
427     }
428
429     //Reset counters between sentences
430     if (varsToStore->firstSpeechActDeg >= 0 &&
431         seg->segType == TA_SEG_PAUSE && seg->segLen >= STMS_MIN_PAUSE_LEN)
432     {
433         varsToStore->firstSpeechActRef = seg->refPos + seg->segLen
434         varsToStore->firstSpeechActDeg = seg->degPos + seg->segLen
435         varsToStore->prevSegRef        = mSegList->end()
436         varsToStore->prevSegDeg        = mUnusedDegSegments->end()
437         varsToStore->minSegLenRef = varsToStore->minSegLenDeg = 0
438     }
439
440     if (seg->segType != TA_SEG_MISSING)
441         return false
442
443     if (seg->segLen < STMS_MIN_SEGLEN) //Segment too short to matter at all.
444         return true
445
446     return false
447 }
448
449 void SegListTraversal::HandleMissingSegment(int &TASearch_bestFrmPos,
450                                             int &minSearchPos, int &maxSearchPos,
451                                             bool &handledRefFrame, bool &handledDegFrame)
452 {
453     if (mCurSegs[mActiveList] == mEndOfLists[mActiveList] ||
454         (handledRefFrame && handledDegFrame) || mSegType != TA_SEG_MISSING)
455         OPTTHROW(( string("ERROR in HandleMissingSegment: Invalid input arguments.\n")))
456
457     mSegReliability = 0.0f  //signal parts not occurring in the other signal always have reliability
    0.
458
459     if (mRefMissingInDeg)
460     {
```

```
461         if ((mCurSegs[SEGLIST]+1) != mEndOfLists[SEGLIST]) //haven't reached the end of the deg
    signal
462         {
463             int TASearch_curFrmPos
464 
465             TASearch_curFrmPos  = max(mCurSegs[SEGLIST]->degPos + mFrameSize/2 - mFrameSize,
    TASearch_bestFrmPos)
466             TASearch_curFrmPos  = limit(TASearch_curFrmPos, 0, (int)mDeg->NrOfSamples()-mFrameSize)
467             TASearch_bestFrmPos = TASearch_curFrmPos  //init to first tried position
468 
469             minSearchPos       = TASearch_curFrmPos
470             maxSearchPos       = min((int)mDeg->NrOfSamples()-mFrameSize,
471                                      (mCurSegs[SEGLIST]+1)->degPos)
472             mFinalSignalPosDeg = mCurSegs[SEGLIST]->degPos
473             handledDegFrame = true
474         }
475         else //Ref segment with signal that doesn't occur in deg, at the end of the deg signal:
476         {
477             int degSigPos
478             degSigPos = max(mCurSegs[mActiveList]->degPos, TASearch_bestFrmPos)
479             degSigPos = max(degSigPos, mSignalPosDeg)
480             degSigPos = min(degSigPos, (int)mDeg->NrOfSamples() - mFrameSize)
481 
482             minSearchPos = maxSearchPos = mFinalSignalPosDeg = degSigPos
483             handledDegFrame = true
484         }
485     }
486 
487     else if (mCurSegs[SEGLIST] != mEndOfLists[SEGLIST]) //we're not at the end of the ref signal
488     {
489         int TASearch_curFrmPos
490 
491         maxSearchPos = (int)mRef->NrOfSamples() - mFrameSize
492         int i
493         for (i = 0
494              i < (int)mDetailedSegList->size() &&
495              ((*mDetailedSegList)[i].segType != TA_SEG_MATCHED ||
496               (*mDetailedSegList)[i].degPos   < mSignalPosDeg)
497              i++)
498         if (i != (int)mDetailedSegList->size())
499             if (mDeg->VADprofile()[mDeg->SamplePosToFrameNum(mSignalPosDeg+mFrameSize/2)] ==
    SQ_VAD_NO_SPEECH &&
500                 ((i > 0 && (*mDetailedSegList)[i-1].segType == TA_SEG_PAUSE) ||
501                  (i > 1 && (*mDetailedSegList)[i-2].segType == TA_SEG_PAUSE)))
502                 maxSearchPos = limit((*mDetailedSegList)[i].refPos - mFrameSize,
503                                      mSignalPosRef, maxSearchPos)
504             else
505                 maxSearchPos = min((*mDetailedSegList)[i].refPos,
506                                      maxSearchPos)
507         else
508             maxSearchPos = min(mCurSegs[SEGLIST]->refPos + round(0.200f * mRef->SamplingFreq()),
    //just make sure we don't jump too far
509                                maxSearchPos)
510         if (maxSearchPos < mSignalPosRef)
511             OPTTHROW ((string("ERROR in HandleMissingSegment: Internal error.\n")))
512 
513         TASearch_curFrmPos  = mSignalPosRef
514         TASearch_curFrmPos  = max(mSignalPosRef, TASearch_bestFrmPos)
515         TASearch_curFrmPos  = limit(TASearch_curFrmPos, 0, maxSearchPos)
516         TASearch_bestFrmPos = TASearch_curFrmPos
517 
518         minSearchPos       = TASearch_curFrmPos
519         mFinalSignalPosRef = max(mCurSegs[SEGLIST]->refPos - mFrameSize/2, TASearch_curFrmPos)
520         mSignalPosRef      = TASearch_curFrmPos
521         handledRefFrame = true
522     }
523 
524     //Deg segment with signal that doesn't occur in ref, after the end of the ref signal:
```

```
525     else
526     {
527         int bestMissingFrmPos = mCurSegs[mActiveList]->refPos + mCurSegs[mActiveList]->segLen/2 -
    mFrameSize/2
528         bestMissingFrmPos = max(bestMissingFrmPos, mSignalPosRef)  //Don't go backwards
529         bestMissingFrmPos = min(bestMissingFrmPos, (int)mRef->NrOfSamples()-mFrameSize)
530 
531         minSearchPos = maxSearchPos = bestMissingFrmPos
532         mFinalSignalPosRef = bestMissingFrmPos
533         mSignalPosRef      = bestMissingFrmPos
534         handledRefFrame = true
535     }
536 }
537 
538 void SegListTraversal::HandleGuessedSegment(int &TASearch_bestFrmPos,
539                                             int &minSearchPos, int &maxSearchPos,
540                                             bool &handledRefFrame, bool &handledDegFrame)
541 {
542     if ((handledRefFrame && handledDegFrame) || mActiveList != SEGLIST)
543         OPTTHROW(( string("ERROR in HandleGuessedSegment: Invalid input arguments.\n")))
544 
545     int   const TA_SEARCH_MIN_INCREMENT =
546         mSegType == TA_SEG_PAUSE ? 0 :
547         mFrameSize / 10
548 
549     int minShift, maxShift, guessedShift, minRefPos, maxRefPos, TASearch_curFrmPos
550 
551     //Determine the ref signal range to try out
552     if (!HandleGuessedSegment_determineTAsearchRange(minShift,
553                                                      maxShift,
554                                                      guessedShift,
555                                                      minRefPos,
556                                                      maxRefPos))
557         return
558 
559     TASearch_curFrmPos  = mSignalPosDeg - maxShift
560     TASearch_bestFrmPos = max(minRefPos, TASearch_bestFrmPos + TA_SEARCH_MIN_INCREMENT)
561     TASearch_curFrmPos  = max(TASearch_curFrmPos, TASearch_bestFrmPos)
562     TASearch_curFrmPos  = limit(TASearch_curFrmPos, minRefPos, maxRefPos)
563 
564     minSearchPos       = max(minRefPos, TASearch_bestFrmPos)
565     maxSearchPos       = min(maxRefPos, mSignalPosDeg - minShift)
566     mFinalSignalPosRef = limit(mSignalPosDeg - guessedShift, minSearchPos, maxSearchPos)
567     mSignalPosRef      = TASearch_curFrmPos
568     handledRefFrame    = true
569 }
570 
571 bool SegListTraversal::HandleGuessedSegment_determineTAsearchRange(int &minShift, int &maxShift,
572                                                                    int &guessedShift,
573                                                                    int &minRefPos, int &maxRefPos)
574 {
575     if (mCurSegs[mActiveList] == mEndOfLists[mActiveList] ||
576         mSegType == TA_SEG_MISSING || mSegType == TA_SEG_MATCHED ||
577         mActiveList != SEGLIST || mDetailedSegList == NULL || mDetailedSegList->size() == 0)
578         OPTTHROW(( string("ERROR in HandleGuessedSegment_determineTAsearchRange: Invalid input
    arguments.\n")))
579 
580     int   const MIN_CONT_FRM_LEN        =
581         (int)(mFrameSize - mFrameStep/2)
582     int   const TA_SEARCH_STEP          =
583         mFrameSize / 10
584     int   const MIN_PAUSE_LEN           =
585         round(0.333f*mRef->SamplingFreq())
586 
587     //Try various ref frame positions based on info from adjacent TA_SEG_MATCHED segments.
588 
589     int i
590     TA_segStruct const *nextMatchedSeg = NULL, *prevMatchedSeg = NULL, *curMatchedSeg = NULL
```

```
591     for (i = 0
592          i < (int)mDetailedSegList->size() &&
593          ((*mDetailedSegList)[i].segType != TA_SEG_MATCHED ||
594           (*mDetailedSegList)[i].degPos + (*mDetailedSegList)[i].segLen <= mSignalPosDeg)
595          i++)
596     if (i != (int)mDetailedSegList->size())
597         nextMatchedSeg = &(*mDetailedSegList)[i]
598
599     for (i = i >= 0 ? i-1 : (int)mDetailedSegList->size()-1
600          i >= 0 &&
601          (*mDetailedSegList)[i].segType != TA_SEG_MATCHED
602          i--)
603     if (i >= 0)
604         prevMatchedSeg = &(*mDetailedSegList)[i]
605
606     if (prevMatchedSeg != NULL &&
607         mSignalPosDeg >= prevMatchedSeg->degPos &&
608         mSignalPosDeg <  prevMatchedSeg->degPos + prevMatchedSeg->segLen)
609         OPTTHROW(( string("I thought this never happened?! What the heck!\n")))
610
611     else if (nextMatchedSeg != NULL &&
612         mSignalPosDeg >= nextMatchedSeg->degPos &&
613         mSignalPosDeg <  nextMatchedSeg->degPos + nextMatchedSeg->segLen)
614         curMatchedSeg = nextMatchedSeg
615
616     if (curMatchedSeg != NULL &&
617         mSignalPosDeg                     >= curMatchedSeg->degPos &&
618         mSignalPosDeg + MIN_CONT_FRM_LEN <= curMatchedSeg->degPos + curMatchedSeg->segLen)
619     {
620         mSegReliability = 1.0f
621         return false
622     }
623
624     if (curMatchedSeg != NULL)
625     {
626         prevMatchedSeg = nextMatchedSeg
627         for (i = 0
628              i < (int)mDetailedSegList->size() &&
629              ((*mDetailedSegList)[i].segType != TA_SEG_MATCHED ||
630               (*mDetailedSegList)[i].degPos  <  prevMatchedSeg->degPos + prevMatchedSeg->segLen)
631              i++)
632         if (i != (int)mDetailedSegList->size())
633             nextMatchedSeg = &(*mDetailedSegList)[i]
634         else
635             nextMatchedSeg = NULL
636     }
637
638     bool pauseSegBefore     = mCurSegs[SEGLIST] != mSegList->begin() &&
639                               (mCurSegs[SEGLIST]-1)->segType == TA_SEG_PAUSE &&
640                               (mCurSegs[SEGLIST]-1)->segLen >= MIN_PAUSE_LEN
641     bool pauseSegAfterwards = (mCurSegs[SEGLIST]+1) != mEndOfLists[SEGLIST] &&
642                               (mCurSegs[SEGLIST]+1)->segType == TA_SEG_PAUSE &&
643                               (mCurSegs[SEGLIST]+1)->segLen >= MIN_PAUSE_LEN
644     pauseSegBefore = pauseSegBefore &&
645         (prevMatchedSeg == NULL ||
646          prevMatchedSeg->refPos + prevMatchedSeg->segLen <= (mCurSegs[SEGLIST]-1)->refPos)
647     pauseSegAfterwards = pauseSegAfterwards &&
648         (nextMatchedSeg == NULL ||
649          nextMatchedSeg->refPos >= (mCurSegs[SEGLIST]+1)->refPos + (mCurSegs[SEGLIST]+1)->segLen)
650
651     //Determine the maximum and minimum shifts, based on the adjacent matched segments.
652     minShift = maxShift = guessedShift = mCurSegs[SEGLIST]->degPos - mCurSegs[SEGLIST]->refPos
653
654     //Use the full delay spread as search range of there is a pause segment just before or after.
655     if (pauseSegAfterwards || pauseSegBefore)
656     {
657         minShift = mTAinfo->MinDelay()
658         maxShift = mTAinfo->MaxDelay()
```

```
659     }
660     else
661     {
662         if (nextMatchedSeg != NULL)
663         {
664             minShift = min(minShift, nextMatchedSeg->degPos - nextMatchedSeg->refPos)
665             maxShift = max(maxShift, nextMatchedSeg->degPos - nextMatchedSeg->refPos)
666         }
667         if (prevMatchedSeg != NULL)
668         {
669             minShift = min(minShift, prevMatchedSeg->degPos - prevMatchedSeg->refPos)
670             maxShift = max(maxShift, prevMatchedSeg->degPos - prevMatchedSeg->refPos)
671         }
672 
673         TA_segStruct const *nextMatchedSegAfterCurMergedSeg = NULL,
674                            *prevMatchedSegAfterCurMergedSeg = NULL
675         for (i = 0
676              i < (int)mDetailedSegList->size() &&
677              ((*mDetailedSegList)[i].segType != TA_SEG_MATCHED ||
678               (*mDetailedSegList)[i].degPos <
679                mCurSegs[SEGLIST]->degPos + mCurSegs[SEGLIST]->segLen)
680              i++)
681         if (i != (int)mDetailedSegList->size())
682             nextMatchedSegAfterCurMergedSeg = &(*mDetailedSegList)[i]
683         for (i = i >= 0 ? i-1 : (int)mDetailedSegList->size()-1
684              i >= 0 &&
685              ((*mDetailedSegList)[i].segType != TA_SEG_MATCHED ||
686               (*mDetailedSegList)[i].degPos + (*mDetailedSegList)[i].segLen >
687                mCurSegs[SEGLIST]->degPos)
688              i--)
689         if (i >= 0)
690             prevMatchedSegAfterCurMergedSeg = &(*mDetailedSegList)[i]
691 
692         if (nextMatchedSegAfterCurMergedSeg != NULL)
693         {
694             minShift = min(minShift, nextMatchedSegAfterCurMergedSeg->degPos -
    nextMatchedSegAfterCurMergedSeg->refPos)
695             maxShift = max(maxShift, nextMatchedSegAfterCurMergedSeg->degPos -
    nextMatchedSegAfterCurMergedSeg->refPos)
696         }
697         if (prevMatchedSegAfterCurMergedSeg != NULL)
698         {
699             minShift = min(minShift, prevMatchedSegAfterCurMergedSeg->degPos -
    prevMatchedSegAfterCurMergedSeg->refPos)
700             maxShift = max(maxShift, prevMatchedSegAfterCurMergedSeg->degPos -
    prevMatchedSegAfterCurMergedSeg->refPos)
701         }
702     }
703 
704     if (!pauseSegBefore && curMatchedSeg == NULL && mSegType != TA_SEG_PAUSE &&
705         prevMatchedSeg == NULL && nextMatchedSeg != NULL)
706     {
707         for (i = mDetailedSegList->findInsLocIdx(nextMatchedSeg->refPos) - 1
708              i >= 0 && (*mDetailedSegList)[i].segType != TA_SEG_MATCHED  i--)
709         if (i < 0)
710         {
711             minShift = min(minShift, (*mDetailedSegList)[0].degPos - (*mDetailedSegList)[0].refPos)
712             maxShift = max(maxShift, (*mDetailedSegList)[0].degPos - (*mDetailedSegList)[0].refPos)
713         }
714     }
715     if (!pauseSegAfterwards && curMatchedSeg == NULL && mSegType != TA_SEG_PAUSE &&
716         nextMatchedSeg == NULL && prevMatchedSeg != NULL)
717     {
718         for (i = mDetailedSegList->findInsLocIdx(prevMatchedSeg->refPos) + 1
719              i < (int)mDetailedSegList->size() && (*mDetailedSegList)[i].segType != TA_SEG_MATCHED
    i++)
720         if (i == mDetailedSegList->size())
721         {
```

```
722             minShift = min(minShift, (*mDetailedSegList)[i-1].degPos -
    (*mDetailedSegList)[i-1].refPos)
723             maxShift = max(maxShift, (*mDetailedSegList)[i-1].degPos -
    (*mDetailedSegList)[i-1].refPos)
724         }
725     }
726
727     for (i = 0
728          i < (int)mDetailedSegList->size() &&
729          (*mDetailedSegList)[i].degPos + (*mDetailedSegList)[i].segLen <= mSignalPosDeg
730          i++)
731     if (i != (int)mDetailedSegList->size() && (*mDetailedSegList)[i].segType == TA_SEG_GUESSED)
732         mSegReliability = (*mDetailedSegList)[i].reliability
733     else
734         mSegReliability = 0.0f
735
736     if (maxShift - minShift < TA_SEARCH_STEP)
737         return false  //No room for different ref frame positions, let FullTraversal() handle this.
738
739     //Avoid intruding into adjacent matched segments.
740     minRefPos = 0
741     maxRefPos = mRef->NrOfSamples() - mFrameSize
742     if (prevMatchedSeg != NULL)
743         minRefPos =
744             max(minRefPos,
745                   max(prevMatchedSeg->refPos,
746                         prevMatchedSeg->refPos + prevMatchedSeg->segLen - mFrameSize))
747     if (nextMatchedSeg != NULL)
748         maxRefPos = min(maxRefPos,
749                           max(minRefPos, nextMatchedSeg->refPos))
750     if (maxRefPos < minRefPos)
751         OPTTHROW(( string("ERROR in HandleGuessedSegment_determineTAsearchRange: Invalid adjacent
    matched segments.\n")))
752
753     return true
754 }
755
756 }
757
758
```