AUTOWADE2.0 – Pseudocode

## Main Procedure: AUTOWADE\_SAR\_2\_0

```pseudocode  
PROCEDURE AUTOWADE\_SAR\_2\_0  
  
 COMPILE\_OPT IDL3  
 ENABLE ENVI headless mode  
  
 // Step 1: Load input data and parameters  
 CALL ReadInputData(RasterSARPost, RasterPre, RasterSlope, RasterLC, RasterLIA, RasterLIApre, ExclusionMask, PermanentWaterMask)  
  
 // Step 2: Stack and align raster inputs  
 CALL StackData(RasterSARPost, RasterPre, RasterLC, RasterSlope, RasterLIA, RasterLIApre, StackedRaster)  
  
 // Step 3: Extract data bands from stacked raster  
 CALL LoadStackedRasterBands(StackedRaster, s0post, s0pre, LC, Slope, LIA, LIApre)  
  
 // Step 4: Normalize and calculate difference image  
 CALL ImageCrossCal(s0post, s0pre, s0post\_norm, s0pre\_norm)  
 Deltas0\_norm = s0post\_norm - s0pre\_norm  
  
 // Step 5: Run ISODATA clustering  
 CALL IsodataClassification(s0post\_norm, FloodMaskPost)  
 CALL IsodataClassification(Deltas0\_norm, FloodMaskDelta)  
  
 // Step 6: Intersect low-backscatter and changed clusters  
 FloodMap\_fromClustering = Intersect(FloodMaskPost, FloodMaskDelta)  
  
 // Step 7: Estimate expected radar response from models  
 CALL s0models(LIA, frequency, x1w, x2w, x1f, x2f, x1a, x2a, mu\_pw, sigma\_pw)  
  
 // Step 8: Compute fuzzy score Maps based on backscattering  
 FuzzyScore\_s0post = zmf(s0post, x1w, x2w)  
 Deltas0 = s0post - s0pre  
 FuzzyScore\_Deltas0 = zmf(Deltas0, x1f, x2f)  
  
 // Step 9: Compute fuzzy score map based on area  
 CALL WaterObjectsArea(FloodMap\_fromClustering, Area)  
 FuzzyScore\_Area = 1 - zmf(Area, x1a, x2a)  
  
 // Step 10: Compute combined fuzzy score  
 FuzzyScoreFinal = FuzzyScoreAggregation(FuzzyScore\_s0post, FuzzyScore\_Deltas0, FuzzyScore\_Area)  
  
 // Step 11: Select high-confidence seed pixels (score ≥ 0.5)  
 SeedPixels = FuzzyScoreFinal ≥ 0.5  
  
 // Step 12: Region growing to produce the final FloodMap  
 CALL WaterMapping(SeedPixels, s0post, mu\_pw + 2sigma\_pw, FuzzyScoreFinal, FloodMap)  
  
 // Step 13: Save output flood map  
 CALL WriteOutputData(FloodMap)  
  
END  
```

## Supporting Procedure: ReadInputData

```pseudocode  
PROCEDURE ReadInputData(RasterSARPost, RasterPre, RasterSlope, RasterLC, RasterLIA, RasterLIApre, ExclusionMask, PermanentWaterMask)  
  
 RasterSARPost = ENVI\_OPEN(INPUT\_FILE\_POST)  
 RasterPre = ENVI\_OPEN(INPUT\_FILE\_PRE)  
 RasterSlope = ENVI\_OPEN(SLOPE\_FILE)  
 RasterLC = ENVI\_OPEN(LANDCOVER\_FILE)  
 RasterLIA = ENVI\_OPEN(LIA\_FILE)  
 RasterLIApre = ENVI\_OPEN(LIApre\_FILE)  
  
 ExclusionMask = (RasterSlope > 7) OR (RasterLC == urban\_class)  
 PermanentWaterMask = (RasterLC == water\_class)  
  
END

## Supporting Procedure: StackData

```pseudocode  
PROCEDURE StackData(RasterSARPost, RasterPre, RasterLC, RasterSlope, RasterLIA, RasterLIApre, StackedRaster)  
  
 Extract bounding box from RasterSARPost  
 Subset all ancillary rasters to match this extent  
 Set GridTask.INPUT\_RASTER = RasterSARPost  
 GridTask.Execute()  
  
 StackedRaster = ENVILayerStack([RasterSARPost, SubsetPre, SubsetLC, SubsetSlope, SubsetLIA, SubsetLIApre])  
 Close all input rasters  
  
END

## Supporting Procedure: LoadStackedRasterBands

```pseudocode  
PROCEDURE LoadStackedRasterBands(StackedRaster, s0post, s0pre, LC, Slope, LIA, LIApre)  
  
 s0post = StackedRaster.getdata(bands=0)  
 s0pre = StackedRaster.getdata(bands=1)  
 LC = StackedRaster.getdata(bands=2)  
 Slope = StackedRaster.getdata(bands=3)  
 LIA = StackedRaster.getdata(bands=4)  
 LIApre = StackedRaster.getdata(bands=5)  
  
 StackedRaster.close  
  
END

## Supporting Procedure: ImageCrossCal

```pseudocode  
PROCEDURE ImageCrossCal(s0post, s0pre, s0post\_norm, s0pre\_norm)  
  
 s0post\_lin = 10^(s0post / 10)  
 s0pre\_lin = 10^(s0pre / 10)  
  
 max\_clip\_post = percentile(s0post\_lin, 98)  
 max\_clip\_pre = percentile(s0pre\_lin, 98)  
  
 Ipost\_clipped = MIN(s0post\_lin, max\_clip\_post)  
 Ipre\_clipped = MIN(s0pre\_lin, max\_clip\_pre)  
  
 s0post\_eq = histogram\_equalize(Ipost\_clipped)  
 s0pre\_eq = histogram\_equalize(Ipre\_clipped)  
  
 s0post\_norm = 10 \* log10(s0post\_eq)  
 s0pre\_norm = 10 \* log10(s0pre\_eq)  
  
END

## Supporting Procedure: IsodataClassification

```pseudocode  
PROCEDURE IsodataClassification(Image, FloodMask)  
  
 MIN\_CLUSTERS = 5  
 MAX\_CLUSTERS = 10  
 ClusterLabels = run\_ISODATA(Image, MIN\_CLUSTERS, MAX\_CLUSTERS)  
 ClusterMeans = compute\_cluster\_means(Image, ClusterLabels)  
 FloodCluster = cluster with minimum mean  
 FloodMask = (ClusterLabels == FloodCluster)  
  
END

## Supporting Function: Intersect

```pseudocode  
FUNCTION Intersect(MaskA, MaskB)  
  
 RETURN (MaskA == 1) AND (MaskB == 1)  
  
END

## Supporting Function: s0models

```pseudocode  
PROCEDURE s0models(theta, frequency, x1w, x2w, x1f, x2f, x1a, x2a, mu\_pw, sigma\_pw)  
  
 sigma0\_water = compute\_SPM(theta, frequency)  
 sigma0\_soil = compute\_OhModel(theta, frequency, mv=0.04, s=s\_rayleigh)  
  
 Gaussfit(s0post in PermanentWaterMask, mu\_pw, sigma\_pw)  
  
 x1w = MIN(mu\_pw, sigma0\_water)  
 x2w = sigma0\_soil  
 x1f = x1w - sigma0\_soil  
 x2f = 0  
 x1a = 1  
 x2a = 5  
  
END

## Supporting Function: zmf

```pseudocode  
FUNCTION zmf(x, x1, x2)  
  
 y = array same size as x, filled with 0.0  
 index1 = where(x <= x1)  
 index2 = where((x > x1) AND (x <= (x1 + x2)/2))  
 index3 = where((x > (x1 + x2)/2) AND (x <= x2))  
  
 y[index1] = 1.0  
 y[index2] = 1.0 - 2 \* ((x[index2] - x1) / (x1 - x2))^2  
 y[index3] = 2 \* ((x2 - x[index3]) / (x1 - x2))^2  
  
 RETURN y  
  
END

## Supporting Procedure: WaterObjectsArea

```pseudocode  
PROCEDURE WaterObjectsArea(FloodMap, Area)  
  
 LabelMap = label\_connected\_components(FloodMap)  
 FOR EACH object in LabelMap:  
 Area[object] = number\_of\_pixels(object) \* pixel\_area  
 Remove objects with Area < 1 hectare  
  
END

## Supporting Function: FuzzyScoreAggregation

```pseudocode  
FUNCTION FuzzyScoreAggregation(FuzzyScore\_s0post, FuzzyScore\_Deltas0, FuzzyScore\_Area)  
  
 FuzzyScoreFinal = (FuzzyScore\_s0post + FuzzyScore\_Deltas0 + FuzzyScore\_Area) / 3.0  
 indices = where((FuzzyScore\_s0post == 0) OR (FuzzyScore\_Deltas0 == 0) OR (FuzzyScore\_Area == 0))  
 IF indices not empty THEN:  
 FuzzyScoreFinal[indices] = 0.0  
  
 RETURN FuzzyScoreFinal  
  
END

## Supporting Procedure: WaterMapping

```pseudocode  
PROCEDURE WaterMapping(SeedMask, s0post, tolerance, FuzzyScoreFinal, FloodMap)  
  
 FloodMap = copy of SeedMask  
 Queue = coordinates where SeedMask == 1  
 NeighborOffsets = 8-connected pixel offsets  
 height, width = dimensions of s0post  
  
 WHILE Queue not empty:  
 (i, j) = dequeue(Queue)  
 FOR EACH (di, dj) in NeighborOffsets:  
 ni = i + di  
 nj = j + dj  
 IF bounds valid AND FloodMap[ni, nj] == 0 AND s0post[ni, nj] <= tolerance AND FuzzyScoreFinal[ni, nj] > 0:  
 FloodMap[ni, nj] = 1  
 enqueue(Queue, (ni, nj))  
  
END

## Supporting Procedure: WriteOutputData

```pseudocode  
PROCEDURE WriteOutputData(FloodMap)  
  
 Metadata = build\_metadata()  
 ENVI\_SAVE(FloodMap, FILENAME="FloodMap\_<DATE>.dat", METADATA=Metadata)  
  
END