% Compute inside-outside maps

if( options.vocal ), tic; fprintf( 'videoRapidSegment: Computing inside-outside maps...\t' ); end

data.inMaps = getInOutMaps( flow );

inRatios = getSuperpixelInRatio( superpixels, data.inMaps );מחזירה את האחוזים של הפיקסלים מתוך הסופר פיקסלים שנמצאים באובייקט ולא ברקע עבור שורה של פריימים ונקראת superpixelInRaion\_modified.cpp

הפונקציה שאני כתבתי עושה את אותו דבר עבור פריים בודד

if( options.vocal ), toc; end

imgs = data.imgs;

frames = length( flow );

% Compute pairwise potentials

if( options.vocal ), tic; fprintf( 'videoRapidSegment: Computing pairwise potentials...\t' ); end

[ superpixels, nodeFrameId, bounds, labels ] = ...

makeSuperpixelIndexUnique( superpixels );

[ colours, centres, ~ ] = ...

getSuperpixelStats( imgs, superpixels, labels );

pairPotentials = computePairwisePotentials( params, superpixels, ...

flow, colours, centres, labels ); מחזירה את המטריצות דאבליו ווי.

if( options.vocal ), toc; end

colours = uint8( round( colours ) );

% Preallocate space for unary potentials

nodes = size( colours, 1 );

unaryPotentials = zeros( nodes, 2 );

יצירת המקום בזיכרון עבור מטריצת UNARY

% Create location priors

if( options.vocal ), tic; fprintf( 'videoRapidSegment: Computing location priors...\t\t' ); end

[ ~, accumulatedInRatios ] = accumulateInOutMap( params, data );

צריך לממש את הפונקציה הזו:

התפקיד של הפונקציה הזו הוא ליצור את הבסיס למטריצת המיקומים. מה שנעשה הוא מעבר על כל הפריימים פעמיים, פעם אחת מהסוף להתחלה ופעם שניה מההתחלה לסוף, כאשר בסוף נעשה שיכלול של הנתונים.

היא בונה את המטריצה בצורה איטרטיבית, מתחילה עם הערכים של ה-INOUTMAP עבור כל סופר פיקסל ומוסיפה לכל אחד מהם משקל בהתאם לאחידות הגרדיאנטים בכל סופרפיקסל על פי נוסחא 12 במאמר.בצורה הזו מפחיתים את השגיאות שנכנסות דרך ה-OPTICAL FLOW.

הכנתי את כל הפונקציות המרכזיות שנמצאות בתוך הפונקציה הזו:

getSuperpixelConnectivity

getSuperpixelInRatio

superPixelMeanFlowMagnitude

getFlowGradient – simple matlab function needs to be implemented in c

locationMasks = cell2mat( accumulatedInRatios );

locationUnaries = 0.5 \* ones( nodes, 2, 'single' );

locationUnaries( 1: length( locationMasks ), 1 ) = locationMasks / ( locationNorm \* max( locationMasks ) );

locationUnaries( locationUnaries > 0.95 ) = 0.999;

for( frame = 1: frames )

start = bounds( frame );

stop = bounds( frame + 1 ) - 1;

frameMasks = locationUnaries( start: stop, 1 );

overThres = sum( frameMasks > 0.6 ) / single( ( stop - start ) );

if( overThres < 0.05 )

E = 0.005;

else

E = 0.000;

end

locationUnaries( start: stop, 1 ) = ...

max( locationUnaries( start: stop, 1 ), E );

end

locationUnaries( :, 2 ) = 1 - locationUnaries( :, 1 );

סיום התעסקות במטריצת המיקום L

תחילת התעסקות במטריצת ההופעה A

if( options.vocal ), toc; end

masks = 0.19 \* ones( nodes, 1 );

masks( 1: bounds( frames + 1 ) - 1 ) = single( cell2mat( inRatios ) );

% Create binary masks for foreground/background initialisation

foregroundMasks = masks > 0.2;

backgroundMasks = masks < 0.05;

% Create fading frame weight

if( options.vocal ), tic; fprintf( 'videoRapidSegment: Neighbour frame weighting...\t\t' ); end

weights = zeros( 1 + 2 \* frames, 1, 'single' );

middle = frames + 1;

for( i = 1: length( weights ) )

weights( i ) = exp( - params.fadeout \* ( i - middle ) ^ 2 );

end

if( options.vocal ), toc; end

% Initialise segmentations

if( options.vocal ), tic; fprintf( 'videoRapidSegment: Computing initial segmentation...\t' ); end

fgColors = colours( foregroundMasks, : );

bgColors = colours( backgroundMasks, : );

for( frame = 1: frames )

ids = nodeFrameId - frame + middle;

fgNodeWeights = masks( foregroundMasks ) .\* ...

weights( ids( foregroundMasks ) );

bgNodeWeights = ( 1 - masks( backgroundMasks ) ) .\* ...

weights( ids( backgroundMasks ) );

עד כאן מימוש קבועים פשוט עם פרמטרים שהוגדרו בהתחלה

עבור הפונקציה הבאה עשיתי את המימוש

[ uniqueFgColours, fgNodeWeights ] = ...

findUniqueColourWeights( fgColors, fgNodeWeights );

[ uniqueBgColours, bgNodeWeights ] = ...

findUniqueColourWeights( bgColors, bgNodeWeights );

startIndex = bounds( frame );

stopIndex = bounds( frame + 1 ) - 1;

if( size( uniqueFgColours, 1 ) < fgMix || ...

size( uniqueBgColours, 1 ) < bgMix )

warning( 'Too few data points to fit GMM...\n' ); %#ok<WNTAG>

unaryPotentials( startIndex: stopIndex, : ) = -log( 0.5 );

else

פונקציית מטלב שצריך לממש ב-C

[ fgModel ] = fitGMM( fgMix, uniqueFgColours, fgNodeWeights );

[ bgModel ] = fitGMM( bgMix, uniqueBgColours, bgNodeWeights );

הפונקציה הבאה מטלבית לגמרי מוצאת את פונקציית PDF על סמך המודלים

appearanceUnary = getUnaryAppearance( ...

single( colours( nodeFrameId == frame, : ) ), ...

fgModel, bgModel );

unaryPotentials( startIndex: stopIndex, : ) = ...

-params.locationWeight \* log( ...

locationUnaries( startIndex: stopIndex, : ) ) + ...

-log( appearanceUnary );

end

end

פונקציה לא ברורה, נראה לי שהיא מאחדת את המטריצות אבל אני לא בטוח איך היא עושה את זה

[ ~, labels ] = maxflow\_mex\_optimisedWrapper( pairPotentials, ...

single( 10 \* unaryPotentials ) );

segmentation = superpixelToPixel( labels, superpixels );

if( options.vocal ), toc; end

בדיקת שפיות נתונים:

% Check that we did not get a trivial, all-background/foreground

% segmentation

if( all( labels ) || all( ~labels ) )

if( options.vocal ), fprintf( 'videoRapidSegment: Trivial segmentation detected, exiting...\n' ); end

return;

end

ביצוע איטרציות לסיגמנטציה (כלומר חוזרים על אותו תהליך של ביצוע ה-GMM עם תנאי התחלה חדשים שהם ה-LABELS שקיבלנו מהאיטרציה הקודמת) עד שמגיעים לאיטרציה האחרונה או עד שמקבלים התכנסות של הפתרון

לאחר מכן מדפיסים את הווידאו כמו שאנחנו מכירים כבר. ההדפסה ויצירה של הווידאו נמצאת בתוך הלולאה.

% Iterating segmentations

for( i = 2: params.maxIterations )

if( options.vocal ), tic; fprintf( 'videoRapidSegment: Iteration: %i...\t\t\t', i ); end

fgColors = colours( labels, : );

bgColors = colours( ~labels, : );

for( frame = 1: frames )

oldLabels = labels;

ids = nodeFrameId - frame + middle;

fgNodeWeights = weights( ids( labels ) );

bgNodeWeights = weights( ids( ~labels ) );

[ uniqueFgColours, fgNodeWeights ] = ...

findUniqueColourWeights( fgColors, fgNodeWeights );

[ uniqueBgColours, bgNodeWeights ] = ...

findUniqueColourWeights( bgColors, bgNodeWeights );

if( size( uniqueFgColours, 1 ) < fgMix || ...

size( uniqueBgColours, 1 ) < bgMix )

warning( 'videoRapidSegment: Too few data points to fit GMM...\n' ); %#ok<WNTAG>

return;

end

[ fgModel ] = fitGMM( fgMix, uniqueFgColours, fgNodeWeights );

[ bgModel ] = fitGMM( bgMix, uniqueBgColours, bgNodeWeights );

appearanceUnary = getUnaryAppearance( ...

single( colours( nodeFrameId == frame, : ) ), ...

fgModel, bgModel );

startIndex = bounds( frame );

stopIndex = bounds( frame + 1 ) - 1;

unaryPotentials( startIndex: stopIndex, : ) = ...

-params.locationWeight \* log( ...

locationUnaries( startIndex: stopIndex, : ) ) + ...

-log( appearanceUnary );

end

[ ~, labels ] = maxflow\_mex\_optimisedWrapper( pairPotentials, ...

single( 10 \* unaryPotentials ) );

segmentation = superpixelToPixel( labels, superpixels );

if( options.vocal ), toc; end

if( ( i == params.maxIterations ) ||...

all( all( oldLabels == labels ) ) )

if( options.vocal ), fprintf( 'videoRapidSegment: Convergence or maximum number of iterations reached\n' ); end

if( options.visualise )

if( options.vocal ), tic; fprintf( 'videoRapidSegment: Creating segmentation video...\t' ); end

videoParams.name = sprintf( 'segmentation%i', data.id );

videoParams.range = data.id;

mode = 'ShowProcess';

data.locationProbability = superpixelToPixel( ...

locationUnaries( :, 1 ), superpixels );

data.appearanceProbability = mapBelief( ...

superpixelToPixel( - log( single( ...

getUnaryAppearance( single( colours ), fgModel, ...

bgModel ) ) ), superpixels ) );

data.unaryPotential = mapBelief( superpixelToPixel( ...

unaryPotentials, superpixels ) );

data.segmentation = segmentation;

createSegmentationVideo( options, videoParams, data, mode );

videoParams.name = sprintf( 'segmentation%i-dominantObject', data.id );

data.segmentation = getLargestSegmentAndNeighbours( ...

segmentation );

createSegmentationVideo( options, videoParams, data, mode );

clear data

if( options.vocal ), toc; end

end

break;

end

% Check that we did not get a trivial, all-background/foreground

% segmentation

if( all( labels ) || all( ~labels ) )

if( options.vocal ), fprintf( 'videoRapidSegment: Trivial segmentation detected, exiting...\n' ); end

return;

end

end

if( options.vocal ), fprintf( 'videoRapidSegment: Algorithm stopped after %i iterations.\n', i ); end

end