References

Agarwal, S., Roth, D., 2002. Learning a sparse representation for object detection, in: European Conference on Computer Vision. Springer, pp. 113–127.

Araujo, A., Chen, D., Vajda, P., Girod, B., 2014. Real-time query-by-image video search system. ACM Press, pp. 723–724. doi:10.1145/2647868.2654867

Bahri, A., Zouaki, H., n.d. A Surf-Color Moments for Images Retrieval Based on Bag-of-Features [WWW Document]. EA J. URL http://www.eajournals.org/journals/european-journal-of-computer-science-and-information-technology-ejcsit/vol-1-issue-1-june-2013/a-surf-color-moments-for-images-retrieval-based-on-bag-of-features/ (accessed 4.23.17).

Eickeler, S., Wallhoff, F., Lurgel, U., Rigoll, G., 2001. Content based indexing of images and video using face detection and recognition methods. IEEE, pp. 1505–1508. doi:10.1109/ICASSP.2001.941217

El-gayar, M.M., Soliman, H., meky, N., 2013. A comparative study of image low level feature extraction algorithms. Egypt. Inform. J. 14, 175–181. doi:10.1016/j.eij.2013.06.003

Elnemr, H.A., 2016. Combining SURF and MSER along with Color Features for Image Retrieval System Based on Bag of Visual Words. J. Comput. Sci. 12, 213–222. doi:10.3844/jcssp.2016.213.222

Fergus, R., Perona, P., Zisserman, A., 2003. Object class recognition by unsupervised scale-invariant learning, in: Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on. IEEE, pp. II–264.

Fitzgibbon, A., Zisserman, A., 2002. On Affine Invariant Clustering and Automatic Cast Listing in Movies, in: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (Eds.), Computer Vision — ECCV 2002, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 304–320.

Hu, W., Xie, N., Li, L., Zeng, X., Maybank, S., 2011. A Survey on Visual Content-Based Video Indexing and Retrieval. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. 41, 797–819. doi:10.1109/TSMCC.2011.2109710

Jing, Y., Liu, D., Kislyuk, D., Zhai, A., Xu, J., Donahue, J., Tavel, S., 2015. Visual Search at Pinterest, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15. ACM, New York, NY, USA, pp. 1889–1898. doi:10.1145/2783258.2788621

Kavitha, H., Sudhamani, M.V., 2013. Object based Image Retrieval from Database using Combined Features. Int. J. Comput. Appl. 76, 38–42.

Kavya, J., Shashirekha, H., 2014. A Novel Approach for Image Retrieval using Combination of Features. Int. J. Comput. Technol. Appl. 6, 323–327.

Khan, N.Y., McCane, B., Wyvill, G., 2011. SIFT and SURF Performance Evaluation against Various Image Deformations on Benchmark Dataset. IEEE, pp. 501–506. doi:10.1109/DICTA.2011.90

Madbouly, A.M.M., Wafy, M., Mostafa, M.-S.M., 2015. Performance Assessment of Feature Detector-Descriptor Combination. Int. J. Comput. Sci. Issues IJCSI 12, 87.

McGuinness, K., McCusker, K., O'Hare, N., O'Connor, N.E., 2012. Efficient storage and decoding of SURF feature points, in: International Conference on Multimedia Modeling. Springer, pp. 440–451.

Patel, B.V., Meshram, B.B., 2012. Content Based Video Retrieval Systems. Int. J. UbiComp 3, 13–30. doi:10.5121/iju.2012.3202

Rathod, G.I., Nikam, D.A., 2013. An algorithm for shot boundary detection and key frame extraction using histogram difference. Int. J. Emerg. Technol. Adv. Eng. 3, 155–163.
Reverse image search - Search Help [WWW Document], n.d. URL https://support.google.com/websearch/answer/1325808?hl=en (accessed 6.14.16).
Sav, S., Johns, G.G.F., Lee, H., 2006. , in: Image and Video Retrieval: 5th Internatinoal Conference, CIVR 2006, Tempe, AZ, USA, July 13-15, 2006, Proceedings. Springer Science & Business Media, pp. 1–10.
Schneiderman, H., Kanade, T., 2000. A statistical method for 3D object detection applied to faces and cars, in: Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on. IEEE, pp. 746–751.
S.G., A., Karibasappa, K., Eswar Reddy, B., 2013. Video Segmentation for Moving Object Detection Using Local Change & Entropy Based Adaptive Window Thresholding. Academy & Industry Research Collaboration Center (AIRCC), pp. 155–166. doi:10.5121/csit.2013.3916
Sivic, J., Schaffalitzky, F., Zisserman, A., 2004. Efficient object retrieval from videos, in: Signal Processing Conference, 2004 12th European. IEEE, pp. 1737–1740.
Sivic, J., Zisserman, A., 2004. Video data mining using configurations of viewpoint invariant regions, in: Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on. IEEE, pp. I–488.
Sivic, J., Zisserman, A., 2003. Video Google: A Text Retrieval Approach to Object Matching in Videos, in: Ninth IEEE International Conference on Computer Vision (ICCV 2003). Presented at the IEEE International Conference on Computer Vision (ICCV 2003), IEEE.
Velmurugan, K., Baboo, L.D.S.S., 2011. Content-based image retrieval using SURF and colour moments. Glob. J. Comput. Sci. Technol. 11.
Weiming Hu, Nianhua Xie, Li Li, Xianglin Zeng, Maybank, S., 2011. A Survey on Visual Content-Based Video Indexing and Retrieval. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. 41, 797–819. doi:10.1109/TSMCC.2011.2109710
Yang, L., Cai, Y., Hanjalic, A., Hua, X.-S., Li, S., 2011. Video-based image retrieval, in: Proceedings of the 19th ACM International Conference on Multimedia. ACM, pp. 1001–1004.
Yildirim, Y., Yazici, A., Yilmaz, T., 2013. Automatic Semantic Content Extraction in Videos Using a Fuzzy Ontology and Rule-Based Model. IEEE Trans. Knowl. Data Eng. 25, 47–61. doi:10.1109/TKDE.2011.189

Appendix A - Approach

A.1 DFD Data Dictionary

Data Member Name	Type	Default Value	Mandatory ?
Video folder path	Character	“” (empty)	Yes
Image path	Character	“” (empty)	Yes
Rated video analysis results	Complex	null	No
Harris_features	Complex	null	No
SURF_features	Complex	null	No
color_features	Complex	null	No
image	Complex	null	Yes
Key frames list	Complex	null	No
Video details	Complex	null	Yes
Video list	Complex	null	No

Table A.1 - DFD data dictionary

Appendix B - Design

B.1 Activity Diagrams

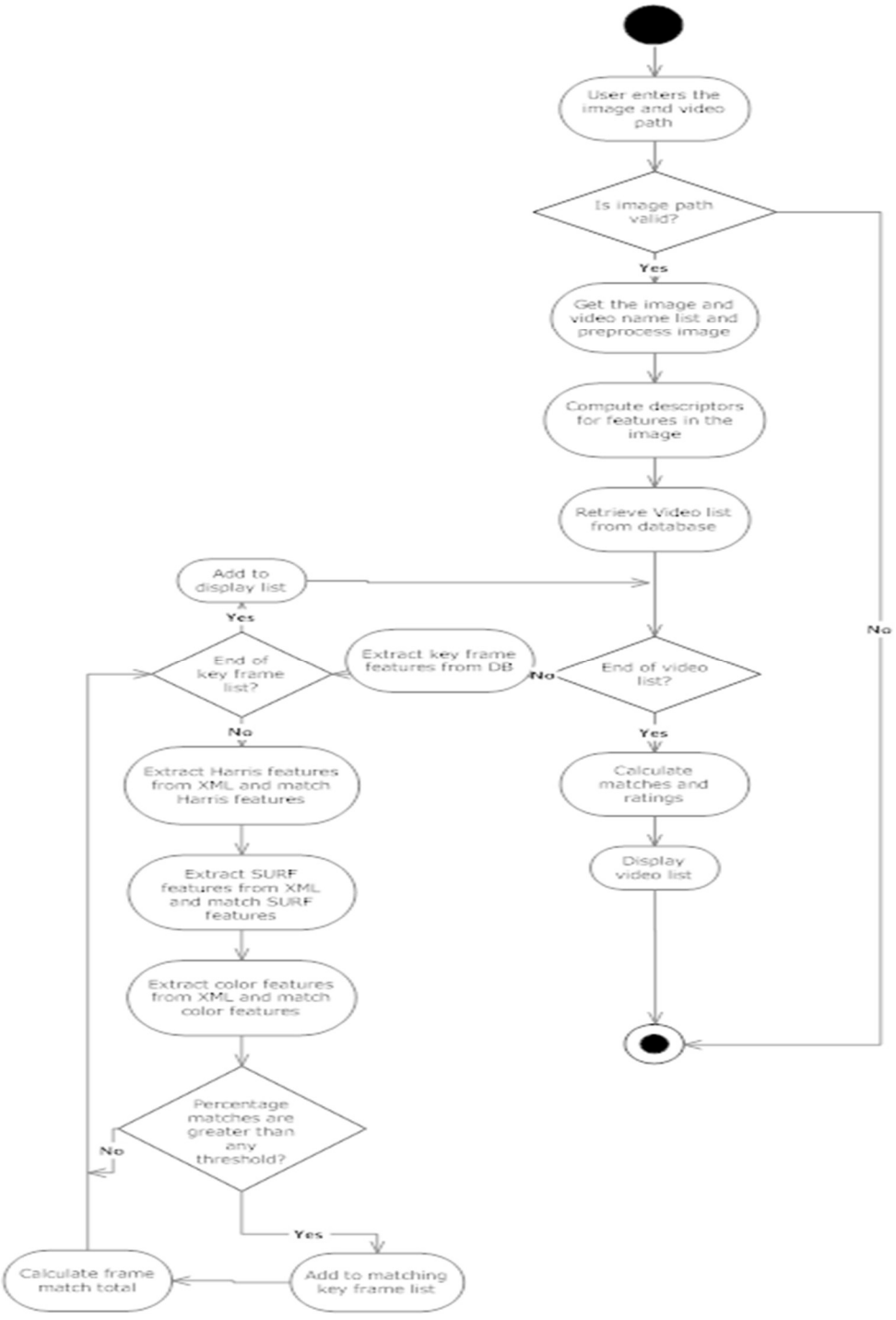

Figure B.1 – Activity diagram of Stored summary searching prototype search senario

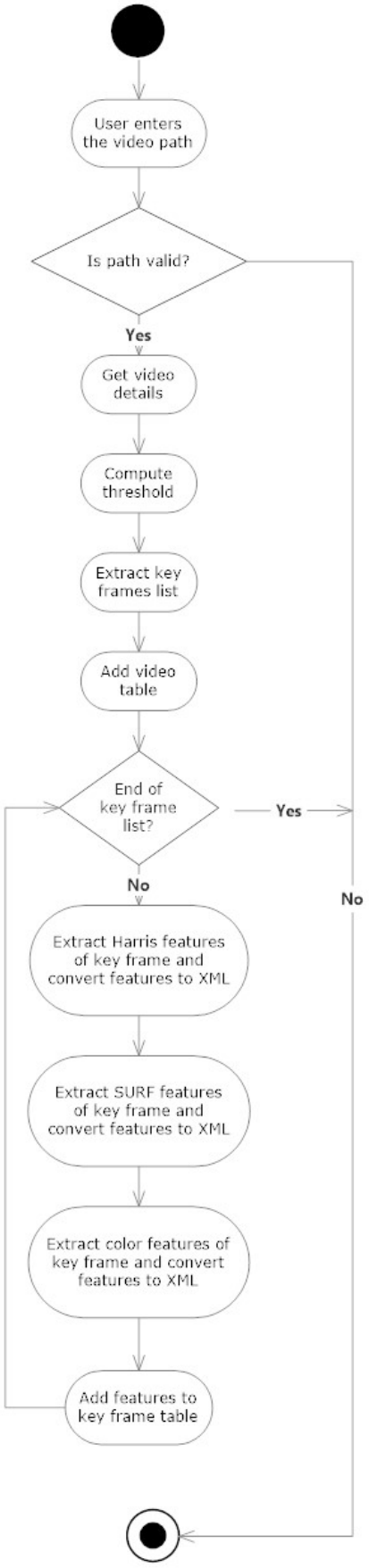

Figure B.2 - Stored summary searching prototype's Add Video scenario

Appendix C - Implementation

C. 1 Code Listing

C.1.1 Get Key Frames

```
function [Keyframes, NOKF] = keyfOtsu(V, keyFramesFolder)
% Copyright (c) 2015, krishnapriya Subramanian
% All rights reserved.
% Modified by H. S. Senevirathna Copyright (c) 2017.

xyloObj = VideoReader(V);   %Using video reader reading video
%Extracting frames
NOF = xyloObj.NumberOfFrames;    % Calculating number of frames
disp(strcat('Number of frames = ',num2str(NOF)));

% special lengths of video
x = 1;
if (xyloObj.Duration >= 15 && 45 > xyloObj.Duration )
        x = 2+((floor(NOF/1000)).^2);
elseif(xyloObj.Duration > 45) % not used
    x = 3+((floor(NOF/1000)).^2);
end
x = floor(x);
NOFhalf = floor(NOF/x);

Diffs =  zeros(NOFhalf,1); Frames = cell(NOFhalf,1); %initialize
Frames{1} = read( xyloObj,x);   % Retrieve data from video and Add to cell
array
Frames{1} = resizeImage(Frames{1});

for g=1:NOFhalf
    if(g~=  NOFhalf)
        y = (g + 1)* x;
        Frames{(g+1)} = read( xyloObj,(y));
        Frames{(g+1)} = resizeImage(Frames{(g+1)});
        diff = imabsdiff(Frames{g},Frames{g+1});
        I=rgb2gray(diff);  % Convert into gray scale
        level = graythresh(I); % Using Otsu's thresolding
        Diffs(g) = level;

%To calculate histogram difference between two frames
        %
    end
end

%calculating mean and standard deviation and extracting key frames
mean=mean2(Diffs);
std = std2(Diffs);
threshold= (mean + std);
NOKF = 0;
Keyframes{1} = cell(1);
lastKFno = 0;
for g=1: NOFhalf
    if(g~=NOFhalf)
        th=Diffs(g);
        if(g == (NOFhalf - 1) || th > threshold)    % Greater than
threshold select as a key frame
```

```
            NOKF = NOKF + 1;
            midKFno = ceil((lastKFno + g + 1)/2);
            if(midKFno ~= lastKFno && midKFno ~= g+1)
                Keyframes{NOKF} = Frames{midKFno};
                NOKF = NOKF + 1;
            end
            lastKFno = g + 1;
            Keyframes{NOKF} = Frames{g+1}; % Add to cell array
        end
    end
end
end
```

C.1.2 Extract Color Features

```
function [features, metrics] = featuresColorExtractor(I)
% Example color layout feature extractor.
% Local color layout features are extracted from truecolor image, I and
% returned in features. The strength of the features are returned in
% metrics.

[~,~,P] = size(I);

isColorImage = P == 3;

if isColorImage

    % Convert RGB images to the L*a*b* colorspace. The L*a*b* colorspace
    % enables you to easily quantify the visual differences between colors.
    % Visually similar colors in the L*a*b* colorspace will have small
    % differences in their L*a*b* values.
    Ilab = rgb2lab(I);

    % Compute the "average" L*a*b* color within 16-by-16 pixel blocks. The
    % average value is used as the color portion of the image feature. An
    % efficient method to approximate this averaging procedure over
    % 16-by-16 pixel blocks is to reduce the size of the image by a factor
    % of 16 using IMRESIZE.
    Ilab = imresize(Ilab, 1/16);

    % Note, the average pixel value in a block can also be computed using
    % standard block processing or integral images.

    % Reshape L*a*b* image into "number of features"-by-3 matrix.
    [Mr,Nr,~] = size(Ilab);
    colorFeatures = reshape(Ilab, Mr*Nr, []);

    % L2 normalize color features
    rowNorm = sqrt(sum(colorFeatures.^2,2));
    colorFeatures = bsxfun(@rdivide, colorFeatures, rowNorm + eps);

    % Augment the color feature by appending the [x y] location within the
    % image from which the color feature was extracted. This technique is
    % known as spatial augmentation. Spatial augmentation incorporates the
    % spatial layout of the features within an image as part of the
    % extracted feature vectors. Therefore, for two images to have similar
    % color features, the color and spatial distribution of color must be
    % similar.
```

```
    % Normalize pixel coordinates to handle different image sizes.
    xnorm = linspace(-0.5, 0.5, Nr);
    ynorm = linspace(-0.5, 0.5, Mr);
    [x, y] = meshgrid(xnorm, ynorm);

    % Concatenate the spatial locations and color features.
    features = [colorFeatures y(:) x(:)];

    % Use color variance as feature metric.
    metrics  = var(colorFeatures(:,1:3),0,2);
else

    % Return empty features for non-color images. These features are
    % ignored by the color feature matching.
    features = zeros(0,5);
    metrics  = zeros(0,1);
end
```

C.1.3 Match SURF Features

```
function  [match, percentage, partial_match] =
matchSurfFeatures(I1,I2,features1, features2, valid_points1, valid_points2,
colour_percentage)
partial_match = false;
indexPairs = matchFeatures(features1,features2);

totalFeatures = length(features1); %baseline number of features
numPairs = size(indexPairs,1); %the number of pairs
if(totalFeatures ~= 0 && numPairs ~= 0)
    status = 1; sizeOfinliers1 = 0;
    matchedPoints1 = valid_points1(indexPairs(:,1),:);
    matchedPoints2 = valid_points2(indexPairs(:,2),:);
    if(numPairs > 2)
        %         disp(size(indexPairs,1));
        [~, inliers1, ~, status] =
estimateGeometricTransform(matchedPoints1, matchedPoints2, 'affine');
        if(status == 0)
            sizeOfinliers1 = size(inliers1,1);
        end
    end
    percentage  = (numPairs * 100)/totalFeatures;
else
    percentage =0; sizeOfinliers1=0;
end

% disp('Surf : ');
if (numPairs == 0)
    %     disp('We do not have this');
    %     disp(percentage);
    match = false;
else
    if(size(I1,2) < size(I2,2))
        Tdist = size(I2,2);
    else
        Tdist = size(I1,2);
    end
```

```
    averageDist = sum(indexPairs(:,2)-indexPairs(:,1))/numPairs;
    percAffine = (sizeOfinliers1*100/numPairs);
    if ((status ~= 0 && percentage >= 8 && (averageDist <= (Tdist/2) * 2.5)
)...
            || (percAffine >= 8 && status == 0) || percentage >= 10)

        match = true;
    else
        %         disp('We do not have this');
        %         disp(percentage);
        match = false;
    end
    if(percentage >= 8)
        partial_match = true;
    end
end

end
```

C.1.4 Match Harris Features

```
function  [match, percentage, partial_match] =
matchHarrisFeatures(I1size,I2size,features1, features2, valid_points1,
valid_points2, colour_percentage)

partial_match = false;
indexPairs = matchFeatures(features1,features2);

totalFeatures = features1.NumFeatures; %baseline number of features
numPairs = size(indexPairs,1); %the number of pairs
if(totalFeatures ~= 0 && numPairs ~= 0)
    status = 1; sizeOfinliers1 = 0;
    matchedPoints1 = valid_points1(indexPairs(:,1),:);
    matchedPoints2 = valid_points2(indexPairs(:,2),:);
    if(numPairs > 2)
        [tform, inliers1, inliers2, status] =
estimateGeometricTransform(matchedPoints1, matchedPoints2, 'affine');
        if(status == 0)
            sizeOfinliers1 = size(inliers1,1);
        end
    end
    percentage  = (numPairs * 100)/totalFeatures;
else
    percentage =0; sizeOfinliers1=0;
end

% disp('Harris : ');
if (numPairs == 0)
    %     disp('We do not have this');
    %     disp(percentage);
    match = false;
else
    if(I1size < I2size)
        Tdist = I2size;
    else
        Tdist = I1size;
    end
    averageDist = sum(indexPairs(:,2)-indexPairs(:,1))/numPairs;
```

```
    percAffine = (sizeOfinliers1*100/numPairs);
    % disp(strcat('max=',num2str(max(indexPairs(:,end)))));
    if ((status ~= 0 && percentage >= 2  && averageDist <= (Tdist/2) * 0.9)
...
            || (percAffine >= 2 && status == 0) || percentage >= 6)
 %         disp('We have this');
        %
disp(percentage);disp(averageDist);disp(size(features1.Features, 2));
        match = true;
    else
        %         disp('We do not have this');
        %         disp(percentage);
        match = false;
    end
    if(percentage >= 4)
        partial_match = true;
    end
end
end
```

C.1.5 Match Color Features

```
function  [percentage, match, partial_match] =
matchColorFeatures(I1,I2,features1, features2)

partial_match = false;
indexPairs = matchFeatures(features1,features2);

totalFeatures = length(features1); %baseline number of features
numPairs = length(indexPairs); %the number of pairs
if(totalFeatures ~= 0)
    percentage  = (numPairs * 100)/totalFeatures;
else
    percentage =0;
end

% disp('Color : ');
if (numPairs == 0)
%     disp('We do not have this');
%     disp(percentage);
    match = false;
else
    if(size(I1,2) < size(I2,2))
        Tdist = size(I2,2);
    else
        Tdist = size(I1,2);
    end
    averageDist = sum(indexPairs(:,2)-indexPairs(:,1))/numPairs;
    if (percentage >= 3.9 && (averageDist <= Tdist/2 * 2.5)) || percentage
>= 50
%         disp('We have this');
%         disp(percentage);disp(averageDist);disp(size(features1, 2));
        match = true;
    else
%         disp('We do not have this');
%         disp(percentage);
        match = false;
    end
```

```
    if(percentage >= 15)
        partial_match = true;
    end
end
end
```

C.1.6 Resize Image

```
function image = resizeImage( image )
if(size(image,2)  > 640)
    p1 = imresize(image(:,:,1), [NaN 640]);
    [m,n,~]=size(p1);
    image=zeros(m,n,3);
    image(:,:,1) = p1;
    image(:,:,2) = imresize(image(:,:,2), [m 640]);
    image(:,:,3) = imresize(image(:,:,3), [m 640]);
elseif(size(image,1)  > 480)
    p1 = imresize(image(:,:,1), [480 NaN]);
    [m,n,~]=size(p1);
    image=zeros(m,n,3);
    image(:,:,1) = p1;
    image(:,:,2) = imresize(image(:,:,2), [480 n]);
    image(:,:,3) = imresize(image(:,:,3), [480 n]);
end
end
```

C.1.7 Connect to Database

```
function conn = ConnectToDB()
dbpath = ['D:\MSc\Project\SurfSearchVid\VidSearchNone.accdb'];
url = [['jdbc:odbc:Driver={Microsoft Access Driver (*.mdb,
*.accdb)};DSN='''';DBQ='] dbpath];
conn = database('','','','sun.jdbc.odbc.JdbcOdbcDriver',url);
end
```

C.1.8 Search Videos

```
function searchVids(imgPath,vidFolder)
disp(datetime('now'));
keyFramesFolder = 'Evaluation\Keyframes';
image = imread(imgPath);
k = dir(strcat(vidFolder,'*.mp4'));
filenames = {k.name}'; % get video files
noOfVids = length(filenames);
if (noOfVids == 0)
    disp('The folder location of videos is not found or empty.');
end
image = resizeImage(image);

% Get features of the image
image_features_Surf = GetSurfFeatures(image);
image_features_Harris = GetHarrisFeatures(image);
[image_features_Color, metrics] = featuresColorExtractor(image);

numOfResults=0;
```

```
% searcing each vid
for g=1: size(filenames)
    video = filenames(g);
    path = strcat(vidFolder,video);
    disp(video);
    pathSring = char(path);

    [keyfArr, NOKF] = keyfOtsu(pathSring, keyFramesFolder);
    disp(strcat('Number of keyframes = ',num2str(NOKF)));

    if (NOKF > 0)
        [sumOfPerc, numMatches] = searchOneVid(image, keyfArr,
keyFramesFolder, image_features_Surf, image_features_Harris,
image_features_Color, video);
    end
    disp(strcat(video, ' Number of results = ',num2str(numMatches)));
    if(NOKF ~= 0 && numMatches ~= 0)
        averageMatch = numMatches/NOKF;
        percMatch = averageMatch * 100;
        avgFeatureMatch = sumOfPerc/(3*numMatches);
    else
        avgFeatureMatch = 0;
        percMatch = 0;
    end
    disp(strcat(video, ' Matching ratio = ',num2str(avgFeatureMatch)));
    if(percMatch >= 10 && avgFeatureMatch >= 12.028) % thresholded
prcentage of matching key frames - empirically derived
        numOfResults = numOfResults+1;
        VideoList{numOfResults,1} = char(video);
        VideoList{numOfResults,2} = avgFeatureMatch;
        VideoList{numOfResults,3} = percMatch;
        VideoList{numOfResults,4} = percMatch+avgFeatureMatch;

    end
end

% Display video list
if(numOfResults>0)
    VideoList = sortrows(VideoList, -4);
    disp(VideoList);
end

disp(strcat('No of matching videos : ',num2str(numOfResults)));
disp(datetime('now'));
end
```

C.1.9 Search One Video

```
function [sumOfPerc, numMatches] = searchOneVid(image,keyfArr,
keyFramesFolder, image_features_Surf, image_features_Harris,
valid_points_Surf, valid_points_Harris, image_features_Color, video)
numMatches = 0;
sumOfPerc = 0;
n = length(keyfArr);
for g=1: n
    %     disp(strcat(num2str(g),' of ',num2str(n)));
    [KF_features_color,metrics] = featuresColorExtractor(keyfArr{(g)});
```

```
    [colour_percentage, match3, partial1] =
matchColorFeatures(image,keyfArr{(g)},image_features_Color,
KF_features_color);

    I1 = rgb2gray(keyfArr{(g)});
    points1 = detectSURFFeatures(I1);
    [KF_features_Surf,KF_valid_points_Surf] = extractFeatures(I1,points1);
    points2 = detectHarrisFeatures(I1);
    [KF_features_Harris,KF_valid_points_Harris] =
extractFeatures(I1,points2);

    [match1, percentage1, partial2] =
matchSurfFeatures(size(image,2),size(keyfArr{(g)},2),image_features_Surf,
KF_features_Surf, valid_points_Surf, KF_valid_points_Surf,
colour_percentage);
    [match2, percentage2, partial3] =
matchHarrisFeatures(size(image,2),size(keyfArr{(g)},2),image_features_Harri
s, KF_features_Harris, valid_points_Harris, KF_valid_points_Harris,
colour_percentage);

    if ((match1 == true || match2 == true || match3 == true) || (partial1
&& partial2 && partial3))
        numMatches = numMatches + 1;
        sumOfPerc = sumOfPerc + percentage1 + percentage2 +
colour_percentage;
    end
end

end
```

C.1.10 Add video to database

```
function AddVidToDB( path )
%insert keyframe features into DB
disp(datetime('now'));

conn = ConnectToDB();
keyFramesFolder = 'Evaluation\Keyframes';
slash_place = strfind(path,'\');
video = path(slash_place(length(slash_place))+1:end);
if exist(path, 'file') ~= 2
    disp(strcat('The file ',video,' does not exist in the path
',path(1:hash_place(1)-1)));
end

[keyfArr, NOKF]  = keyfOtsu(path, keyFramesFolder); % extract key frames
% disp(size(keyfArr,2));
disp(strcat('Number of keyframes = ',num2str(NOKF)));

if(NOKF >0)
    KFsize = size(keyfArr{1},2);
else
    KFsize = 0;
end
colnames = {'VidName', 'FrameWidth'};
data = {video, KFsize};
tablename = 'TblVideo';
```

```
datainsert(conn,tablename,colnames,data);
result = exec(conn,'SELECT Distinct @@Identity FROM TblVideo');
result = fetch(result);
ID = result.Data{1};

colnames2 = {'VideoId', 'SurfFeatures', 'HarrisFeatures', 'ColorFeatures',
'SurfPoints', 'HarrisPoints'};
tablename2 = 'TblKeyFrame';

featuresSet = cell(NOKF,1);
for g=1: NOKF
    featuresC = featuresColorExtractor(keyfArr{(g)});
    I1 = rgb2gray(keyfArr{(g)});
    points1 = detectSURFFeatures(I1);
    [featuresS,KF_valid_points_Surf] = extractFeatures(I1,points1);
    points2 = detectHarrisFeatures(I1);
    [featuresH,KF_valid_points_Harris] = extractFeatures(I1,points2);

    strfS = strcat(int2str(size(featuresS,2)),'#',mat2str(featuresS));
    strfH =
strcat(int2str(size(featuresH.Features,2)),'#',mat2str(featuresH.Features))
;
    strfC = strcat(int2str(size(featuresC,2)),'#',mat2str(featuresC));
    strVPS =
strcat(int2str(size(KF_valid_points_Surf.Location,1)),'#',mat2str(KF_valid_
points_Surf.Location));
    strVPH =
strcat(int2str(size(KF_valid_points_Harris.Location,1)),'#',mat2str(KF_vali
d_points_Harris.Location));

    featuresSet{g} = {ID strfS strfH strfC strVPS strVPH};
%     disp(g);
end
disp('features extracted');
for g=1: NOKF
    data3 = featuresSet{g};
    datainsert(conn,tablename2,colnames2,data3);
%     disp('KF inserted.');
    disp(g);
end
disp('End.');

close(result);
close(conn);
disp(datetime('now'));
end
```

C.1.11 Search Videos from Database

```
function searchVidsFromDB(imgPath)
disp(datetime('now'));

keyFramesFolder = 'Evaluation\Keyframes';
image = imread(imgPath);
image = resizeImage(image);

% Get features of the image
```

```
I1 = rgb2gray(image);
points1 = detectSURFFeatures(I1);
[image_features_Surf,valid_points_Surf] = extractFeatures(I1,points1);
points2 = detectHarrisFeatures(I1);
[image_features_Harris,valid_points_Harris] = extractFeatures(I1,points2);
[image_features_Color, metrics] = featuresColorExtractor(image);

conn = ConnectToDB();
result = exec(conn,'SELECT VidName, ID, FrameWidth FROM TblVideo');
setdbprefs('DataReturnFormat','cellarray');
result = fetch(result,30);
videos = result.Data;
keyfArr = cell(5);
dims = size(videos);

% searcing each vid
numOfResults=0;
if (dims(1) > 0 && dims(2) > 0)

    for g=1: dims(1)
        numMatches = 0;
        video = videos(g,:);
        vidId = uint32(video{2});
        frameWidth = uint32(video{3});
        disp(video{1});
        query = strcat('SELECT SurfFeatures, HarrisFeatures, ColorFeatures,
SurfPoints, HarrisPoints FROM TblKeyFrame WHERE VideoId =
',num2str(vidId));
        resultVid = exec(conn,query);
        setdbprefs('DataReturnFormat','cellarray');
        resultVid = fetch(resultVid);
        images = resultVid.Data;
        dims2 = size(images);
        sumOfPerc = 0;
        noOfKFs = dims2(1);
        disp(strcat('Number of keyframes = ',num2str(noOfKFs)));

        %Get key frames
        featuresSet = cell(noOfKFs,1);
        for i=1: noOfKFs
            vid_image = images(i,:);
            featuresS = single(str2matrix(vid_image{1},1));
            featureVectH = uint8(str2matrix(vid_image{2},1));
            featuresH = binaryFeatures(featureVectH);
            featuresC = str2matrix(vid_image{3},1);
            pointsMatS = single(str2matrix(vid_image{4},3));
            pointsS = SURFPoints(pointsMatS);
            pointsMatH = single(str2matrix(vid_image{5},3));
            pointsH = cornerPoints(pointsMatH);
            featuresSet{i} = {featuresS featuresH featuresC pointsS
pointsH};
        end

        %searching each image
        for i=1: noOfKFs
%             disp(strcat(num2str(i),' of ',num2str(noOfKFs)));
            [sumOfPerc, numMatches] = searchOneImageFromDB(numMatches,
sumOfPerc, image, keyfArr, keyFramesFolder,...
```

```
                image_features_Surf, image_features_Harris,
image_features_Color, featuresSet{i}, frameWidth, valid_points_Surf,
valid_points_Harris);
        end
        disp(strcat(video{1}, ' Number of results =
',num2str(numMatches)));
        if(noOfKFs ~= 0 && numMatches ~= 0)
            averageMatch = numMatches/noOfKFs;
            percMatch = averageMatch * 100;
            avgFeatureMatch = sumOfPerc/(3*numMatches);
        else
            avgFeatureMatch = 0;
            percMatch = 0;
        end
        disp(strcat(video{1}, ' Matching ratio =
',num2str(avgFeatureMatch)));
        if(percMatch >= 10 && avgFeatureMatch >= 12) % thresholded
prcentage of matching key frames
            numOfResults = numOfResults+1;
            VideoList{numOfResults,1} = video{1};
            VideoList{numOfResults,2} = avgFeatureMatch;
            VideoList{numOfResults,3} = percMatch;
            VideoList{numOfResults,4} = percMatch+avgFeatureMatch;
        end
%         end
    end

    % Display video list
    if(numOfResults > 0)
        VideoList = sortrows(VideoList, -4);
        disp(VideoList);
    end
end
disp(strcat('No of matching videos : ',num2str(numOfResults)));
disp(datetime('now'));
end
```

C.1.12 Convert String to Matrix

```
function M = str2matrix( fulltext, mode )%,x,y
% creates a matrix document for the string

if (mode == 1)
    %     if(fulltext ~= '')
    hash_place = strfind(fulltext,'#');
    if(numel(hash_place)~=0)
        y = str2double(fulltext(1:hash_place(1)-1)); %get cols
        if(fulltext(hash_place(1)+1:hash_place(1)+5) ~= char('zeros'))
            text = fulltext(hash_place(1)+2:end-1); % remove #s and []
            row_end = strfind(text,';');
            x = length(row_end)+1;
            M = zeros(x,y);

            % extract row contents
            if x>0
                for i=1:x
                    if i ~= 1
                        a = row_end(i-1)+1;
```

```
                    else
                        a = 1;
                    end
                    if i ~= x
                        b = row_end(i)-1;
                    else
                        b = length(text);
                    end
                    row = text(a:b);
                    ele_end = strfind(row,' ');
                    if y>0
                        for j=1:y
                            if j ~= 1
                                c = ele_end(j-1)+1;
                            else
                                c = 1;
                            end
                            if(numel(ele_end)==0)
                                disp('end');
                            end
                            if j ~= y
                                d = ele_end(j)-1;
                            else
                                d = length(row);
                            end
                            M(i,j) = str2double(row(c:d));  %get element
contents
                        end
                    end
                end
            end
        else
            M = zeros(0,y);
        end
    else
        M = zeros(0,1);
    end
elseif (mode == 2)
    hash_place = strfind(fulltext,'#');
    if(numel(hash_place)~=0)
        if(fulltext(hash_place(1)+1:hash_place(1)+5) ~= char('zeros'))
            text = fulltext(hash_place(1)+2:end-1); % remove #s and []
            rows = strsplit(text,';');
            n = length(rows);
            M = zeros(n,1);
            % extract element contents
            if n>0
                for i=1:n
                    M(i,1) = str2double(rows{i});
                end
            end
        else
            M = zeros(0,y);
        end
    else
        M = zeros(0,1);
    end
elseif (mode == 3) % 2 column array
    hash_place = strfind(fulltext,'#');
    if(numel(hash_place)~=0)
        if(fulltext(hash_place(1)+1:hash_place(1)+5) ~= char('zeros'))
```

```
            text = fulltext(hash_place(1)+2:end-1); % remove #s and []
            row_end = strfind(text,';');
            x = length(row_end)+1;
            M = zeros(x,2);

            % extract row contents
            if x>0
                for i=1:x
                    if i ~= 1
                        a = row_end(i-1)+1;
                    else
                        a = 1;
                    end
                    if i ~= x
                        b = row_end(i)-1;
                    else
                        b = length(text);
                    end
                    row = text(a:b);
                    ele_end = strfind(row,' ');
                    c = 1;
                    d = ele_end(1)-1;
                    M(i,1) = str2double(row(c:d));
                    c = ele_end(1)+1;
                    d = length(row);
                    M(i,2) = str2double(row(c:d));
                end
            end
        else
            M = zeros(0,2);
        end
    else
        M = zeros(0,2);
    end
end
```

C.1.13 Search an Image from Database

```
function [sumOfPerc, numMatches] = searchOneImageFromDB(numMatches,
sumOfPerc, image, keyfArr, keyFramesFolder, image_features_Surf,
image_features_Harris,...
    image_features_Color, key_frame, frameWidth, valid_points_Surf,
valid_points_Harris)
% perform matching

imageWidth = size(image,2);
[colour_percentage, match3, partial1] = matchColorFeatures(imageWidth,
frameWidth, image_features_Color, key_frame{3});
[match1, percentage1, partial2] =
matchSurfFeatures(imageWidth,frameWidth,image_features_Surf, key_frame{1},
valid_points_Surf, key_frame{4}, colour_percentage);
[match2, percentage2, partial3] =
matchHarrisFeatures(imageWidth,frameWidth,image_features_Harris,
key_frame{2}, valid_points_Harris, key_frame{5}, colour_percentage);

if ((match1 == true || match2 == true || match3 == true) || (partial1 &&
partial2 && partial3))
    sumOfPerc = sumOfPerc + percentage1 + percentage2 + colour_percentage;
```

```
end
end
```

Appendix D - Evaluation

D.1 bulkRunReal

```
function bulkRunReal(imgFolderPath,vidFolder1,vidFolder2,vidFolder3)
diary('diaryReal.txt');
    k = dir(strcat(imgFolderPath,'*.jpg'));
    filenames = {k.name}'; % get img files
%     matching each img
disp(vidFolder1);
    for g=1: size(filenames)
        disp(filenames{g});
        imgPath = strcat(imgFolderPath, filenames{g});
        searchVids(imgPath,vidFolder1);
    end
disp(vidFolder2);
    for g=1: size(filenames)
        disp(filenames{g});
        imgPath = strcat(imgFolderPath, filenames{g});
        searchVids(imgPath,vidFolder2);
    end
disp(vidFolder3);
    for g=1: size(filenames)
        disp(filenames{g});
        imgPath = strcat(imgFolderPath, filenames{g});
        searchVids(imgPath,vidFolder3);
    end
diary off;
end
```

D.2 bulkRun

```
function bulkRun(FolderPath,mode)
diary('DB_prot_log2.txt');
if(mode==2)
    k = dir(strcat(FolderPath,'*.mp4'));
    filenames = {k.name}'; % get files
    %  each video
    for g=1: size(filenames)
        disp(filenames{g});
        vidPath = strcat(FolderPath, filenames{g});
        AddVidToDB(vidPath);
    end
elseif(mode==3)
    k = dir(strcat(FolderPath,'*.jpg'));
    filenames = {k.name}'; % get img files
    % matching each img
    for g=1: size(filenames)
        disp(filenames{g});
        imgPath = strcat(FolderPath, filenames{g});
        searchVidsFromDB(imgPath);
    end
end
diary off;
end
```

D.3 Detailed Evaluation Data

D.3.1 Comparison of Speeds of the Two Prototypes

		Image	Start time	End time	Time difference
Stored summary	Good match	**1**	19:22:22	19:49:58	0:27:36
		2	19:49:58	20:18:01	0:28:03
		3	20:18:01	20:47:00	0:28:59
		4	20:47:00	21:15:20	0:28:20
		5	21:15:20	21:43:51	0:28:31
	Moderate match	**1**	21:45:13	22:13:34	0:28:21
		2	22:13:34	22:42:08	0:28:34
		3	22:42:08	23:10:28	0:28:20
		4	23:10:29	23:38:33	0:28:04
		5	11:38:33	12:06:41	0:28:08
	No match	**1**	0:35:23	1:21:38	0:46:15
		2	1:21:38	2:07:57	0:46:19
		3	2:07:57	2:54:06	0:46:09
		4	2:54:07	3:40:13	0:46:06
		5	3:40:13	4:27:28	0:47:15
Real-time	Good match	**1**	5:31:08	5:52:08	0:21:00
		2	5:52:08	6:13:16	0:21:08
		3	6:13:16	6:34:40	0:21:24
		4	6:34:40	6:55:48	0:21:08
		5	18:55:48	19:16:55	0:21:07
	Moderate match	**1**	7:16:55	7:44:58	0:28:03
		2	7:44:59	8:12:28	0:27:29
		3	8:12:28	8:40:31	0:28:03
		4	8:40:32	9:08:21	0:27:49
		5	9:08:22	9:36:00	0:27:38
	No match	**1**	9:36:01	9:50:32	0:14:31
		2	9:50:32	10:05:02	0:14:30
		3	10:05:02	10:19:42	0:14:40
		4	10:19:42	10:34:09	0:14:27
		5	10:34:09	10:48:34	0:14:25

Table D.1 - Evaluation details of prototype comparison

D.3.2 Search Results For Different Image Sets

Image No	0% set		mid% set		>90% set	
	Matches	%	Matches	%	Matches	%
5 images						
1	1	3.333333333	2	6.666667	0	0
2	5	16.66666667	11	36.66667	6	20
3	5	16.66666667	11	36.66667	6	20
4	1	3.333333333	1	3.333333	1	3.333333
5	5	16.66666667	1	3.333333	1	3.333333
		11.33333333		**17.33333**		**9.333333**
10 images						
1	5	16.66666667	11	36.66667	6	20
2	5	16.66666667	11	36.66667	6	20
3	5	16.66666667	11	36.66667	6	20
4	5	16.66666667	11	36.66667	6	20
5	0	0	2	6.666667	2	6.666667
6	5	16.66666667	11	36.66667	6	20
7	6	20	11	36.66667	6	20
8	5	16.66666667	11	36.66667	6	20
9	5	16.66666667	11	36.66667	6	20
10	6	20	11	36.66667	6	20
		15.66666667		**33.66667**		**18.66667**
15 images						
1	5	16.66666667	11	36.66667	6	20
2	5	16.66666667	11	36.66667	6	20
3	5	16.66666667	11	36.66667	6	20
4	5	16.66666667	11	36.66667	6	20
5	1	3.333333333	1	3.333333	5	16.66667
6	5	16.66666667	11	36.66667	6	20
7	5	16.66666667	11	36.66667	6	20
8	2	6.666666667	2	6.666667	3	10
9	5	16.66666667	11	36.66667	6	20
10	5	16.66666667	11	36.66667	6	20
11	5	16.66666667	11	36.66667	6	20
12	6	20	11	36.66667	6	20
13	9	30	5	16.66667	9	30
14	5	16.66666667	11	36.66667	6	20
15	12	40	5	16.66667	10	33.33333
		17.77777778		**29.77778**		**20.66667**
20 images						
1	0	0	1	3.333333	0	0
2	4	13.33333333	1	3.333333	1	3.333333
3	5	16.66666667	2	6.666667	0	0
4	4	13.33333333	3	10	2	6.666667

6	6	20	11	36.66667	5	16.66667
6	6	20	11	36.66667	5	16.66667
7	6	20	11	36.66667	5	16.66667
8	6	20	11	36.66667	5	16.66667
9	6	20	11	36.66667	5	16.66667
10	6	20	11	36.66667	5	16.66667
11	6	20	11	36.66667	5	16.66667
12	6	20	11	36.66667	5	16.66667
13	6	20	11	36.66667	5	16.66667
14	6	20	11	36.66667	5	16.66667
15	6	20	11	36.66667	5	16.66667
16	12	40	6	20	11	36.66667
17	6	20	11	36.66667	5	16.66667
18	3	10	1	3.333333	4	13.33333
19	7	23.33333333	2	6.666667	2	6.666667
20	10	33.33333333	6	20	11	36.66667
		19.5		**25.66667**		**15.16667**

Table D.2 – Evaluation details of video results for different image sets

D.3.2 Comparison of Results of the Two Prototypes

		Image	Matches
Stored summary	Good match	**1**	0
		2	1
		3	1
		4	0
		5	1
	Moderate match	**1**	0
		2	7
		3	7
		4	0
		5	0
	No match	**1**	0
		2	5
		3	5
		4	0
		5	0
Real-time	Good match	**1**	1
		2	3
		3	3
		4	1
		5	5
	Moderate match	**1**	1
		2	8
		3	8
		4	0
		5	0
	No match	**1**	0
		2	5
		3	5
		4	0
		5	0

Table D.3 – Evaluation details of video results for different video sets

Glossary of Terms

- BRISK: Binary Robust Invariant Scalable Key-points
- CBVR: Content based video retrieval
- FAST: Features from Accelerated Segment Test
- FREAK: Fast Retina Keypoint
- F-SIFT: Fast-SIFT
- GPU: Graphics Processing Unit
- LSI: Latent semantic analysis
- MSAC: M-estimator SAmple Consensus
- MSER: Maximally Stable Extremal Regions
- PCA: Principal Component Analysis
- SIFT: Scale-invariant feature transform
- SURF: Speeded Up Robust Features
- SVM: Support Vector Machine