Evan Rolfe
COM3004

Assignment 2: Word Search

Searching the Labels

Given a word to search for, the search_labels function iterates through each character in the grid, now it generates eight different strings each with the same length as the search word: the word starting from that character moving horizontally to the right across the grid, the word starting with that character moving vertically down the grid and two words formed from moving diagonal north-west and diagonal south-west. It then takes these four strings and computes their reversed string, now we have eight strings with which to compare against the search word.

Dimensionality Reduction

In order to reduce the number of features used in classification, principle component analysis was used. Each 30x30 slice of the image which represents a character can be represented as a singled point in 900-d space. Visualizing this cluster, a vector can be computed such that the amount of variance along this vector is the greatest. Computing the 10 vectors with the greatest amount of spread allows us to now write each point in 900-d space as the distance from the mean point in terms of these vectors. This is just a simple linear transform which is easy to compute since the principle components are just the eigenvectors of the data's covarience matrix.

Search Labels Function

```
function [ start_row, start_col, end_row, end_col ] = search_labels(grid,
word)
    found = false;
    for r=1:15
       for c=1:15
                      s = 1;
            % 1. Search horizontally, forwards and backwards
            if(c+length(word)-1 <= 15)
                labels_word = grid(r,c:(c+length(word)-1));
                if(strcmpi(labels_word,word))
                    start_row = r;
                    start_col = c;
                    end_row = r;
                    end_col = c+length(word)-1;
                elseif(strcmpi(fliplr(labels_word),word))
                    start_row = r;
                    start_col = c+length(word)-1;
                    end_row = r;
                    end_col = c;
                end
            end

            % 2. Search vertically, forwards and backwards
            if(r+length(word)-1 <= 15)
                labels_word = char(grid(r:(r+length(word)-1),c));
                labels_word = reshape(labels_word,1,[]);

                 if(strcmpi(labels_word,word))
                    start_row = r;
                    start_col = c;
```

```
                end_row = r+length(word)-1;
                end_col = c;
            elseif(strcmpi(fliplr(labels_word),word))
                start_row = r+length(word)-1;
                start_col = c;
                end_row = r;
                end_col = c;
            end
        end

        % 3. Search Diagonally, forwards and backwards
        grid_word = '';
                    grid_word2 = '';
        start_r = r;
        start_c = c;

                    % iterate through each character in the search word
        for i=0:length(word)-1

                            % 3.1 Search south-east and north-west
(backwards and forwards)
                            % if there is still enough space on the grid
            if(start_r+i <= 15 && start_c+i <= 15)
                grid_word = strcat(grid_word,
grid(start_r+i,start_c+i));           %concatenate the next word onto the
string
            end

                            % 3.2 Search north east and south-west
(backwards and forwards)
            if(start_r-i >= 1 && start_c+i <= 15)
                grid_word2 = strcat(grid_word2, grid(start_r-
i,start_c+i));
            end
        end

                    % south-east direction
        if(strcmpi(grid_word,word))
            start_row = start_r;
            start_col = start_c;
            end_row = start_row+length(word)-1;
            end_col = start_col+length(word)-1;

                    % north-west direction
        elseif(strcmpi(fliplr(grid_word),word))
            start_row = start_r+length(word)-1;
            start_col = start_c+length(word)-1;
            end_row = start_r;
            end_col = start_c;

                    % north east
        elseif(strcmpi(grid_word2,word))
            start_row = start_r;
            start_col = start_c;
            end_row = start_row-length(word)+1;
            end_col = start_col+length(word)-1;

                    % south-west
        elseif(strcmpi(fliplr(grid_word2),word))
            start_row = start_r-length(word)+1;
            start_col = start_c+length(word)-1;
```

```
                end_row = start_r;
                end_col = start_c;
            end
        end
    end
end
```

Trial1 Code

```
function trial1()
        load assignment2.mat

        imagesc(reshape(test1,450,450));
        for i=1:size(words)(1)
                search_word(words(i),test1, train_data, train_labels)
        end
end

function search_word(word,img_vector, train_data, train_labels)
alphabet = ['A' 'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J' 'K' 'L' 'M' 'N' 'O' 'P'
'Q' 'R' 'S' 'T' 'U' 'V' 'W' 'X' 'Y' 'Z' ' '];


        % 1. Preprocessing
        pixel_features = [];    % each row is a sample of 900 pixel values

        % 1.1 Loop through each 30x30 square
        for r=1:15
                for c=1:15
                    r_start = ((r*30)-29);
                    r_end = r*30;
                    c_start = ((c*30)-29);
                    c_end = c*30;

                        % 1.2 Extract the 30x30 matrix of pixels which are
sqaure slice of the image and it to bottom of pixel_features matrix
                    img_matrix =
reshape(img_vector(r_start:r_end,c_start:c_end),1,[]);
                    pixel_features = vertcat(pixel_features,img_matrix);
                end
        end

        % 2. Classification
        % 2.1 Convert to a vector labels of 225 letters

        % find reduced training set
        labels = classify(train_data,train_labels,pixel_features);
        labels = alphabet(labels);

        % 2.2 Reshape this vector to a 15x15 matrix of letters
        labels = transpose(reshape(labels,15,15));

        % 3. Select a search word
        word = char(word)

        % 4. Perform the search
        [ start_row, start_col, end_row, end_col ] =
search_labels(labels,word);

        % 5. Display the result
```

```
        % imagesc(reshape(img_vector,450,450));

        % 5.1 Convert the start/end rows/cols to pixel coordinates where
each pixel is the center of that character square
        y1 = 14+30*(start_row-1)
        x1 = 14+30*(start_col-1)
        y2 = 14+30*(end_row-1)
        x2 = 14+30*(end_col-1)


        hold on
        % plot([1,450],[450,1]);
        plot([x1,x2],[y1,y2])
        colormap(gray);
end
```

Trial2 Code

```
function trial2()
        load assignment2.mat

        % 1. Preprocessing
        pixel_features = [];    % each row is a sample of 900 pixel values

        imagesc(reshape(test1,450,450));
        % 1.1 Loop through each 30x30 square
        for r=1:15
                for c=1:15
                    r_start = ((r*30)-29);
                    r_end = r*30;
                    c_start = ((c*30)-29);
                    c_end = c*30;
                    img_matrix =
reshape(test1(r_start:r_end,c_start:c_end),1,[]);
                    pixel_features = vertcat(pixel_features,img_matrix);
                end
        end

        % Compute the pca coponents
        covx = cov(train_data);
        [V,d] = eigs(covx,11);

        % project the data onto the pca axes
        pca_train_data = (train_data - repmat(mean(train_data), 699, 1)) *
V;
        pca_test_data = (pixel_features - repmat(mean(train_data),225,1)) *
V;

        % select 10 features
        pca_test_reduced = pca_test_data(:,2:11);
        pca_train_reduced = pca_train_data(:,2:11);

        % perform the search
        for i=1:size(words)(1)
                search_word(words(i),test1, pca_train_reduced,
train_labels, img_matrix, pca_test_reduced)
        end
end
```

```
function search_word(word,img_vector, train_data,
train_labels,img_matrix,pixel_features)
	alphabet = ['A' 'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J' 'K' 'L' 'M' 'N'
'O' 'P' 'Q' 'R' 'S' 'T' 'U' 'V' 'W' 'X' 'Y' 'Z' ' '];

	% 2. Classification
	% find reduced training set
	labels = classify(train_data,train_labels,pixel_features);
	labels = alphabet(labels);

	% 2.2 Reshape this vector to a 15x15 matrix of letters
	labels = transpose(reshape(labels,15,15));

	% 3. Select a search word
	word = char(word);

	% 4. Perform the search
	[ start_row, start_col, end_row, end_col ] =
search_labels(labels,word);

	% 5. Display the result
	% imagesc(reshape(img_vector,450,450));

	% 5.1 Convert the start/end rows/cols to pixel coordinates where
each pixel is the center of that character square
	y1 = 14+30*(start_row-1);
	x1 = 14+30*(start_col-1);
	y2 = 14+30*(end_row-1);
	x2 = 14+30*(end_col-1);

	hold on
	plot([x1,x2],[y1,y2])
	colormap(gray);
end
```