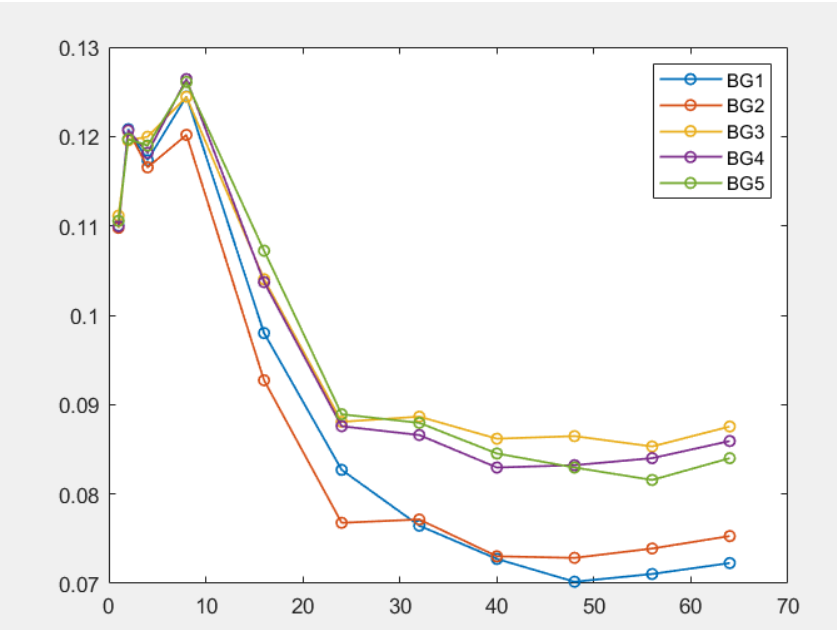
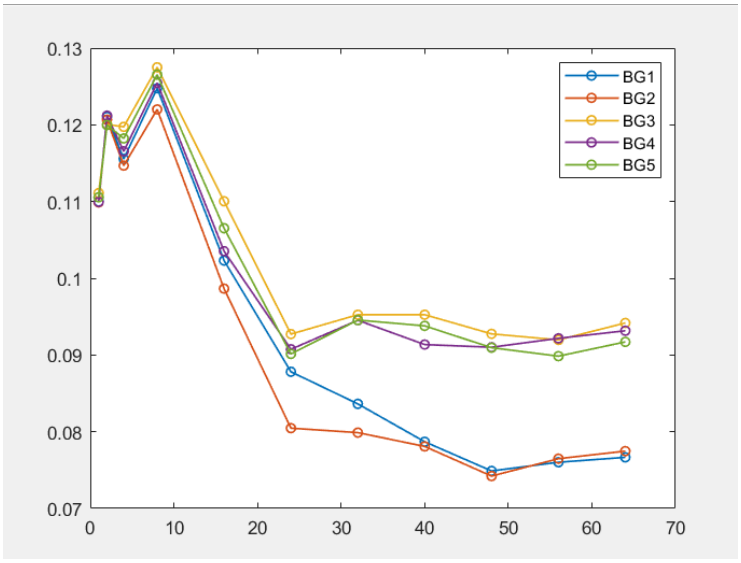
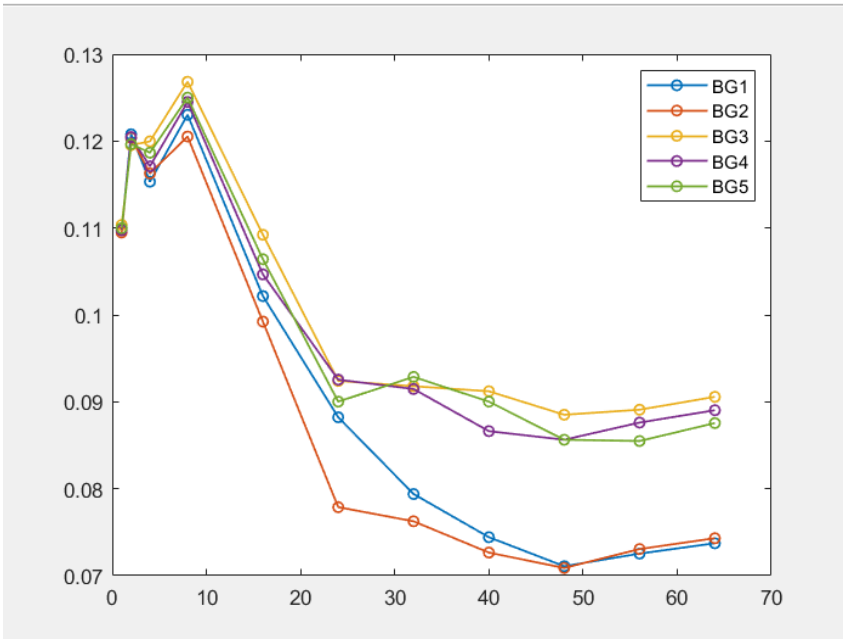
a) The way we combine the foreground and background affects how accurate the results are because the algorithm starts randomly each time. From the graph, we see that using more features usually improves accuracy, but using all 64 dimensions doesn't work as well due to there being some additional unnecessary features. It looks like the best results are normally under 60 and above 40.



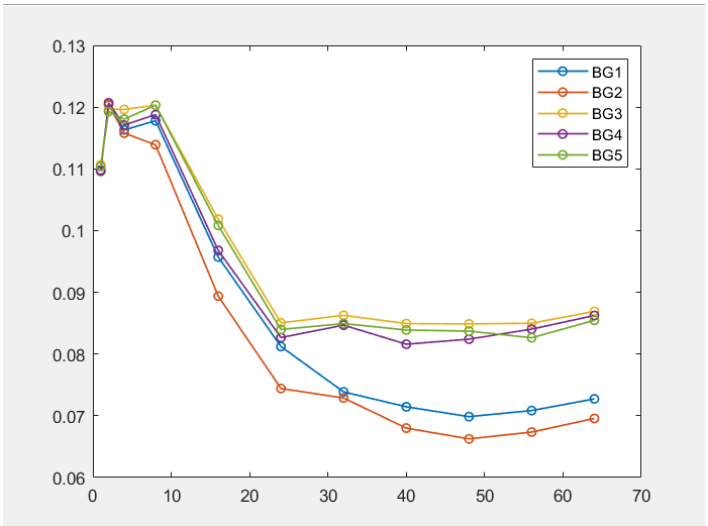
Error plot for foreground set 1



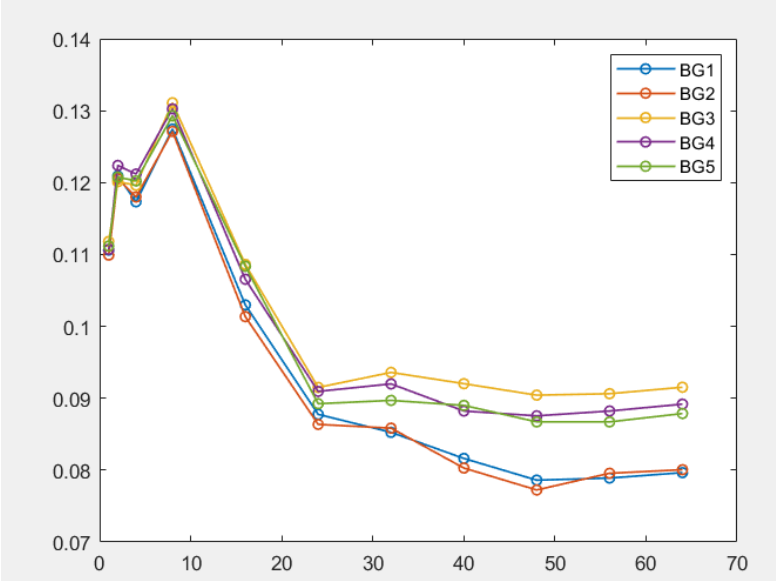
Error plot for foreground set 2



Error plot for foreground set 3

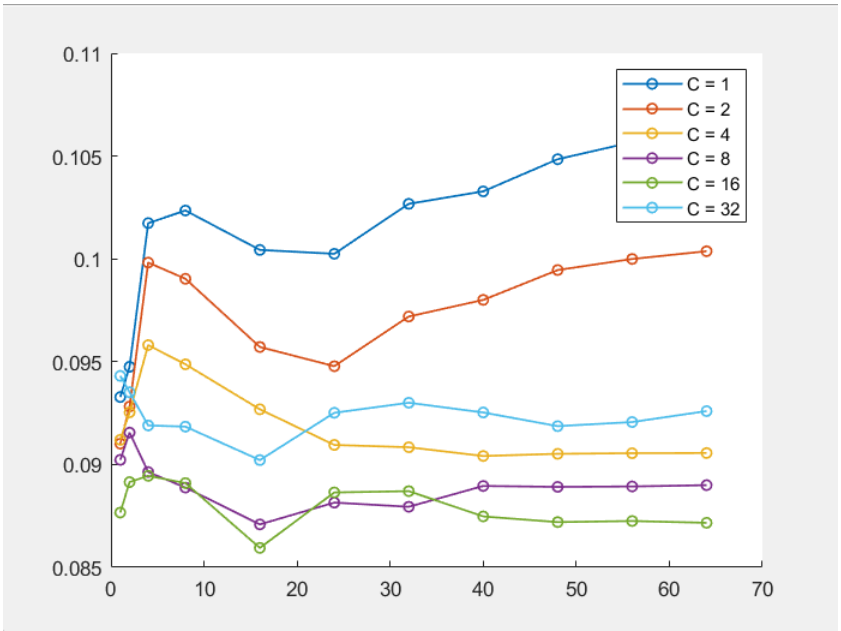


Error plot for foreground set 4



Error plot for set 5

b) The probability of error decreases as the number of mixture components increase because the real data distribution uses a mix of Gaussian distributions.



Mixture Components Plot

Code:

```
%a)
load('TrainingSamplesDCT_8_new.mat');
imageData = double(imread('cheetah.bmp')) / 255;
maskData = imread('cheetah_mask.bmp');
pattern = readmatrix('Zig-Zag Pattern.txt');
paddedImg = padarray(imageData, [4, 4]);
imageRows = size(imageData, 1);
imageCols = size(imageData, 2);
imagePadding = zeros(imageRows * imageCols, 64);

for i = 1:imageRows
    for j = 1:imageCols
        tempBlock = paddedImg(i:i+7, j:j+7);
        dct2Block = dct2(tempBlock);
        flattenedBlock = zeros(1, 64);

        for row = 1:8
            for col = 1:8
                flattenedBlock(1, pattern(row, col) + 1) = dct2Block(row, col);
            end
        end
        imagePadding((i - 1) * imageCols + j, :) = flattenedBlock;
    end
end

C = 8;
dimList = [1, 2, 4, 8, 16, 24, 32, 40, 48, 56, 64];
CList = [1, 2, 4, 8, 16, 32];
maxIter = 200;
```

```
for fgIdx = 1:5
    figure;
    piFG = rand(1, C);
    piFG = piFG / sum(piFG);
    muFG = TrainsampleDCT_FG(randi([1 size(TrainsampleDCT_FG, 1)], 1, C), :);
    fgCovariance = repmat(eye(size(TrainsampleDCT_FG, 2)), 1, 1, C);


    for iter = 1:maxIter
        %E-Step
        joint_prob_FG = zeros(size(TrainsampleDCT_FG, 1), C);

        for j = 1:C
            diff = TrainsampleDCT_FG - muFG(j, :);
            invCov = inv(fgCovariance(:, :, j));
            detCov = det(fgCovariance(:, :, j));
            exponent = -0.5 * sum((diff * invCov) .* diff, 2);
            normFactor = (2 * pi)^(size(muFG, 2) / 2) * sqrt(detCov);
            joint_prob_FG(:, j) = exp(exponent) / normFactor * piFG(j);
        end

        h_ij = joint_prob_FG ./ sum(joint_prob_FG, 2);
        %M-Step
        piFG = sum(h_ij) / size(TrainsampleDCT_FG, 1);

        for j = 1:C
            muFG(j, :) = sum(h_ij(:, j) .* TrainsampleDCT_FG, 1) / sum(h_ij(:, j));
            diffFG = TrainsampleDCT_FG - muFG(j, :);
            fgCovariance(:, :, j) = diag(diag((diffFG' * (diffFG .* h_ij(:, j))) / sum(h_ij(:, j)) + 1e-3 * eye(size(muFG, 2))));
        end

    end
```

```
for bgIdx = 1:5
    piBG = rand(1, C);
    piBG = piBG / sum(piBG);
    muBG = TrainsampleDCT_BG(randi([1 size(TrainsampleDCT_BG, 1)], 1, C), :);
    bgCovariance = repmat(eye(size(TrainsampleDCT_BG, 2)), 1, 1, C);


    for iter = 1:maxIter
        % E-step
        jointBG = zeros(size(TrainsampleDCT_BG, 1), C);
        for j = 1:C
            diff = TrainsampleDCT_BG - muBG(j, :);
            invCov = inv(bgCovariance(:, :, j));
            detCov = det(bgCovariance(:, :, j));
            exponent = -0.5 * sum((diff * invCov) .* diff, 2);
            normFactor = (2 * pi)^(size(muBG, 2) / 2) * sqrt(detCov);
            jointBG(:, j) = exp(exponent) / normFactor * piBG(j);
        end
        h_ij = jointBG ./ sum(jointBG, 2);
        % M-step
        piBG = sum(h_ij) / size(TrainsampleDCT_BG, 1);
        for j = 1:C
            muBG(j, :) = sum(h_ij(:, j) .* TrainsampleDCT_BG, 1) / sum(h_ij(:, j));
            diffBG = TrainsampleDCT_BG - muBG(j, :);
            bgCovariance(:, :, j) = diag(diag((diffBG' * (diffBG .* h_ij(:, j))) / sum(h_ij(:, j)) + 1e-3 * eye(size(muBG, 2))));
        end
    end

    error_arr = zeros(1, length(dimList));

    for i = 1:length(dimList)
        dim = dimList(i);
        result = zeros(imageRows * imageCols, 1);
```

```
for x = 1:length(imagePadding)
    probFG = 0;
    probBG = 0;

    for y = 1:C
        diffFG = imagePadding(x, 1:dim) - muFG(y, 1:dim);
        covFGInv = inv(fgCovariance(1:dim, 1:dim, y));
        covFGDet = det(fgCovariance(1:dim, 1:dim, y));
        exponentFG = -0.5 * (diffFG * covFGInv * diffFG');
        normFactorFG = (2 * pi)^(dim / 2) * sqrt(covFGDet);
        probFG = probFG + exp(exponentFG) / normFactorFG * piFG(y);


        diffBG = imagePadding(x, 1:dim) - muBG(y, 1:dim);
        covBGInv = inv(bgCovariance(1:dim, 1:dim, y));
        covBGDet = det(bgCovariance(1:dim, 1:dim, y));
        exponentBG = -0.5 * (diffBG * covBGInv * diffBG');
        normFactorBG = (2 * pi)^(dim / 2) * sqrt(covBGDet);
        probBG = probBG + exp(exponentBG) / normFactorBG * piBG(y);
    end

    if probBG <= probFG
        result(x) = 1;
    end
end

for x = 1:imageRows
    for y = 1:imageCols
        if maskData(x, y) ~= result((x - 1) * imageCols + y, 1)
            error_arr(1, i) = error_arr(1, i) + 1;
        end
    end
end
```

```
            error_arr(1, i) = error_arr(1, i) / (imageRows * imageCols);
            disp("Plot"+ fgIdx +"Step"+ bgIdx+"Dimension"+dim);
         end

            hold on;
            plot(dimList, error_arr, 'o-', 'linewidth', 1, 'markersize', 5);
            legendEntries{bgIdx} = ['BG' num2str(bgIdx)];
    end
        legend(legendEntries);
end

%b)
clear;
load('TrainingSamplesDCT_8_new.mat');
imageData = imread('cheetah.bmp');
imageData = double(imageData) / 255;
cheetahMask = imread('cheetah_mask.bmp');
pattern = readmatrix('Zig-Zag Pattern.txt');
paddedImage = zeros(263, 278);
for row = 5 : 259
    for col = 5 : 274
        paddedImage(row, col) = imageData(row - 4, col - 4);
    end
end

numFGSamples = size(TrainsampleDCT_FG, 1);
numBGSamples = size(TrainsampleDCT_BG, 1);
bgDim = size(TrainsampleDCT_BG, 2);
imageRows = size(imageData, 1);
imageCols = size(imageData, 2);
imageDCTData = zeros(imageRows * imageCols, 64);
for row = 1 : imageRows
    for col = 1 : imageCols
```

```
        tempBlock = zeros(8,8);
        for r = row : row + 7
            for c = col : col + 7
                tempBlock(r - row + 1, c - col + 1) = paddedImage(r, c);
            end
        end

        dctBlock = dct2(tempBlock);
        flattenedBlock = zeros(1, 64);

        for r = 1 : 8
            for c = 1 : 8
                flattenedBlock(1, pattern(r, c) + 1) = dctBlock(r, c);
            end
        end

        imageDCTData((row - 1) * imageCols + col, :) = flattenedBlock;
    end
end


clusterCounts = [1, 2, 4, 8, 16, 32];
dimensions = [1, 2, 4, 8, 16, 24, 32, 40, 48, 56, 64];
maxIterations = 500;
figure;
for cIdx = 1 : length(clusterCounts)
    C = clusterCounts(cIdx);

    piFG = randi(1, C);
    piFG = piFG / sum(piFG);
    muFG = TrainsampleDCT_FG(randi([1 numFGSamples], 1, C), :);
    fgCovariance = zeros(size(TrainsampleDCT_FG, 2), size(TrainsampleDCT_FG, 2), C);
```

```
for i = 1 : C
    fgCovariance = repmat(eye(size(TrainsampleDCT_FG, 2)), [1, 1, C]) .* rand(1, size(TrainsampleDCT_FG, 2), C);
end

fgJoint = zeros(numFGSamples, C);
for iter = 1 : maxIterations
    numSamples = size(TrainsampleDCT_FG, 1);
numDimensions = size(TrainsampleDCT_FG, 2);
fgJoint = zeros(numSamples, C);

for k = 1:C
    invCov = inv(fgCovariance(:, :, k));
    detCov = det(fgCovariance(:, :, k));
    diff = TrainsampleDCT_FG - muFG(k, :);
    mahalDist = sum((diff * invCov) .* diff, 2);
    fgJoint(:, k) = (1 / sqrt((2 * pi) ^ numDimensions * detCov)) * exp(-0.5 * mahalDist) * piFG(k);
end

    fgH = fgJoint ./ sum(fgJoint, 2);
    logLikelihood(iter) = sum(log(sum(fgJoint, 2)));

    % M-step
    piFG = sum(fgH) / numFGSamples;
    muFG = (fgH' * TrainsampleDCT_FG) ./ sum(fgH, 1)';

    for k = 1:C
        fgCovariance(:, :, k) = diag(diag(transpose(TrainsampleDCT_FG - muFG(k, :)) .* transpose(fgH(:, k)) * (TrainsampleDCT_FG - muFG(k, :)) ./ sum(fgH(:, k), 1)) + 1e-7);
    end

    if iter > 1
        if abs(logLikelihood(iter - 1) - logLikelihood(iter)) <= 1e-4
            break;
        end
    end
end
```

```
piBG = randi(1, C);
piBG = piBG / sum(piBG);
muBG = TrainsampleDCT_BG(randi([1 numBGSamples], 1, C), :);
bgCovariance = zeros(bgDim, bgDim, C);

for i = 1 : C
    bgCovariance = repmat(eye(bgDim), [1, 1, C]) .* rand(1, bgDim, C);
end

bgJoint = zeros(numBGSamples, C);
for iter = 1 : maxIterations
    % E-step
    for k = 1:C
        invCov = inv(bgCovariance(:, :, k));
        detCov = det(bgCovariance(:, :, k));
        diff = TrainsampleDCT_BG - muBG(k, :);
        expTerm = sum((diff * invCov) .* diff, 2);
        bgJoint(:, k) = (1 / sqrt((2 * pi)^size(TrainsampleDCT_BG, 2) * detCov)) * exp(-0.5 * expTerm) * piBG(k);
    end


    bgH = bgJoint ./ sum(bgJoint, 2);
    logLikelihood(iter) = sum(log(sum(bgJoint, 2)));

    % M-step
    piBG = sum(bgH) / numBGSamples;
    muBG = (transpose(bgH) * TrainsampleDCT_BG) ./ transpose(sum(bgH));
    for k = 1:C
        bgCovariance(:, :, k) = diag(diag(transpose(TrainsampleDCT_BG - muBG(k, :)) .* transpose(bgH(:, k)) * (TrainsampleDCT_BG - muBG(k, :)) ./ sum(bgH(:, k), 1)) + 1e-7);
    end


    if iter > 1
        if abs(logLikelihood(iter - 1) - logLikelihood(iter)) <= 1e-4
            break;
        end
    end
end

error_arr = zeros(1, length(dimensions));
```

```
for dimIdx = 1 : length(dimensions)
    dim = dimensions(dimIdx);
    result = zeros(imageRows * imageCols, 1);

    for x = 1 : length(imageDCTData)
        probFG = 0;
        probBG = 0;

        for y = 1:C
            invFGCov = inv(fgCovariance(1:dim, 1:dim, y));
            invBGCov = inv(bgCovariance(1:dim, 1:dim, y));
            detFGCov = det(fgCovariance(1:dim, 1:dim, y));
            detBGCov = det(bgCovariance(1:dim, 1:dim, y));
            diffFG = imageDCTData(x, 1:dim) - muFG(y, 1:dim);
            diffBG = imageDCTData(x, 1:dim) - muBG(y, 1:dim);
            mahalDistFG = diffFG * invFGCov * diffFG';
            mahalDistBG = diffBG * invBGCov * diffBG';
            probFG = probFG + (1 / sqrt((2 * pi)^dim * detFGCov)) * exp(-0.5 * mahalDistFG) * piFG(y);
            probBG = probBG + (1 / sqrt((2 * pi)^dim * detBGCov)) * exp(-0.5 * mahalDistBG) * piBG(y);
       end


        if probBG <= probFG
            result(x) = 1;
        end
    end

    for x = 1 : imageRows
        for y = 1 : imageCols
            if cheetahMask(x, y) ~= result((x - 1) * imageCols + y)
                error_arr(dimIdx) = error_arr(dimIdx) + 1;
            end
        end
    end

    error_arr(dimIdx) = error_arr(dimIdx) / (imageRows * imageCols);
   %disp("C= " + C + " dim= " + dim);
end

hold on;
plot(dimensions, error_arr, 'o-', 'LineWidth', 1, 'MarkerSize', 5);
```

```
        %disp("C= " + C + " dim= " + dim);
    end

    hold on;
    plot(dimensions, error_arr, 'o-', 'LineWidth', 1, 'MarkerSize', 5);
end

legend('C = 1', 'C = 2', 'C = 4', 'C = 8', 'C = 16', 'C = 32');
```