Applied Computer Vision

David Vernon
Carnegie Mellon University Africa

vernon@cmu.edu
www.vernon.eu

Lecture 18

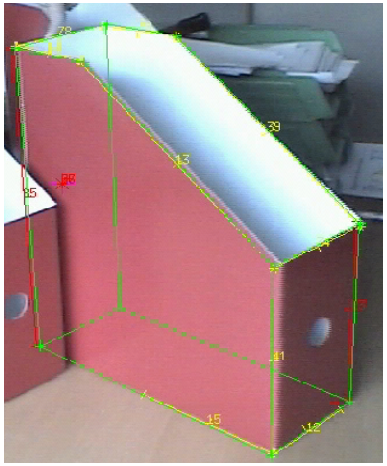
Object Recognition

Colour Histogram Matching & Backprojection

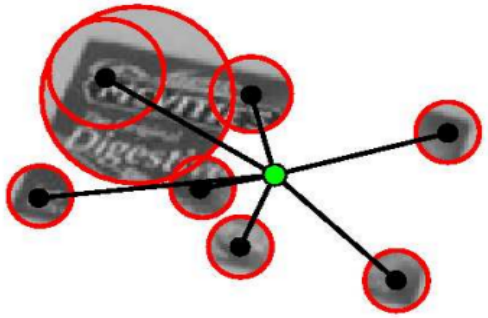
Approaches to Object Recognition

- Generic Gestalt Principles
  - The world is structured, extract features
  - perceptual grouping

- Model based
  - CAD model of object
  - Geometric features
  - Locate features and their arrangement



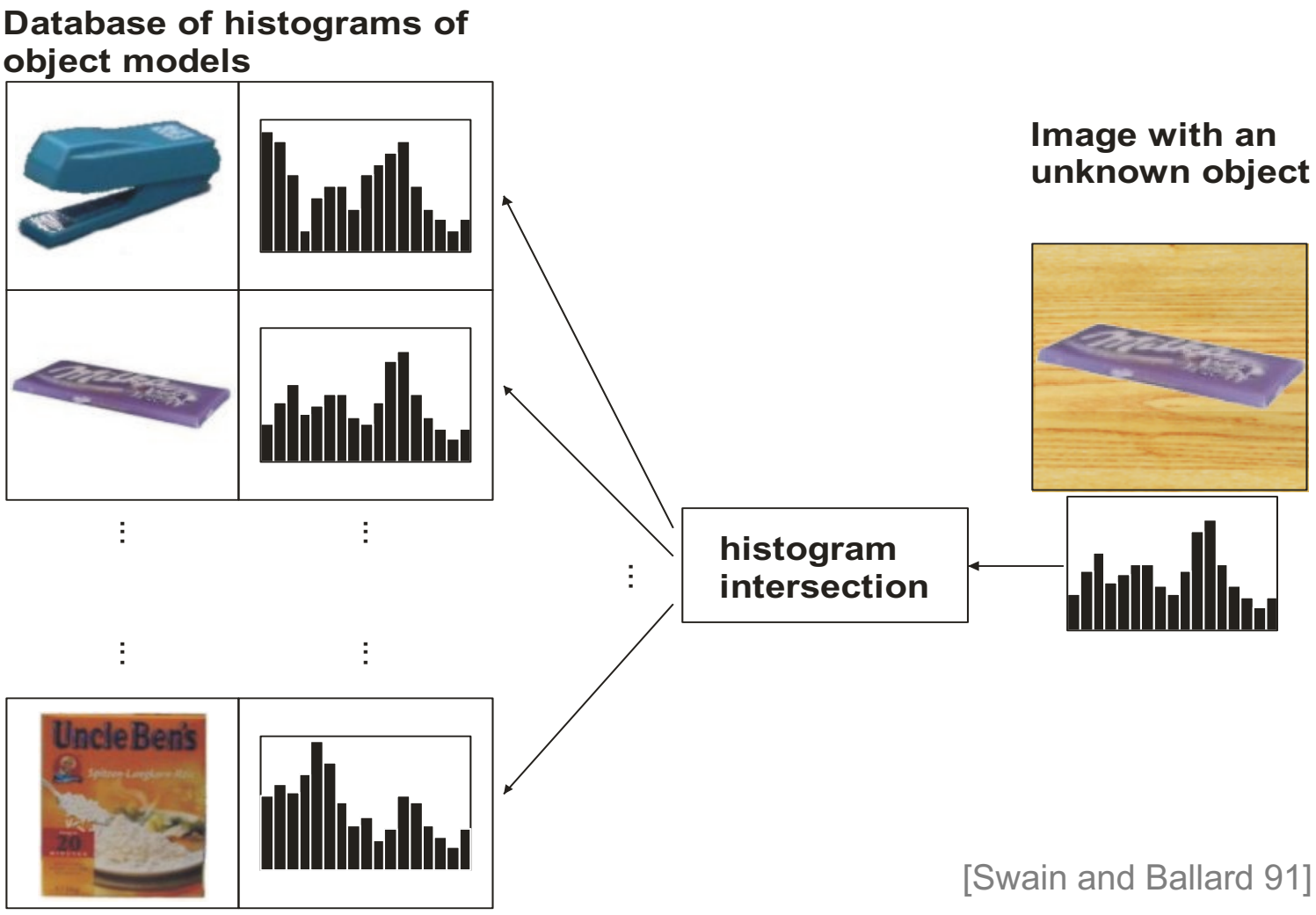
- Appearance based
  - Interest points / point features
  - or “whole” object



Credit: Markus Vincze, Technische Universität Wien

Colour Histogram Matching

Colour Histogram Matching

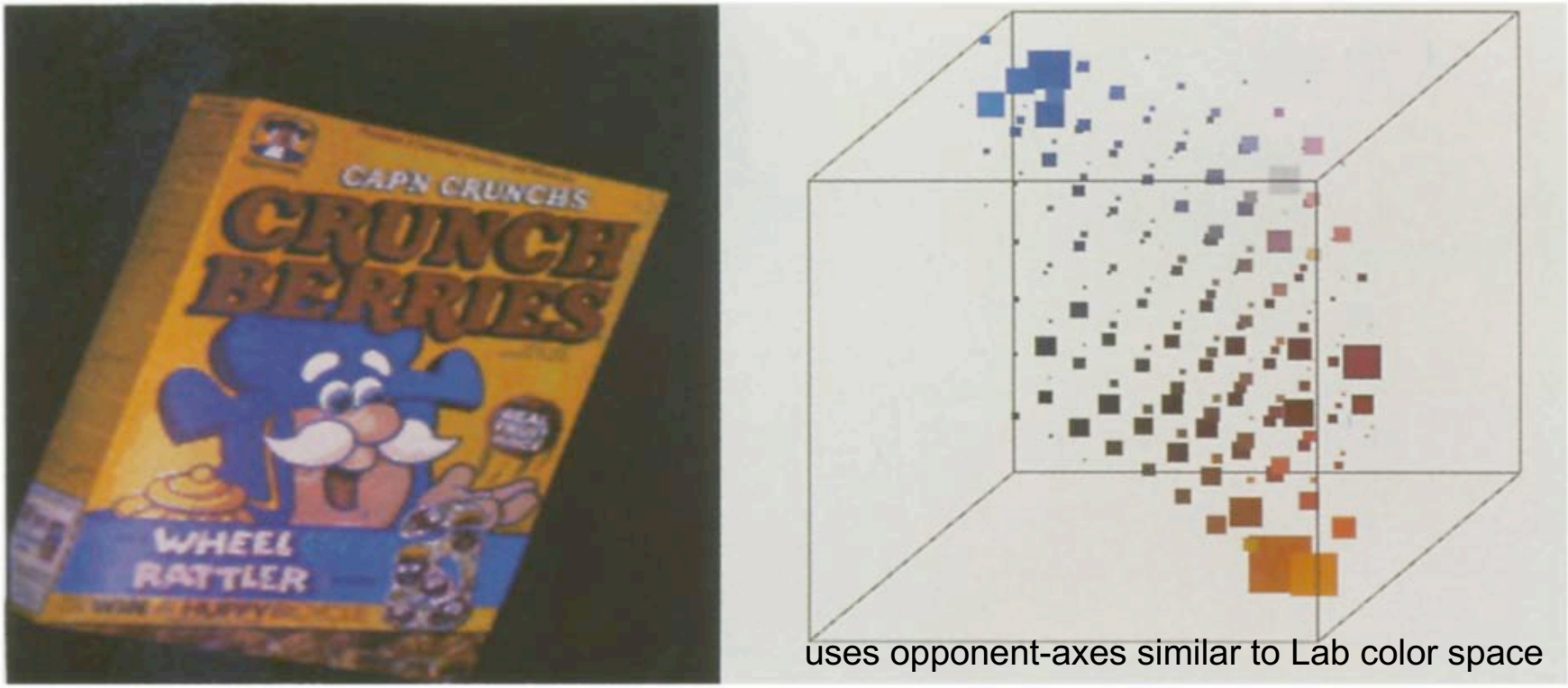


[Swain and Ballard 91]

Credit: Markus Vincze, Technische Universität Wien

Colour Histogram Intersection

Originally: indexing and image search in large DBs

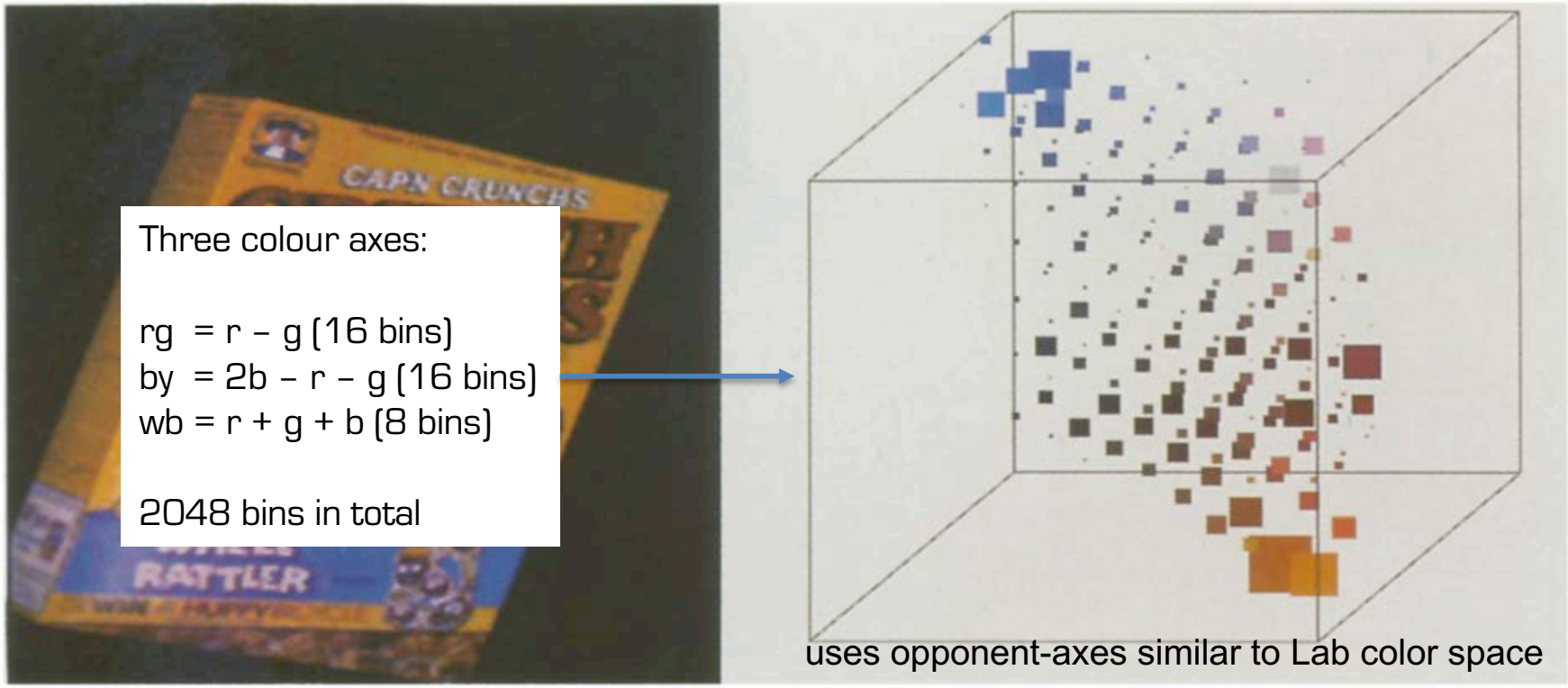


Swain and Ballard, Color Indexing, IJCV 1991.

Credit: Markus Vincze, Technische Universität Wien

Colour Histogram Intersection

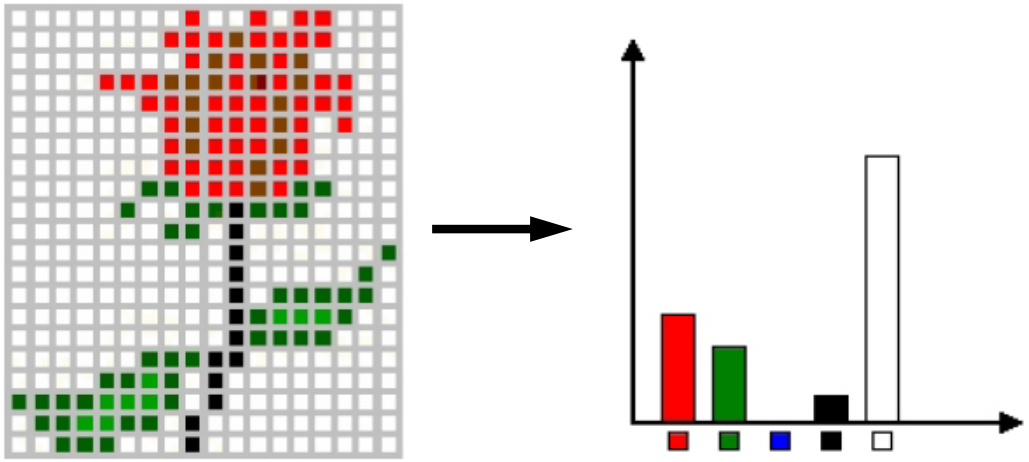
Originally: indexing and image search in large DBs



Swain and Ballard, Color Indexing, IJCV 1991.

Credit: Markus Vincze, Technische Universität Wien

Colour Histogram Matching



[Sebastian Bertram]

(1) Select colour space (RGB, HSI, Lab, ...)

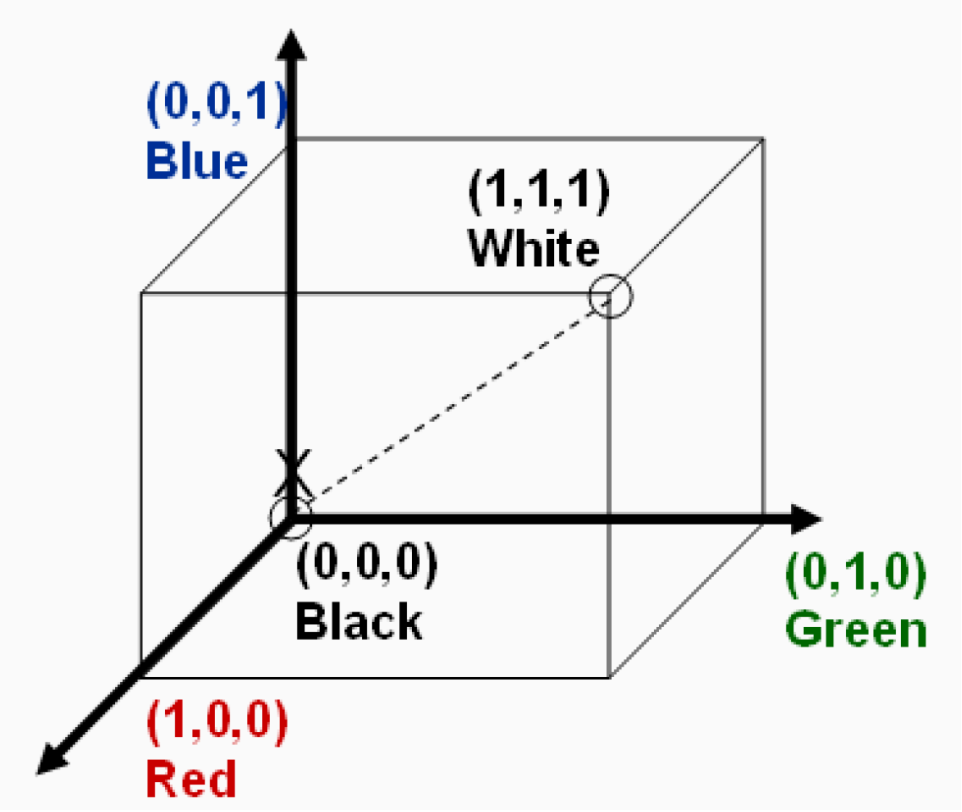
(2) Quantisation of colour space (number of bins)

(3) Calculate the histograms

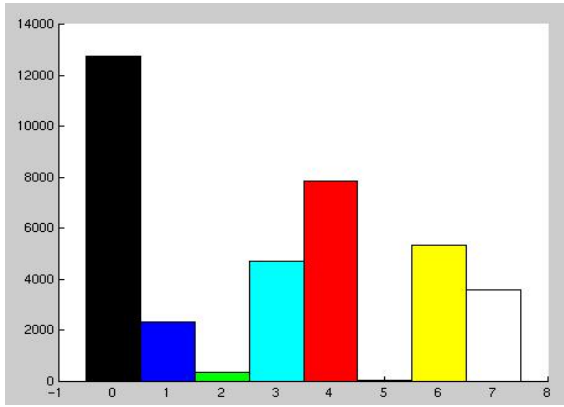
(4) Calculate the histogram distance function

(5) Identify nearest index

Credit: Markus Vincze, Technische Universität Wien

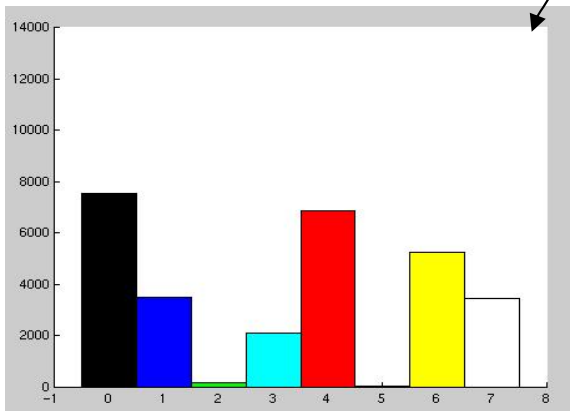


Model

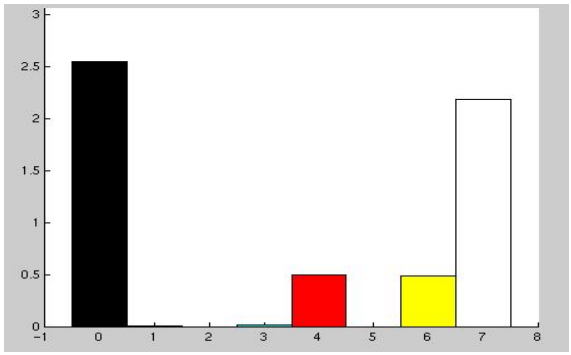


feature vectors

Test 1



Test 2



How many bins (classes)?

Credit: Markus Vincze, Technische Universität Wien

Colour Histogram Matching

Histograms H_1 and H_2, normalised to [0,1]

Correlation

$$d_{\text{correl}}(H_1, H_2) = \frac{\sum_i H_1'(i) \cdot H_2'(i)}{\sqrt{\sum_i H_1'^2(i) \cdot H_2'^2(i)}}$$

with $$H_k'(i) = H_k(i) - (1/N)\left(\sum_j H_k(j)\right)$$

and N the number of classes (bins) in the histogram

Value: 1 is perfect correlation, 0 smallest value

Colour Histogram Matching

Chi-Square

$$d_{\text{chi-square}}(H_1,\ H_2) = \sum_i \frac{(H_1(i) - H_2(i))^2}{H_1(i) + H_2(i)}$$

Value: 0 is perfect hit, values go to infinity

Colour Histogram Matching

Intersection [Swain and Ballard1991]

Value: 1 is perfect correlation, 0 smallest value

$$d_{\text{intersection}}(H_1,\ H_2) = \sum_i \min(H_1(i),\ H_2(i))$$

Colour Histogram Matching

Bhattacharyya

$$d_{\text{Bhattacharyya}}(H_1, H_2) = \sqrt{1 - \sum_i \frac{\sqrt{H_1(i) \cdot H_2(i)}}{\sqrt{\sum_i H_1(i) \cdot \sum_i H_2(i)}}}$$

- Value: 0 is perfect hit, 1 is no agreement
- Important: normalisation!

Colour Histogram Matching

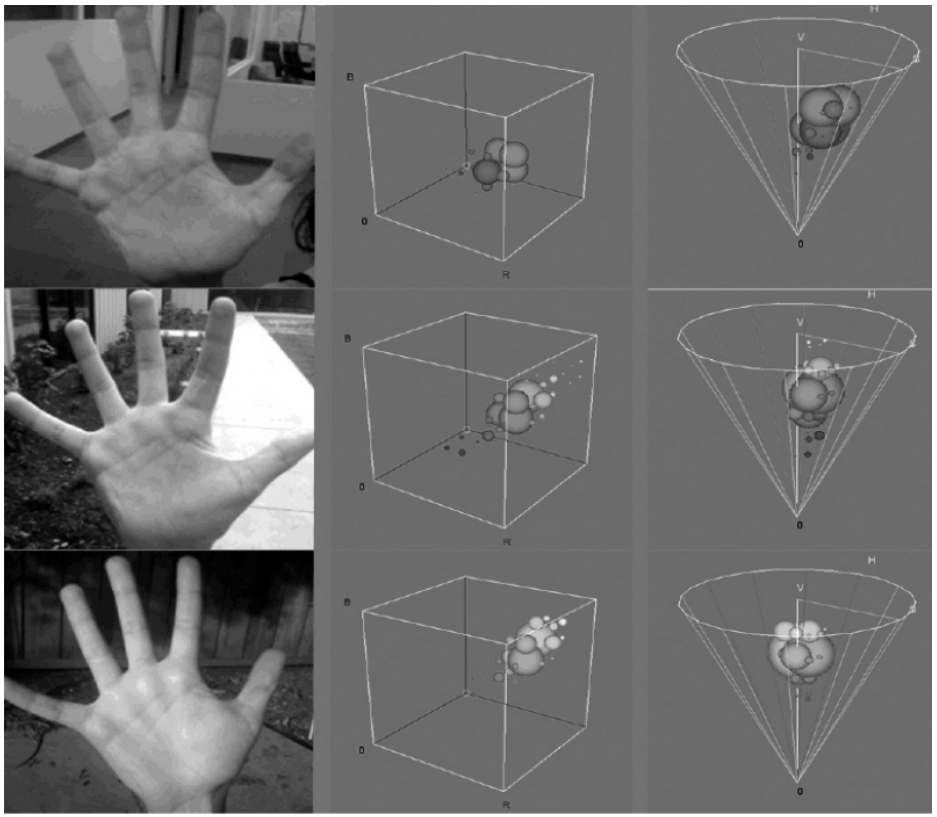
Comparison of distance measures

[Prednasky, Bratislava]

1. Skin colour in three illuminations
   - Indoors
   - Outdoors
   - Artificial light

2. Centre: RGB

3. Right: HSV colour model



Credit: Markus Vincze, Technische Universität Wien

Colour Histogram Matching

Comparison for the hand with the upper part of the hand indoors

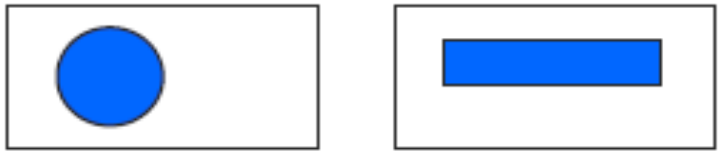
Perfect agreement:

	1.0	0.0	1.0	0.0
Comparison	CORREL	CHISQR	INTERSECT	BHATTACHARYYA
Indoor lower half	0.96	0.14	0.82	0.2
Outdoor shade	0.09	1.57	0.13	0.8
Outdoor sun	–0.0	1.98	0.01	0.99

Credit: Markus Vincze, Technische Universität Wien

Colour Histogram Matching

- Efficient, simple, rotation and translation invariant
- Requires segmentation from background or costly search over entire image
- Different images/objects can have the same colour histogram
- Good if few colours on object



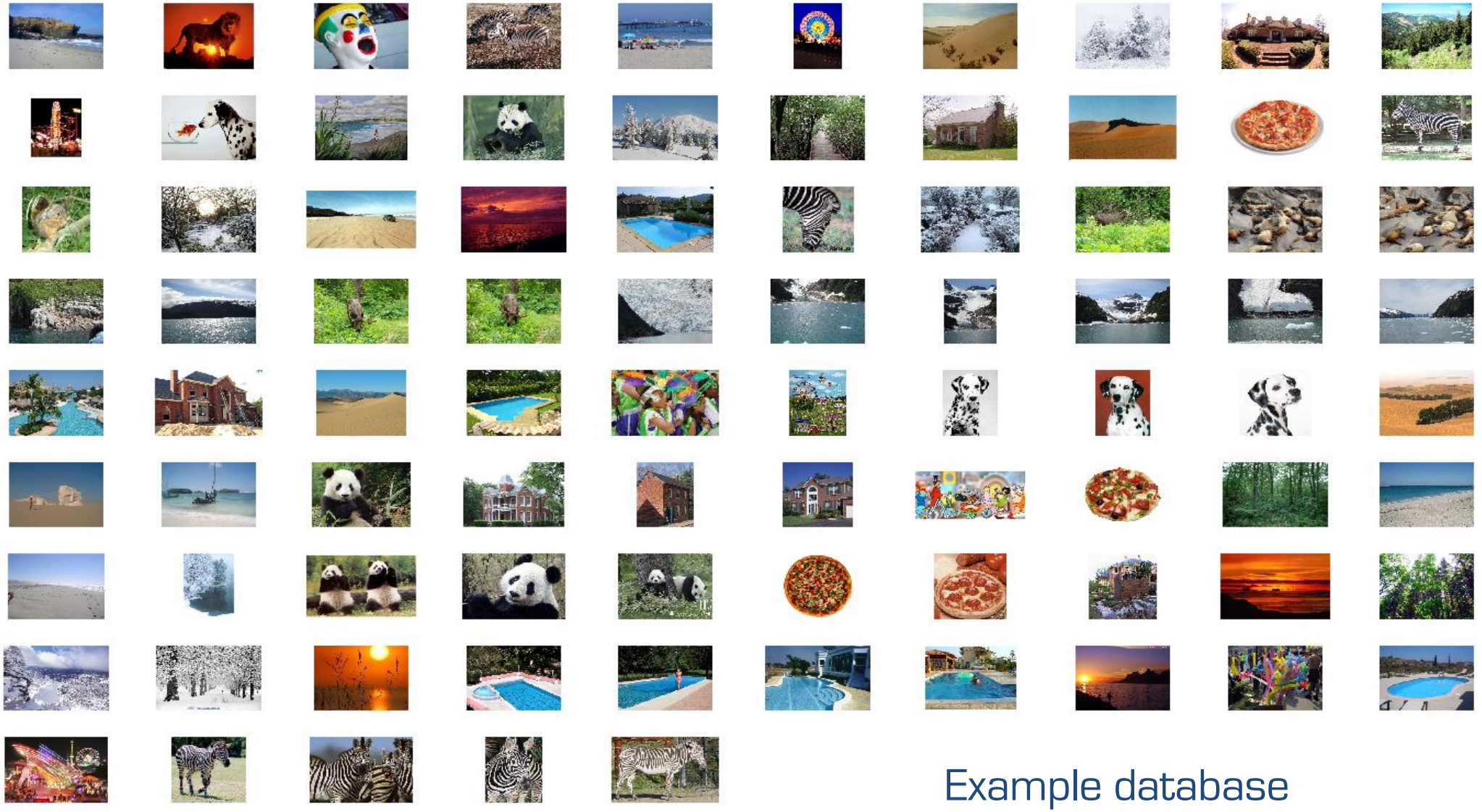
- Bad with texture
- Many parameters: bins, colour model, ...

Credit: Markus Vincze, Technische Universität Wien

Colour Histogram Matching

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

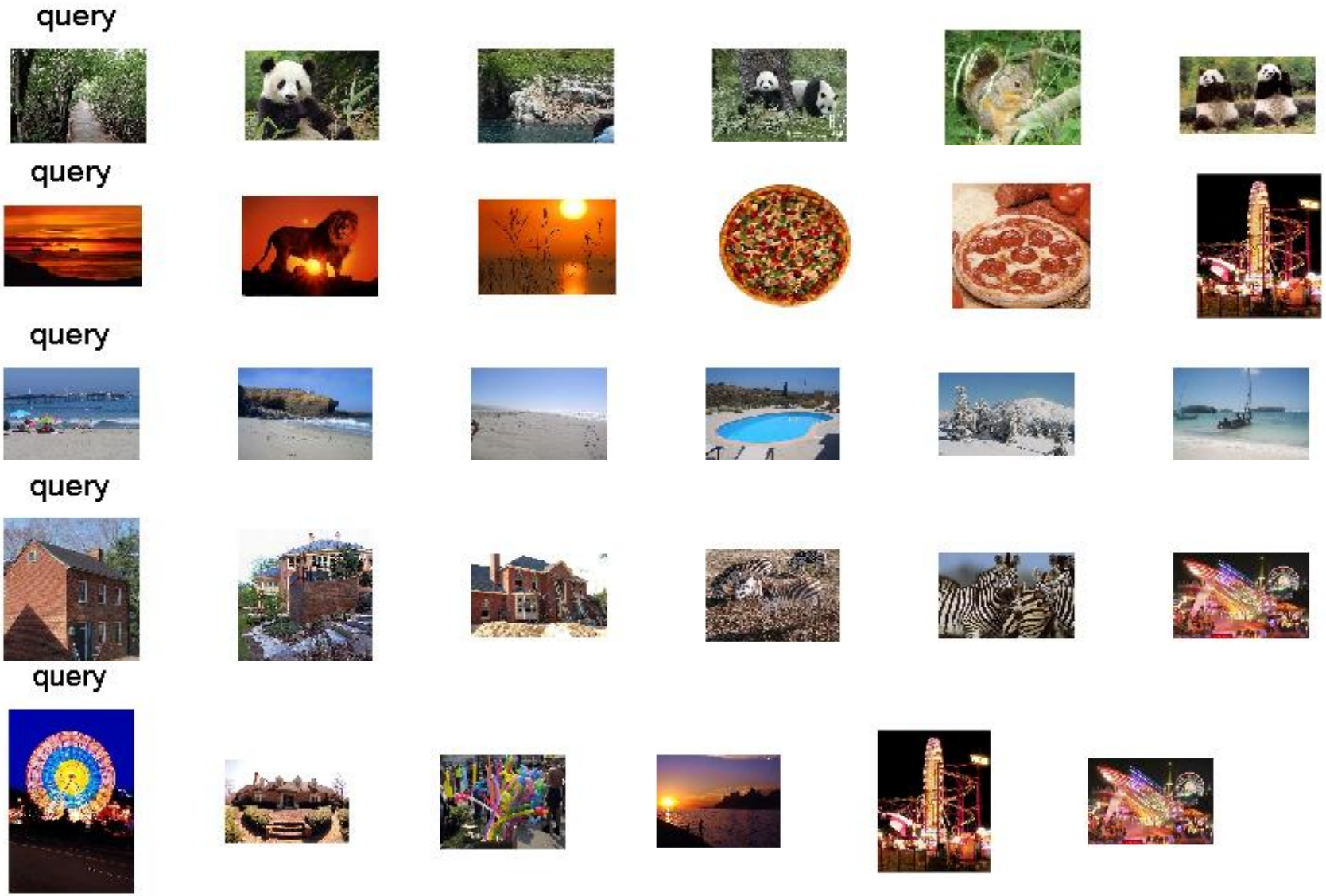
Colour Histogram Matching



Example database

Credit: Markus Vincze, Technische Universität Wien

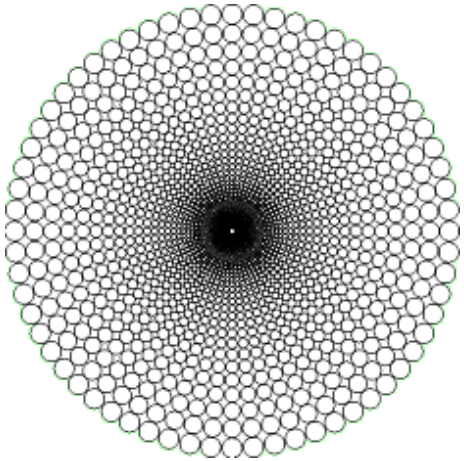
Colour Histogram Matching



Example retrievals

Credit: Markus Vincze, Technische Universität Wien

Colour Histogram Matching with Log-polar Images



Credit: Markus Vincze, Technische Universität Wien

Colour Histogram Backprojection

Backproject a (normalized) colour histogram onto an image

Produces a image indicating where that image best matches the colour distribution in the histogram

Image of faces

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Colour Histogram Backprojection

Histogram Backprojection (Swain and Ballard 1991)

Let $H(c)$ be the histogram function that maps
a colour c (a three-dimensional value)
to a histogram bin (a three-dimensional value)

Let D^r be a disk of radius r

$$D^r_{x,y} = \begin{cases} 1 & \text{if } (x^2 + y^2)^{1/2} < r \\ 0 & \text{otherwise} \end{cases}$$

Colour Histogram Backprojection

Histogram Backprojection (Swain and Ballard 1991)

Define the "loc" function to return to pixel (x, y) the value of its argument

Let M and I be the multi-dimensional colour histogram of a model and search image, respectively

Define a ratio histogram R: $R_i = \min(M_i / I_i, 1)$

Colour Histogram Backprojection

For each histogram bin j

$$R_j = \min(M_j / I_j, 1)$$

For each x, y // for all pixels in the image

$b_{x,y} = R_{h(c_{x,y})}$ // get the ratio histogram value for that pixel
// and add it to a match image

$b = D^r * \mathrm{b}$ // Convolve match image with disk approximately
//same size as object

$(x_t, y_t) = \mathrm{loc}(\max_{x,y} b_{x,y})$ // find the coordinates of the maximum

Demos

The following code is taken from the colourhistogramMatching project in the lectures directory of the ACV repository

See:

```
colourhistogramMatching.h
colourhistogramMatchingImplementation.cpp
colourhistogramMatchingApplication.cpp
```

```
/*
  Example use of openCV to perform colour histogram matching
  ----------------------------------------------------------

  Read a list of image filenames from the input file and compute the histogram match value for all pairs of images.

  The matrix of results is written to the output file.

  A 2-D array of images thumbnails is also generated, with the thumbnail images in each row ordered by how well they
  match the first thumbnail image on that row. This array of thumbnails is then displayed.

  David Vernon
  31 May 2017
*/

#include "colourHistogramMatching.h"

int main() {

   int  end_of_file;
   bool debug = false;
   char filename[MAX_FILENAME_LENGTH];
   char image_filenames[MAX_NUMBER_OF_IMAGES][MAX_FILENAME_LENGTH];
   int  numberOfImages;
   int  i,j, k;

   FILE *fp_in;
   FILE *fp_out;
```

```
Mat inputImage;
Mat images[MAX_NUMBER_OF_IMAGES];
Mat thumbnailImage;
Mat comparisonImage;

comparisonType similarity[MAX_NUMBER_OF_IMAGES][MAX_NUMBER_OF_IMAGES];
comparisonType temp;

char inputWindowName[MAX_STRING_LENGTH]              = "Test Image";
char outputWindowName[MAX_STRING_LENGTH]             = "Similarity Images";

namedWindow(inputWindowName,   CV_WINDOW_AUTOSIZE);
namedWindow(outputWindowName,  CV_WINDOW_AUTOSIZE);

if ((fp_in = fopen("../data/colourHistogramMatchingInput.txt","r")) == 0) {
  printf("Error can't open input colourHistogramMatchingInput.txt\n");
  prompt_and_exit(1);
}

if ((fp_out = fopen("../data/colourHistogramMatchingOutput.txt","w")) == 0) {
  printf("Error can't open output colourHistogramMatchingOutput.txt\n");
  prompt_and_exit(1);
}

printf("Example of how to use openCV to perform colour histogram matching.\n\n");
```

```
printf("Example of how to use openCV to perform colour histogram matching.\n\n");

/* read the list of images */

numberOfImages = 0;

do {
   end_of_file = fscanf(fp_in, "%s", filename);

   if ((numberOfImages < MAX_NUMBER_OF_IMAGES) && (end_of_file != EOF)) {
      if (debug) printf ("Reading image %s\n",filename);

      strcpy(image_filenames[numberOfImages], filename);              // store the filename

      /* read this image file */
      images[numberOfImages] = imread(filename, CV_LOAD_IMAGE_COLOR);  // Read the image

      if (!(images[numberOfImages].data)) {                           // Check for invalid input
        printf("Error: failed to read image %s\n",filename);
        prompt_and_exit(-1);
      }
      numberOfImages++;
   }
} while (end_of_file != EOF);
```

```
/* build the image of thumbnail versions of the input images */
thumbnailImage.create(Size(THUMBNAIL_IMAGE_ROWS, THUMBNAIL_IMAGE_COLS), CV_8UC3);

for (i=0; i<numberOfImages; i++) {
   insertThumbnailImage(images[i], thumbnailImage, numberOfImages, i);
}

imshow(inputWindowName,  thumbnailImage);
waitKey(30);

/* build a comparison matrix by comparing all pairs of images */
for (i=0; i<numberOfImages; i++) {
   for (j=0; j<numberOfImages; j++) {
      similarity[i][j].image_number = j; // the column number is the number of the image being matched for this row
      similarity[i][j].match_value = colourHistogramMatching(images[i], images[j]);
   }
}

/* write the similarity values to the output file */
for (i=0; i<numberOfImages; i++) {
   fprintf(fp_out, "%30s ", image_filenames[i]);
   for (j=0; j<numberOfImages; j++) {
      fprintf(fp_out, "%5.2f ",similarity[i][j].match_value);
   }
   fprintf(fp_out, "\n");
}
```

```
/* sort each row by match value, in the order required by the comparison metric */
for (k=0; k < numberOfImages; k++) { // for each row
   for (i=0; i < numberOfImages-1; i++) {  // bubble sort
      for (j=i; j >= 0; j--) {
         if (similarity[k][j].match_value >= similarity[k][j+1].match_value) { // choose relational operator to match comparison me
            temp = similarity[k][j+1]; // swap
            similarity[k][j+1] = similarity[k][j];
            similarity[k][j] = temp;
         }
      }
   }
}

/* write the sorted similarity values to the output file */
fprintf(fp_out, "\n");
for (i=0; i<numberOfImages; i++) {
   fprintf(fp_out, "%30s ", image_filenames[i]);
   for (j=0; j<numberOfImages; j++) {
      fprintf(fp_out, "%2d: %4.2f ",similarity[i][j].image_number, similarity[i][j].match_value);
   }
   fprintf(fp_out, "\n");
}
```

```
    /* build the image of 2-D matrix of thumbnail versions of the input images */
    comparisonImage.create(Size(THUMBNAIL_IMAGE_ROWS, THUMBNAIL_IMAGE_COLS), CV_8UC3);

    for (i=0; i<numberOfImages; i++) {
       for (j=0; j<numberOfImages; j++) {

          k = i * numberOfImages + j;
          insertThumbnailImage(images[ similarity[i][j].image_number ], comparisonImage, numberOfImages*numberOfImages, k);

       }
    }

    imshow(inputWindowName,  thumbnailImage);
    imshow(outputWindowName, comparisonImage);

    do{
       waitKey(30);
    } while (!_kbhit());

    getchar(); // flush the buffer from the keyboard hit

    fclose(fp_in);
    fclose(fp_out);

    destroyWindow(inputWindowName);
    destroyWindow(outputWindowName);

    return 0;
}
```

```
/*****************************************************************************************************
colourHistogramMatching()

Compute the similarity of two images using colour histogram matching

David Vernon
31 May 2017
*****************************************************************************************************/

double colourHistogramMatching(Mat &image1, Mat &image2) {

   Mat    hls_image1;
   Mat    hls_image2;
   double matching_score;

   cvtColor(image1, hls_image1, CV_BGR2HLS);
   cvtColor(image2, hls_image2, CV_BGR2HLS);

   ColourHistogram histogram1(hls_image1,4);
   ColourHistogram histogram2(hls_image2,4);

   histogram1.NormaliseHistogram();
   histogram2.NormaliseHistogram();

   /* CV_COMP_CORREL: score of 1 implies perfect match - reflect this when sorting the matches in the application file */
   //matching_score = compareHist(histogram1.getHistogram(),histogram2.getHistogram(),CV_COMP_CORREL);

   /* CV_COMP_BHATTACHARYYA: score of 0 implies perfect match - reflect this when sorting the matches in the application file */
   matching_score = compareHist(histogram1.getHistogram(),histogram2.getHistogram(),CV_COMP_BHATTACHARYYA);

   /* CV_COMP_INTERSECT: the bigger the score the better the match - reflect this when sorting the matches in the application file */
   //matching_score = compareHist(histogram1.getHistogram(),histogram2.getHistogram(),CV_COMP_INTERSECT);

   return(matching_score);
}
```

Demos

The following code is taken from the colourhistogramBackprojection project in the lectures directory of the ACV repository

See:

```
colourhistogramBackprojection.h
colourhistogramBackprojectionImplementation.cpp
colourhistogramBackprojectionApplication.cpp
```

```
void colourHistogramBackprojection(char *referenceImageFilename, char *targetImageFilename) {

   char referenceWindowName[MAX_STRING_LENGTH]  = "Reference Image";
   char targetWindowName[MAX_STRING_LENGTH]     = "Target Image";
   char outputWindowName[MAX_STRING_LENGTH]     = "Backprojection Image";

   Mat referenceImage;
   Mat targetImage;
   Mat backprojectionImage;
   Mat hlsImage;
   Mat back_projection_probabilities_display;
   std::vector<cv::Mat> input_planes(3);

   const int* channel_numbers = { 0 };
   float channel_range[] = { 0.0, 255.0 };
   const float* channel_ranges = channel_range;
   int numberOfBins = 64;

   namedWindow(referenceWindowName,   CV_WINDOW_AUTOSIZE);
   namedWindow(targetWindowName,      CV_WINDOW_AUTOSIZE);
   namedWindow(targetWindowName,      CV_WINDOW_AUTOSIZE);

   referenceImage = imread(referenceImageFilename, CV_LOAD_IMAGE_COLOR); // Read the file

   if (!referenceImage.data) {                                          // Check for invalid input
      printf("Error: failed to read image %s\n",referenceImageFilename);
      prompt_and_exit(-1);
   }

   targetImage = imread(targetImageFilename, CV_LOAD_IMAGE_COLOR); // Read the file
```

```
    if (!targetImage.data) {                                    // Check for invalid input
        printf("Error: failed to read image %s\n",targetImageFilename);
        prompt_and_exit(-1);
    }

    printf("Press any key to continue ...\n");

    CV_Assert(referenceImage.type() == CV_8UC3 );    // colour image
    CV_Assert(targetImage.type()    == CV_8UC3 );    // colour image

    cvtColor(referenceImage, hlsImage, CV_BGR2HLS);

    ColourHistogram histogram3D(hlsImage,8);
    histogram3D.NormaliseHistogram();

    cvtColor(targetImage, hlsImage, CV_BGR2HLS);

    Mat back_projection_probabilities = histogram3D.BackProject(hlsImage);

    back_projection_probabilities = StretchImage(back_projection_probabilities);

    cvtColor(back_projection_probabilities, back_projection_probabilities_display, CV_GRAY2BGR);

    imshow(referenceWindowName, referenceImage);
    imshow(targetWindowName,    targetImage);
    imshow(outputWindowName,    back_projection_probabilities_display);


    do{
        waitKey(30);
    } while (!_kbhit());

    getchar(); // flush the buffer from the keyboard hit

    destroyWindow(referenceWindowName);
    destroyWindow(targetWindowName);
    destroyWindow(outputWindowName);
}
```

Reading

A. Hanbury, “The Taming of the Hue, Saturation, and Brightness Colour Space”, Proc. Computer Vision Winter Workshop (CVWW), Austria, 2002.

M. Swain and D. Ballard. “Color indexing:, International Journal of Computer Vision, 7(1):11-32, 1991.

M. J. Swain and D. H. Ballard. ”Indexing via colour histograms”, International Conference on Computer Vision, ICCV90, pp. 390-393, 1990.

.