HSV Histogram Calculation and Comparison on Video Stream

```cpp
#include <iostream>
#include "opencv2/opencv.hpp"

cv::Mat grab_an_image_from_camera() {
    // Start time
    int64 t0 = cv::getTickCount();
    cv::VideoCapture cap_template;
    // open the default camera
    cap_template.open(0);

    cv::Mat frame_template;
    cv::namedWindow("livepreview", 1);

    for (;;) {
        // Current time
        int64 t1 = cv::getTickCount();
        double timeRunning = static_cast<double>(t1-t0) / cv::getTickFrequency();
        if (timeRunning >= 10.0) {
            // wait 10 seconds before capture an image from the camera
            cap_template >> frame_template;
            cap_template.release();
            break;
        }

        cv::Mat mat_live_preview;
        cap_template >> mat_live_preview;
        imshow("livepreview", mat_live_preview);

        // preview a video at 30 ms per frame
        if (cv::waitKey(30) >= 0) break;
    }

    return frame_template;
}

cv::Mat load_template_image() {
    // Load template image and convert it from BGR to HSV
    cv::Mat sourceImage;
    cv::Mat outputImage;
    sourceImage = cv::imread("./ait.jpg", cv::IMREAD_COLOR);
    cv::cvtColor(sourceImage, outputImage, cv::COLOR_BGR2HSV);

    return outputImage;
}

cv::Mat construct_hsv_hist(cv::Mat matImage) {
    std::vector<cv::Mat> hsv_planes;
    split(matImage, hsv_planes);

    int histSize = 256;
    float range[] = {0, 255};
    const float* histRange = {range};
```

```
    cv::Mat h_hist, s_hist, v_hist;

    calcHist(&hsv_planes[0], 1, 0, cv::Mat(), h_hist, 1, &histSize, &histRange);
    calcHist(&hsv_planes[1], 1, 0, cv::Mat(), s_hist, 1, &histSize, &histRange);
    calcHist(&hsv_planes[2], 1, 0, cv::Mat(), v_hist, 1, &histSize, &histRange);

    // Draw the histograms for B, G and R
    int hist_w = 512;
    int hist_h = 400;
    int bin_w = cvRound(static_cast<double>(hist_w / histSize));
    cv::Mat histImage(hist_h, hist_w, CV_8UC3, cv::Scalar(0, 0, 0));
    // Normalize the result to [ 0, histImage.rows ]
    normalize(h_hist, h_hist, 0, histImage.rows, cv::NORM_MINMAX, -1, cv::Mat());
    normalize(s_hist, s_hist, 0, histImage.rows, cv::NORM_MINMAX, -1, cv::Mat());
    normalize(v_hist, v_hist, 0, histImage.rows, cv::NORM_MINMAX, -1, cv::Mat());
    // Draw for each channel
    for (int i = 1; i < histSize; i++) {
        line(histImage, cv::Point(bin_w * (i - 1), hist_h - cvRound(h_hist.at<float>(i - 1))),
             cv::Point(bin_w * (i), hist_h - cvRound(h_hist.at<float>(i))),
             cv::Scalar(255, 0, 0), 2, 8, 0);
        line(histImage, cv::Point(bin_w * (i - 1), hist_h - cvRound(s_hist.at<float>(i - 1))),
             cv::Point(bin_w * (i), hist_h - cvRound(s_hist.at<float>(i))),
             cv::Scalar(0, 255, 0), 2, 8, 0);
        line(histImage, cv::Point(bin_w * (i - 1), hist_h - cvRound(v_hist.at<float>(i - 1))),
             cv::Point(bin_w * (i), hist_h - cvRound(v_hist.at<float>(i))), cv::Scalar(0, 0, 255), 2, 8, 0);
    }

    return histImage;
}

double calculateSimilarity(cv::Mat matSrc, cv::Mat matDst) {
    // Declare channels and bin size
    int channels[] = {0, 1, 2};
    int h_bins = 60;
    int s_bins = 60;
    int v_bins = 60;
    int histSize[] = {h_bins, s_bins, v_bins};
    float h_ranges[] = {0, 255};
    float s_ranges[] = {0, 255};
    float v_ranges[] = {0, 255};
    const float *ranges[] = {h_ranges, s_ranges, v_ranges};
    // Use the o-th and 1-st and 2-nd channels int channels[] = { 0, 1, 2 };
    cv::Mat histSrc, histDst;
    // Calculate the histograms
    calcHist(&matSrc, 1, channels, cv::Mat(), histSrc, 2, histSize, ranges);
    normalize(histSrc, histSrc, 0, 1, cv::NORM_MINMAX, -1, cv::Mat());
    calcHist(&matDst, 1, channels, cv::Mat(), histDst, 2, histSize, ranges);
    normalize(histDst, histDst, 0, 1, cv::NORM_MINMAX, -1, cv::Mat());

    // method: 3, HISTCMP_BHATTACHARYYA (from imgproc.hpp)
    // Bhattacharyya distance
    // (In fact, OpenCV computes Hellinger distance, which is related to Bhattacharyya coefficient.)
    return compareHist(histSrc, histDst, 3);
}
```

```cpp
// HSV histogram calculation and comparison on video stream
int main(int argc, char *argv[]) {
    // 2) At the beginning of the program before infinity loop for grabbing frame from USB or network camera,
    // you need to load one color image and convert this image into HSV color space,
    // and then construct HSV histogram
    // and keep it as a template.
    cv::Mat image_template;
    image_template = load_template_image();
    cv::imwrite("./image_template_hsv.jpg", image_template);
    cv::Mat hsv_hist;
    hsv_hist = construct_hsv_hist(image_template);
    cv::imwrite("./image_template_hist.jpg", hsv_hist);

    // 1) Create a program to grab frame from USB camera or network camera connected to your computer
    // and display the grabbed frame.
    cv::Mat grabbed_frame;
    grabbed_frame = grab_an_image_from_camera();

    // 3) Every grabbed frame must be converted to HSV color space
    // and must be extracted HSV histogram with the same number of bins used in step 2).
    // In this step, you can use different numbers of channel for calculating histogram similarity
    // such as using both H and S for calculating histogram similarity,
    // or using only H for calculating histogram similarity,
    // or using all H, S, and V for calculating histogram similarity.
    cv::Mat grabbed_frame_hsv;
    cv::cvtColor(grabbed_frame, grabbed_frame_hsv, cv::COLOR_BGR2HSV);
    cv::Mat grabbed_image_hist;
    grabbed_image_hist = construct_hsv_hist(grabbed_frame_hsv);
    cv::imwrite("./grabbed_image_hsv.jpg", grabbed_frame_hsv);
    cv::imwrite("./grabbed_image_hist.jpg", grabbed_image_hist);
    double similarity = 1.0 - calculateSimilarity(image_template, grabbed_frame_hsv);

    // 4) Compare similarity
    // between template HSV histogram
    // and HSV histogram of frame grabbed from camera.
    // Whenever the HSV histogram similarity is less than some thresholds,
    // such as similarity is greater than 75%,
    // you need to save that frame into your hard disk.
    if (similarity > 0.5) {
        std::cout << "Similarity: " << similarity * 100.0 << "%" << std::endl;
        std::cout << "Saved the similar image to disk" << std::endl;
        cv::imwrite("./similar_image.jpg", grabbed_frame);
    } else {
        std::cout << "Similarity is less than 50%" << std::endl;
    }

    return 0;
}
```