```
/**
 * OpenCV video streaming over TCP/IP
 * Server: Captures video from a webcam and send it to a client
 * by Isaac Maia
 */

#include "opencv2/opencv.hpp"
#include <iostream>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <sys/ioctl.h>
#include <net/if.h>
#include <unistd.h>
#include <string.h>

using namespace cv;

void *display(void *);

int capDev = 0;

    VideoCapture cap(capDev); // open the default camera




int main(int argc, char** argv)
{

    //--------------------------------------------------------
    //networking stuff: socket, bind, listen
    //--------------------------------------------------------
    int                 localSocket,
                        remoteSocket,
                        port = 4097;

    struct  sockaddr_in localAddr,
                        remoteAddr;
    pthread_t thread_id;


    int addrLen = sizeof(struct sockaddr_in);


    if ( (argc > 1) && (strcmp(argv[1],"-h") == 0) ) {
          std::cerr << "usage: ./cv_video_srv [port] [capture device]\n" <<
```

```
                  "port           : socket port (4097 default)\n" <<
                  "capture device : (0 default)\n" << std::endl;

      exit(1);
}

if (argc == 2) port = atoi(argv[1]);

localSocket = socket(AF_INET , SOCK_STREAM , 0);
if (localSocket == -1){
     perror("socket() call failed!!");
}

localAddr.sin_family = AF_INET;
localAddr.sin_addr.s_addr = INADDR_ANY;
localAddr.sin_port = htons( port );

if( bind(localSocket,(struct sockaddr *)&localAddr , sizeof(localAddr)) < 0) {
     perror("Can't bind() socket");
     exit(1);
}

//Listening
listen(localSocket , 3);

std::cout <<  "Waiting for connections...\n"
          <<  "Server Port:" << port << std::endl;

//accept connection from an incoming client
while(1){
//if (remoteSocket < 0) {
//    perror("accept failed!");
//    exit(1);
//}

 remoteSocket = accept(localSocket, (struct sockaddr *)&remoteAddr, (socklen_t*)&addrLen);
  //std::cout << remoteSocket<< "32"<< std::endl;
if (remoteSocket < 0) {
    perror("accept failed!");
    exit(1);
}
std::cout << "Connection accepted" << std::endl;
 pthread_create(&thread_id,NULL,display,&remoteSocket);

 //pthread_join(thread_id,NULL);

}
```

```
    //pthread_join(thread_id,NULL);
    //close(remoteSocket);

    return 0;
}

void *display(void *ptr){
    int socket = *(int *)ptr;
    //OpenCV Code
    //----------------------------------------------------------

    Mat img, imgGray;
    img = Mat::zeros(480 , 640, CV_8UC1);
     //make it continuous
    if (!img.isContinuous()) {
        img = img.clone();
    }

    int imgSize = img.total() * img.elemSize();
    int bytes = 0;
    int key;


    //make img continuos
    if ( ! img.isContinuous() ) {
          img = img.clone();
          imgGray = img.clone();
    }

    std::cout << "Image Size:" << imgSize << std::endl;

    while(1) {

            /* get a frame from camera */
                cap >> img;

                //do video processing here
                cvtColor(img, imgGray, CV_BGR2GRAY);

                //send processed image
                if ((bytes = send(socket, imgGray.data, imgSize, 0)) < 0){
                     std::cerr << "bytes = " << bytes << std::endl;
                     break;
                }
    }

}
```