1 Visual Studio

UTD students can get this software for free. See: https://www.dreamspark.com.
Computer Science students are also eligible for DreamSpark Premium. It can be accessed in the following link to DreamSpark Premium.

2 Setting up Visual Studio for OpenCV

The main link to OpenCV is:

www.opencv.org.

The installation that we support is the one distributed by Ceemple, which comes compiled and ready to use in visual studio. Follow this link for additional information:

http://opencv.org/ceemple-opencv-distribution-for-visual-studio.html

The installer can be downloaded here.
The advantage of using the Ceemple distribution is that it includes all the necessary OpenCV libraries. The only disadvantage is that it compiles the project into a Win32 application while we will typically need a Console application. The instructions below show how to properly set up the console mode with Ceemple.

3 Example programs

3.1 Display an image

The program below reads an displays an image called "fruits.jpg". Assume that the file containing the program is called example1.cpp.

```
#include "opencv2/opencv.hpp"
#include "opencv2/highgui.hpp"
#include <iostream>
using namespace cv;
using namespace std;

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

  Mat image = imread("fruits.jpg", CV_LOAD_IMAGE_UNCHANGED);  // Read
  if(image.empty()) { // invalid input
   cerr << "Couldn't open or find the image " << "fruits.jpg" << endl;
   return(-1);
  }

  namedWindow("Display window",CV_WINDOW_AUTOSIZE); // create window
  imshow( "Display window", image); // Show image inside it.

  waitKey(0); // Wait for a keystroke
  destroyWindow("Display window");
}
```

Here are detailed steps that can be followed to compile and run it within visual studio and Ceemple.

1. Create a project by pointing to "New" on the "File menu", and then clicking "Project".

2. In the "Visual C++" project types pane, click "Ceemple OpenCV Project".

3. Type a name for the project. For example, "example1".

4. By default, the solution that contains the project has the same name as the project, but you can type a different name. You can also type a different location for the project. We assume that the selected project location is "MyProjFolder".

5. Click OK to create the project.

6. For the following steps we need the "Solution Explorer". If it doesn't show, click "Solution Explorer" on the "View" menu.

7. The following steps make sure a Console App is created:

- In the "Solution Explorer" right-click on the project in question (in our example it is example1) and select "properties".
- Under "Configuration Properties", select "Linker→System".
- Change the "SubSystem" option from Windows to "Console (/SUBSYSTEM:CONSOLE)".

For additional information see this link.

8.1. a. In "Solution Explorer", right-click the "Source Files" folder, point to "Add", and then click "New Item".

b. In the "Code" node, click "C++ File (.cpp)", type a name for the file, for example, "example1", and then click "Add".

c. The .cpp file appears in the "Source Files" folder in "Solution Explorer", and the file opens in the Visual Studio editor.

d. Type or paste the program.

e. Save the file.

8.2. This is an alternative to 8.1.

a. In "Solution Explorer", right-click the "Source Files" folder, point to "Add", and then click "Existing Item".

b. Navigate to the location of the file example1.cpp and select it.

c. Double click on the file example1.cpp in the "Solution Explorer". The file is opened in the Visual Studio editor.

8.3. This is an alternative to 8.1, 8.2.

a. Copy the source file (example1.cpp) to the project folder "MyProjFolder".

b. In "Solution Explorer", right-click the Source Files folder, point to Add, and then click Existing Item.

c. Double click on the file example1.cpp in the "Solution Explorer". The file is opened in the Visual Studio editor.

9. On the "Build" menu, click "Build Solution".

10 Copy an image file fruits .jpg to the project folder "MyProjFolder".

11. On the "Debug" menu, click "Start without Debugging".

3.2 Display live video

The program below reads from the default camera of the computer (if a camera is available).

```
#include "opencv2/opencv.hpp"
#include "opencv2/highgui.hpp"
#include <iostream>
using namespace cv;
using namespace std;

int main(int argc, char **argv)
{
  VideoCapture videocapture(0); // open the default camera
  if(!videocapture.isOpened()) {
    cout << "Can't open default video camera" << endl;
    return(-1);
  }
  namedWindow("video", CV_WINDOW_AUTOSIZE);

  Mat frame;
  bool showframe = true;
  while(showframe) {
    if(!videocapture.read(frame)) {
      cout << "Can't capture frame" << endl;
      break;
    }
    imshow("video", frame);
    if(waitKey(30) >= 0) showframe = false;
  }
  return(0);
}
```