# Install Instructions

## Overview

The videos in Chapter 1 cover how to install OpenCV for a variety of platforms. In many cases, there are long URLs or commands that are easy to copy and paste rather than write out by hand. Alternatively, if you prefer to not follow the videos at all, you can use this guide to get up and running with a working OpenCV install in no time.

## Install on macOS

The following steps work on both Intel and Apple silicon chips.

1. Read and accept the Xcode license.
   ```
   sudo xcodebuild -license
   ```
2. Install the Xcode command-line tools.
   ```
   sudo xcode-select --install
   ```
3. Install Homebrew by copying the install command from https://brew.sh/, which at the time of writing is:
   ```
   /bin/bash -c "$(curl -fsSL
   https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
   ```
4. Run `brew update` to ensure you're up to date.
5. Install Python 3, or a specific version of Python; if you already have a recent version of Python 3 (such as 3.7 or higher), you can skip the next two steps.
   ```
   brew install python3  # any latest version of python
   brew install python3.9 # A specific major version of python
   ```

Want to manage multiple installs of Python? You can use Brew to install specific versions, or you could also use pyenv. You can install pyenv with brew as well.

```
brew install pyenv
```

6. Restart your shell, then validate you have the version expected.
   ```
   python3 --version
   ```

It's a good idea to also validate where this version was installed to see if it is indeed coming from a brew.

```
which python3
```

7. Create a virtual environment, in this example named "opencv."
   ```
   python3 -m venv ~/opt/opencv
   ```

8. Activate your virtual environment from anywhere on your machine (now or in the future).
   `source ~/opt/opencv/bin/activate`

9. Optional: Add an alias to your Bash profile (e.g. ~/.zshenv) to quickly run this command in the future.
   `alias activate_opencv='source ~/opt/opencv/bin/activate'`

10. Make sure your environment is active (it should have the OpenCV prefix in the command prompt, or whatever you named your environment).

11. Now, update pip as needed.
   `pip install --upgrade pip`

12. Finally, install OpenCV.
   `pip install opencv-contrib-python`

## Install on Windows

The following steps will install Python and OpenCV into a recent version of Windows. There are many ways to install Python, but for this module we showed how to install it using Anaconda. This has the benefit of easily managing your Python environments through the conda commands later on.

1. Download the anaconda installer from https://www.anaconda.com/ - note will start a download size in the 100s of MBs.

2. Start the exe, reading the license.

3. Decide to install for yourself or for the overall system (Admin rights needed).

4. Choose and remember where you are installing Anaconda.

5. If you don't check the box for "Add to environment path", then you will need to add three path environment variables to your system.

6. Open the Anaconda command prompt, which was newly installed.

7. Create a new conda environment, in this example named "opencv."
   `conda create -n opencv`

8. Activate your virtual environment from anywhere on your machine (now or in the future).
   `conda activate opencv`

9. Make sure your environment is active (it should have the OpenCV prefix in the command prompt, or whatever you named your environment).

10. Check that you have a functioning recent version of Python available (such as 3.10).
   `python3 --version`

11. Now, update pip as needed.
   `pip install --upgrade pip`

12. Finally, install OpenCV.
    `pip install opencv-contrib-python`

## Install on Linux (The Quick Way)

Note: This method of installing is not represented in its own video, as the Linux install video is a demo of building in source. In most cases, this is not necessary if you can manage to use pre-existing builds with pip.

1. Ensure you have a version of Python 3 installs.
   `python3 –version`

If not found, install Python.
`sudo apt-get install python3`

2. Create a virtual environment named “opencv.”
   `python3 –m venv ~/opt/opencv`

In the event your console complains that virtualenv is not installed, run the suggested command it provides for the version of Python you have locally, and then try the line above again. See an example here:
`sudo apt install python3.10-venv`

3. Activate your virtual environment from anywhere on your machine (now or in the future).
   `source ~/opt/opencv/bin/activate`

4. Optional: Add an alias to your Bash profile (e.g. ~/.bashrc) to quickly run this command in the future.
   `alias activate_opencv='source ~/opt/opencv/bin/activate'`

5. Make sure your environment is active (it should have the OpenCV prefix in the command prompt, or whatever you named your environment).

6. Now, update pip as needed.
   `pip install ––upgrade pip`

7. Finally, install OpenCV.
   `pip install opencv-contrib-python`

## Building from Source (on Linux)

The following steps demonstrate how to build OpenCV from source files. When you are working on a platform where existing binaries do not exist, your only option might be to build from source. We are building on Linux Ubuntu for this example, but the same general steps will apply to most platforms.

*Prepare dependencies*

1. Install build essentials.
   ```
   sudo apt-get install build-essential cmake git pkg-config
   ```

2. Install image formats.
   ```
   sudo apt-get install libjpeg-dev libtiff-dev libpng-dev openexr
   libopenexr-dev
   ```

3. Install video codecs.
   ```
   sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev
   libv4l-dev
   ```

4. Install libraries used for interface drawing features.
   ```
   sudo apt-get install libgtk-3-dev
   ```

5. Install commands to optimize OpenCV commands.
   ```
   sudo apt-get install libatlas-base-dev gfortran
   ```

6. There are some additional libraries you can optionally install.
   ```
   supt apt-get install python3-dev python3-numpy \
   libtbb2 libtbb-dev libdc1394-22-dev \
   libgstreamer-plugins-base1.0-dev libgstreamer1.0-dev \
   libxvidcore-dev libx264-dev
   ```

7. Check `python3 --version` to ensure you have a recent version of Python 3 (for example, 3.10).

8. Test that you have a version of pip installed or install it fresh if needed.
   ```
   sudo apt-get install python3-pip
   ```

9. Check that pip is using the same expected version of Python.
   ```
   pip --version
   ```

Alternate to make sure you are using the same pip as your Python.
```
python3 -m pip --version
```

10. Contrary to other install approaches shown in this course, notice that we aren't using a Python environment here, as we will ultimately be installing OpenCV for the overall system when linking builds and .so files. For OpenCV to install correctly, we need to manually install a core Python library with NumPy.
    ```
    sudo pip install numpy
    ```

*Run the install*

1. Create a folder to stage our intermediate files and builds.
   ```
   mkdir ~/opencv_build
   ```

2. Move into that folder.
   ```
   cd ~/opencv_build
   ```

3. Clone both libraries needed.
   ```
   git clone https://github.com/opencv/opencv.git
   git clone https://github.com/opencv/opencv_contrib.git
   ```

4. Using git tag, find the latest released version to use, then in each folder check out that tag.
   ```
   cd opencv
   git checkout 4.6.0
   cd ../opencv_contrib
   git checkout 4.6.0
   ```

5. Now move back into the main OpenCV git folder.
   ```
   cd ../opencv
   ```

6. Create and move into a build folder, where we will run the install.
   ```
   mkdir build
   cd build
   ```

7. Create a reference to the contributions module for use in the next step (notice we are selecting the modules subfolder).
   ```
   CONTRIB_PATH=~/opencv_build/opencv_contrib/modules
   ```

8. Formulate the cmake command, noting the last two dots represent how we're using the cmakelist in the directory up above this build folder.
   ```
   cmake -D CMAKE_BUILD_TYPE=RELEASE \
       -D CMAKE_INSTALL_PREFIX=/usr/local \
       -D INSTALL_C_EXAMPLES=ON \
       -D INSTALL_PYTHON_EXAMPLES=ON \
       -D OPENCV_GENERATE_PKGCONFIG=ON \
       -D OPENCV_EXTRA_MODULES_PATH=$CONTRIB_PATH \
       -D BUILD_EXAMPLES=ON ..
   ```

Notice the last two periods, indicating to cmake that the makelist file is in the directory above.

9. With the cmake cache ready, we are ready to run the primary install make command. Depending on your system, this step could take quite a while. The number here is up to you, but it's recommended to match or slightly exceed the number of cores on your system (use a tool like `nproc` if you're unsure).
   ```
   make -j4
   ```

10. Finally, run the faster install step, which should take a couple minutes at most.
    ```
    sudo make install
    ```

11. Finish linking any dependencies.
    ```
    sudo ldconfig
    ```

12. Validate the .so installs ended up in the correct place.
    ```
    ls l/usr/local/lib/python3.10/dist-packages/
    ```

13. Furthermore, use the "Test the install" steps below to further validate if the install worked.

## Test the Install

Regardless of how you install OpenCV, the following steps should be relevant.

1. If you set up an environment as a part of your install, activate it now (see operating system–specific commands above).
2. Start Python, using either the command **`python`** or **`python3`**
3. Validate the expected version of Python is printed in this interactive console, such as 3.10
4. Test importing NumPy.

   ```
   import numpy as np
   np.__version__
   ```

5. Import the cv2 library (the 2 is a misnomer and does not indicate the version of OpenCV installed). Validate that you get the expected version of OpenCV, such as 4.6.0.

   ```
   import cv2
   cv2.__version__
   ```

## Troubleshooting

If you find your Python command doesn't get you the version you expected, troubleshoot further by seeing which Python environment the command may be linked with.

```
which python3
```

If the above command doesn't yield you the Python executable path you expect, then you have a linking problem where your more recently installed Python executable is not being picked up.

Are you getting errors when trying to use conda commands after an Anaconda install? It is possible you need to upgrade pip, or patch pip with an extra dependency. In some narrow cases, this can be resolved with the following command:

```
pip install --upgrade pywin32==228
```

Continuing to have an error installing via Anaconda? You could try installing Python directly (https://www.python.org/downloads/windows/), and once installed, follow similar steps to the quick Linux approach where you create an environment using the Python command and install OpenCV via pip.

Make sure that the install of pip you are using matches the Python executable you are intending. If you are unsure, you can convert any pip install command to be a call of your Python executable.

```
pip install --upgrade pip → python -m pip install --upgrade pip
```

Be sure to activate your Python virtual environment if you used one to install OpenCV. Your current environment is local to a given console session and needs to be reactivated every time you open a new console.

Likewise, take care to ensure you run all of your pip install commands while active in a specific environment; otherwise, you may accidentally install the library globally on your machine.

You can tell an environment is active if there is a prefix in the command-line prompt, such as:

```
(opencv) $
```

If you encounter any errors, search for the specific error message online to find what others found as a solution.