**The State of Maryland had caught me red handed (or lead footed, rather) — doing 81mph in a 65mph zone.**

More to the point, *they had pictures to prove it!*

You see, I was caught by a camera positioned along the I-95 highway to catch speeding drivers, similar to the one pictured below:


(https://gurus.pyimagesearch.com/wp-content/uploads/2015/03/traffic_camera_example.jpg)

**FIGURE 1:** AN EXAMPLE OF A TRAFFIC CAMERA USED TO CAPTURE THE LICENSE PLATES OF SPEEDING MOTORISTS.

Once my car drove past over the speed limit, the camera snapped a photo of the rear of the car, which has a clear view of my license plate.

From there, an automatic license plate recognition computer vision system was used to analyze the photo of my car, extract my license plate number, look my information up in the State of Maryland database, and issue me a speeding ticket.

When I received that citation three weeks later, I couldn't help but chuckle to myself.

*I knew the exact computer vision algorithms they were using to detect and recognize my license plate* — but that couldn't stop me from getting a ticket!

I paid the ticket and moved on.

But the key takeaway here is that **Automatic Number Plate Recognition** is an **excellent example** of **computer vision systems being deployed into our everyday lives.**

# 6.1: What is ANPR?

I'd like to start our module on Automatic Number/License Plate Recognition with a little story.

About 2 years ago I was traveling back and forth between Maryland and Connecticut (which are both states on the east coast of the United States, separated by a couple hundred miles, for our international readers) for monthly work.

The main interstate highway that connects Maryland and Connecticut, along with pretty much the rest of the east coast of the United States, is I-95. There is always road-work happening on some parts of I-95 — not to mention, lots of traffic. Always. It's practically unavoidable.

One Friday, I finished up my work ahead of schedule and thought I would dip out a little early and try to beat the rush hour commute.

It was a nice Friday afternoon in Maryland that summer day. The sun high in the sky, but not too hot, around 85 degrees Fahrenheit (29 degrees Celsius), which is actually quite low for a Maryland summer.

Excited about the beautiful weather (and leaving early, of course), I lowered all the windows on my beat up old Honda Civic that I had since high school and took off down I-95, music blasting.

It was arguably ***the best*** drive I have ever had on I-95. No traffic. No roadwork to speak of.

...And then I received a letter in the mail from the Comptroller of Maryland three weeks later.

Just a quick note on terminology: Throughout this course I will be using the terms *Automatic Number Plate Recognition* and *Automatic License Plate Recognition* interchangeably. In the United States, we use the term *license plate*; however, overseas I know it's common to use the term *number plate*. I tend to prefer the term *license plate*, since *number plate* seems too constraining to just "digits" and not the characters of the English alphabet.

# Objectives:

In this lesson we will:

- Familiarize ourselves with the terms *Automatic Number Plate Recognition* and *Automatic License Plate Recognition*.
- Review the four steps of building *any* ANPR system:
    1. Acquisition of a photo
    2. Localization
    3. Segmentation
    4. Recognition

# What is ANPR?

At the very core, "Automatic License Plate Recognition" (ANPR) systems are used to automatically *detect* and *recognize* license plates in images. From there, the identified license plate can be used to look up information on the owner of the car.

ANPR systems are mainly used for automatic toll collection, and in the United States, the primary use of ANPR is within law enforcement to issue speeding citations.

ANPR systems tend to be **heavily region specific** on a state-by-state (or province-by-province) level. In the United States, each State has a different and unique license plate design. And as we travel across the world, we continue to see various forms of license plate designs.

**It's important to understand that there is *no one-size-fits-all solution* to ANPR!**

Instead, smaller, mini-ANPR systems are developed to localize and segment license plates from images on a state-by-state basis.

Some people may believe developing an ANPR system on a per-state basis is cheating or a limitation of computer vision as a whole. But I tend to disagree with this statement.

Any time we can apply *a priori* knowledge about our domain when solving *any* problem (whether it's computer vision, machine learning, or crocheting a scarf for our grandmother), that knowledge will help us solve the problem more effectively and more accurately.

For example, if we know our grandmother's favorite color is blue, perhaps we crochet her a blue scarf. And if we are the State of Maryland deploying an ANPR system, we know that most images we will capture will be of Maryland license plates — so we better be damn good at recognizing those Maryland plates.

So now that we know what ANPR actually *is*, let's briefly review the **4 steps required to build any ANPR system.** We'll obviously be reviewing each of these steps in more detail in the rest of this module (along with code for each of the steps), but I think it's best that we introduce them now.

# Step 1: Photo acquisition

The first step in any ANPR system is to actually *acquire* a photo of a car that (presumably) has a license plate that we want to detect and recognize.

Most production-level ANPR systems deployed in the real-world utilize infrared cameras so photos can be captured of vehicles regardless of time of day.

Furthermore, these cameras could be part of a closed-circuit system, such as security feeds for a large corporation. These cameras could be connected to a much larger network, such as a law enforcement agency. Or perhaps you have just mounted a Raspberry Pi to a street lamp in your neighborhood.

There are a variety of endless ways to capture our original image of a vehicle. But the point here is that we must (1) consider our surroundings, (2) determine which camera/setup will work best, and (3) deploy our camera in the wild.

An intermediary step between *photo acquisition* and *localization* is how to actually **trigger** our camera to capture a photo of the passing vehicle. There are many methods to accomplish this, the main ones being radar and motion detection. I tend to include the triggering of the camera into the photo acquisition step, but others may break this into an entirely independent step.

# Step 2: Localization

So now that we have an actual image containing a vehicle, we need to *find* or *localize* the region(s) of the image that contain the license plate(s).

Below is an example image of vehicle, followed by finding the license plate and drawing a bounding box surrounding it:


(https://gurus.pyimagesearch.com/wp-content/uploads/2015/03/license_plate_bounding_box.jpg)

**FIGURE 2:** FINDING AND LOCALIZING A LICENSE PLATE IN AN IMAGE, FOLLOWED BY DRAWING A BOUNDING BOX AROUND IT.

Again, there are a variety of methods to accomplish the localization tasks. Perhaps surprisingly, we do not need any fancy machine learning algorithms to detect license plates in images (although machine learning can help in certain situations).

In general, we can leverage clever combinations of basic image processing techniques to reveal regions of an image that *could* contain a license plate. We call these regions *license plate candidates*, and use further image processing to filter out the false-positives.

We then take the license plate regions and pass them on to the *segmentation* step.

# Step 3: Segmentation

So we have found the region of our image that contains the license plate. What now?

Well, in order to identify each of the characters on the license plate, we first need to *segment* them from the license plate background.

In general, you'll find that license plate images are too noisy to apply Optical Character Recognition (http://en.wikipedia.org/wiki/Optical_character_recognition) (OCR) algorithms on directly, so we'll further need leverage our image processing skills.

Most techniques perform some sort of *adaptive thresholding* or *scissoring* to "cut" the characters from the license plate, almost as if we were cutting a piece of paper into strips:


(https://gurus.pyimagesearch.com/wp-content/uploads/2015/03/license_plate_segmentation.jpg)

**FIGURE 3:** PERFORMING SEGMENTATION ON THE LICENSE PLATE CHARACTERS AND EXTRACTING EACH CHARACTER ONE-BY-ONE. THESE CHARACTERS ARE THEN FED INTO THE CHARACTER CLASSIFIER FOR RECOGNITION.

# Step 4: Recognition

Finally, given each (cleanly segmented) individual character on the license plate, we apply a bit of machine learning to recognize it.

Each character is *quantified* via *feature extraction* (see the Image Descriptors Module for more information on feature extraction). There are many procedures you could use to quantify each character, including using only the raw, thresholded pixel intensities, quantifying the contour/outlined region of the character, or extracting more abstract representations.

As you'll see later in this module, we'll be using the *Block Binary Pixel Sum* descriptor, which is a simple yet effective method to characterize and quantify license plate characters in images:


(https://gurus.pyimagesearch.com/wp-content/uploads/2015/03/license_plate_identified.jpg)

**FIGURE 4:** THE OUTPUT OF OUR ANPR SYSTEM — THE LICENSE PLATE IS NOW FULLY IDENTIFIED AND ACCESSIBLE VIA A PLAIN-TEXT VARIABLE.

Once we have the feature vector associated with each character, we then pass it to our machine learning model to classify and recognize the character. After recognizing each character, we will have obtained our final license plate recognition stored as a plain-text variable.

# Summary

Throughout the rest of this module we will be building a computer vision system to *localize*, *segment*, and *identify* the characters in a license plate. We'll accomplish this goal using clever (basic) image processing techniques, followed by a bit of machine learning. By the end of this module we'll have a **fully working ANPR system** that works on a dataset of license plate images.

However, keep in mind what I mentioned earlier in this article — ANPR systems are developed on a state-by-state basis. Don't be surprised if, when you go outside and snap a photo of your own car's license plate, the license plate identification fails. While this *will* be a working ANPR system, *you'll also have to tweak it a bit to get it working for your own license plate images*. Again, this lends back to what I was saying regarding *a priori* knowledge of license plates.

Finally, remember that ANPR is a challenging task. Even if we focus on identifying a single type of license plate, we'll still have a lot of work to do and many challenges to overcome. But don't worry, I'll be walking you through these challenges one by one. By the time you're done this module, you'll be able to build your own ANPR system for your home state.

| | Quizzes | Status |
|---|---|---|
| 1 | What is ANPR Quiz (https://gurus.pyimagesearch.com/quizzes/what-is-anpr-quiz/) | |

← Previous Lesson (https://gurus.pyimagesearch.com/lessons/the-complete-face-recognition-pipeline/) Next Lesson → (https://gurus.pyimagesearch.com/lessons/the-problem-with-anpr-datasets/)

# Course Progress

## Ready to continue the course?

Click the button below to **continue your journey to computer vision guru**.

I'm ready, let's go! (/pyimagesearch-gurus-course/)

## Resources & Links

- PyImageSearch Gurus Community (https://community.pyimagesearch.com/)
- PyImageSearch Virtual Machine (https://gurus.pyimagesearch.com/pyimagesearch-virtual-machine/)
- Setting up your own Python + OpenCV environment (https://gurus.pyimagesearch.com/setting-up-your-python-opencv-development-environment/)
- Course Syllabus & Content Release Schedule (https://gurus.pyimagesearch.com/course-syllabus-content-release-schedule/)
- Member Perks & Discounts (https://gurus.pyimagesearch.com/pyimagesearch-gurus-discounts-perks/)
- Your Achievements (https://gurus.pyimagesearch.com/achievements/)
- Official OpenCV documentation (http://docs.opencv.org/index.html)

## Your Account

- Account Info (https://gurus.pyimagesearch.com/account/)
- Support (https://gurus.pyimagesearch.com/contact/)
- Logout (https://gurus.pyimagesearch.com/wp-login.php?action=logout&redirect_to=https%3A%2F%2Fgurus.pyimagesearch.com%2F&_wpnonce=5736b21cae)

Search

Feedback