**Introduction to Computer Vision**

Using software to parse the world’s visual content is as big of a revolution in computing as mobile was 10 years ago, and will provide a major edge for developers and businesses to build amazing products.

Computer Vision is the process of using machines to understand and analyze imagery (both photos and videos). While these types of algorithms have been around in various forms since the 1960’s, recent advances in [Machine Learning](https://blog.algorithmia.com/introduction-to-machine-learning/), as well as leaps forward in data storage, computing capabilities, and cheap high-quality input devices, have driven major improvements in how well our software can explore this kind of content.

**What is Computer Vision?**

Computer Vision is the broad parent name for any computations involving visual content – that means images, videos, icons, and anything else with pixels involved. But within this parent idea, there are a few specific tasks that are core building blocks:

* In **object classification**, you train a model on a dataset of specific objects, and the model classifies new objects as belonging to one or more of your training categories.
* For **object identification**, your model will recognize a specific instance of an object – for example, parsing two faces in an image and tagging one as Tom Cruise and one as Katie Holmes.

A classical application of computer vision is handwriting recognition for digitizing handwritten content (we’ll explore more use cases below). Outside of just recognition, other methods of analysis include:

* Video **motion analysis** uses computer vision to estimate the velocity of objects in a video, or the camera itself.
* In **image segmentation**, algorithms partition images into multiple sets of views.
* **Scene reconstruction** creates a 3D model of a scene inputted through images or video (check out [Selva](https://www.selva3d.com/)).
* In **image restoration**, noise such as blurring is removed from photos using Machine Learning based filters.

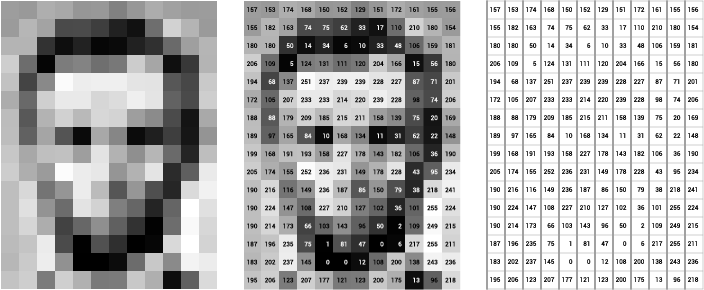
Any other application that involves understanding pixels through software can safely be labeled as computer vision.

**How Computer Vision Works**

One of the major open questions in both Neuroscience and Machine Learning is: how exactly do our brains work, and how can we approximate that with our own algorithms? The reality is that there are very few working and comprehensive theories of brain computation; so despite the fact that Neural Nets are supposed to “mimic the way the brain works,” nobody is quite sure if that’s actually true. Jeff Hawkins has an [entire book on this topic called On Intelligence](https://www.amazon.com/Intelligence-Understanding-Creation-Intelligent-Machines/dp/0805078533).

The same paradox holds true for computer vision – since we’re not decided on how the brain and eyes process images, it’s difficult to say how well the algorithms used in production approximate our own internal mental processes. For example, [studies have shown](https://www.technologyreview.com/s/508376/in-a-frogs-eye/) that some functions that we thought happen in the brain of *frogs* actually take place in the eyes. We’re a far cry from amphibians, but similar uncertainty exists in human cognition.

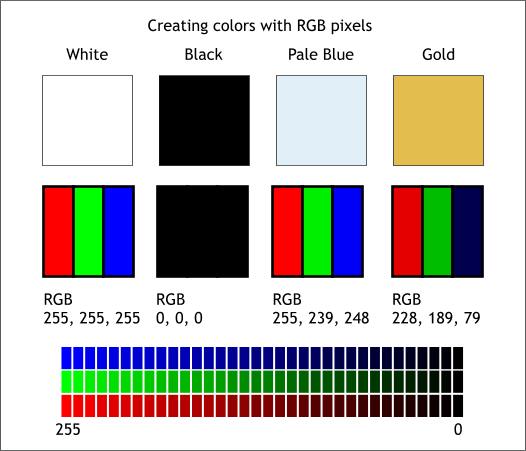
Machines interpret images very simply: as a series of pixels, each with their own set of color values. Consider the simplified image below, and how grayscale values are converted into a simple array of numbers:



Source: [Openframeworks](http://openframeworks.cc/ofBook/chapters/image_processing_computer_vision.html)

Think of an image as a giant grid of different squares, or pixels (this image is a very simplified version of what looks like either Abraham Lincoln or a Dementor). Each pixel in an image can be represented by a number, usually from 0 – 255. The series of numbers on the right is what software sees when you input an image. For our image, there are 12 columns and 16 rows, which means there are 192 input values for this image.

When we start to add in color, things get more complicated. Computers usually read color as a series of 3 values – red, green, and blue (RGB) – on that same 0 – 255 scale. Now, each pixel actually has 3 values for the computer to store in addition to its position. If we were to colorize President Lincoln (or Harry Potter’s worst fear), that would lead to 12 x 16 x 3 values, or 576 numbers.



Source: [Xaraxone](http://archive.xaraxone.com/webxealot/workbook35/page_5.htm)

For some perspective on how computationally expensive this is, consider this tree:

* Each color value is stored in 8 bits.
* 8 bits x 3 colors per pixel = 24 bits per pixel.
* A normal sized 1024 x 768 image x 24 bits per pixel = almost 19M bits, or about 2.36 megabytes.

That’s a lot of memory to require for one image, and a lot of pixels for an algorithm to iterate over. But to train a model with meaningful accuracy – especially when you’re talking about [Deep Learning](https://blog.algorithmia.com/introduction-to-deep-learning/) – you’d usually need tens of thousands of images, and the more the merrier. Even if you were to use [Transfer Learning](https://en.wikipedia.org/wiki/Transfer_learning) to use the insights of an already trained model, you’d still need a few thousand images to train yours on.

With the sheer amount of computing power and storage required just to train deep learning models for computer vision, it’s not hard to understand why advances in those two fields have driven Machine Learning forward to such a degree.

**Business Use Cases for Computer Vision**

Computer vision is one of the areas in Machine Learning where core concepts are already being integrated into major products that we use every day. [Google is using maps](https://research.googleblog.com/2017/05/updating-google-maps-with-deep-learning.html) to leverage their image data and identify street names, businesses, and office buildings. Facebook is using computer vision to identify people in photos, and do a number of things with that information.

But it’s not just tech companies that are leverage Machine Learning for image applications. Ford, the American car manufacturer that has been around [literally since the early 1900’s](https://en.wikipedia.org/wiki/Ford_Motor_Company), is [investing heavily in autonomous vehicles (AVs)](https://media.ford.com/content/fordmedia/fna/us/en/news/2016/08/16/ford-targets-fully-autonomous-vehicle-for-ride-sharing-in-2021.html). Much of the underlying technology in AVs relies on analyzing the multiple video feeds coming into the car and using computer vision to analyze and pick a path of action.

Another major area where computer vision can help is in the medical field. Much of diagnosis is image processing, like reading x-rays, MRI scans, and other types of diagnostics. [Google has been working with medical research teams](https://research.google.com/teams/brain/healthcare/) to explore how deep learning can help medical workflows, and have made significant progress in terms of accuracy. To paraphrase from their research page:

“*Collaborating closely with doctors and international healthcare systems, we developed a state-of-the-art computer vision system for reading retinal fundus images for diabetic retinopathy and determined our algorithm’s performance is on par with U.S. board-certified ophthalmologists. We’ve recently published some of our research in the* [*Journal of the American Medical Association*](https://research.google.com/pubs/archive/45732.pdf) *and summarized the highlights in a* [*blog post*](https://research.googleblog.com/2016/11/deep-learning-for-detection-of-diabetic.html)*.*”

But aside from the groundbreaking stuff, it’s getting much easier to integrate computer vision into your own applications. A number of high-quality third party providers like Clarifai offer [a simple API for tagging and understanding images](https://www.clarifai.com/), while Kairos [provides functionality around facial recognition](https://www.kairos.com/). We’ll dive into the open-source packages available for use below.

**Computer Vision on Algorithmia**

Algorithmia makes it easy to deploy computer vision applications as scalable microservices. Our marketplace has a few algorithms to help get the job done:

* [SalNet](https://algorithmia.com/algorithms/deeplearning/SalNet) automatically identifies the most important parts of an image
* [Nudity Detection](https://algorithmia.com/algorithms/sfw/NudityDetectioni2v) detects nudity in pictures
* [Emotion Recognition](https://algorithmia.com/algorithms/deeplearning/EmotionRecognitionCNNMBP) parses emotions exhibited in images
* [DeepStyle](https://demos.algorithmia.com/deep-style/) transfers next-level filters onto your image
* [Face Recognition](https://algorithmia.com/algorithms/cv/FaceRecognition)…recognizes faces.
* [Image Memorability](https://algorithmia.com/algorithms/deeplearning/LargescaleImageMemorability) judges how memorable an image is.

A typical workflow for your product might involve passing images from a security camera into Emotion Recognition and raising a flag if any aggressive emotions are exhibited, or using Nudity Detection to block inappropriate profile pictures on your web application.

For a more detailed exploration of how you can use the Algorithmia platform to implement complex and useful computer vision tasks,

### Computer Vision Resources

##### **Packages and Frameworks**

[OpenCV](https://opencv.org/) – “OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.”

[SimpleCV](http://simplecv.org/) – “SimpleCV is an open source framework for building computer vision applications. With it, you get access to several high-powered computer vision libraries such as OpenCV – without having to first learn about bit depths, file formats, color spaces, buffer management, eigenvalues, or matrix versus bitmap storage.”

[Mahotas](http://mahotas.readthedocs.io/en/latest/) – “Mahotas is a computer vision and image processing library for Python. It includes many algorithms implemented in C++ for speed while operating in numpy arrays and with a very clean Python interface. Mahotas currently has over 100 functions for image processing and computer vision and it keeps growing.”