EW309 Computer Vision

Estimating Scale

Goal

The units of a digital image are natively in pixels. To use information from a camera in the real world, we need to estimate a relationship between pixels and some linear unit (e.g. centimeters or millimeters).

Observation

What happens in your video when you move the target closer to the camera? Similarly, what happens when you move the target farther from the camera? Even though the target is the same size, it appears larger when closer; and smaller when farther away. This means the relationship between pixels and linear units is dependent on the distance between the target and camera!

For an "ideal" (pinhole) camera, this relationship can be represented mathematically given a distance between the camera and the point it is looking at, defined as s:

$$x_{pixels} = a\frac{x_{linearUnits}}{s} + b \quad (1)$$

$$y_{pixels} = c\frac{y_{linearUnits}}{s} + d \quad (2)$$

In these equations, x_{pixels} and y_{pixels} represent the pixel coordinate relative to the upper left of the picture (Fig. 1) and $x_{linearUnits}$ and $y_{linearUnits}$ represents the x and y coordinates in the real world. Note that using a common unit like centimeters for $x_{linearUnits}$, $y_{linearUnits}$, and s is advisable. In these equations, the a and c terms scale units to pixels, and the b and d terms represent the pixel offset between the image origin (in the upper left) and the center of the image. The position described by b and d is typically referred to as the principal point.

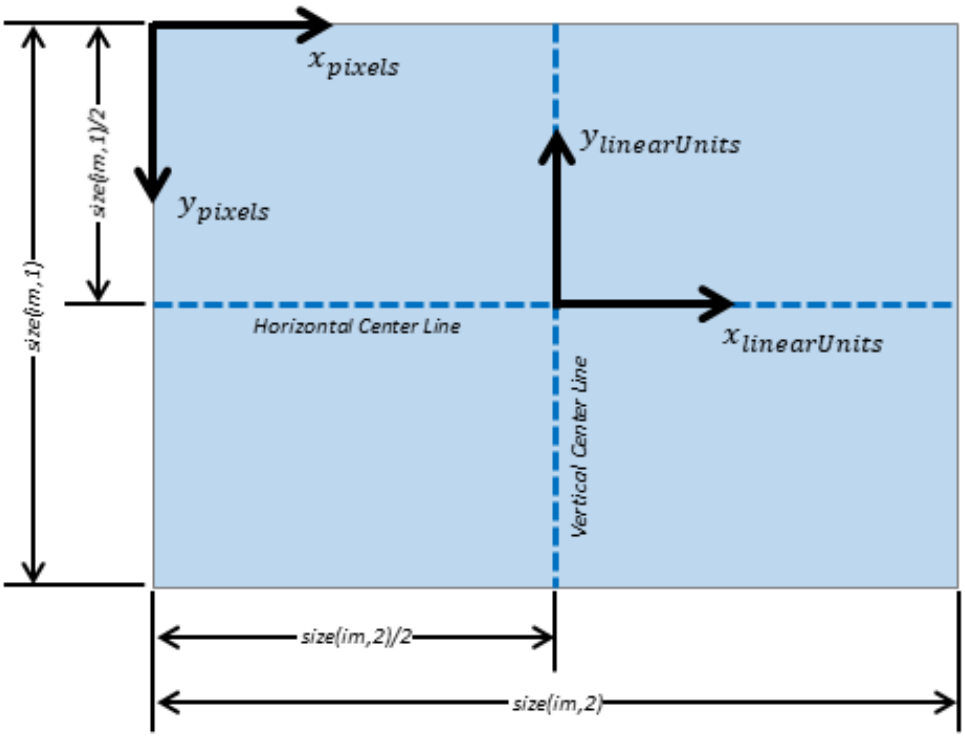

Figure 1: Recommended coordinate frame assignments highlighting the native image frame and image dimensions in MATLAB syntax.

Referencing Data

For our application, we will be using camera mounted to the rail of our weapon as a targeting mechanism. This means we expect our targets to be near the center of the image prior to firing. With this in mind, using the center of the image as a reference for measuring linear units seems to make sense (Fig 1).

Collecting Data

Create a uniform grid on the board (a crosshair with large tick marks labelled with a ruler or tape measure will work), center your camera on the crosshair, and take images at a series of known distances. Be sure to fixture your camera (or hold it very still) when taking the images; and write down the distance associated with each image.

When collecting data to solve for the constants a, b, c, and d as given in (1) and (2), you will need the following:

1. The distance to the grid (s), measured in linear units (e.g. centimeters).
2. The x and y tick mark locations referenced to the center of the crosshair in linear units (e.g. centimeters). Note that the crosshair must be aligned with the center of the image.
3. The x and y tick mark locations referenced to the image coordinate frame in the upper left of the image in pixels.

Data for Item 3 can be collected using the function "`ginput.m`" in MATLAB. Given an image plotted in a figure window whose handle is saved in "`fig`", "`ginput.m`" can be used to collect a single x and y location in pixels using the following:

```
figure(fig);
[x,y] = ginput(1);
```

Here, "figure(fig)" brings the figure containing your image to the foreground, and "`[x,y] = ginput(1)`" allows you to select the location of one pixel saving the x information in the variable `x`, and the y information in the variable `y`.

Processing Data

Using the tools from PART 1 – PART 3, create a test script that will display an image and allow the user to manually align the crosshair with the center of the image. Once aligned, the script should capture a static image and allow the user to measure the x and y pixel location of each tick-mark relative to the image coordinate frame in the upper left of the image.

Using this data, and the known location of each tick mark in linear units (which you will need to manually record), create a table consisting of the x and y values of each point on the crosshair in both linear units (referenced to the center of the crosshair) and in pixels (referenced to the upper left of the image). For each point, you should also record a distance value s in linear units.

Using this table with data collected from at least three distances, use MATLAB's fitting tools to solve for the constants a, b, c, and d given in (1) and (2).

Packaging Your Results

Once you have defined the relationship between pixels and linear units, package your equations in a MATLAB function for later use. The inputs to this function should be s, x_{pixels}, and y_{pixels}; and the outputs to this function should be $x_{linearUnits}$ and $y_{linearUnits}$. Be sure to use a descriptive function name, and document your function! As an example, *if all of your linear units are in centimeters*, you may want to define your function as follows:

```
function [x_cm,y_cm] = pixelsToCentimeters(s_centimeters,x_pixels,y_pixels)
```

Documenting Your Function

Take the time to write help documentation for your function. This will be a long semester, and it is easy to forget how to use even code that you have written. The commonly used format for MATLAB help documentation is as follows:

```
function [out1, out2, …] = funcName(in1, in2, …)
%FUNCNAME A brief description of what the function does
%    [out1, …] = FUNCNAME(in1, …) an in depth description of how the
%    function works given the specified input/output combination.
%    Any additional notes/comments/details about the aforementioned function
%    call.
%
%    [out1, …] = FUNCNAME(in1, in2, …) an in depth description of how the
%    function works given the specified input/output combination.
%    …
%
%    See also RELATEDFUNC1, FELATEDFUNC2, …
%
%    Contributor(s) names & date initially created
```