Where Is Waldo?

Ben Douek
#0684070

CIS 4720

In this folder you will find two programs that use similar but slightly different approaches to finding Waldo -or Wally for the Brits.

The first program you will find enclosed in the folder is named the “waldoKeyFinder.py” the second would be the “waldoAvgKey.py”; Along with these programs are 8 sample images stored in .tif format; Lastly, there are two “Waldo keys”. Using the keys enables each program to identify Waldo.

The **“waldoKeyFinder.py”** uses a key image that has **four** small 3*2 snippets of the unique orange sampled from a book in Waldo’s book stack. It also has another collection of **four** 3*2 snippets sampled Waldo’s shirt(s).

This program loops through these snippets comparing each of the keys with the whole image 6 pixels at a time. This program checks only for exact matches of these 3*2 combinations meaning its as secure as a combination lock with 255^3 different combos. So since each key was taken from a different image it is guaranteed to find Waldo in all **eight** files usually in less then 5 minutes.

Below you can see the very small .tif file with the first key (fig1). This program was a good **starting point** but unfortunately if you throw it a curve ball using an image it has not learned from it is highly unlikely to find Waldo.

Fig 1 Fig 2

The **“waldoAvgKey.py”** uses a similar key matching technique but this time with only one snippet from Waldo’s shirt (fig2). This time the program takes in a **tolerance** as an argument as well. It will then proceed to check for matches, but this time it will not only accept matches that are exact but those that fall within the tolerance too. The tolerance is a number that will judge how similar the combinations must be to be accepted as a match. 0 – is an exact match; 255 – would match absolutely anything.

This key was sampled from the “weresWaldo3.tif” file but it will also match Waldo in “weresWaldo4.tif” utilizing a tolerance of 30, and it may well find Waldo in others with the settings adjusted. This program could easily be setup to use a bigger key library like the first program, in which case if a match was not found it could move on to the next key. I did not implement this larger key library to demonstrate that the program could interpret Waldo using only the tolerance feature and a key that was not an exact match. This program will take significantly longer to run than the first, but runtime should not exceed 5 minutes.

Both programs show that Waldo can be identified using a simple 3*2 key, and that 6 pixels is enough to identify Waldo, and using only 6 and can greatly reduce comparison processing time.

To Run:

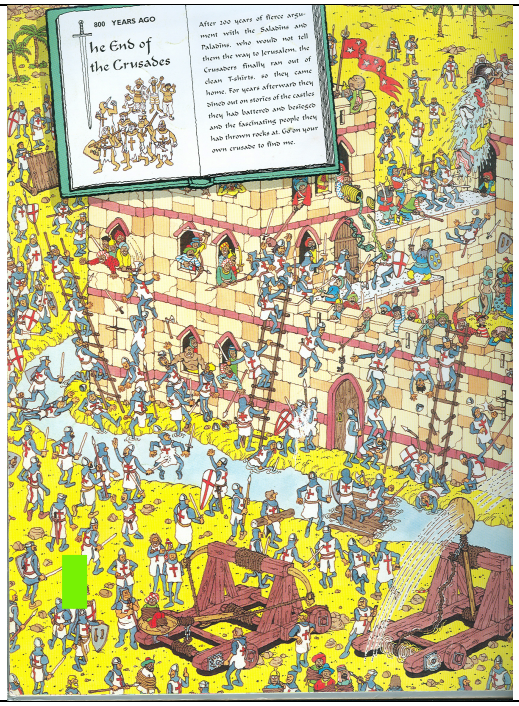
Assuming the numpy, PIL, sys, and math libraries are installed open the terminal in the folder and simply type:

```
python ./waldoKeyFinder.py wheresWaldoX.tif
OR
python ./waldoAvgKey.py wheresWaldoX.tif 30
```

X would indicate the number or letter to use, 30 is usually I good number I have found but as mentioned above 0-255 could be used.

Output of program 1

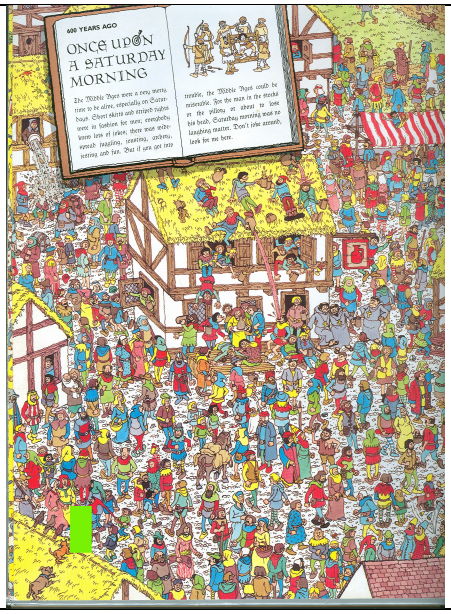
Tested using wheresWaldo3.tif with a non-variable tolerance of 0:



As you can see Waldo's location has turned green and he has been cropped out of the image.

Output of program 2

Tested using wheresWaldo4.tif with a tolerance of 30:



This is not the same file used to make the key but using the tolerance it can still locate Waldo.