There's Waldo!
Using Computer Vision and a Neural Net to Find Waldo

Diego Gonzalez, John Freeman, Jose Rivas-Garcia
Williams College

May 18, 2017

1 Introduction

In the last decade, there has been little research done in the AI community towards solving the age old question, Wheres Waldo? Most research that revolves around the infamous Waldo has come from the realm of psychology or human computer interaction, which focuses on the strategies used by people towards finding the hidden, white-and-red striped character [2]. Thus, the goal of our project was to finally expose Waldo to the world through Artificial Intelligence and Computer Vision. Though we did not fully achieve our goal, our program is able to find Waldo on most Wheres Waldo images by narrowing down the search to 50 ranked potential Waldo locations.

2 Original Plan

Our original plan was to narrow down the search space through two approaches. First we would break down the image into some number of subimages. We would then compare the color distribution in that image to what we would expect from an image of Waldo. More specifically, we would look for appropriate levels of adjacent red and white pixels in the image. We would throw away subimages that strayed too far from what we expected an image of Waldo to contain and feed the remaining subimages to a neural net. This neural net would be trained on images of Waldo in order to determine whether or not a given sub image contained Waldo. Ideally, we would be left with a single image, but otherwise we would rank our images on their Waldinity, how much they resembled Waldo according to our neural net. This original plan was inspired by the method by which humans find

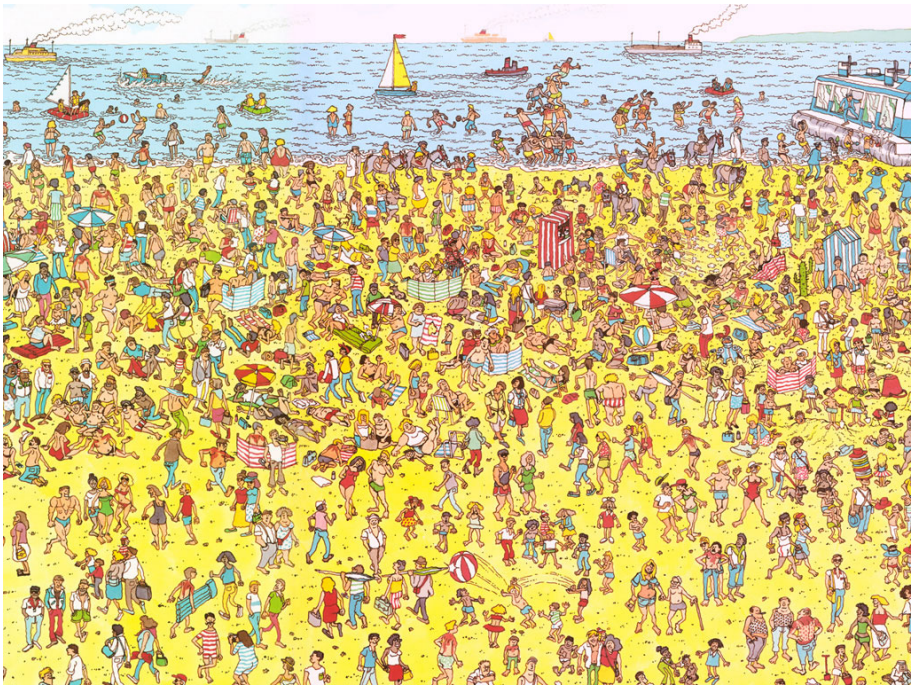

Figure 1: The Waldo Beach Image. Can You Find Waldo?

Waldo; first search for places with red and white and then look for Waldos iconic face.

2.1 Step 1: Using Local Color Histograms

Francois Ennesser and Gerard Medioni developed an algorithm for determining Waldos most probable locations [1]. The algorithm they implemented used local color information to determine if any particular area contained the red-and-white stripes that so characterize Waldo. They themselves adopted and changed a different algorithm, the Backprojection algorithm of Swain and Ballard, which did a point by point (pixel by pixel) statistical analysis of an image to determine if a pixel belonged to the object they wanted. This statistical analysis relied on the full color histogram of the image and the object in question. Ennesser and Medioni exploited the information from this algorithm, but instead of analyzing the image as a whole, they analyzed the image using local histograms that are the size of the expected aspect ratio of the object. This helped them narrow down the search space and become much more accurate. By using an algorithm inspired by this, we were able to narrow down our search space considerably.

2.2 Step 2: Using a Neural Net for Cartoon Facial Recognition

In a paper published in 1996, researchers Paul Juell and Ron Marsh outlined a process by which they trained a neural net to recognize faces by performing edge enhancement and processing by a hierarchical neural network [3]. The promising aspect of their work is that their neural net was able to both detect faces with glasses and cartoon faces. This is relevant to us since we want to train a neural net to find a cartoon face with glasses. Unfortunately, they mention that the net has a tendency to have false positive, especially with noisy images. We hoped to overcome this by supporting the neural net with our color analysis.

We largely followed our original plan with some modifications. We will discuss these in the following sections.

3 Implementation

In this section, we will describe how we implemented our *"Where's Waldo"* finder.

3.1 Color Histograms

The algorithm we developed to find Waldo is similar to Ennesser and Medionis algorithm, discussed in the previous section, in that we also search for locations with a high percentage of red-and-white pixels. However, while Ennesser and Medioni use their Waldo model(s) to determine the probability that a certain pixel belongs to Waldo, we decided to generalize the search a bit further. Instead, we make five distinctive passes through our Waldo image, in order to narrow down the locations that Waldo could appear. Only once we have found these locations and taken snapshots of them, do we use our Waldo models to rank them. We explain these passes below, but first we must define a larger issue, which plagued us at the beginning. What does red and white look like?

3.1.1 The Trouble With Colors

Color, by definition, is a subjective medium through which we view the world. Color may not only be perceived differently by different people, but the names we associate with certain shades of colors vary, as well. Certain shades of violet, for example, may appear more purple than blue. It is no surprise, then, that defining color to a computer, which views them solely through 8 bit red, blue, green, and alpha channels, is especially challenging, and the definitions we provided made a huge difference between spotting Waldo or not. Thus, for the purposes of this lab, we followed the recommendations of Michael Mason and Zoran Duric [4] and

converted all colors in our Waldo image to 12 bit color, ignoring the alpha channel, and providing the other three channels with 4 bits each, allowing for a crisper distinction between colors, because the maximum value for those channels drops from 255 to 15.

Furthermore, to find Waldo, we also had to define red and white for our program. Given that Waldo may vary in size, and pixel colors tend to bleed through wherever two colors meet, we decided to take a more liberal approach to defining these colors. For our program, red is defined as any color whose red channel is at least 4 units higher than the blue and green channels, and whose blue and green channels differ by at most 2 units, or any color that appears to be close to the predefined Pink, Red, or Magenta colors in Java. Likewise, white is defined as any color whose blue and green channels differ by at most 2 units, and whose red channel is at least 3 units to avoid black or dark grey. Furthermore, to catch pinker shades, we allow the red channel to be slightly higher than the blue and green channels, as well.

3.1.2 Parsing the Image

As mentioned previously, in order to find Waldo we perform five distinct passes through a given *Wheres Waldo?* image.

The first two passes through the *Wheres Waldo?* image are a simple edge detector to mark where red and white meet. This marking is done by first converting each pixel to one of three colors: red, blue or black, wherein red represents a red pixels, blue represents a white, and black represents any other color. The image is then looped through again, and at each pixel a 3 * 3 grid is created around the pixel to search for the locations where red and white pixels have gathered together. If both red and white pixels appear in the grid, the proportion of red and white pixels found are then stored in another image at that pixels location for future use.

Following these initial passes, the next two passes create larger grids to search for local areas where red and white meet. These grids are 7*7 and 11*11 pixels in size respectively. Each pass uses information from the previous one to determine if that location could potentially contain Waldo. For example, if at a particular location there appears to be over 90% red pixels with few white pixels, it is safe to assume that Waldo may not appear at that location. Likewise, if there appears to be over 90% white pixels with few red pixels, then it is safe to assume that Waldo will not be there. At each stage, the proportions found for these larger grids are also stored in the secondary image for the next passes to use.

The fifth and final pass takes the final passs information and reports a probability at each pixel location that that pixel belongs to Waldo. This calculation is similar to the calculation done in the previous passes, but its grid is smaller at 5*5

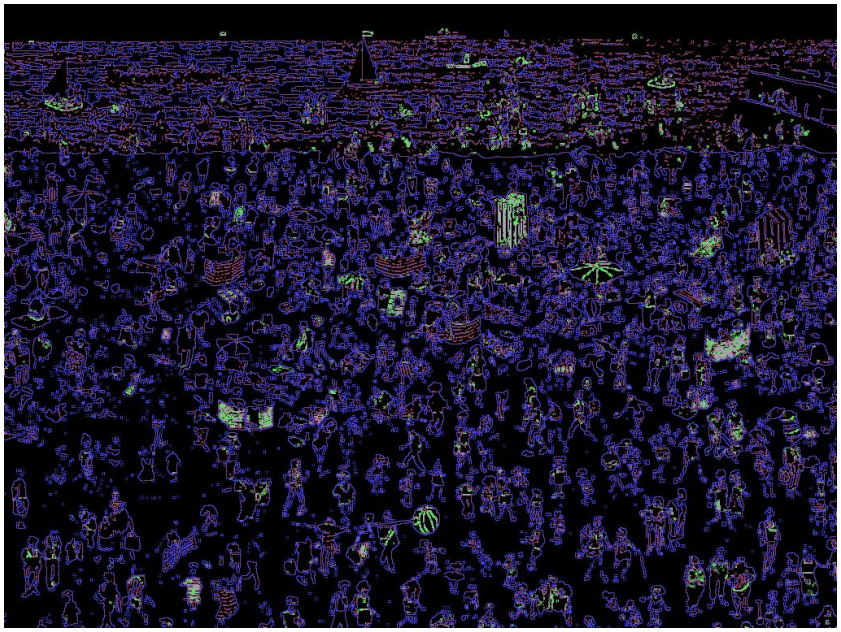

Figure 2: Output from Initial Passes on the Waldo Beach Image.

pixels, and it uses an average of the red and white pixel proportions found around each pixel to determine the probability that Waldo is at a particular location.

3.1.3 Taking Snapshots

Once the image has been completely parsed, and all the possible locations Waldo can appear have been determined, it is still necessary to grab snapshots of those locations to provide to our neural net. This subimage production is done through a final parser that takes the previous histogram as input, and parses it, looking for locations with a high enough probability to contain Waldo. This parser, much like all of the passes used to create the histogram, averages the probabilities around each pixel to determine whether that pixel could be a pixel of Waldo. Once such a pixel is found, snapshots of that location are taken, the histogram is blacked out at that grid, and the parsing continues.

Due to the way our neural net is formed, the snapshots our subimage creator produces have to be of the same size, and they must provide an accurate presentation of Waldo's face, meaning we must grab more than just his hat, hair or chin. Since our histogram is based off of red and white pixel locations, it is not necessarily true that we will be able to grab Waldo's face by just looking around a marked

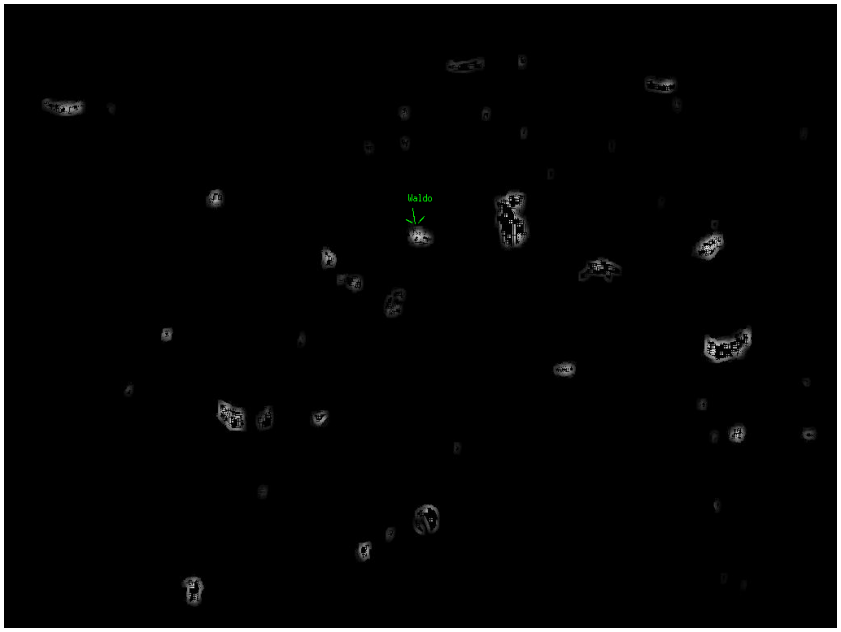

Figure 3: Output from Final Pass on the Waldo Beach Image.

pixel's location. In fact, because our histogram is so general, it may well be that we marked only the center of Waldo's shirt, and his face is much higher up. To adjust for this, our subimage producer takes four snapshots containing that pixel. These snapshots are: 1) a snapshot with the current pixel at the bottom right corner; 2) a snapshot with the current pixel at the bottom left corner; 3) a snapshot with the current pixel at the top left corner; and 4) a snapshot with the current pixel at the center. With these four snapshots, we are guaranteed to grab Waldo's face, if we have marked his shirt or hat.

3.1.4 Ranking Images Based on Waldo Models

The subimages we created from our general probabilistic search are not always images of Waldo or Waldo look-a-likes. In fact, many of them are images of cars, red/white piping, and other inanimate objects. Thus, in order to determine how likely it is an image contains Waldo, we developed a ranking system based off of Ennesser and Medionis idea of using a Waldo model to determine a subimage's waldinity.

To perform this ranking, we saved a certain number of Waldo subimages in a separate folder as models for our color histogram classifier to use. These mod-

els are then parsed through in a similar fashion to the larger *"Where's Waldo?"* image by searching for locations where red-and-white meet. Once these hotspots are found, the proportion of pixels left are calculated. Once each model has its proportion found, the mean and standard deviation are calculated.

Finally, for each snapshot taken from our original *"Where's Waldo?"* image, the same histogram function used on the models is performed on them. Each snapshot is then ranked according to how many standard deviations from the mean their red and white proportion falls.

3.2 Neural Net

We created and used a neural network which takes an image, converts each pixel to a greyscale-like value, and returns a confidence of that subimage containing Waldo. We used the MultiLayerPerceptron class from the Weka open source library to create it. We use it by running all of the images we have flaqged as possible Waldo images based on histogram through the net and use the nets confidence value to help rank which image we believe is the most likely to be the actual Waldo image.

3.2.1 Creating the Neural Net

We created a NeuralNetCreator class that is responsible for creating, training, testing, and saving the neural net to be read and used in Main. The net is saved as a .model file. Currently, the net is a two hidden layer neural net with 160 node in the first layer and 8 nodes in the second. It takes in 200 input values and was trained with a 0.15 learning rate that decays over time and 0.2 momentum. The training set we used on for the neural net is a set of 1300 total image where 700 contain Waldo and 600 are random images from Waldo puzzles but dont contain Waldo. These images were created from known Waldo images which we added to photoshop to move Waldos face around and filled the background with a single color we could add noise to. We did this for the hope that the neural net would recognize Waldos face, glasses and hat from the images rather than some detail in the background of the image. On the testing set we are using to rate the neural net comprising 100 Waldo images and 300 non-Waldo image, it is testing at 85% accuracy overall. With more training images and more tuning of the parameters, we believe this neural net could be achieve even better results than this.

3.2.2 Using the Neural Net

We also have a BWArffGenerator class that is responsible for translting image data to Instances that the neural net can train and test on. For the training and testing

set, we generate .arff files from folders of images that the neural net can train and test on using our iterateFolders method that writes the file to a filename given on the command line. We also have a createInstance method that will take an image, generate the attributes corresponsponding to the pixels of that image and return a weka Instance. That instance can then be applied to the neural net using the distributionForInstance method which returns a Waldo confidence value for the image based on the neural net. The values we generate from an image for use in the neural net is a modified graycode from the pixels where white pixels should remain white with high values, red pixels are of their greycode value, and non red or white pixels are their graycode value. This allows our neural net to distinguish between red, white, and other pixels as red and white pixels are very present with Waldo.

3.3 Bringing The Two Together: Determining Waldinity

By ranking the subimages left by the histogram analysis using the neural net alone, we were not able to get the results that we had originally hoped for. To adjust for this, we performed two operations, adjacent potential Waldo consolidation and a combined waldinity ranking.

3.3.1 A Combined Ranking

Originally, we were ranking using only the neural net. However, this ranking was not achieving desirable results and we noticed that many highly ranked Waldo subimages did not have a significant amount of red and white pixels. Part of this may be due to our liberal definition of red and white, but another cause was that the neural net was not necessarily looking for red and white in the image; many of the training images used for the neural net were of Waldos face, and not of his iconic clothes. To adjust for this, we devised a system by which the potential waldo's are ranked by both the color histograms waldinity confidence and the neural nets waldinity confidence. We further weighted the color histograms confidence levels by their deviation from the mean measured in standard deviations. Using this combined ranking, we generated more favorable results.

3.3.2 Adjacent Consolidation

By nature of how we created our subimages, there were many overlapping subimages. One phenomena that we observed was that among the top 50 potential waldo's ranked by waldinity, there were many areas circled various times because they were being considered multiple times through adjacent subimages. To correct

for this, we combined subimages that overlapped so that each area was only considered once in the ranking. When combined, the resulting subimage would take the higher confidence rating, for both neural confidence and histogram confidence, of the two subimages combined. This ensured that combined subimages were not weighted more heavily than solitary subimages.

4 Results

Our program is able to consistently catch Waldo or a fake Waldo in the top 50 potential locations. This falls short of our original goal, but considering that we can start with as many as 12,000 subimages, it is able to cut down the search space significantly. In the beach image, we are even able to flag Waldo in the top ranked spot, meaning that most likely Waldo is Waldo himself. He may still be elusive in certain settings, but our work makes his discovery by a person much easier.

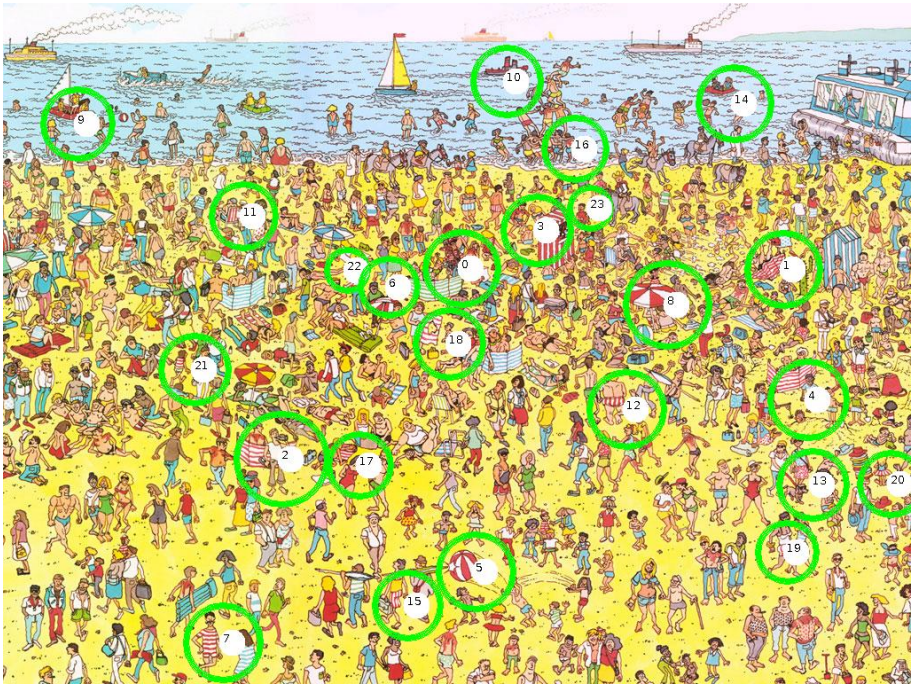

Figure 4: Potential Waldo locations ranked on their waldinity.

Figure 1 shows our program's output from the *"Where's Waldo?"* beach scene and Figure 2 shows a selection of Waldo's and potential Waldos found by our program. We were not able to find Waldo himself in all images, but we were able

to find Waldo or a fake Waldo in all our test images except for WW7, an image of Waldo at a film set for a concert. This is likely due to the fact that WW7 is a worst case for our program, with a large amount of red and white throughout.

(a) (b) (c) (d) (e) (f) (g)

Figure 5: Waldo and Waldo Look-A-Likes in Different "Where's Waldo?" Images

5 Conclusion

All in all, we are happy with the results that we have acquired through our project. We experimented with various approaches and constantly changed our implementation plan in order make our program as effective as possible. That being said, there are limitations to our implementation that could be rectified through other possible procedures. We outline these changes and limitations in the following sections.

5.1 Changes to our Implementation Plan

When writing our "Where's Waldo?" solver, we found ourselves constantly updating and revising our implementation. Our original plan to simply use the neural net to find Waldo in a subset of subimages created by the color histogram algorithm. However, we quickly realized that the neural net alone was not enough to find Waldo. Instead we transitioned to the combined ranking system that takes into account both our classifiers. This allowed us to rank the Waldo images and look through the top 50 images, instead of expecting Waldo's exact location to be found.

There were also modules that did not end up being used in the final version of our program. An early approach to divide the image into subimages first determined the average size of a person through edge detection and then used that size to break the larger image into some number of subimages. These images would later be scaled to the appropriate size needed for the other parts of our program. This approach allowed us to expect a certain size for Waldo proportional to the subimage, but was phased out by a more efficient approach that performed a color anaylsis first in order to determine the locations of subimages.

5.2 Limitations

While we are happy with the results of our program, there are still some limitations it has that can be improved. While we tend to find Waldo in most of the images we gave to our program, there were a small number of cases where we did not find him and the times we did, it was rare for Waldo himself to have the highest ranking by our program.

5.2.1 Limitations with Colors and Histograms

One of the most severe limitations comes from our definition of red and white pixels. Although we made our definitions as liberal as possible, it is entirely possible that a Waldo image uses different hues of red and white. This difference could cause our histogram to be unable to mark Waldo for our subimage creator to snapshot him. Also, this liberal definition causes many other colors to be flagged, such as brown, which is generally very red. One great way to get over this limitation is to find a way to define red and white more flexibly, allowing it to change according to the image.

Another severe limitation is the sensitivity of the histogram to red and white hotspots. From the way our histogram attempts to decrease the spaces that Waldo could appear, it could happen that we accidentally delete Waldo on further passes, if, for example, Waldo is too large to fit in the grid we create. One solution to this problem that we considered was determining the exact size of a human in the image, and using that to determine the color histograms.

Another solution to this problem could be more directly implementing Ennesser and Medioni's algorithm in our implementation. Nevertheless, we felt that while their solution worked when you knew models, it could have many potential flaws in images where the models were not readily available, or only certain areas of Waldo were viewable. While most "Where's Waldo?" images show his hat, they may not show any part of his shirt, or only show a sleeve. We wanted to ensure that we did not have to rely entirely on a model to find Waldo, and instead focus on spots where red and white do occur to determine if they are really the Waldo we are looking for.

5.2.2 Limitations with the Neural Net

There is also some limitations with the current neural net. The biggest limitation is the small number of training images as it was hard to create good training images of Waldo from the images we had. There is also the possibility that there is a better way to represent the images to the neural net than how it's currently being done like adding the ColorHistogram's confidence value as another parameter so the neural

net has some more knowledge about the program giving it the images to classify. There are also some more neural net tricks, like adding more than 2 categories for instances, that might have helped us increase our accuracy and make the neural net more robust to the slight changes in Waldo's appearance, but still recognize him.

References

[1] Ennesser, Franois, and Grard Medioni. "Finding waldo, or focus of attention using local color information". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17.8 (1995): 805-809.

[2] Chang, Hung-Cheng, Stephen Grossberg, and Yongqiang Cao. "Wheres Waldo? How Perceptual, Cognitive, and Emotional Brain Processes Cooperate during Learning to Categorize and Find Desired Objects in a Cluttered Scene". *Frontiers in Integrative Neuroscience* 8 (2014): 43. PMC. Web. 18 May 2017. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4060746/

[3] P. Juell and R. Marsh. "A hierarchical neural network for human face detection." *Pattern Recognition*, vol. 29, no. 5, pp. 781787, 1996. http://www.sciencedirect.com/science/article/pii/0031320395001298

[4] Mason, Michael, and Zoran Duric. "Using histograms to detect and track objects in color video." Applied Imagery Workshop, AIPR 2001 30th. IEEE, 2001.

[5] Raphael B, Smith IFC "A probabilistic search algorithm for finding optimally directed solutions". Icelandic Building Research Institute, Reykjavik, Iceland; 2003. Web. 18 May 2017. https://www.irbnet.de/daten/iconda/CIB20187.pdf