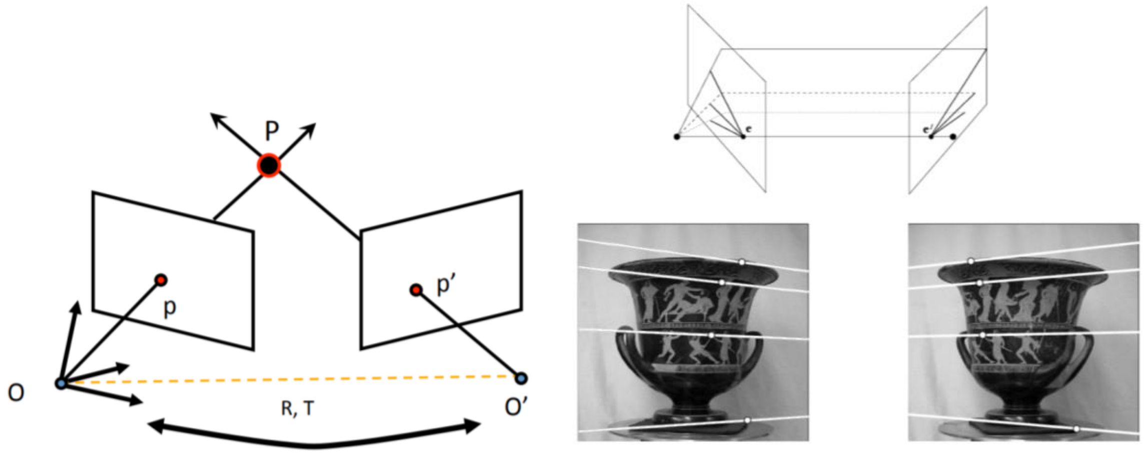
Determining the distance to the object

Second Interim Report

Yana Kryshchuk, Yaroslav Borys, Danylo Sahaidak

General topic

In general, the stereo vision problem is recreating the form and position of 3-dimensional objects by it's multiple (at least two) 2-dimensional images.

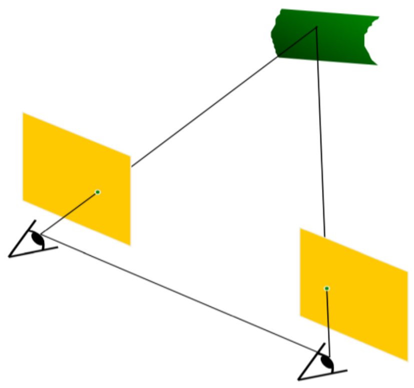


Picture 1

(a) and (b) - Examples of stereo vision procession

The two pictures above are from the Stanford University`s lection (http://vision.stanford.edu/teaching/cs131_fall1314_nope/lectures/lecture9_10_stereo_cs131.pdf).

We are considering the case when cameras are directed parallel to each other, it means that image planes of cameras are parallel to each other. Also we assume that the pictures are aligned in the way that cameras are shifted only in the horizontal axis and there is not a vertical shift. We reckon that the object is located far from the cameras. In this case, there are no pieces which are visible only for one camera and are in the blind area for another one.



Picture 2

Our assumption of how the object and two cameras must be located

Unless there are a lot of restrictions, the examples of such conditions we can often see in our everyday life. Around half of smartphones have at least two cameras, and they are already using the stereo vision to blur the background of the photo, to recognize persons on photos and even create pseudo 3d images. Furthermore, it's usually much easier to place two cameras so they are parallel to each other, as it takes less space. In addition, if we'll start moving such a camera the process of finding the shape of objects or recreating the area 3d map tends to be much simpler in comparison to a single camera.

So the topic isn't new and a lot is already done, but it's connected to our everyday life, but it means that now it's the best time to explore and develop the algorithms and methods to improve and use this technology. And that's what we are doing.

Short overview of related work and possible approaches to solution

We investigated detailed research about Fast Approximate Energy Minimization via Graph Cuts (http://www.cs.cornell.edu/rdz/Papers/BVZ-pami01-final.pdf). The information which we have found there made us change our view on the algorithm a little bit and use another approach. It is all about using energy functions to determine whether pixels are correctly labelled (each pixel should be "attached to its object"). Taking correct energy function results in significant sharpness increase of objects` contours. After finding one we need to find such labelling that minimizes energy function. This can be achieved by creating a weighted graph with labels as its terminal nodes and pixels as its regular nodes and running a minimum graph cut algorithm on it.

Another possible approach is described here:

https://habr.com/ru/post/152553/.

The main difference is that the Particle Filter algorithm doesn't care about the whole image: it sticks to a selected pixel and tries to find its position on the shifted picture. At first its searching area is quite big, but with more pixels selected it learns to optimize the searching area. The author claims repetitive objects cannot affect the algorithm.

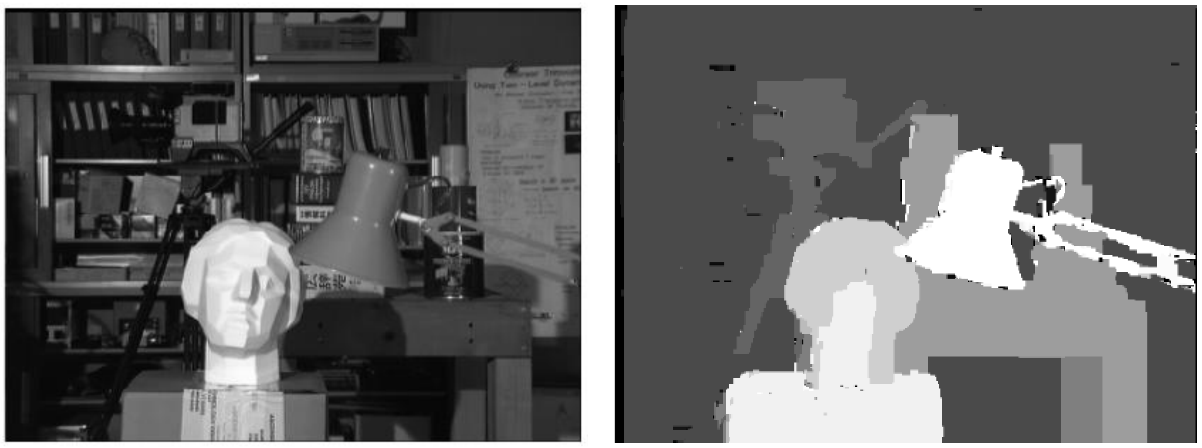
Pros and cons of the algorithm

The accuracy of image splitting is very high with the relatively simple realization of the algorithm. Also, the most significant thing which distinguishes this method among others is that many local optimization techniques use what we will call standard moves, where only one pixel can change its label at a time, while this model encourages labellings consisting of several regions where pixels in the same region have similar labels.

What is more, we are able to choose energy function which would suit a specific picture the best.

Talking about the disadvantages of the approach, the worst part is that this method is iterative - in each cycle, the algorithm performs an iteration for every pair of labels, in a certain order that can be fixed or random. A cycle is successful if strictly better labelling is found at any iteration. As practice shows, these iterations take an extremely big amount of time (up to 10 minutes, but usually 1-2) , which is not rational.

Another disadvantage is that the structure of the object is neglected. The algorithm just extracts the flat objects without any shapes or gradient offsets, which can be observed at the following pictures:

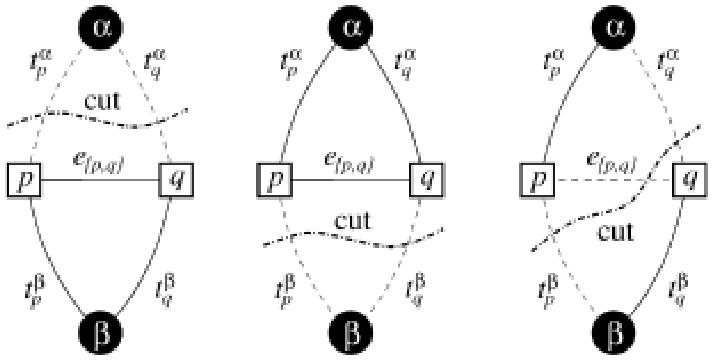


Picture 3

(a) - default image converted to black and white

(b) - result obtained by minimal graph cut algorithm

Yet another drawback is that we cannot influence the amount of different types of objects. What I mean is even starting with say 12 labels one cannot guarantee the depth map would have them all, as graph cut may “ignore” some of them (See more on graph cuts here: https://en.wikipedia.org/wiki/Minimum_cut)



Picture 4

(a) and (b) resulting in no connection to a label

Theoretical part:

The first and essential part of the algorithm is the way we can detect if it is efficient and compare the results. So we need a function that can tell us those things. In the research we have explored it is considered an Energy function that contains two components: first one that evaluates how good pixels from pictures fit to each other;

and the second that corresponds to the idea that different objects should belong to different layers and should be represented as colored in one color. The second summon leads to loss of the structure and shape of objects on a picture. If we consider a ball this method will lead to the one sircle of the same color representing it, even if it is possible to get the ball's shape from pictures. But we can change that functions, tone their contribution to the result. The easiest example of Energy function is:

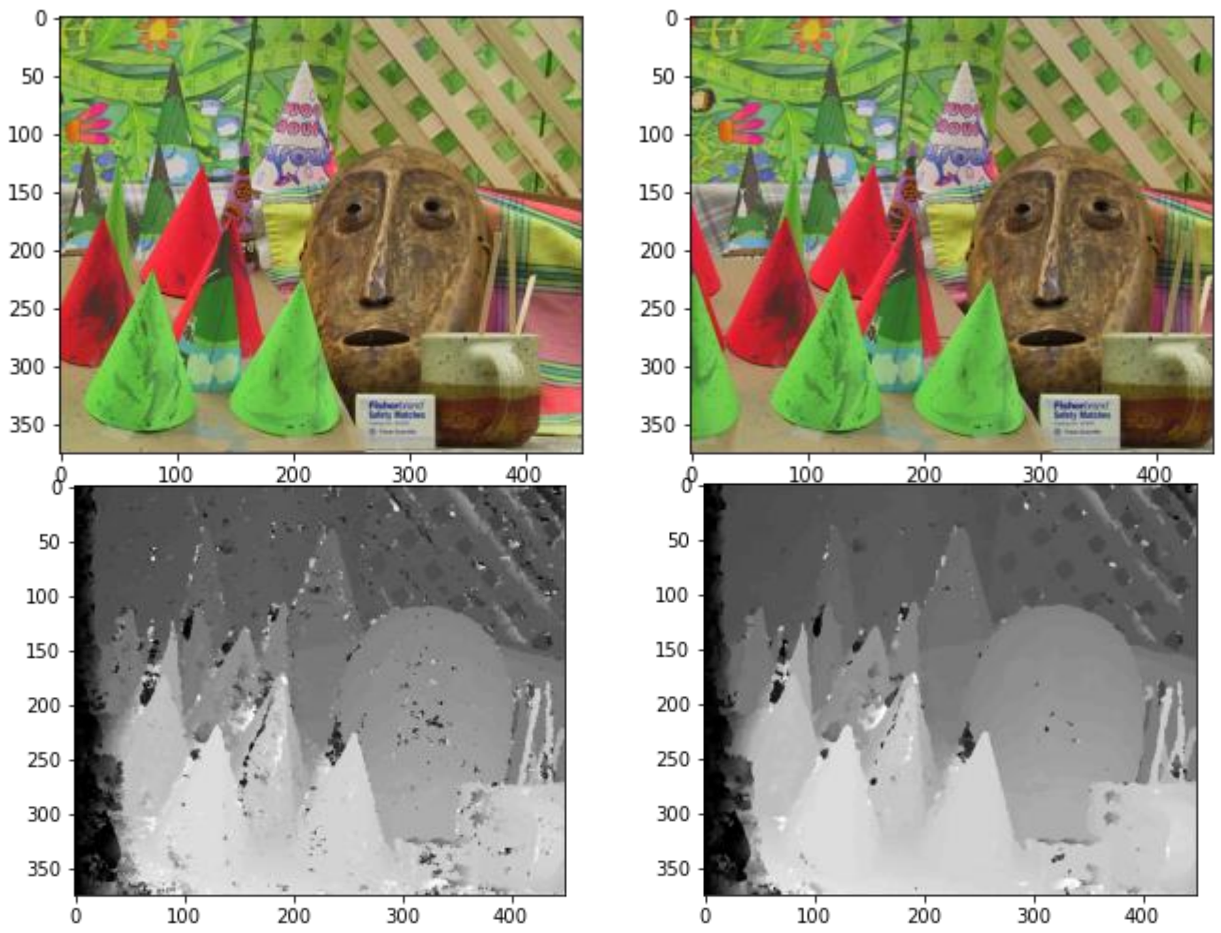
$$E(f) = \sum_{\{p \in P\}} |I(p) - I(f(p))| + \sum_{\{p_1, p_2 \in N\}} k * min(3, |f(p_1) - f(p_2)|)$$

After that we need to construct the starting point and create some a graph with particular structure:

- Every pixel on picture is a vertex
- Adjacent pixels are connected with edges
- We have N vertices (labels) for n distance layers or shift
- Each pixel is connected to one or more label vertix
- Each edge have it's weight due to Energy function

Then we can run an algorithm which will find the best way each pixel is labeled (find all min cuts for each two labels and choose one with the biggest weight). After that try to use swap algorithm or expansion algorithm from the article to improve the result. We will repeat this step until it's giving better results.

Pipeline: First thing first, we decided to check what can be obtained by simply subtracting pictures. Whenever pixels differ too much (meaning they are "taken" from different objects or that the offset is not correct), the resulting pixel would be painted black, otherwise - white.

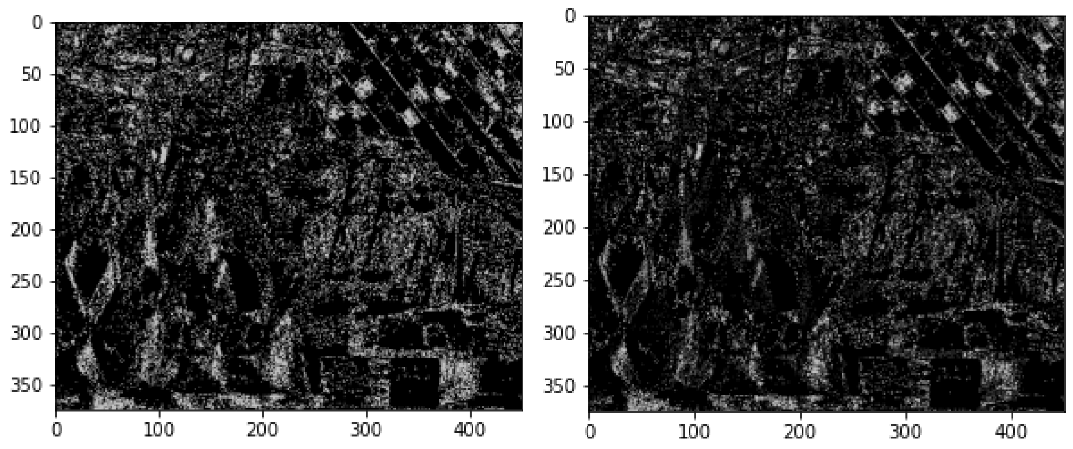


Picture 5

Parts (a), (b) - original images from different perspective

Parts (c) , (d) - depth maps with different smoothness functions

As we didn’t know the exact offset of the pictures, we were continuously shifting one and then subtracting to determine the right one. Below is an example (Picture 6).



Picture 6

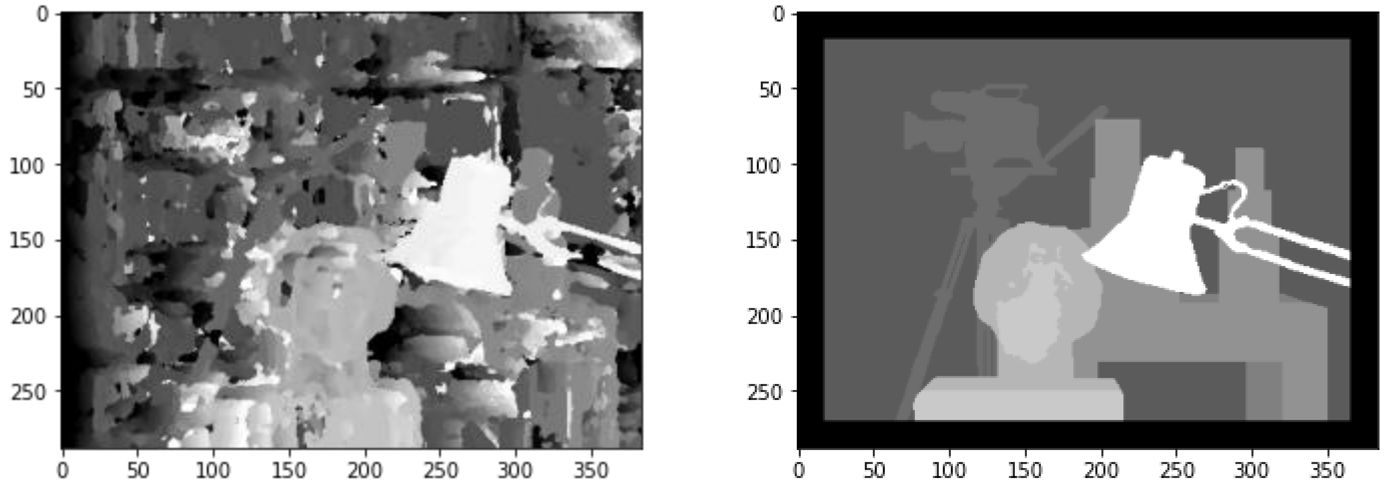
The (b) part has its noise decreased and is more precise

If the pixels coincide we paint it in white, otherwise it’s black. So we can see the ‘wave’ that is passing the whole image. The objects that hit the ‘wave’ at the same time are on the same distance, so on the same shift.

We have achieved a pretty decent result on an easy and less-detailed picture. However, this approach fails dramatically when the number of objects rises, they have complex texture and are approximately the same color. Consider the Picture 3.a.

The results we obtain are far from ideal ones: many objects are being treated the wrong way. Although we can somehow interpret the final result (Picture 7), there is a need for a more precise algorithm, as the objects` contours are blurry.

The one way we can improve the result is to decrease the number of distance layers we need to recognize (in that example we consider 60 possible shifts or distance layers). Otherwise we need to improve the way we are detecting the wave, so it will be more persize.



Picture 7

(a) - our result; (b) - the ground truth image from dataset.

Here is when we can use a minimal graph cut algorithm. After our first image recognition we can run the repetitive function that will improve the results and get a better depth map (Picture 3.b).

Right now we are searching for the best way to implement the algorithm of finding all min-cuts of the graph (we consider Stoer-Wagner algorithm as a good one) and to suit it to our needs (as we have not only to find a minimal cut but also to store altered pixels and their new label).