Images of the Russian Empire

OVERVIEW

Project 1

The goal of Project 1 is to take the digitized Prokudin-Gorskii glass plate images and, using image processing techniques, automatically produce a color image with as few visual artifacts as possible.

PART 1

Approach

Preprocessing

After loading and splitting up the image using the provided boilerplate code I went on to crop the images. As suggested in the task description, the borders do not provide a lot of insights into the alignment of the different channels and for the computation of similarity only the inner parts were used. I set a parameterized cutoff value of 10% of the pixels for each side.

Similarity metrics

Next, I implemented a simple function calculating the Euclidean Distance of two images. The formula was provided in the task description, and the function is a simple one-liner.

Alignment

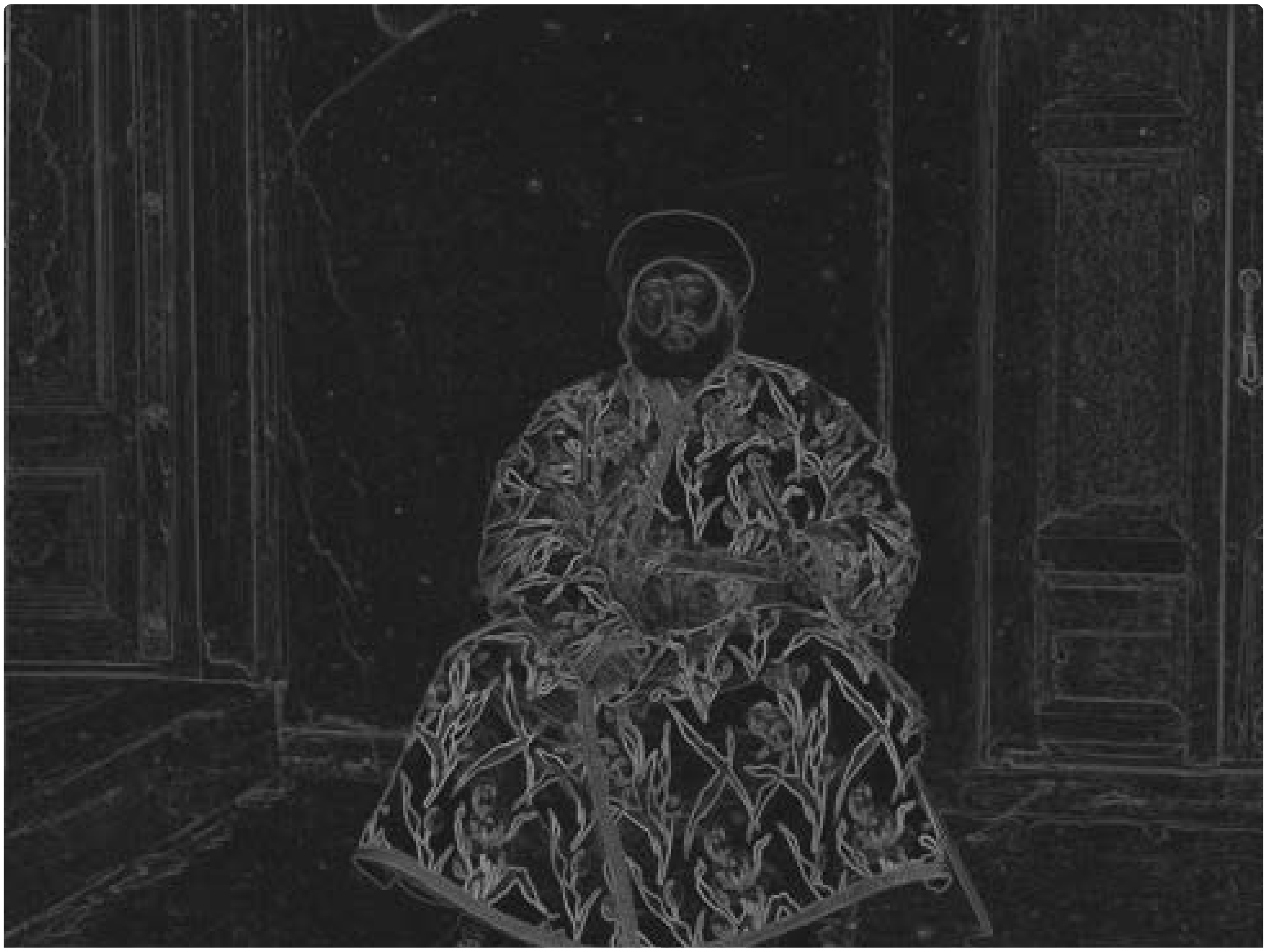
To calculate the right offset for two given channels, I implemented the naive solution of iterating over a predefined range of offset values (in my case 15) for both axis using two nested for-loops. If the current Euclidean Distance is lower than the saved global minimum, the offset values are saved. Once the similarity is computed for the whole range, the offsets corresponding to the minimum distances are returned.

Pyramid

To reduce computation time on the large tif images, I implemented a recursive pyramid search as suggested in the task description. As a first step I created the image pyramid using a function that downsamples the image at every k-th point, where k = 2**i. I then iteratively called the brute force alignment function, starting with the coarsest level of the pyramid (in my case 5, only using every 2**5 = 32nd pixel). The returned shift then needs to be scaled by a factor of 2 for the next level in the pyramid as the resolution increases. I then rolled the channel on the next level using that offset and called the alignment function again.

Using edges

As the results on some of the images are quite bad (i.e. 'emir', or 'melons'), I tried to align the images using their edges and not the raw values. My hope was that this would allow the algorithm to align the channels on the edges. I used a Sobel Edge detection to preprocess both the rolled and reference channel before calling the alignment function. The input image for the alignment function can be seen below, the final alignment results on edge input can be found in the next section (section 3).



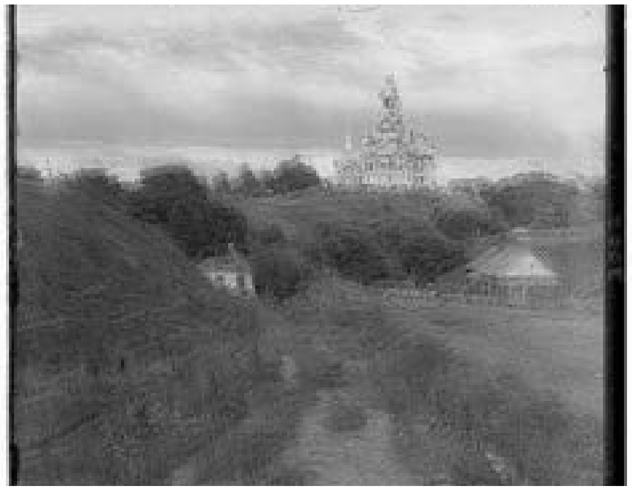
Example input image (Emir) after Sobel edge detection

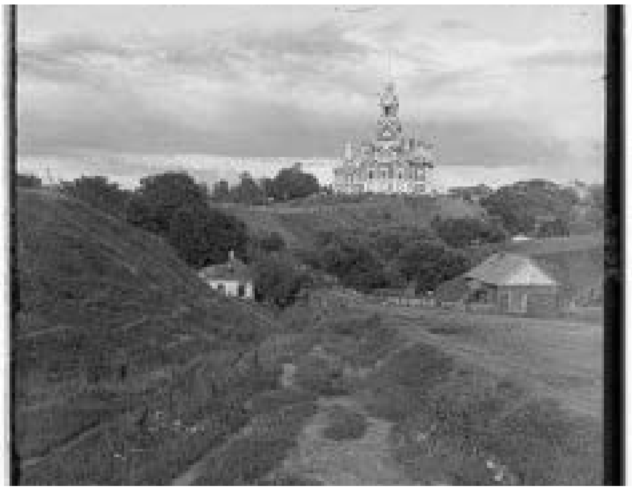
PART 2

Examples

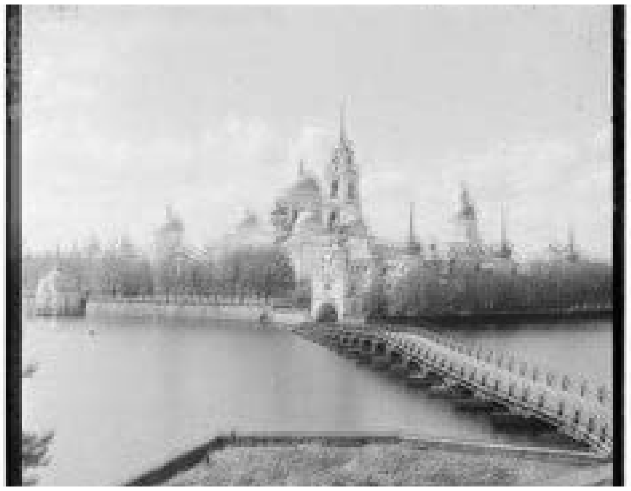
For all images the aligned version is compared to the naive solution of just stacking the original channels onto each other.

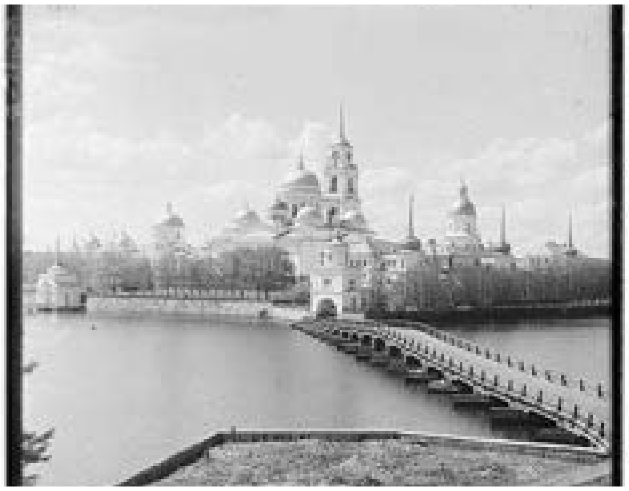
Low resolution (jpg) images


Naive solution

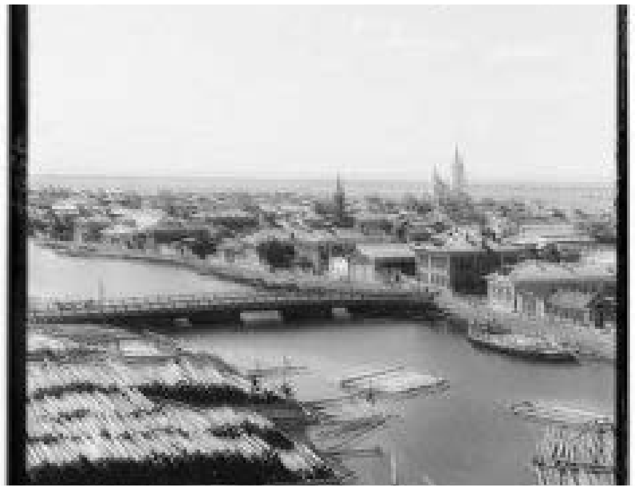

Aligned solution

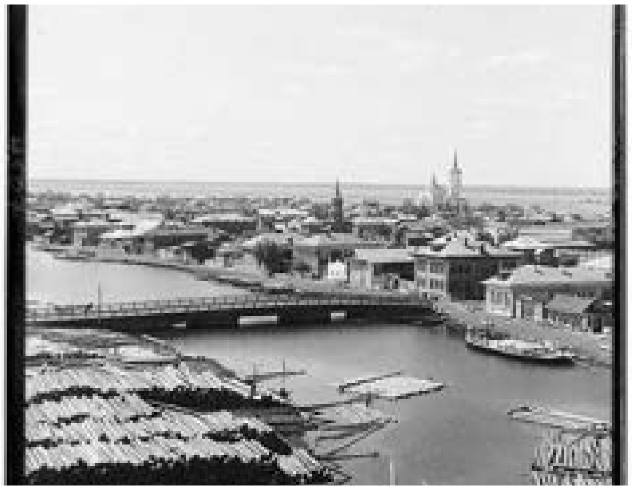
Cathedral: G shift: (5,2) R shift: (12,3)


Naive solution


Aligned solution

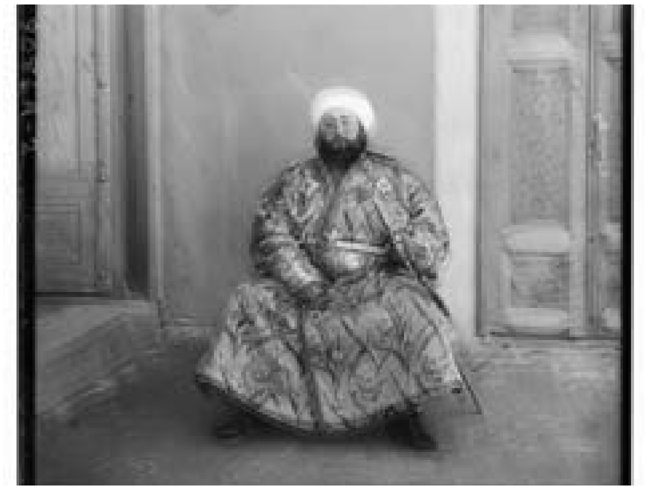
Monastery: G shift: (-3,2) R shift: (3,2)

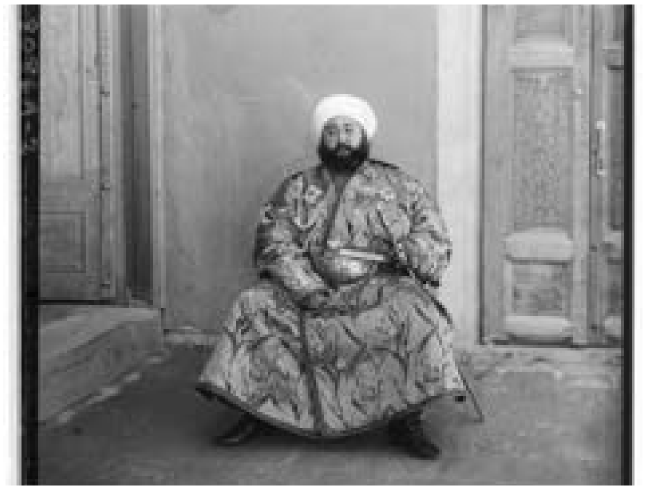

Naive solution


Aligned solution

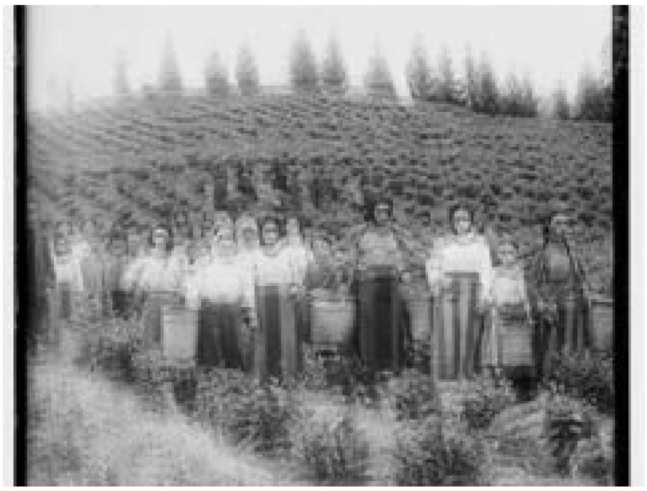
Tobolsk: G shift: (3,3) R shift: (7,3)

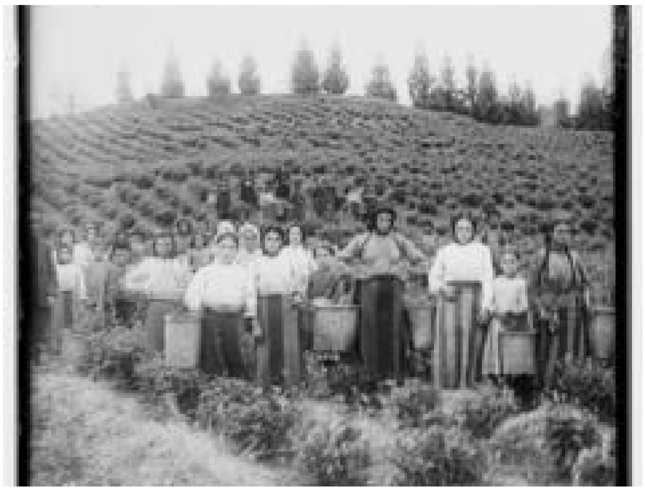
High resolution (tif) images


Naive solution

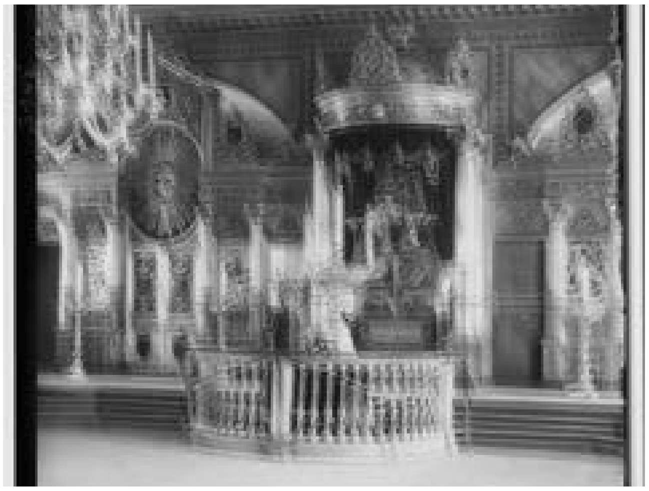

Aligned solution

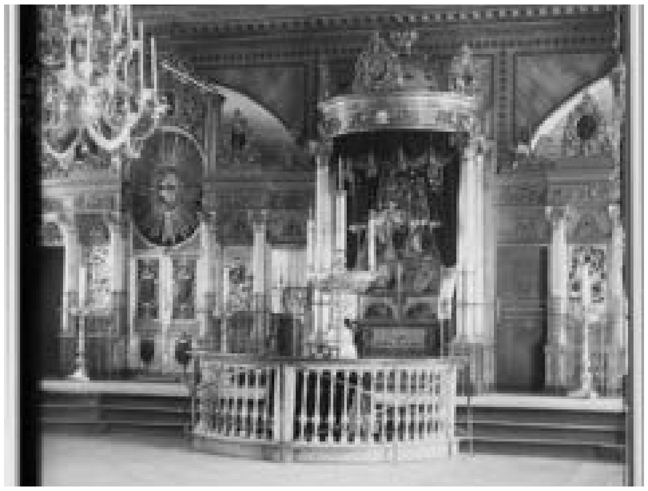
Emir: G shift: (44,24) R shift: (52,-48)


Naive solution

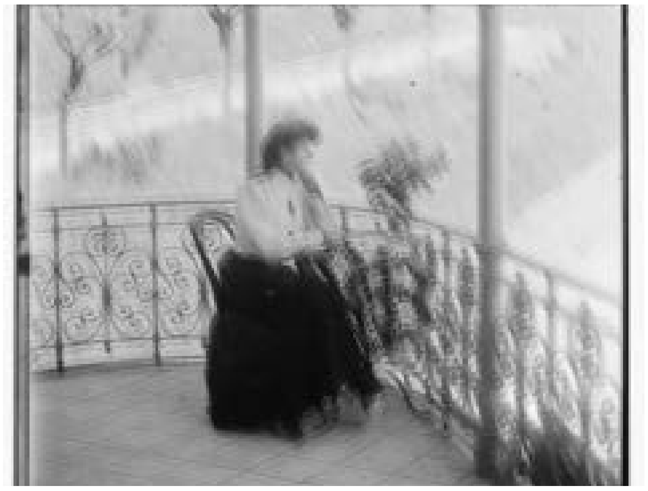

Aligned solution

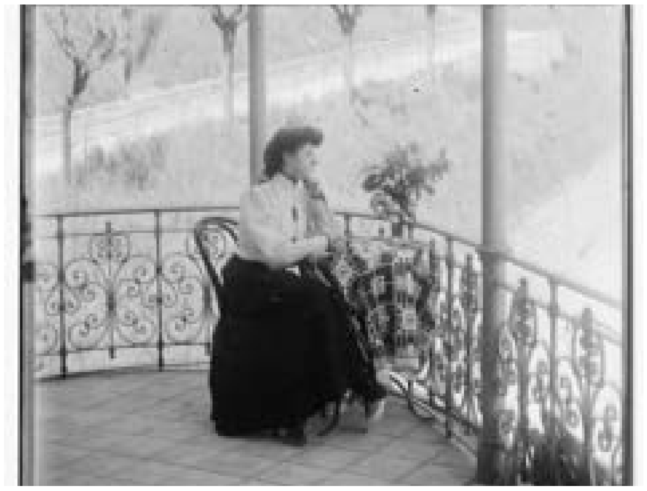
Harvesters: G shift: (50,22) R shift: (60,10)


Naive solution

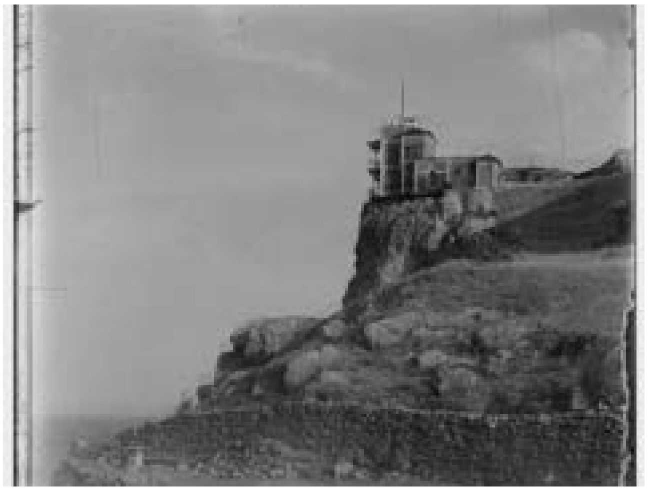

Aligned solution

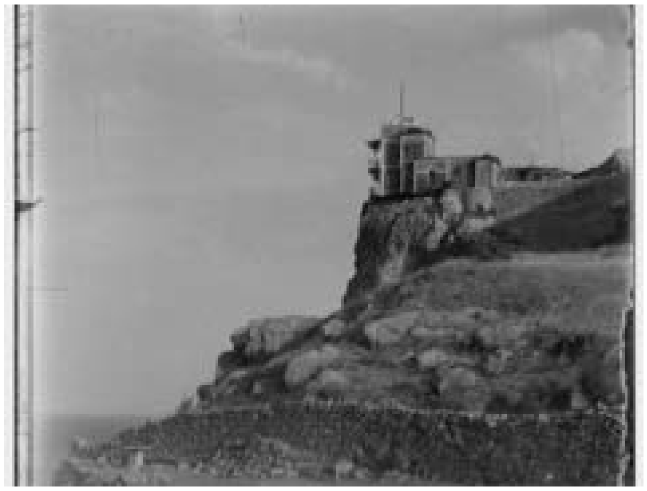
Icon: G shift: (38,22) R shift: (56,30)


Naive solution

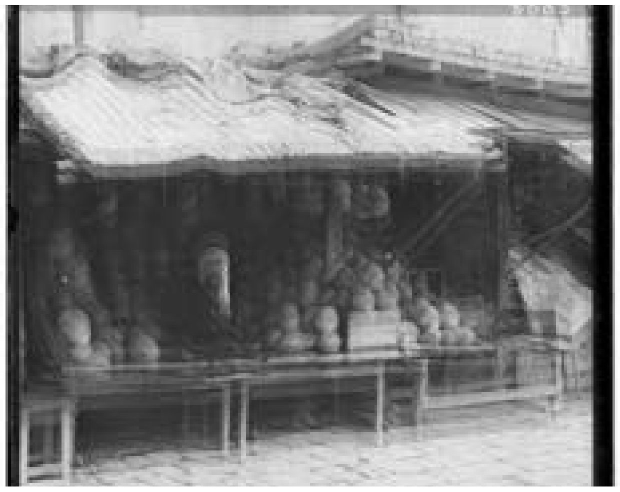

Aligned solution

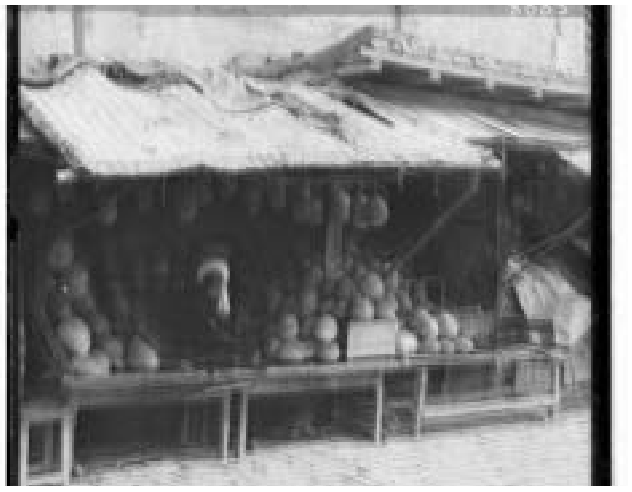
Italil: G shift: (38,24) R shift: (52,34)


Naive solution


Aligned solution

Lastochikino: G shift: (38,24) R shift: (52,34)


Naive solution


Aligned solution

Melons: G shift: (54,6) R shift: (70,14)

Naive solution

Aligned solution

Self Portrait: G shift: (54,22) R shift: (68,36)

Naive solution

Aligned solution

Siren: G shift: (44,-10) R shift: (58,-30)

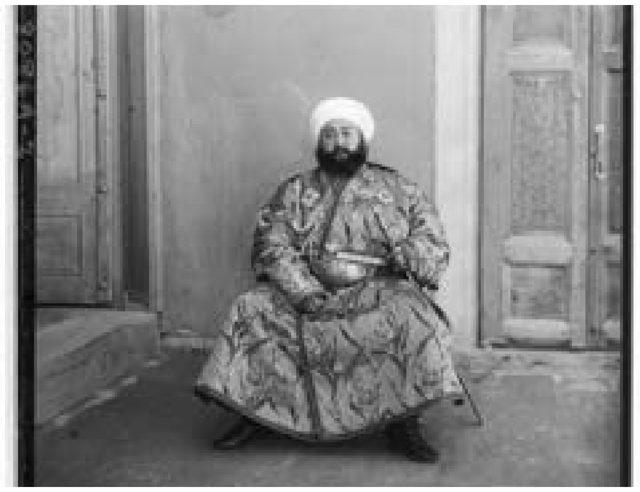
Naive solution

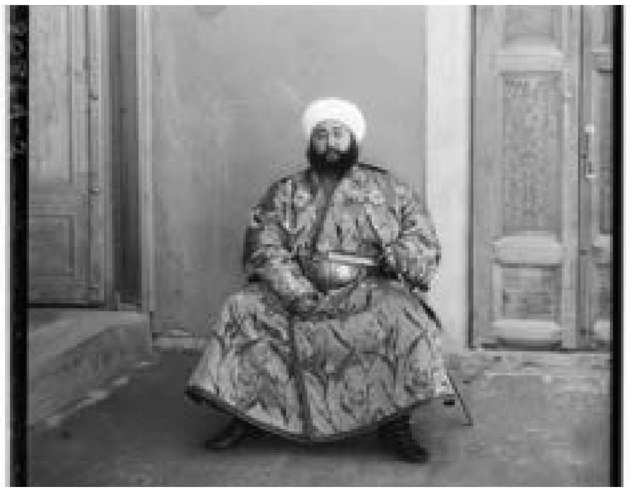
Aligned solution

Three Generations: G shift: (44,18) R shift: (58,16)

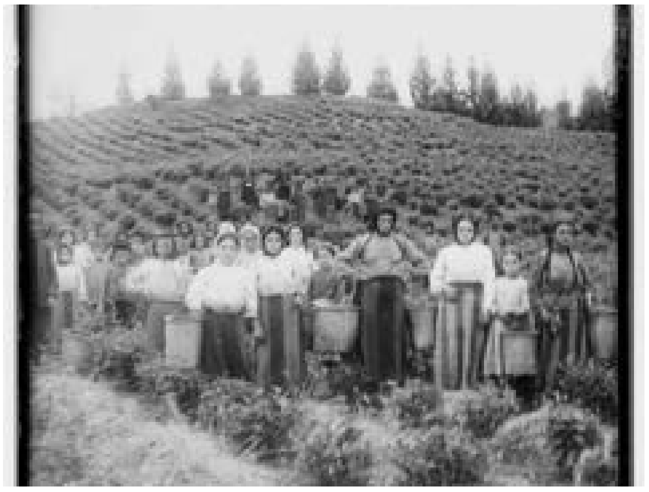
High resolution (tif) images using edges

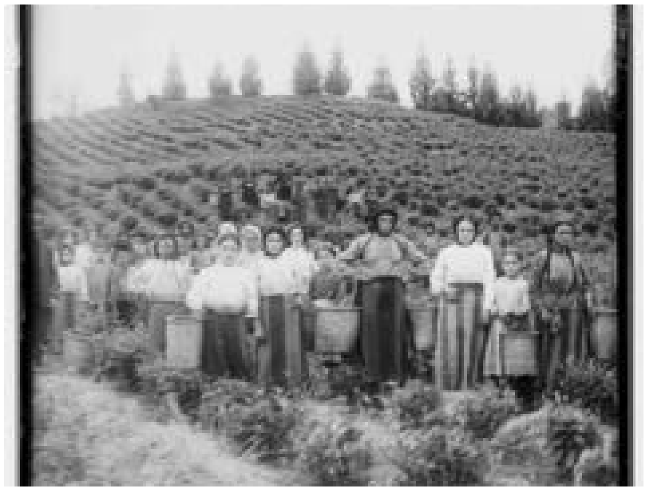
Using edge images as input for the alignment function sometimes improves the outcome, but only in some cases.


Raw value aligned solution

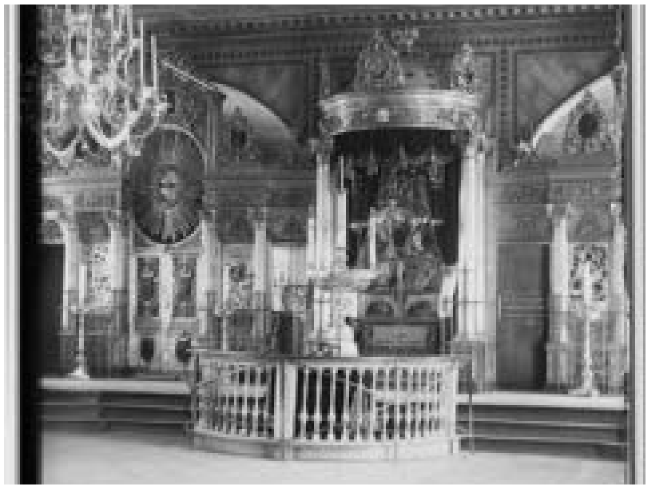

Edge aligned solution

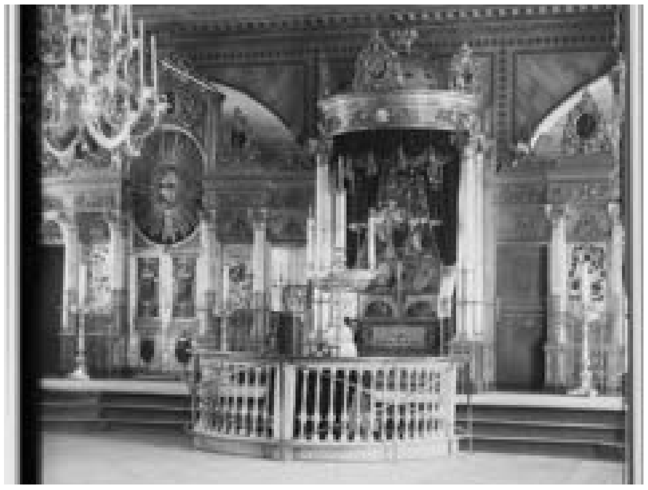
Emir: G shift: (50,26) R shift: (68,40)


Raw value aligned solution


Edge aligned solution

Harvesters: G shift: (54,28) R shift: (70,14)





Raw value aligned solution

Edge aligned solution

Icon: G shift: (42,24) R shift: (64,32)

Raw value aligned solution

Edge aligned solution

Italil: G shift: (44,28)R shift: (64,40)

Raw value aligned solution

Edge aligned solution

Lastochikino: G shift: (-6,-2)R shift: (62,-10)

Raw value aligned solution

Edge aligned solution

Melons: G shift: (64,18) R shift: (72,24)

Raw value aligned solution

Edge aligned solution

Self Portrait: G shift: (64,34) R shift: (76,48)

Raw value aligned solution

Edge aligned solution

Siren: G shift: (48,-12) R shift: (66,-18)

Raw value aligned solution

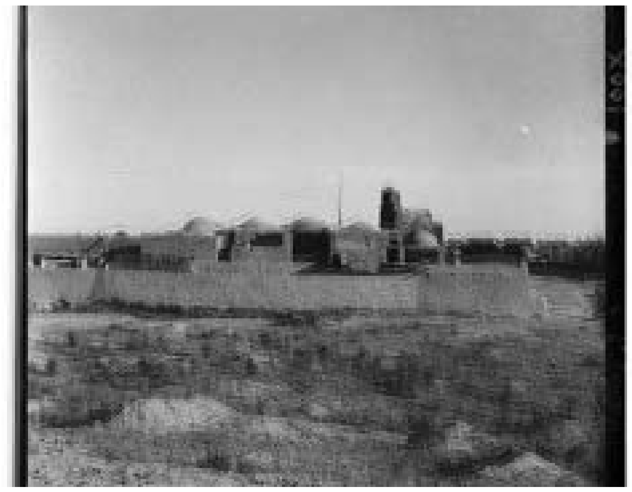
Edge aligned solution

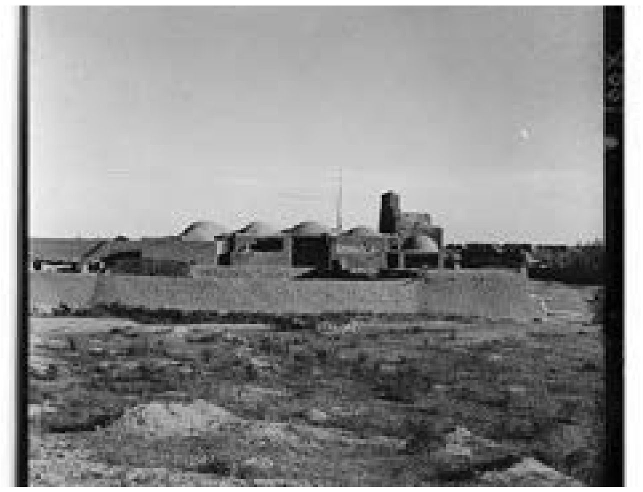
Three generations: G shift: (52,16) R shift: (68,20)

PART 3

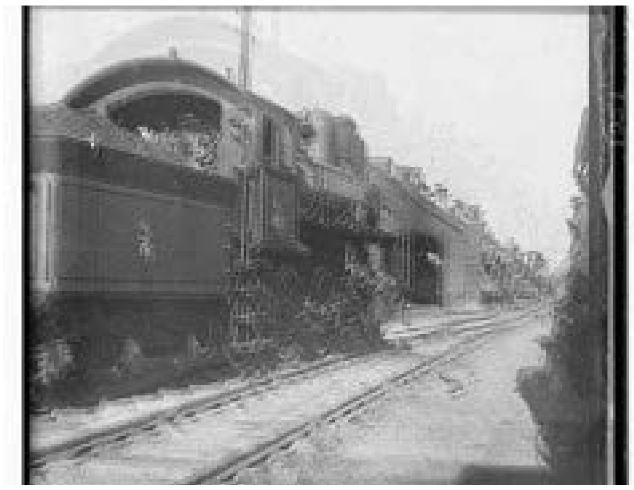
Additional examples from the Prokudin-Gorskii collection

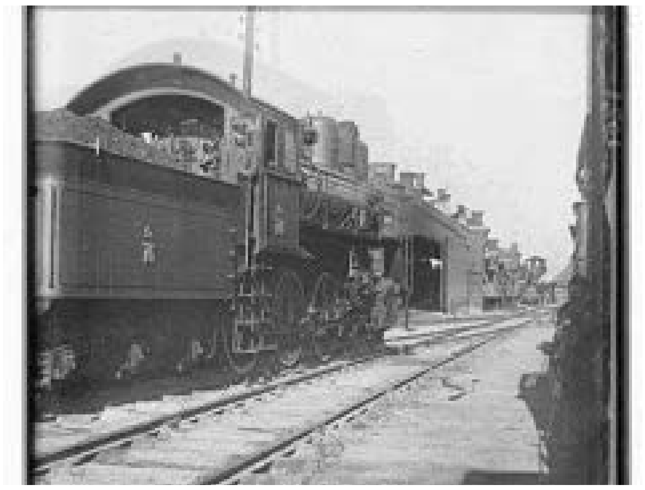
I have used the following three pictures as examples of my own choosing from the Prokudin-Gorskii collection


Naive solution


Raw value aligned solution

Mosque


Naive solution


Raw value aligned solution

Train

Naive solution

Raw value aligned solution

Stone gate

PART 4

Limitations of my algorithm

The alignment algorithm sometimes failed to align the channels for the large tif images. Even with the more advanced usage of the edges it sometimes failed to correctly align the channels. This is probably due to the search area being too small and the Euclidean distance returning local minima that result in incorrect shifts.

PART 5

Bells and whistles implemented

Better features: I implemented an alignment approach on edges using a Sobel edge detection as preprocessing (see section 2).