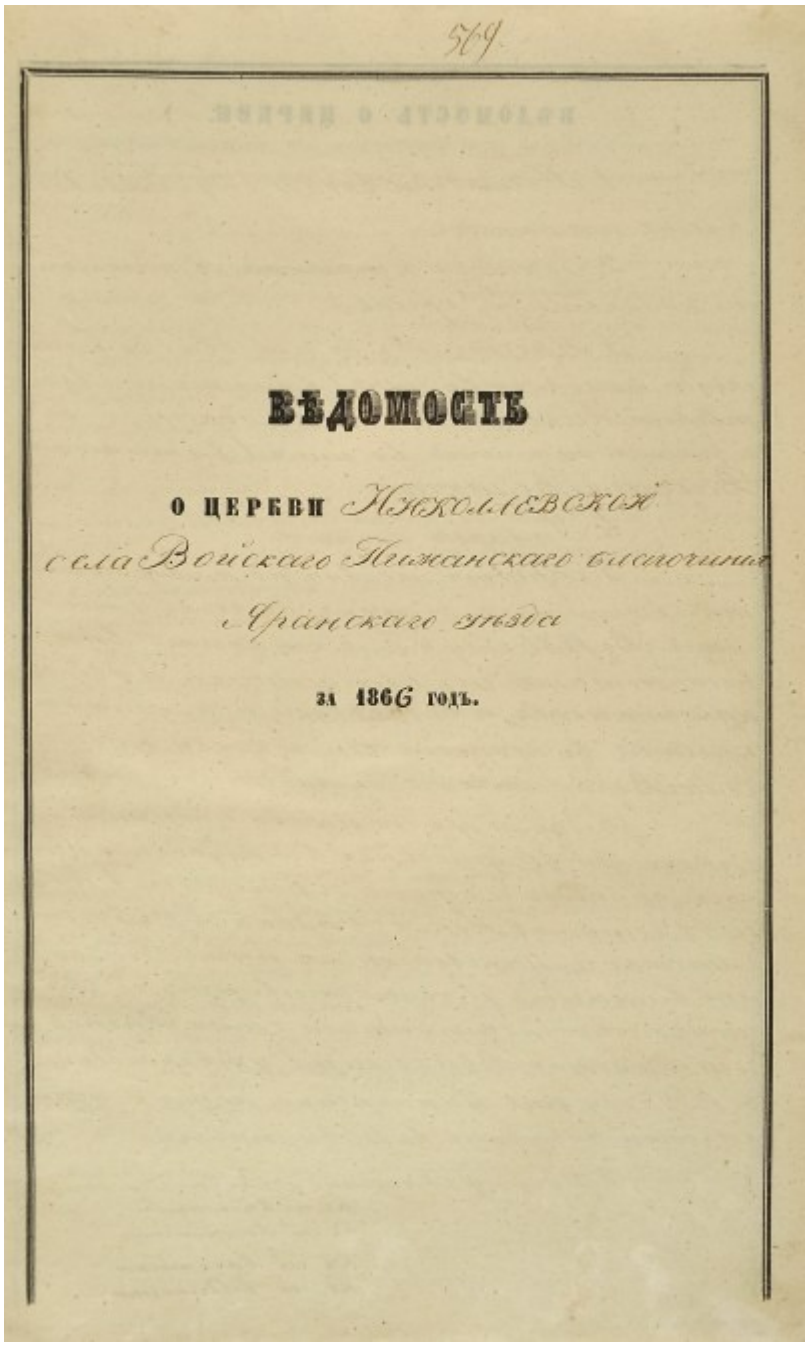
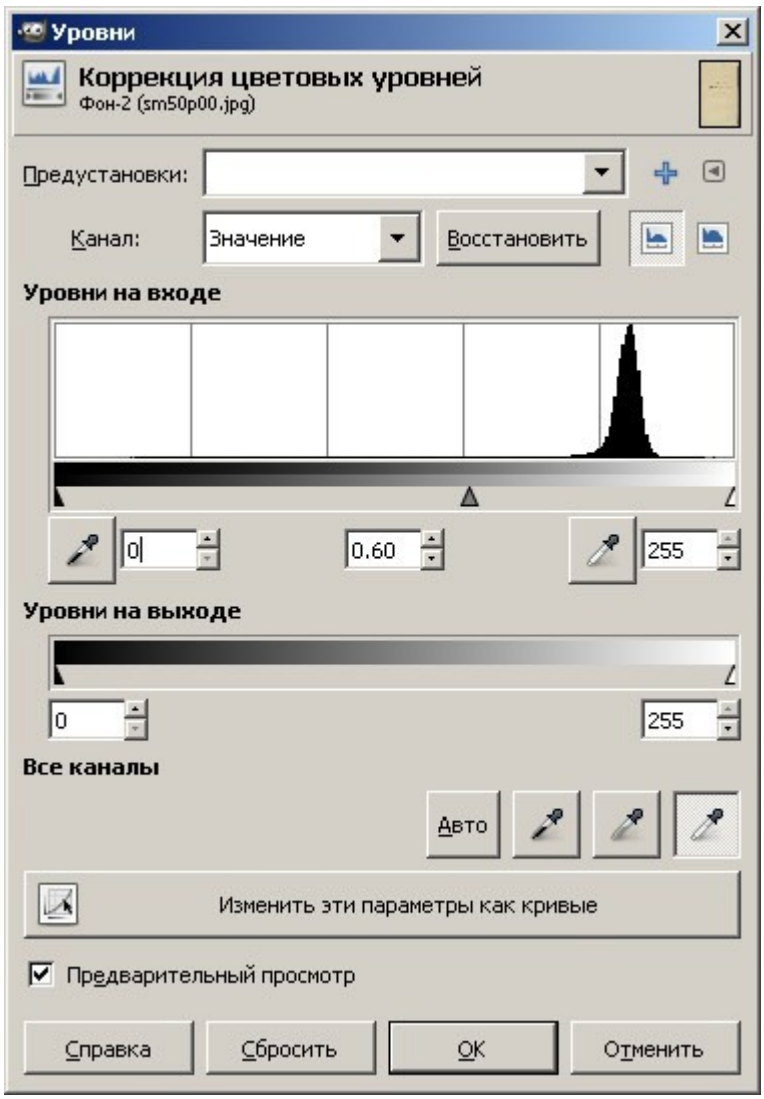
And again about Script-Fu in GIMP

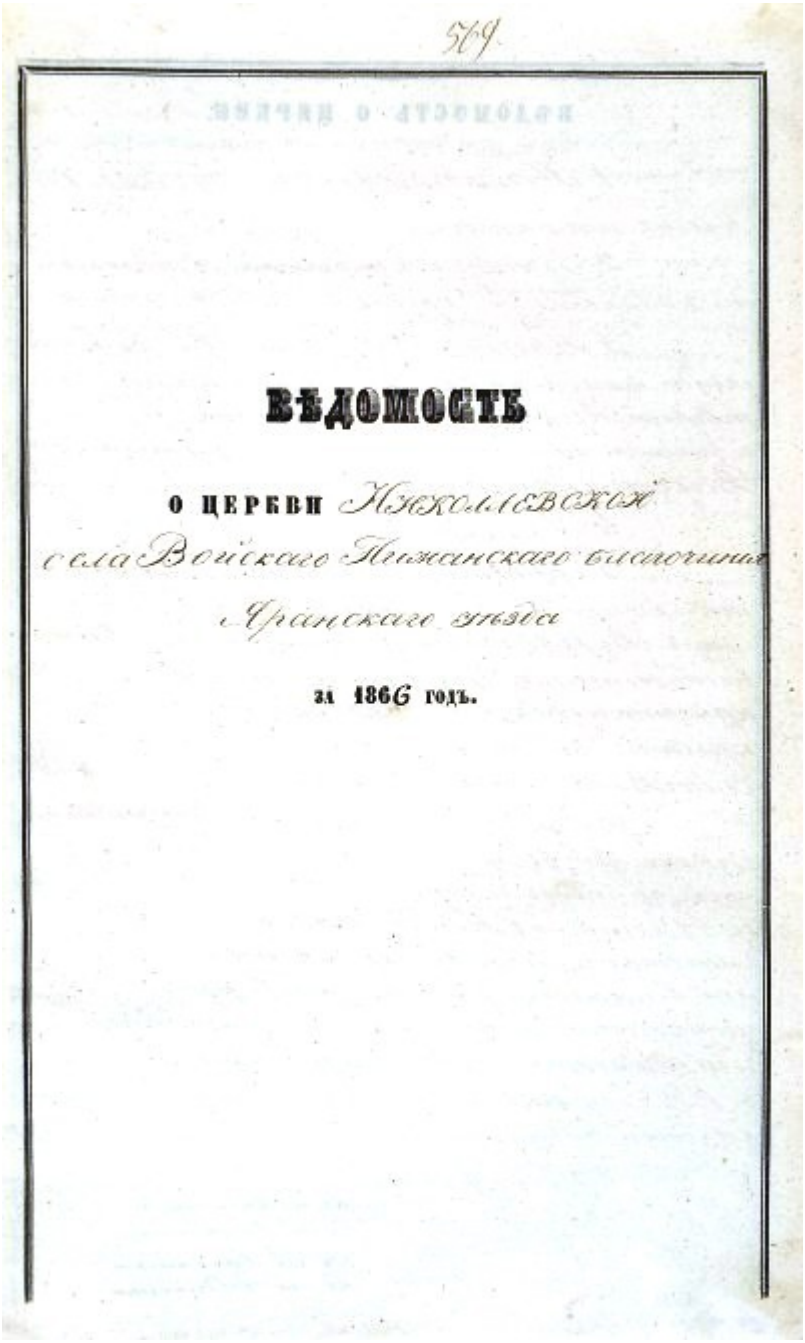
The next set of automation GIMP started with the trivial, in general, puzzles. I brought a bunch of scanned documents, where the scanner was incorrectly displayed color. The result - instead of white paper, obtained some brown slime. Here, for example:



is treated in the GIMP is simple: the tool is taken "levels" in colors of white with a pipette to a point where poking should be white, and gamma of 0.6 is placed to make the gradient color in the "darkness."



That's all. It turns out that something like this:



ВѢДОМОСТЬ

О ЦЕРКВИ Николаевской
села Войского Нижнаго благочинія
Ирбитскаго уѣзда

за 1866 годъ.

The only problem is that such files - not a dozen, and do so every - otuplyayusche and tedious. Conclusion - to be done so that the editor himself did all the steps. At the very least. As a maximum - that it will open files and itself-recorded.

So, what to do minimal script:

1. to find a "white" point of the picture;
2. set it to "levels" as white;
3. set the range of 0.6;
4. processed image.

There is only one question - how do we define "white" point? Decisions "in the forehead" - two: take any fixed point (such as a sheet in the middle), and run diagonally across the sheet, find the brightest. The first option would work, but except in cases where the middle point of the leaf black. The second would be too slow - Scans the "big".

By combining both methods of implementing such an idea: take the 100 points at random pictures, and calculate the average of them. In the first place - a relatively fast (only 100 points), and secondly - the average will be the background color (it is also the expectation, it is the most probable value, that is precisely the background color of the sheet with the text), in the third it is really the background - by random selection of points - to get to the point of "text" is much less likely than in the "background".

The average of each color will be assumed separately, and levels of govern accordingly separately. Well, add a separate scale for simplicity.

```
(While (<y 100)
(Set! i 0)
(Set! px (cadr (gimp-drawable-get-pixel dr
(Rand (- (car (gimp-drawable-width dr)) 1))
(Rand (- (car (gimp-drawable-height dr)) 1))
))
)
(While (<i 3)
(Aset xc i (+ (aref xc i) (aref px i)))
(Set! i (+ i 1))
)
(Set! y (+ y 1))
(Gimp-progress-update (/ y 100))
)
```

pre-xc must be initialized with zeros. in implementing the scheme in gimp (or in my opinion) there is a glitch, so the initialization had to be done explicitly:

```
(Set! i 0)
(While (<i 3)
(Aset xc i 0)
(Set! i (+ i 1))
)
```

And finally - we calculate the average, and adjusted levels:

```
(Set! i 0)
(While (<i 3)
(Set! y (round (/ (aref xc i) 100)))
(Gimp-levels dr (+ i 1) 0 y 1.0 0255)
(Set! i (+ i 1))
)
```

everything in place, adding a "standard" harness looks like this:

```
(Define (script-fu-back-to-white img)

(Gimp-context-push)
(Gimp-image-undo-group-start img)

(Gimp-progress-set-text "Correcting background")
```

```
(Let
(
(Xc # (0 0 0))
(Px # (0 0 0))
(Dr 0)
(Y 0)
(I 0)
)

(Set! xc # (0 0 0))
(Set! dr (car (gimp-image-get-active-drawable img)))

(Set! i 0)
(While (<i 3)
(Aset xc i 0)
(Set! i (+ i 1))
)

; (Write_xc_message "before:" xc)

(While (<y 100)
(Set! i 0)
(Set! px (cadr (gimp-drawable-get-pixel dr
(Rand (- (car (gimp-drawable-width dr)) 1))
(Rand (- (car (gimp-drawable-height dr)) 1))
))
)
(While (<i 3)
(Aset xc i (+ (aref xc i) (aref px i)))
(Set! i (+ i 1))
)
(Set! y (+ y 1))
(Gimp-progress-update (/ y 100))
)

; (Write_xc_message "after:" xc)

(Set! i 0)
(While (<i 3)
(Set! y (round (/ (aref xc i) 100)))
(Gimp-levels dr (+ i 1) 0 y 1.0 0255)
(Set! i (+ i 1))
)
(Gimp-levels dr 0 0 255 0.60 0255)
)


(Gimp-image-undo-group-end img)
(Gimp-displays-flush)
(Gimp-progress-update 1.0)
(Gimp-context-pop)
)
```

Now the small remake of the tutorial function in a script-fu from Ghimpu for the files:

```
(Define (batch-back-to-white pattern)
(Let *
(
(Filelist (cadr (file-glob (string-append pattern "\ \ *. jpg") 1)))
)
(While (not (null? Filelist))
(Let *
(
(Filename (car filelist)
)
(Image (car (gimp-file-load RUN-NONINTERACTIVE filename filename)))
(Drawable (car (gimp-image-get-active-layer image))))
(Script-fu-back-to-white image)
(Gimp-file-save RUN-NONINTERACTIVE image drawable filename filename)
(Gimp-image-delete image)
)
(Set! filelist (cdr filelist))
)
)
)
```

and it remains only to register our new plug-in editor:

```
(Script-fu-register "script-fu-back-to-white"
"Reset background to white"
"Reset background, by changing channels levels upper value to average"
"Leonid Koninin"
"Leonid Koninin"
"2011"
"RGB *, GRAY *"
SF-IMAGE "Image" 0
)

(Script-fu-register "batch-back-to-white"
"Reset background to white (all *. jpg in directory)"
"Reset background, by changing channels levels upper value to average"
"Leonid Koninin"
"Leonid Koninin"
"2011"
"RGB *, GRAY *"
SF-DIRNAME "Directory" ""
)

(Script-fu-menu-register "script-fu-back-to-white"
"<Image> / Filters / Leon")

(Script-fu-menu-register "batch-back-to-white"
"<Image> / Filters / Leon")
```

that's all, though for such a small script I fiddled badly - not enough experience. the script can be downloaded here .