Problem formulation.

Formulation

As it was mentioned above, we work with the images which after some homography transformation can be represented as a sum of the low rank matrix A and a sparse error matrix E. So we can fromulate the optimization objective in the following way

$$\min rk(A) + \|E\|_0$$

Than we add the constraint

$$\tau \bigcirc I = A + E$$

where τ is the homograpy we are searching for and I is the matrix of the original image. As it was show recently [1], this problem usually can be reformulated in the following way:

$$\begin{gathered}\min \|A\|_\sigma + \|E\|_1 \\ \tau \bigcirc I = A + E\end{gathered}$$

Working with nuclear and the first norm is much more convinient than with rank and zero norm.

Summary

Our task will be to write python implementation of TILT, and try to evaluate performa*nce.*

Data.

Actually data for this task is floating arround us every second. We used some pictures from original paper[2], and also some pics of Skoltech, chessboard, e.t.c.

Evaluation.

Homography

To strart with let us precisely define the transformations we are working with. Homography is a coordinate transformation which transform lines to lines. In homogenius coordinates it can be represented as 3×3 matrix. We can think of our image as of the continious function of two variables $I(x, y)$, where I defines intensity in target point. Let H be the matrix of homography, than we transform the coordinates in the following way

$$\begin{bmatrix} x \\ y \end{bmatrix} \to \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \to H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \to \begin{bmatrix} u \\ v \\ s \end{bmatrix} \to \begin{bmatrix} \frac{u}{s} \\ \frac{v}{s} \\ 1 \end{bmatrix} \to \begin{bmatrix} \frac{u}{s} \\ \frac{v}{s} \end{bmatrix}$$

And thus intensity is transformed like this

$$\tau \bigcirc I(x, y) = I(\frac{u}{s}, \frac{v}{s})$$

Later we will need a Jacobi matrix of this transformation.

Optimization

$$\min_{\tau,A,E} \|A\|_\sigma + \|E\|_1$$
$$\tau \bigcirc I = A + E$$

Objective is convex, but the constrains are strange. So the first idea is to linearise constraints around τ_o , solve covex problem using ALM, shift tau, and repeat until convergence. So on each step of outer loop we formulate new problem:

$$\min_{\Delta\tau,A,E} \|A\|_\sigma + \|E\|_1$$
$$\tau_0 \bigcirc I + \nabla I_\tau \Delta\tau = A + E$$

This problem is solved using augmented lagrangian.

$$L = \|A\|_\sigma + \|E\|_1 + \langle Y, C\rangle + \frac{\mu}{2}\|C\|_F^2$$

where $C = \tau_0 \bigcirc I + \nabla I_\tau \Delta\tau - A - E$.

Basic iteration looks like this:

$$A_k, E_k, \Delta\tau_k = argmin L(Y = Y_{k-1})$$
$$Y_k = Y_{k-1} - \mu_{k-1} C_k$$
$$\mu_{k+1} = \rho\mu_k$$

And the first minimization is solved one by one for each variable.

Algorithm 1 (TILT via ALM)

Input: Initial rectangular window $I \in \mathbb{R}^{m \times n}$ in the input image, initial transformations τ in a certain group $\mathbb{G}$ (affine or projective), $\lambda > 0$.

While not converged **Do**

Step 1: normalize the image and compute the Jacobian w.r.t. transformation:

$$I \circ \tau \leftarrow \frac{I \circ \tau}{\|I \circ \tau\|_F}, \quad \nabla I \leftarrow \frac{\partial}{\partial \zeta}\left(\frac{I \circ \zeta}{\|I \circ \zeta\|_F}\right)\Big|_{\zeta=\tau};$$

Step 2: solve the linearized convex optimization (4):

$$\min_{I^0, E, \Delta\tau} \|I^0\|_* + \lambda \|E\|_1 \quad \text{subject to} \quad I \circ \tau + \nabla I \Delta\tau = I^0 + E,$$

with the initial conditions: $Y_0 = 0, E_0 = 0, \Delta\tau_0 = 0, \mu_0 > 0, \rho > 1, k = 0$:

While not converged **Do**

$$\begin{aligned}
(U_k, \Sigma_k, V_k) &\leftarrow \operatorname{svd}(I \circ \tau + \nabla I \Delta\tau_k - E_k + \mu_k^{-1} Y_k),\\
I^0_{k+1} &\leftarrow U_k \mathcal{S}_{\mu_k^{-1}}[\Sigma_k] V_k^T,\\
E_{k+1} &\leftarrow \mathcal{S}_{\lambda\mu_k^{-1}}[I \circ \tau + \nabla I \Delta\tau_k - I^0_{k+1} + \mu_k^{-1} Y_k],\\
\Delta\tau_{k+1} &\leftarrow (\nabla I^T \nabla I)^{-1} \nabla I^T (-I \circ \tau + I^0_{k+1} + E_{k+1} - \mu_k^{-1} Y_k),\\
Y_{k+1} &\leftarrow Y_k + \mu_k (I \circ \tau + \nabla I \Delta\tau_{k+1} - I^0_{k+1} - E_{k+1}),\\
\mu_{k+1} &\leftarrow \rho \mu_k,
\end{aligned}$$

End While

Step 3: update transformations: $\tau \leftarrow \tau + \Delta\tau_{k+1}$;

End While

Output: I^0, E, τ.

Figure 1:

Figure 1, is from original paper [2], describes algorithm in details.

A little bit more attention should be payed to how we linearize, we need to calculate $\nabla_\tau I$, where τ should be considered as a parametres of transformation. We calculate it the following way:

$$\nabla_\tau I = \frac{\partial}{\partial \tau_i} I(x, y) = \frac{\partial}{\partial \tau_i} I(H(\tau)(u, v)) = \frac{\partial I}{\partial u}\frac{\partial u}{\partial \tau_i} + \frac{\partial I}{\partial v}\frac{\partial v}{\partial \tau_i}$$

Additional improvments

First of all we use kind of branch and bound approach to define best initial transformation. We initialize few different rotation matrixes, apply corresponding transformations to the images, calculate the objective, and choose the one which has lowest objective. We do the same for a set of shift matrix, and set initial matrix as a multiplication of choosen shifts and rotates.

The second idea is to blur image before processing, this makes structures "more low ranked", for example text becomes more low ranked, after bluring.

The last idea is to use pyramid approach. To proceed resized image first and than use transformation we obtained as initial for the whole sized picture. This works well, especcially if we need to proceed large images.

Related work

The list of related articles can be found in literature. Actually this exact algorithm was realised on matlab by Visual Computing Group, Microsoft Research Asia, Beijing and Coordinated Science Lab, University of Illinois at Urbana-Champaign [2]. Also some job was done to recover the shapes of cilindrical objects with low ranked structers on them[3]

Our results

Code

Working code of TILT on python can be found here https://github.com/Anton-Rykachevskiy/pytilt

How it looks

Performance

First of let's see how algorithm without pyramid, blur and branch-and-bound works. Some times it fails to converge to what we expect, and converges to some local minimum, even on quite small error tollerance.

For some pictures it can be solved appliying blur to image.

We didn't see any convergence failtures when we applied branch-and-bound, cause it usually finds rather good initial approximation.

Features

Some times we can apply algorithm to pictures which are not actually lowranked, but have at least some symmetry. For example we can rotate faces.

Also we made an experimet with video generating.

Difficulties and solutions

We faced a lot of troubles with correct image processing on python. The main difficulty was that openCV image transformation routines are able to loose information about image if after transformation it goes otside the frame. To solve this, we wrote a few routines, which shift coordinates by multiplying transformation matrix to special shift matrix, and resize image frame so the significant information is not lost.

Now we have fully working open source code on python which (maybe after little cleaning) can be applied, and improved for further tasks.

\
\
\

[1] Candes, E., Li, X., Ma, Y., Wright, J.: Robust principal component analysis preprint (2009)

[2] Zhengdong Zhang y , Xiao Liang y , Arvind Ganesh z , and Yi Ma: TILT: Transform Invariant Low-rank Textures.