CPSC 542G - Project

by

Edoardo A. Dominici

August 2020

Introduction

For this project I implemented the image retargeting algorithm as described in [1], focusing mostly on solving the resulting constrained quadratic program. Starting from a simplified version with only equality constraints, I experimented with a couple of different approaches and then moved on to add the inequality constraints. I am reporting here the findings.

Model overview

This section contains a quick overview of the problem, mostly in order to make sense of the notation used. For any other details see [1].
The problem faced is that of image retargeting, when rescaling a two-dimensional image of source width W and height H and destination width W' and height H', we wish to find a transformation for the pixels such that the resulting image preserves the salient region of the image. Such regions are represented by a saliency map of dimension WxH which is the only input supplied.
The approach followed is that presented in [1], where they cast it as an energy minimization problem over a MxN mesh placed over the image. We optimize $M + N$ variables which respectively represent the width and height of the mesh cells. This makes the transformation inherently axis-aligned. Two energies are discussed in the paper: ARAP (As Rigid As Possible) and ASAP (As Similar As Possible). The former tries penalizes uniform and non-uniform scaling and the latter just non-uniform scaling. Laplacian regularization can also be added to the system, which favors homogeneous scaling.
The size of the model is usually not very big, grids as coarse as $25x25$ and $50x50$ produce near indistinguishable results from higher resolutions and the energies can be written in the quadratic form:

$$min_x \ \boldsymbol{x}^T Q \boldsymbol{x} + \boldsymbol{x}^T q \tag{1}$$

Which need to be subject to some equality constraints:

$$\sum_{i=0}^{M} x_i = W' \sum_{i=M}^{M+N} x_i = H' \tag{2}$$

Inequality constraints can also be used to prevent x_i from going below a certain L. Optimizing without any inequality constraint can incur the risk of the variables having negative values. This appeared only for extreme deformations without any form of

regularization, it converged to the positive solution in most other cases.

$$x_i > L_i \ \forall i \in [0 \,..\, K) \ where \ K = M + N \tag{3}$$

The constraints can also be written in matrix form as

$$\begin{aligned} A_e x &= b_e \\ A_i x &\leq b_i \end{aligned} \tag{4}$$

A_e has dimension $(2, (N + M))$ and is zero everywhere expect for A_{1i} $1 \leq i \leq N$ and A_{2i} $N \leq i \leq (N + M)$. $A_i = -I$, $b_i = -L$ and $b_e = [W' \ H']^T$.
Minimizing non-uniform scaling means penalizing the difference between height and width of each cell.

$$E_{ASAP} = \sum_{i=0}^{M-1} \sum_{j=M}^{M+N-2} \Omega_{i,j} (f_w x_i - f_h x_j)^2 \tag{5}$$

Where f_w and f_h take care of normalizing the aspect ratio and Ω is the input saliency integrated in the MxN grid. We can see that Q is symmetric and also positive-definite, this really simplifies the problem by making it convex.
The Laplacian regularization term penalizes differences in size between neighboring cells.

$$E_{lapl} = \sum_{i=0}^{M-2} f_w (x_i - x_{i+1})^2 + \sum_{i=M}^{N-2} f_h (x_i - x_{i+1})^2 \tag{6}$$

We can simply add this term to the energy being minimized with a blend factor W_{lapl}. The results shown in the document minimize the As-Similar-As-Possible energy with an added Laplacian regularization term $E = E_{ASAP} + W_{lapl} E_{lapl}$. The weights used for W_{lapl} are different than those in the paper, as I found those way too strong to produce meaningful deformations.

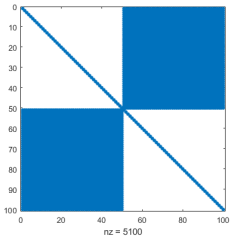

FIGURE 1: Non-zero structure for $\boldsymbol{Q}$ for ASAP energy.

The input saliency map used for all the presented results marks the top-right quadrant of the image to have maximum importance while the remaining pixels have 0, it is a very unnatural saliency map, but allows for quick visual debugging of the results. The input resolution is $2000x2000$.

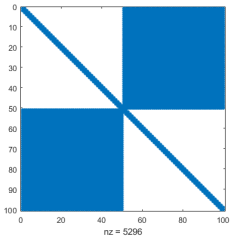

FIGURE 2: Non-zero structure for $\boldsymbol{Q}$ for ASAP energy and Laplacian regularization.

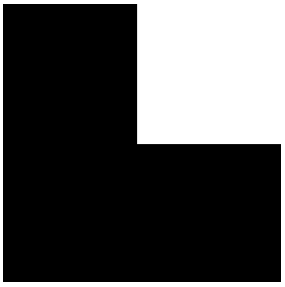

FIGURE 3: Input saliency map

Solving the KKT system

I started by initially ignoring the inequality constraints and trying to solve the equality-only version of the problem. While this in theory simplifies the problem, it makes the solution finding less reliable as the function can now have multiple minimizers. $x^T Qx$ is not guaranteed to be non-negative anymore since we allow for negative x values. In practice, this did not seem to be a particular problem and appeared only for extreme deformations without any form of regularization.

When minimizing a non-linear function under equality and inequality constraints it's common to start from the first-order necessary conditions for x to be a local minimizer of a non-linear function (Karush-Kuhn-Tucker). If x^* is a local solution of $f(x)$ under a set of equality constraints $c_e(x)$, given that both functions are differentiable, there exist a Lagrange multiplier λ^* such that the following conditions are satisfied at (x^*, λ^*):

$$\begin{aligned} \nabla \mathcal{L}(x^*, \lambda^*) &= 0 \\ c_e(x^*) &= 0 \end{aligned} \tag{7}$$

Where $\mathcal{L}$ is the Lagrangian of $f(x)$

$$\mathcal{L}(x^*, \lambda^*) = f(x) - \sum_{i \in E} \lambda_i c_i(x) \tag{8}$$

We can find the minimum of the function by solving for the KKT system as described in equation 7, using the same notation as in the model description we obtain the following linear system:

$$\begin{bmatrix} Q & -A_e^T \\ A_e & 0 \end{bmatrix} \begin{bmatrix} x^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} -q \\ b_e \end{bmatrix}$$

We can also express it in an iterative form by setting $x + p = x^*$, which yields

$$\begin{bmatrix} Q & A_e^T \\ A_e & 0 \end{bmatrix} \begin{bmatrix} p \\ \lambda^* \end{bmatrix} = \begin{bmatrix} q + Qx \\ A_e x - b_e \end{bmatrix}$$

Given the size of the problem, the system above can be solved directly and any naive algorithm is quick enough. Although the system is presented in its iterative form, one step is enough to have a solution with enough ($1e^{-12}$) accuracy.
Some of the results and numbers are shown in the pictures below 11 for a mesh of 10x10, I am using quite different values for W_{lapl} from the ones reported in the paper, but similar results are obtained. Although I did not numerically compare the solutions.

The choice of a feasible or unfeasible starting point does not influence the solution

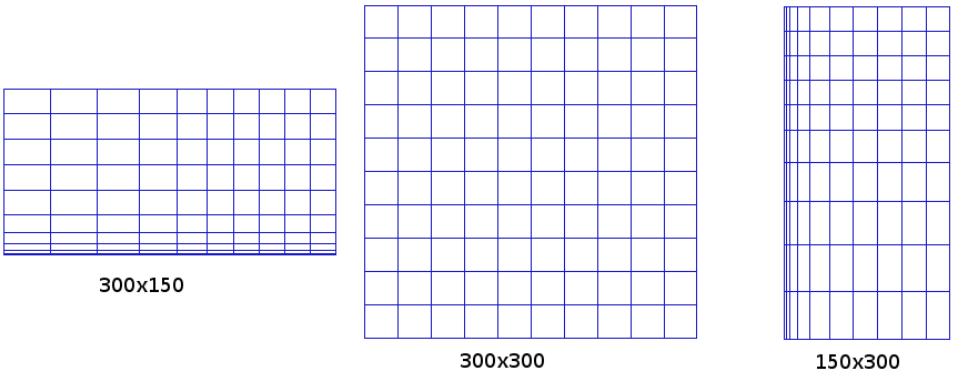

FIGURE 4: Equality-only retargeting result solving the KKT system.

finding.

Penalty method

Another common approach to handing constraints when minimizing functions is the use of penalty functions. Instead of trying to minimize $f(x)$ under $c_e(x)$ constraints, we look for the minimum of the unconstrained function

$$min_x \ \widetilde{f}(x) = f(x) + \beta g(c_e(x)) \tag{9}$$

```
Target Resolution 150 300
Iteration: 0
Error (Residual): 6.35529e-14
Error (Step norm): 112.265
Iteration: 1
Error (Residual): 2.84217e-14
Error (Step norm): 5.21687e-13
----------
Target Resolution 300 150
Iteration: 0
Error (Residual): 2.84217e-14
Error (Step norm): 112.265
Iteration: 1
Error (Residual): 2.84217e-14
Error (Step norm): 4.0616e-13
----------
Target Resolution 300 300
Iteration: 0
Error (Residual): 0
Error (Step norm): 134.164
Iteration: 1
Error (Residual): 5.68434e-14
Error (Step norm): 6.7327e-13
----------
Average iteration count 2
```

FIGURE 5: Equality-only iterations. It takes 2 since the step norm is also used as termination.

Where an artificial penalty β is introduced for violating the constraints, which weights a penalty function $g(f(x))$. By minimizing this function with a large β we hope to obtain a solution which minimizes the original function while satisfying the equality constraints (The contribution of the added term is 0). As we want to greatly penalize any constraint violation, a common loss function which allows for a nice derivation is the quadratic loss

$$g(x) = \frac{1}{2}|A_e x - b_e|^2 \tag{10}$$

We can now rewrite $\widetilde{f}(x)$ in matrix form and simplify:

$$\begin{aligned}\widetilde{f}(x) &= \frac{1}{2}x^T Q x + q^T x + \frac{\beta}{2}(A_e x - b_e)^T(A_e x - b_e) \\ &= \frac{1}{2}x^T Q x + q^T x + \frac{\beta}{2}(x^T A_e^T A_e x - 2A_e^T b_e + k) \\ &= \frac{1}{2}(x^T Q x + q^T x + \beta A_e^T A_e) + (q - \beta A_e^T b_e) + k\end{aligned} \tag{11}$$

To find the minimum we set $\nabla \widetilde{f}(x) = 0$

$$\begin{aligned}\nabla \widetilde{f}(x) &= 0 \\ (Q + \beta A_e^T A_e)x + (q - \beta A_e^T b_e) &= 0\end{aligned} \tag{12}$$

It can also be written in its iterative version. It is interesting to notice that the starting point does matter here. Retargeting to the same resolution as in the previous method converges when starting from a more feasible solution, but finds a different negative minimizer $x \geq 0$ when starting from a unfeasible solution. The number of iterations required to achieve $1e^{-12}$ precision is around 4.

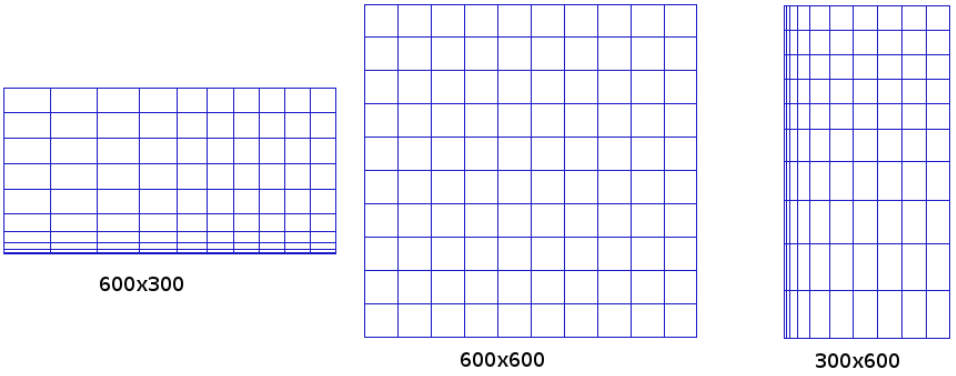

FIGURE 6: Equality-only retargeting result using penalty functions.

```
Target Resolution 300 600
Iteration 1
Error (step norm) 224.536
Iteration 2
Error (step norm) 0.0422935
Iteration 3
Error (step norm) 0.0269874
Iteration 4
Error (step norm) 0.0411645
Iteration 5
Error (step norm) 3.8928e-14
----------
Target Resolution 600 300
Iteration 1
Error (step norm) 224.529
Iteration 2
Error (step norm) 0.0131087
Iteration 3
Error (step norm) 0
----------
Target Resolution 300 300
Iteration 1
Error (step norm) 134.164
Iteration 2
Error (step norm) 0.0269874
Iteration 3
Error (step norm) 0.122562
Iteration 4
Error (step norm) 0.0381432
Iteration 5
Error (step norm) 0
----------
Average iteration count 4.33333
```

FIGURE 7: Equality-only iterations using penalty functions.

Adding Inequality Constraints

In order to always obtain valid solutions we need to enforce inequality constraints. As we transformed the equality constraints into penalty functions, we can do the same for inequality constraints, and minimize

$$min_x \ \widetilde{f}(x) = f(x) + \beta g(c_e(x)) + \gamma g(c_i(x)) \tag{13}$$

The inequalities $A_i x \geq L$ prevent the x_i to go below a certain minimum size L_i: $\forall i \ x_i \geq L_i$. Although from $\gamma g(c_i(x))$, we want the same behavior as from $\beta g(c_e(x))$

the inequalities require us to take an iterative approach in order to guarantee an acceptable solution. The inequality constrains are to be carefully activated only for those variables which are violating them. When all the constraints are active A_i is a negative identity matrix (See the first section).

$$g(c_e(x)) = max(0, -x + L)^2 \tag{14}$$

From an implementation perspective the use of $max()$ requires a per-variable evaluation to check if it's inside or outside the valid solution space. This is achieved by updating the inequality constraints matrix A_i with a 0 at position (i, i) when the constraint is not violated or a 1 in case it is. We can then write

$$\begin{aligned}\widetilde{f}(x) &= \frac{1}{2}x^TQx + q^Tx + \frac{\beta}{2}(A_ex - b_e)^T(A_ex - b_e)\frac{\gamma}{2}(-A_ix + L)^T(-A_ix + L) \\ &= \frac{1}{2}x^TQx + q^Tx + \frac{\beta}{2}(x^TA_e^TA_ex - 2A_e^Tb_e + k) + \frac{\gamma}{2}(x^TA_i^TA_ix - 2A_i^TL + k) \\ &= \frac{1}{2}(x^TQx + q^Tx + \beta A_e^TA_e + \gamma A_i^TA_i) + (q - \beta A_e^Tb_e - \gamma A_i^TL) + k\end{aligned} \tag{15}$$

Taking the gradient and setting it to 0

$$\begin{aligned}\nabla\widetilde{f}(x) &= 0 \\ (Q + \beta A_e^TA_e + \gamma A_i^TA_i)x + (q - \beta A_e^Tb_e - \gamma A_i^TL) &= 0\end{aligned} \tag{16}$$

We can now derive an update step by setting $x_{k+1} = x_k + p$ and rewriting the above equation. Solving for the update step $\boldsymbol{p}$ follows the direction of the function gradient to 0.

$$(q - \beta A_e^Tb_e - \gamma A_i^TL)\boldsymbol{p} = q + \beta A^Tb + \gamma A_i^TL - (Gx + \beta A_e^TAx + \gamma A_i^TAx) \tag{17}$$

One of the main problem faced when implementing the method was the convergence near the solution, by only using the gradient size as termination criterion instability occurred which caused the algorithm to run up to the maximum number of iterations. When x is near x^* the penalty function causes a big step size if violated, which results in the termination criterion never being met (jumping around the solution).
Something as simple as reducing the step size as x^* is approached solved the problem. Another way is that instead of starting with a very large number for β and γ (Any value between $1e^6$ and $1e^{10}$ seemed to work well), we can start with lower values and make the penalty factors exponentially grow. Once the minimum for a fixed β and γ has been found, the solution is used as a starting point for the next iteration with updated $\beta_{k+1} = K\beta_k$ and $\gamma_{k+1} = K\gamma_k$ for $K > 1$.
Numerical instabilities when working with penalty functions at the last iterations seem to be a known problem and slowly increasing the penalty factors provided a better

convergence. The error threshold used as termination criteria is the norm of the step $|\nabla x|^2$. Independently from the feasibility of the initial solution, the penalty method in all the tests converges in around 5-7 iterations to the minimum solution, with starting values $\beta = \gamma = 100$. When uniform scaling is applied the function appears to converge very quickly.

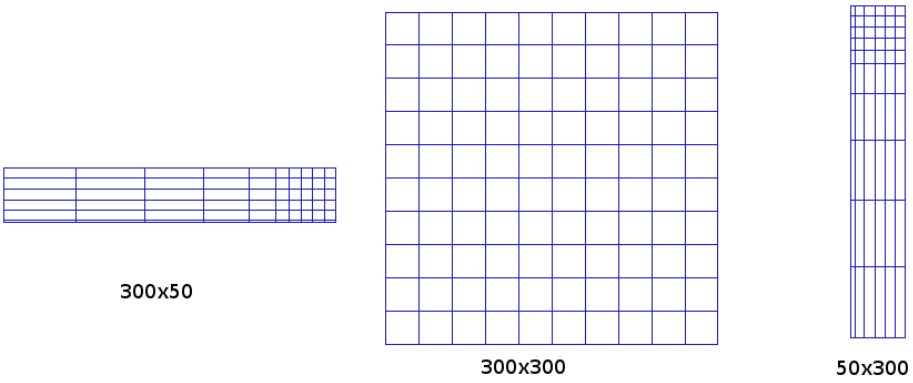

FIGURE 8: Inequality-constrained version solved using penalty functions. Note that the deformations here are more extreme than the previous examples.

Primal-Dual path following

Optimizing adding a penalty function for the constraints, while converging in 5-7 iterations, is not a particularly sophisticated method. The problem is question might not be a good benchmark given its simplicity, but there are faster and more elegant approaches to handle constrained optimization. One such family is the Interior-Point methods, they can have feasible or unfeasible starting guesses and converge to the solution without lying on the boundary between the feasible and unfeasible regions, which avoid spurious solutions solving the KKT system, but not satisfying $x, s \geq 0$. They are called Interior-Point as they remain inside the starting feasibility region.

```
Target Resolution 300 300
Iteration 1
Error (step norm) 166.75
Iteration 2
Error (step norm) 12.1358
Iteration 3
Error (step norm) 0
----------
Target Resolution 50 300
Iteration 1
Error (step norm) 126.58
Iteration 2
Error (step norm) 61.9297
Iteration 3
Error (step norm) 37.0865
Iteration 4
Error (step norm) 2.3228e-07
Iteration 5
Error (step norm) 2.3228e-08
Iteration 6
Error (step norm) 2.32279e-09
Iteration 7
Error (step norm) 2.3231e-10
Iteration 8
Error (step norm) 2.32108e-11
Iteration 9
Error (step norm) 2.299e-12
Iteration 10
Error (step norm) 1.72285e-13
----------
Target Resolution 300 50
Iteration 1
Error (step norm) 126.58
Iteration 2
Error (step norm) 61.9297
Iteration 3
Error (step norm) 42.9192
Iteration 4
Error (step norm) 2.64036e-06
Iteration 5
Error (step norm) 2.99026e-08
Iteration 6
Error (step norm) 2.99024e-09
Iteration 7
Error (step norm) 2.99017e-10
Iteration 8
Error (step norm) 2.99323e-11
Iteration 9
Error (step norm) 2.93764e-12
Iteration 10
Error (step norm) 2.13587e-13
----------
Average iteration count 7
```

FIGURE 9: Inequality-constrained iterations using penalty functions. Note that the deformations here are more extreme than the previous examples.

They are derived from the first order optimality conditions. Given the quadratic program as written in the first section, its KKT conditions can be written as

$$\begin{aligned} Qx + q + A_e^T z + A_e^T y &= 0 \\ A_i x + s &= b_i \\ A_e x &= b \\ zs &= 0 \\ s \geq 0, z &\geq 0 \end{aligned} \tag{18}$$

Since Q is positive definite the above conditions are sufficient and we can solve for them to find the solution of the quadratic program [2]. The augmented solution vector X is composed of the grid dimensions x the slack variables $s \in \mathbb{R}^{N+M}$ added to the inequalities to transform them into equalities, and the dual variables $z \in \mathbb{R}^{N+M}$ and $y \in \mathbb{R}^2$ associated with the Lagrange-dual problem

$$\begin{aligned} min_x \ f(x) + yc_e(x) + zc_i(x) \\ y, z \geq 0 \end{aligned} \tag{19}$$

We shall now solve the system of equations 18 by applying Newton's method and obtain a search direction by solving $J(x,s,z,y)[x\ s\ z\ y]^T = -f(x,s,z,y)$ [3]. The step direction obtained is often referred to as *affine scaling direction.*
Fixing $\mu = 0$ and using the derivation as in [2] adding equality constraints we obtain

$$\begin{bmatrix} Q & 0 & A_i^T & -A_e^T \\ 0 & Z & S & 0 \\ A_i & I & 0 & 0 \\ A_e & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{aff} \\ s_{aff} \\ z_{aff} \\ y_{aff} \end{bmatrix} = \begin{bmatrix} -(Qx + q - A_i^T z - A_e^T y \\ -Sz \\ -(A_i x - s - b_i) \\ -(A_e x - b) \end{bmatrix}$$

The term $\mu = \frac{x^T s}{(N+M)}$ can be used as a quality measure for a solution, every step we could look for a solution point with $\mu_{k+1} < \mu_k$ which would make us approach the problem from the interior of the existing region, avoiding the boundaries. We first calculated the affine step by fixing mu, we now want to calculate a more gentle step, as we don't want to take a full Newton's step since it could violate $x, s \geq 0$.
[2] presents an extension to quadratic optimization of the Mehrotra predictor-corrector method for linear programming [4], which combines (sums) a centering step and a corrector step. The centering step is the same Newton's step calculated using the resulting affine step and a centering parameter $\sigma \in [0\ 1]$. Moving along the affine scaling direction is likely to violate the $x, s > 0$ constraints and the corrector step direction corrects for this error by solving for the right hand side $[\ 0\ 0\ 0\ \ -diag(x_{aff})s_{aff}]$. Summing the steps and adding the equality constraints, we obtain a centering-corrector step of:

$$\begin{bmatrix} Q & 0 & A_i^T & -A_e^T \\ 0 & Z & S & 0 \\ A_i & I & 0 & 0 \\ A_e & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_c c \\ s_c c \\ z_c c \\ y_c c \end{bmatrix} = \begin{bmatrix} -(Qx + q - A_i^T z - A_e^T y \\ -\sigma - S_{aff} z_{aff} - Sz \\ -(A_i x - s - b_i) \\ -(A_e x - b) \end{bmatrix}$$

The affine step size α_{aff} and centering parameter σ are calculated as in [2]. Finally, the current solution can be advanced forward by following $X_{k+1} = X_k + \alpha \nabla X$ where alpha is the maximum real $\in [0\ 1]$ that mantains the non-negativity of z and s.

$$\begin{aligned} \nabla X &= \nabla X_{aff} + \nabla X_{cc} \\ \alpha &= min\{\alpha \geq 0 | s + \nabla s \geq 0, z + \nabla z \geq 0\} \end{aligned} \tag{20}$$

The above systems can likely be simplified and made symmetric, which would be very important for efficient decomposition when solving bigger systems. The current implementation solves them fully without any particular treatment. The error is calculated as the maximum between the duality gap, the equality constraint residual $|-A_e + b_e|^2$ and

the inequality residual $||-A_i x - s + b_i||$. The last two converge to 0 almost immediately, while the difference between the primal and dual solution is more carefully reduced to 0. This causes the average number of iterations required to converge with accuracy $1e^{-10}$ to be around 10, which is much higher than the previous penalty methods.

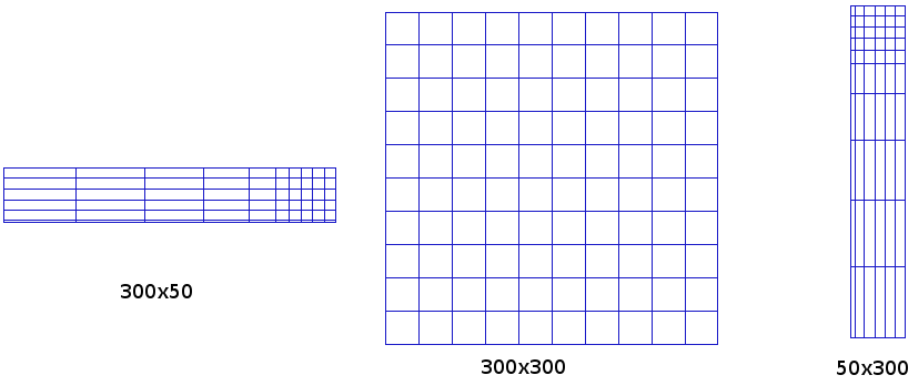

FIGURE 10: Inequality-constrained version solved using a primal-dual path-following algorithm.

```
Target Resolution 300 300
Iteration 1
Error (step norm) 166.75
Iteration 2
Error (step norm) 12.1358
Iteration 3
Error (step norm) 0
----------
Target Resolution 50 300
Iteration 1
Error (step norm) 126.58
Iteration 2
Error (step norm) 61.9297
Iteration 3
Error (step norm) 37.0865
Iteration 4
Error (step norm) 2.3228e-07
Iteration 5
Error (step norm) 2.3228e-08
Iteration 6
Error (step norm) 2.32279e-09
Iteration 7
Error (step norm) 2.3231e-10
Iteration 8
Error (step norm) 2.32108e-11
Iteration 9
Error (step norm) 2.299e-12
Iteration 10
Error (step norm) 1.72285e-13
----------
Target Resolution 300 50
Iteration 1
Error (step norm) 126.58
Iteration 2
Error (step norm) 61.9297
Iteration 3
Error (step norm) 42.9192
Iteration 4
Error (step norm) 2.64036e-06
Iteration 5
Error (step norm) 2.99026e-08
Iteration 6
Error (step norm) 2.99024e-09
Iteration 7
Error (step norm) 2.99017e-10
Iteration 8
Error (step norm) 2.99323e-11
Iteration 9
Error (step norm) 2.93764e-12
Iteration 10
Error (step norm) 2.13587e-13
----------
Average iteration count 7
```

FIGURE 11: Inequality-constrained iterations using a primal-dual path-following algorithm.

Conclusion

All the algorithms implemented converged to a feasible solution in a limited number of iterations. The penalty methods proved to be more unstable, but very well performing. Each step of the iterative penalty method is less costly than the Interior-Point alternative since we are just solving an $(N + M)$ by $(N + M)$ linear system. On the other hand, moving along the Primal-Dual path while avoiding region boundaries provides a very elegant convergence, which also does not "jump" between the feasible and unfeasible region like the exterior penalty method does. As the size of the system does not weight in favor of the unconstrained approach I would choose the Interior-Point method for a final implementation.

The code for the solvers and used to generate the pictures is attached. I have used Eigen's linear algebra routines and all the resulting linear systems are solved using LU decomposition with pivoting as exposed by *Eigen::FullPivLU*.

Bibliography

[1] Daniele Panozzo, Ofir Weber, and Olga Sorkine. Robust image retargeting via axis-aligned deformation. *Comput. Graph. Forum*, 31(2pt1):229–236, May 2012. ISSN 0167-7055. doi: 10.1111/j.1467-8659.2012.03001.x. URL http://dx.doi.org/10.1111/j.1467-8659.2012.03001.x.

[2] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, second edition, 2006.

[3] U.M. Ascher and C. Greif. *A First Course on Numerical Methods*. Computational Science and Engineering. Society for Industrial and Applied Mathematics, 2011. ISBN 9780898719970. URL https://books.google.ca/books?id=eGDMSIqPYdYC.

[4] E.D. Andersen, C. Roos, and T. Terlaky. On implementing a primal-dual interior-point method for conic quadratic optimization. *Mathematical Programming*, 95(2): 249–277, Feb 2003. ISSN 1436-4646. doi: 10.1007/s10107-002-0349-3. URL https://doi.org/10.1007/s10107-002-0349-3.