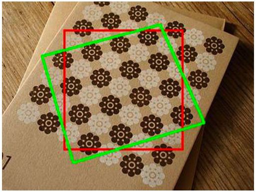
Computational Principles for High-dim Data Analysis

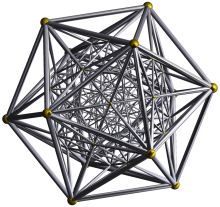
(Lecture Twenty)

Yi Ma

EECS Department, UC Berkeley

November 9, 2021





Structured Nonlinear Low-Dimensional Models

Transform Invariant/Equivariant Low-Rank Texture

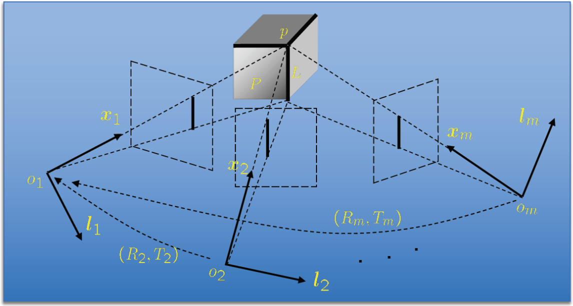
> *"What humans do with the language of mathematics is to describe patterns."*
>
> *– Lynn A. Steen*

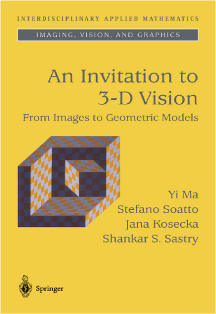
Importance of Mathematical Modeling

If you formulate a problem correctly,
you have more than halfway solved it!

Example: Rank Conditions for Multiple-View Geometry

Geometric and algorithmic foundations for 3D Reconstruction from images.



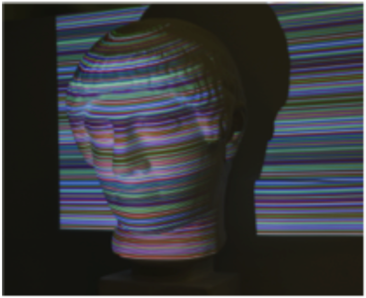


All multi-view incidence conditions among points, lines, planes and symmetric objects are captured by the same rank condition (Ma, 2003):

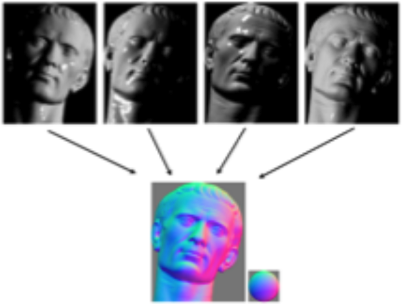
$$\text{rank}(\boldsymbol{M}) = 1 \ (\text{or } 2). \tag{1}$$

Example: Rank Conditions for Photometric Stereo

Geometric and algorithmic foundations for 3D Reconstruction from images.



(b) structured lights



(c) photometric stereo

Images of a (Lambertian) object under different lighting conditions, viewed as columns of a matrix, always satisfy a rank condition (Chapter 14):

$$\mathrm{rank}(\boldsymbol{M}) = 3. \tag{2}$$

Example: Low-Rank Structures in a Single Image

Low-dimensional structures arise at all spatial scales even in an individual image, especially that of man-made objects.

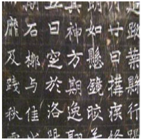
(a) a calibration rig

(b) a carpet

(c) windows

(d) a door

(e) a license plate



(f) characters

(g) a car

(h) a face

Be aware though: only at a rectified view and without any occlusion!

Mathematical Model for Low-Rank Image (Regions)

An image (region) $\boldsymbol{I}_o(x, y)$ can be explicitly factorized as as the combination of r rank-1 functions:

$$\boldsymbol{I}_o(x, y) \doteq \sum_{i=1}^{r} u_i(x) \cdot v_i(y). \tag{3}$$

Symmetry: two low-rank textures *equivalent* if they are scaled and translated versions of each other:

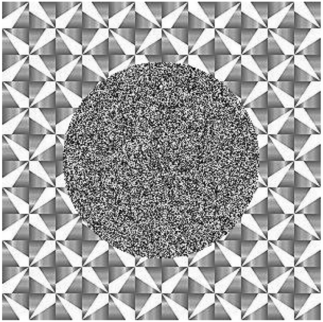
$$\boldsymbol{I}_o(x, y) \sim \alpha \cdot \boldsymbol{I}_o(ax + t_1, by + t_2),$$

for some $\alpha, a, b, c \in \mathbb{R}_+, t_1, t_2 \in \mathbb{R}$.

Image Inpainting as Matrix Completion

Example: Low-rank image inpainting as matrix completion:

$$\min_{\boldsymbol{L}} \ \mathrm{rank}(\boldsymbol{L}) \quad \text{s.t.} \quad \boldsymbol{L}(i,j) = \boldsymbol{I}_o(i,j) \ \ \forall (i,j) \in \Omega. \tag{4}$$



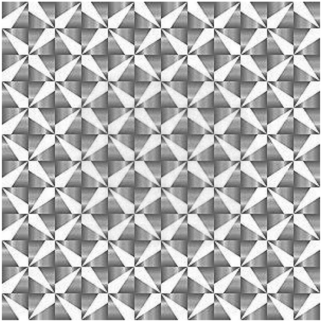
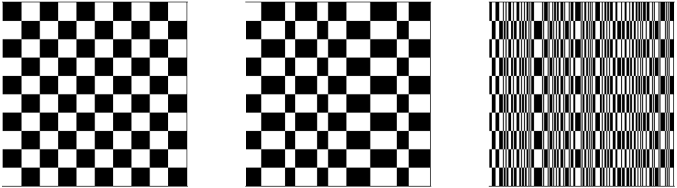


Image Inpainting as Matrix Completion



Is low-rank the only low-dim structure in such images?

$$\boldsymbol{I}_o(x,y) = \sum_{i=1}^{r} u_i(x)v_i(y) = \boldsymbol{U}\boldsymbol{V}^*.$$

$\boldsymbol{U}$ and $\boldsymbol{V}$ might themselves be sparse in some bases
$\boldsymbol{U} = \boldsymbol{B}_1\boldsymbol{X}_1, \boldsymbol{V} = \boldsymbol{B}_2\boldsymbol{X}_2$:

$$\boldsymbol{I}_o = \boldsymbol{B}_1\boldsymbol{X}_1\boldsymbol{X}_2^*\boldsymbol{B}_2^* \doteq \boldsymbol{B}_1\boldsymbol{W}_o\boldsymbol{B}_2^*, \text{ with } \boldsymbol{W}_o = \boldsymbol{X}_1\boldsymbol{X}_2^* \text{ sparse.}$$

Image Inpainting as Matrix Completion

Impose low-rank and sparse structures simultaneously (not RPCA!):

$$\min_{\boldsymbol{L},\boldsymbol{W}} \ \mathrm{rank}(\boldsymbol{L}) + \lambda\|\boldsymbol{W}\|_0 \quad \text{s.t.} \quad \mathcal{P}_\Omega[\boldsymbol{L}] = \mathcal{P}_\Omega[\boldsymbol{I}],\ \boldsymbol{L} = \boldsymbol{B}_1\boldsymbol{W}\boldsymbol{B}_2^*. \tag{5}$$

Relaxation to a corresponding convex program (why?):

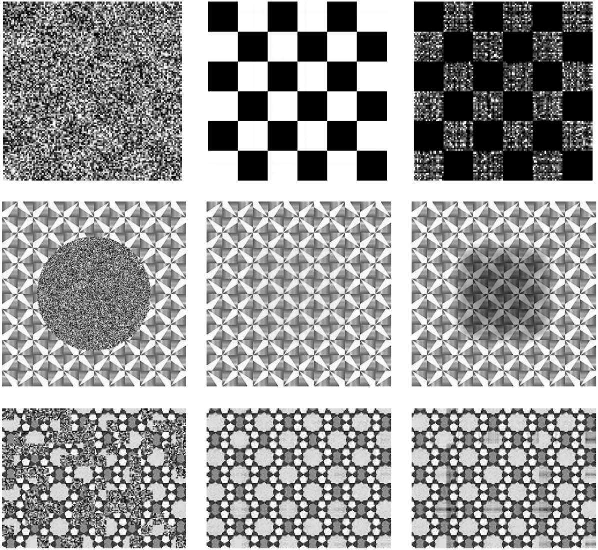
$$\min_{\boldsymbol{W}} \ \|\boldsymbol{W}\|_* + \lambda\|\boldsymbol{W}\|_1 \quad \text{s.t.} \quad \mathcal{P}_\Omega[\boldsymbol{B}_1\boldsymbol{W}\boldsymbol{B}_2^*] = \mathcal{P}_\Omega[\boldsymbol{I}]. \tag{6}$$

Reformulation for a better optimization algorithm:

$$\min_{\boldsymbol{L},\boldsymbol{W}} \ \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{W}\|_1 \quad \text{s.t.} \quad \boldsymbol{L} = \boldsymbol{W},\ \mathcal{P}_\Omega[\boldsymbol{B}_1\boldsymbol{W}\boldsymbol{B}_2^*] = \mathcal{P}_\Omega[\boldsymbol{I}]. \tag{7}$$

Note that the above problem can be solved by ALM + ADMM!

Image Inpainting: Improved Performance



(a) Input image (b) Sparse + Low-rank (c) Low-rank only

Image Recovery

Robustness to corruption $\boldsymbol{I} = \boldsymbol{I}_o + \boldsymbol{E}_o$ for unknown sparse $\boldsymbol{E}_o$:

$$\min_{\boldsymbol{W}} \|\boldsymbol{W}\|_* + \lambda\|\boldsymbol{W}\|_1 + \alpha\|\boldsymbol{E}\|_1 \quad \text{s.t.} \quad \mathcal{P}_\Omega[\boldsymbol{B}_1\boldsymbol{W}\boldsymbol{B}_2^* + \boldsymbol{E}] = \mathcal{P}_\Omega[\boldsymbol{I}]. \quad (8)$$

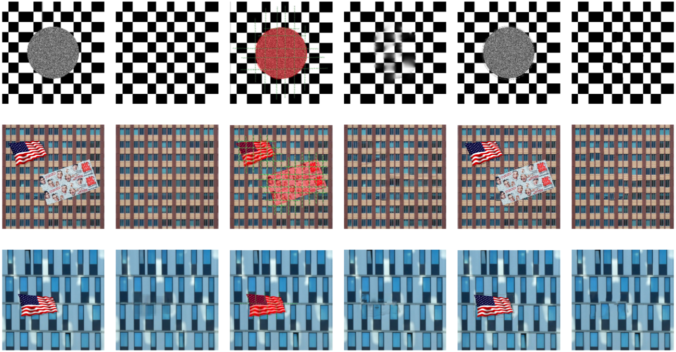


Figure: Left: Low-rank; Middle: Microsoft; Right: Adobe.

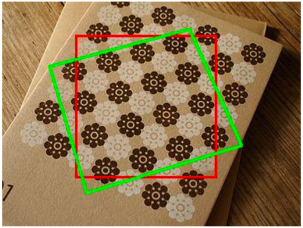
Deformation and Corruption

In practice, the observe image $\boldsymbol{I}$ is typically a transformed version of the (rectified) low-rank image $\boldsymbol{I}_o$: $\boldsymbol{I} \circ \tau(x, y) = \boldsymbol{I}_o(x, y)$ or:

$$\boldsymbol{I}(x, y) = \boldsymbol{I}_o \circ \tau^{-1}(x, y) = \boldsymbol{I}_o\left(\tau^{-1}(x, y)\right), \quad \tau \in \mathbb{G}.$$

(a) Low-rank texture $\boldsymbol{I}_o$

$\xleftarrow{\tau}$



(b) Its image $\boldsymbol{I}$ under a different viewpoint

Note: the observed image might also be corrupted: $\boldsymbol{I} = (\boldsymbol{I}_o + \boldsymbol{E}_o) \circ \tau^{-1}$.

Algorithm Development

Original problem: transformed low-rank and sparse decomposition

$$\min_{\boldsymbol{L},\boldsymbol{E},\tau} \operatorname{rank}(\boldsymbol{L}) + \gamma\|\boldsymbol{E}\|_0 \quad \text{subject to} \quad \boldsymbol{I}\circ\tau = \boldsymbol{L}+\boldsymbol{E}. \tag{9}$$

Relaxation: use convex surrogates for rank and sparsity

$$\min_{\boldsymbol{L},\boldsymbol{E},\tau} \underbrace{\|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{E}\|_1}_{\text{convex}} \quad \text{subject to} \quad \underbrace{\boldsymbol{I}\circ\tau = \boldsymbol{L}+\boldsymbol{E}}_{\text{still nonlinear!}}. \tag{10}$$

Linearization to deal with nonlinearity: $\boldsymbol{I}\circ\tau + \nabla\boldsymbol{I}\cdot d\tau \approx \boldsymbol{L}+\boldsymbol{E}$ and

$$\min_{\boldsymbol{L},\boldsymbol{E},d\tau} \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{E}\|_1 \quad \text{subject to} \quad \boldsymbol{I}\circ\tau + \nabla\boldsymbol{I}\cdot d\tau = \boldsymbol{L}+\boldsymbol{E}. \tag{11}$$

Note: this is exactly a Compressive PCP problem (in Chapter 5)!

The TILT Algorithm

INPUT: Input image $\boldsymbol{I} \in \mathbb{R}^{w \times h}$, initial transformation $\tau \in \mathbb{G}$ (affine or projective), and a weight $\lambda > 0$.

WHILE not converged **DO**

Step 1: Normalization and compute Jacobian:

$$\boldsymbol{I} \circ \tau \leftarrow \frac{\boldsymbol{I} \circ \tau}{\|\boldsymbol{I} \circ \tau\|_F}; \quad \nabla \boldsymbol{I} \leftarrow \frac{\partial}{\partial \zeta} \left(\frac{\mathrm{vec}(\boldsymbol{I} \circ \zeta)}{\|\mathrm{vec}(\boldsymbol{I} \circ \zeta)\|_2} \right)\Big|_{\zeta = \tau}; \tag{12}$$

Step 2 (inner loop): Solve the linearized CPCP problem:

$$\begin{array}{rcl} (\boldsymbol{L}_\star, \boldsymbol{E}_\star, d\tau_\star) & \leftarrow & \arg\min_{\boldsymbol{L}, \boldsymbol{E}, d\tau} \quad \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{E}\|_1 \\ & & \text{subject to} \quad \boldsymbol{I} \circ \tau + \nabla \boldsymbol{I} \cdot d\tau = \boldsymbol{L} + \boldsymbol{E}; \end{array} \tag{13}$$

Step 3: Update the transformation: $\tau \leftarrow \tau + d\tau_\star$;

END WHILE

OUTPUT: Converged solution $\boldsymbol{L}_\star$, $\boldsymbol{E}_\star$, $\tau_\star$ to problem (10).

Inner Loop of TILT

Apply ALM + ADMM to solve the inner loop CPCP problem (13):

INPUT: The current (deformed and normalized) image $\boldsymbol{I} \circ \tau \in \mathbb{R}^{m \times n}$ and its Jacobian $\nabla \boldsymbol{I}$ against current deformation τ (from the outer loop), and $\lambda > 0$.
Initialization: $k = 0, \boldsymbol{Y}_0 = 0, \boldsymbol{E}_0 = 0, d\tau_0 = 0, \mu_0 > 0, \rho > 1$;
WHILE not converged **DO**

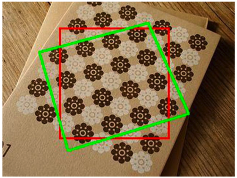
$$(\boldsymbol{U}_k, \boldsymbol{\Sigma}_k, \boldsymbol{V}_k) = \mathrm{SVD}\big(\boldsymbol{I} \circ \tau + \nabla \boldsymbol{I} \cdot d\tau_k - \boldsymbol{E}_k + \mu_k^{-1} \boldsymbol{Y}_k\big);$$
$$\boldsymbol{L}_{k+1} = \boldsymbol{U}_k \mathsf{soft}(\boldsymbol{\Sigma}_k, \mu_k^{-1}) \boldsymbol{V}_k^*;$$
$$\boldsymbol{E}_{k+1} = \mathsf{soft}(\boldsymbol{I} \circ \tau + \nabla \boldsymbol{I} \cdot d\tau_k - \boldsymbol{L}_{k+1} + \mu_k^{-1} \boldsymbol{Y}_k, \lambda \mu_k^{-1});$$
$$d\tau_{k+1} = (\nabla \boldsymbol{I})^\dagger (-\boldsymbol{I} \circ \tau + \boldsymbol{L}_{k+1} + \boldsymbol{E}_{k+1} - \mu_k^{-1} \boldsymbol{Y}_k);$$
$$\boldsymbol{Y}_{k+1} = \boldsymbol{Y}_k + \mu_k (\boldsymbol{I} \circ \tau + \nabla \boldsymbol{I} \cdot d\tau_{k+1} - \boldsymbol{L}_{k+1} - \boldsymbol{E}_{k+1});$$
$$\mu_{k+1} = \rho \mu_k;$$

END WHILE
OUTPUT: Converged solution $(\boldsymbol{L}_\star,\ \boldsymbol{E}_\star,\ d\tau_\star)$ to problem (11).

Rectifying Planar Low-Rank Textures

(a) Low-rank texture $\boldsymbol{I}_o$

$\xleftarrow{\tau}$



(b) Its image $\boldsymbol{I}$ under a different viewpoint

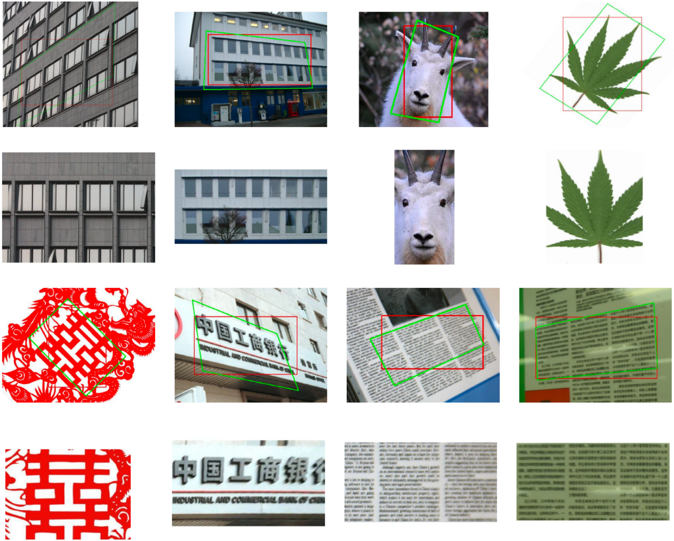
$\boldsymbol{I}_o$ is on a planar surface and $\boldsymbol{I}$ is seen from a different view:

$$\boldsymbol{I}(x,y) = \boldsymbol{I}_o \circ \tau^{-1}(x,y) = \boldsymbol{I}_o\left(\tau^{-1}(x,y)\right)$$

where τ is any planar homography $\boldsymbol{H} \in \mathrm{GL}(3)$:

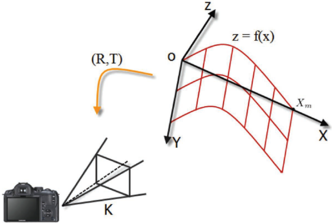
$$\tau(x,y) = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{14}$$

Empirical Observations: large range of attraction



Rectifying Generalized Cylinder Surfaces

In practice, very often the low-rank texture $\boldsymbol{I}_o$ does not lie on a planar surface, instead say on a cylindrical surface:



The overall deformation consists of composition of three mappings:

- from a flat 2D plane to the 3D cylindrical surface (shape);
- from the surface to the image plane (camera pose and projection);
- from the image plane to the pixel coordinates (intrinsic calibration).

Rectifying Generalized Cylinder Surfaces

Then $\boldsymbol{I}_o = \boldsymbol{I} \circ g$ where the transformation g from the texture coordinates $\boldsymbol{I}_o(x, y)$ to the image coordinates $\boldsymbol{I}(u, v)$ is a composition of the three mappings specified above:

$$g : (x, y) \mapsto (X_c, Y_c, Z_c) \mapsto (x_n, y_n) \mapsto (u, v). \tag{15}$$

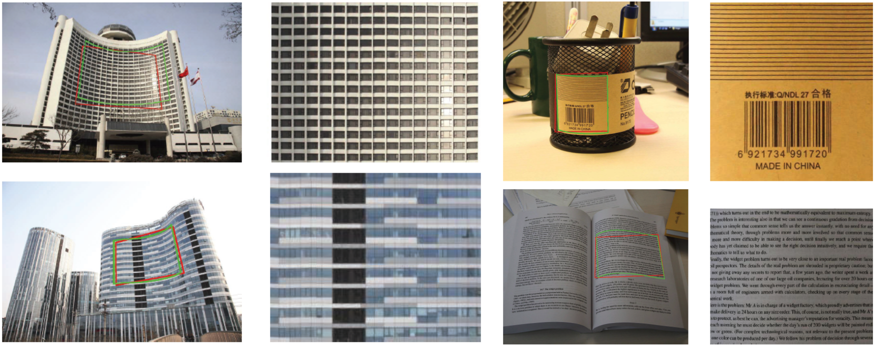
Hence we solve the following nonlinear optimization problem:

$$\min_{\boldsymbol{L},\boldsymbol{E},\boldsymbol{c},\boldsymbol{R},\boldsymbol{T}} \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{E}\|_1 \quad \text{s.t.} \quad \boldsymbol{I} \circ g = \boldsymbol{L} + \boldsymbol{E}. \tag{16}$$

As in TILT, this can be solved iteratively from its linearized version:

$$\min_{\boldsymbol{L},\boldsymbol{E},dg} \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{E}\|_1 \quad \text{s.t.} \quad \boldsymbol{I} \circ g + \nabla \boldsymbol{I}_g \cdot dg = \boldsymbol{L} + \boldsymbol{E}, \tag{17}$$

Empirical Results



Camera Calibration

Three most important factors in robotics and augmented reality:

calibration, calibration, calibration!

Figure 15.10 Left: typical image of a fisheye camera. **Right:** image of a perspective camera.

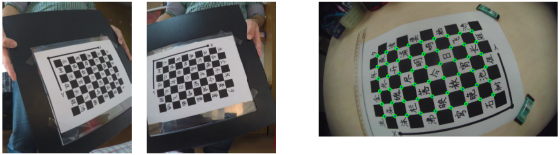


Figure 15.11 Left two: Images of a typical calibration rig. **Right:** Corners need to be marked (or detected) for conventional calibration methods.

Camera Calibration: from a calibration rig to the image

Extrinsic parameters τ_i (camera pose – different for each image i):

rotation $\boldsymbol{R} \in SO(3)$ and translation $\boldsymbol{T} \in \mathbb{R}^3$.

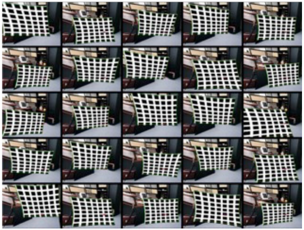
Intrinsic parameters τ_0 (focal length etc. – common for all images):

$$\boldsymbol{K} \doteq \begin{bmatrix} f_x & \theta & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \quad \in \mathbb{R}^{3\times 3}. \tag{18}$$

Lens distortion τ_0 (radial distortion etc. – common for all images):

$$\begin{cases} r & \doteq & \sqrt{x_n^2 + y_n^2}, \\ f(r) & \doteq & 1 + k_c(1)r^2 + k_c(2)r^4 + k_c(5)r^6, \\ \boldsymbol{p}_d & = & \begin{bmatrix} f(r)x_n + 2k_c(3)x_n y_n + k_c(4)(r^2 + 2x_n^2) \\ f(r)x_n + 2k_c(4)x_n y_n + k_c(3)(r^2 + 2y_n^2) \end{bmatrix}. \end{cases} \tag{19}$$

Camera Calibration



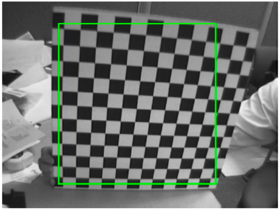
Multiple images $\boldsymbol{I}_i$ of the calibration rig $\boldsymbol{I}_o$ at different views τ_i are related by:

$$\boldsymbol{I}_i \circ (\tau_0 \circ \tau_i) = \boldsymbol{I}_o + \boldsymbol{E}_i, \quad i = 1, 2, \ldots, N. \tag{20}$$

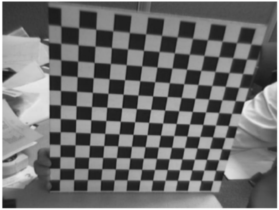
$$\min \sum_{i=1}^{N} \|\boldsymbol{L}_i\|_* + \|\boldsymbol{E}_i\|_1, \quad \text{s.t.} \quad \boldsymbol{I}_i \circ (\tau_0 \circ \tau_i) = \boldsymbol{L}_i + \boldsymbol{E}_i, \ \boldsymbol{L}_i = \boldsymbol{L}_j. \tag{21}$$

Or $\quad \min \ \|\boldsymbol{L}_c\|_* + \|\boldsymbol{L}_r\|_* + \lambda\|\boldsymbol{E}\|_1, \quad \text{s.t.} \quad \boldsymbol{I}_i \circ (\tau_0 \circ \tau_i) = \boldsymbol{L}_i + \boldsymbol{E}_i.$ (22)

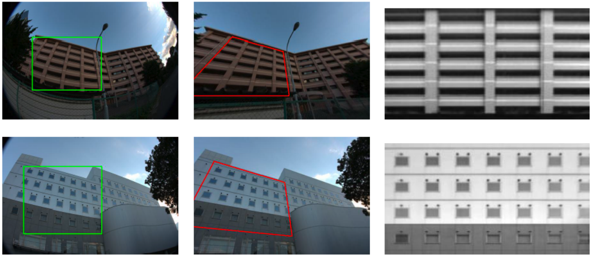
Calibration Results



(a) Input image with an initial window



(b) Lens distortion removed



Summary

A good idea deserves a good implementation.

The better the idea, the better should be the implementation.

Assignments

- Reading: Chapter 15.
- Programming Homework #4.