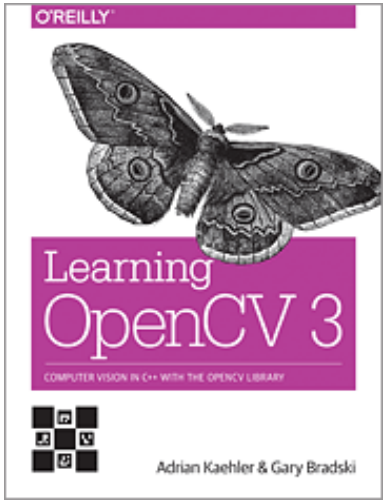
Lab 2 – Image blending with OpenCV

02.02.2017

Litt om datatypene i OpenCV

Ressurser

- Dokumentasjonen
  - http://docs.opencv.org/3.2.0/
- Tutorials
  - http://docs.opencv.org/3.2.0/d9/df8/tutorial_root.html
- Learning OpenCV 3, 1st Edition
  - Gary Bradski, Adrian Kaehler



Laplace blending

Steg 1: Les og konverter bildene

- Åpne `lab_2_2_image_blending_unfinished`
- Les inn to bilder
  - `cv::imread(...)`
- Konverter bildene til `CV_32F`
  - Skaler bildene slik at de får verdier i intervallet [0, 1]

Steg 2: Lag maske med blandingsvekter

- Lag maske med rampe
  - Samme størrelse som inputbildene
  - Første halvdelen av kolonnene svart (0.0)
  - Siste halvedelen av kolonnene hvit (1.0)
  - Hvordan lage rampe?

Steg 3: Enkel lineær blanding av bilder

- Implementer enkel blanding av to bilder vektet med masken

```
// TODO: Blend the two images according to the weights.
cv::Mat linearBlending(const cv::Mat& img_1, const cv::Mat& img_2, const cv::Mat& weights)
{
  return cv::Mat();
}
```

- Bruk funksjonen og vis frem resultatet
  - Prøv med forskjellige masker
    - Andre sømmer, sirkler, større rampe

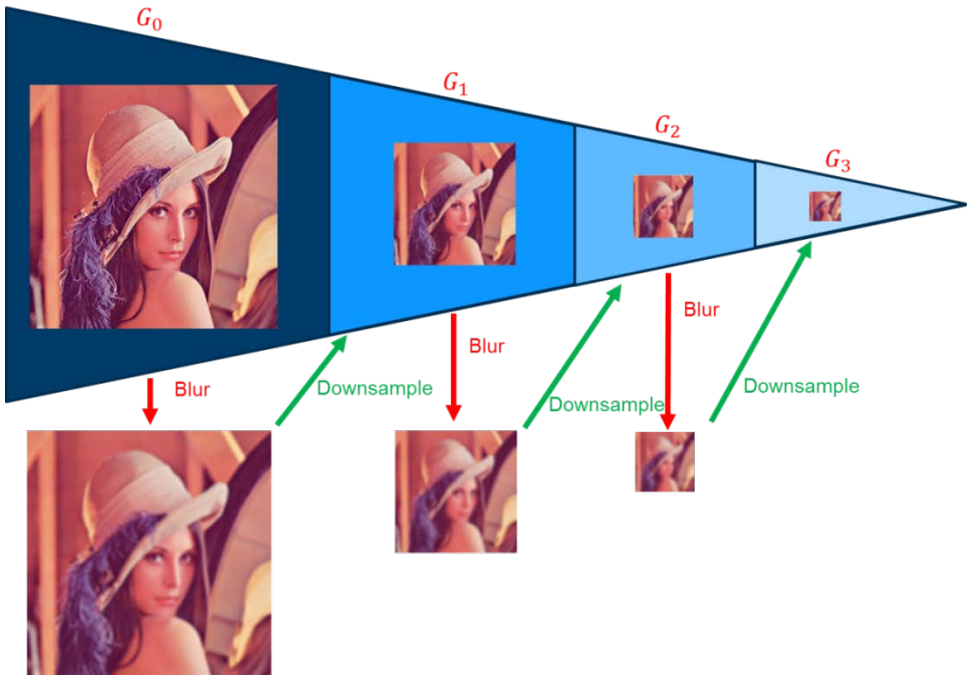
Steg 4: Laplaceblanding

- Lag en gaussisk pyramide

```
std::vector<cv::Mat, std::allocator<cv::Mat>> constructGaussianPyramid(const cv::Mat& img)
{
  std::vector<cv::Mat> pyr = {img.clone()};

  while(pyr.back().cols > 16)
  {
    // TODO: Add the next level in the pyramid.
  }

  return pyr;
}
```

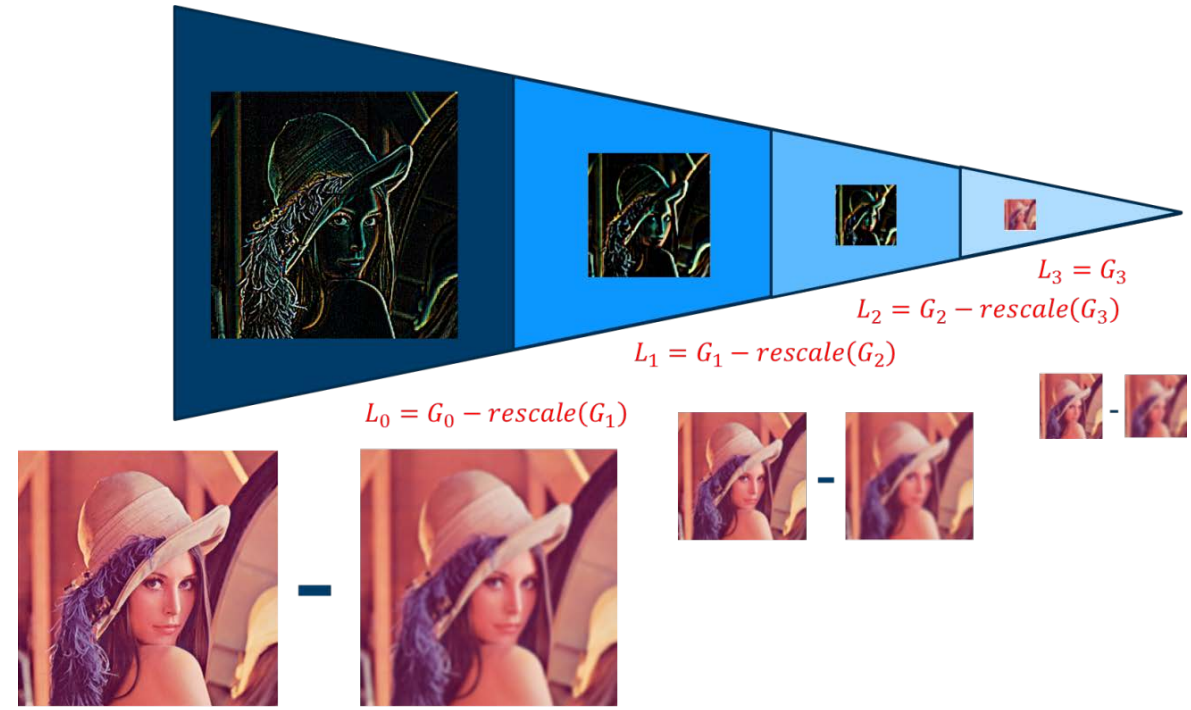


- `cv::pyrDown()`

Steg 4: Laplaceblanding

- Lag en laplacepyramide

```
std::vector<cv::Mat> constructLaplacianPyramid(const cv::Mat& img)
{
  // TODO: Use the gaussian pyramid to construct a laplacian pyramid.
  std::vector<cv::Mat> pyr;
  return pyr;
}
```

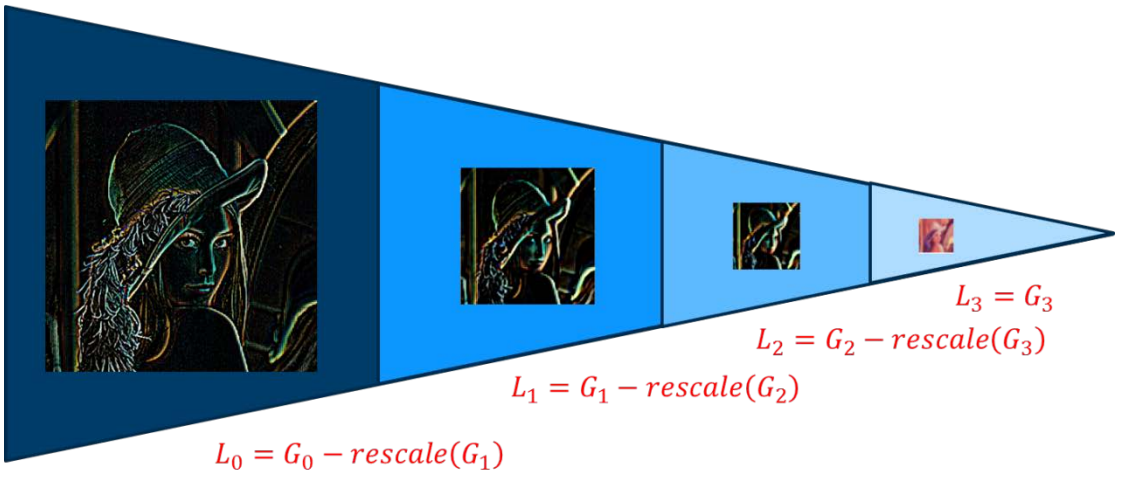


- `cv::pyrUp()`

Steg 4: Laplaceblanding

- Rekonstruer et bilde ved å kollapse en laplacepyramide

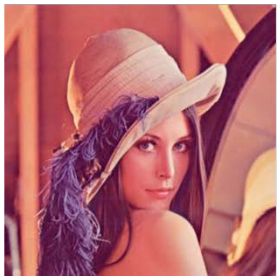
```
cv::Mat collapsePyramid(const std::vector<cv::Mat>& pyr)
{
  // TODO: Collapse the pyramid.
  return cv::Mat();
}
```



- `cv::pyrUp()`

Collapsing the Laplacian pyramid:

$$rescale(rescale(rescale(L_3) + L_2) + L_1) + L_0 =$$



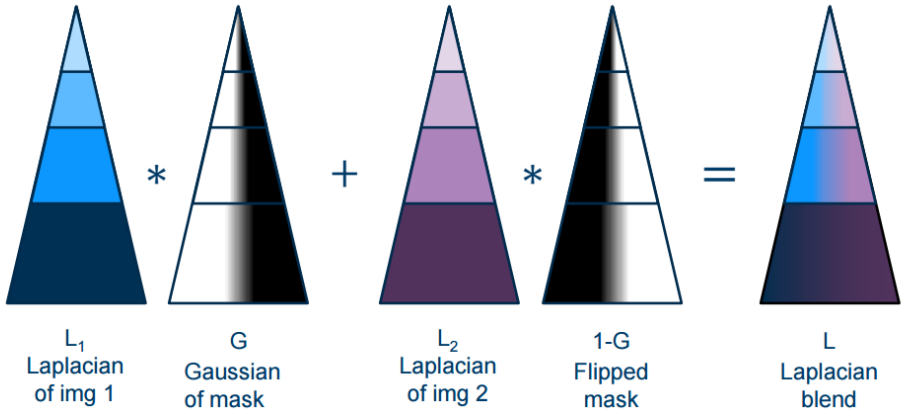
Steg 4: Laplaceblanding

- Gjør selve laplaceblendingen

```
cv::Mat laplaceBlending(const cv::Mat& img_1, const cv::Mat& img_2, const cv::Mat& mask)
{
  std::vector<cv::Mat> mask_pyr = constructGaussianPyramid(mask);
  std::vector<cv::Mat> img_1_pyr = constructLaplacianPyramid(img_1);
  std::vector<cv::Mat> img_2_pyr = constructLaplacianPyramid(img_2);

  std::vector<cv::Mat> blend_pyr(img_1_pyr.size());
  for (int i = 0; i < static_cast<int>(img_1_pyr.size()); i++)
  {
    // TODO: Blend the images using linear blending on each pyramid level.
  }

  return collapsePyramid(blend_pyr);
}
```



Steg 4: Laplaceblanding

- Hvordan er resultatet i forhold til enkel blanding?
  - Prøv forskjellige rampestørrelser

Steg 5: Moroplukk

- Prøv andre bilder
  - Ta bilder med kameraet
  - Finn bilder på nett
  - Samregistrer bildene

```
cv::Point2f pts_1[] = {{321, 200}, {647, 200}, {476, 509}};
cv::Point2f pts_2[] = {{441, 726}, {780, 711}, {615, 1142}};
cv::Mat trans_mat = cv::getAffineTransform(pts_2, pts_1);
cv::warpAffine(img_2, img_2, trans_mat, img_1.size());
```

- Andre masker
  - Last ned GIMP for å tegne finere masker

Steg 5: Dypdykk

- Implementer pyramiden selv
  - Ikke bruk `cv::pyrDown()` eller `cv::pyrUp()`
- Ta en titt på `cv::seamlessClone()`
- Prøv å implementere warpingen selv