Low Level Vision

Doug Blank
Androids: Design and Practice
Bryn Mawr College
Fall 2011

Low Level Vision

- Written in C++ or C
- Upennalizers/Player/Vision/Lib/ImageProc/
- Pointer arithmetic
- Images as 1D or 2D arrays
- Often uses matrix[m][n] rather than x,y
- All matrix indexing is done in 1D
  - matrix[m + n * COLS]

color_count.cpp

```
int *color_count(uint8_t *x, int n) {
  static std::vector<int> count(nColor);
  for (int i = 0; i < nColor; i++) {
    count[i] = 0;
  }
  for (int i = 0; i < n; i++) {
    int label = x[i];
    count[label]++;
  }
  return &count[0];
}
```

Lua Wrapper

```
static int lua_color_count(lua_State *L) {
  uint8_t *label = (uint8_t *) lua_touserdata(L, 1);
  if ((label == NULL) || !lua_islightuserdata(L, 1)) {
    return luaL_error(L, "Input LABEL not light user data");
  }
  int n = luaL_checkint(L, 2);
  int *count = color_count(label, n);
  lua_createtable(L, nColor, 0);
  for (int i = 0; i < nColor; i++) {
    lua_pushinteger(L, count[i]);
    lua_rawseti(L, -2, i);
  }
  return 1;
}
```

Called from Lua

```
-- determine total number of pixels of each color/label
colorCount = ImageProc.color_count(labelA.data,
                                   labelA.npixel);
```

2D Matrix with a 1D Index

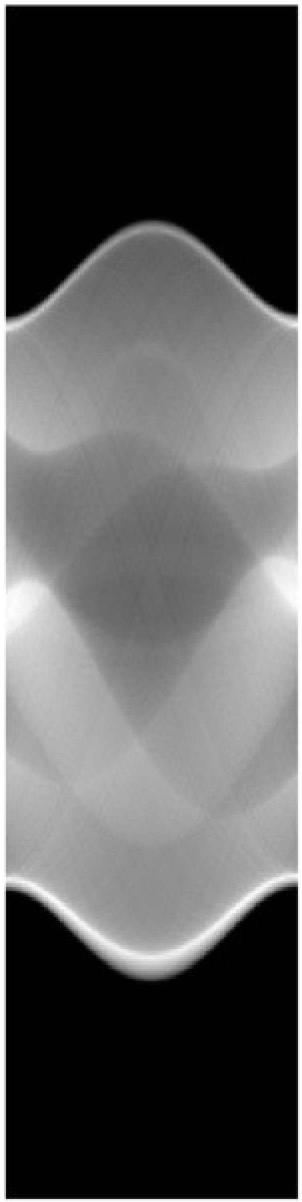
```
uint8_t *block_bitor(uint8_t *x, int mx, int nx, int msub, int nsub) {
  static std::vector<uint8_t> block;
  int my = 1+(mx-1)/msub;
  int ny = 1+(nx-1)/nsub;
  block.resize(my*ny);
  for (int iy = 0; iy < my*ny; iy++) {
    block[iy] = 0;
  }
  for (int jx = 0; jx < nx; jx++) {
    int jy = jx/nsub;
    for (int ix = 0; ix < mx; ix++) {
      int iy = ix/msub;
      block[iy+jy*my] |= *x++;
    }
  }
  return &block[0];
}
```

Radon Transform

"The Radon transform is widely applicable to tomography, the creation of an image from the scattering data associated to cross-sectional scans of an object. If a function f represents an unknown density, then the Radon transform represents the scattering data obtained as the output of a tomographic scan. Hence the inverse of the Radon transform can be used to reconstruct the original density from the scattering data, and thus it forms the mathematical underpinning for tomographic reconstruction, also known as image reconstruction."

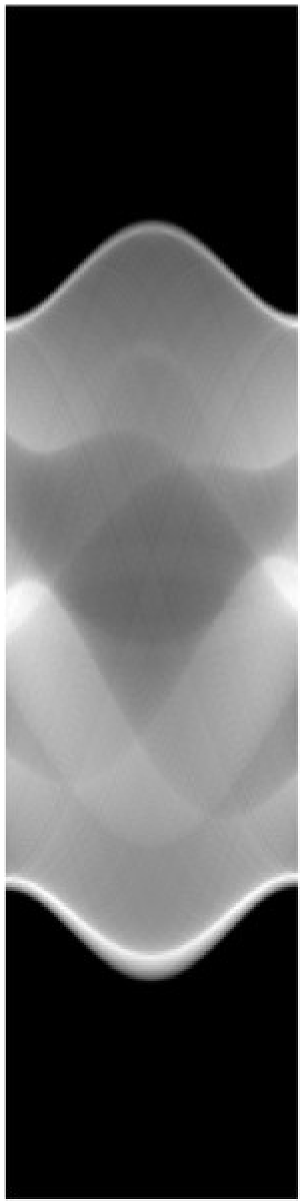
http://en.wikipedia.org/wiki/Radon_transform

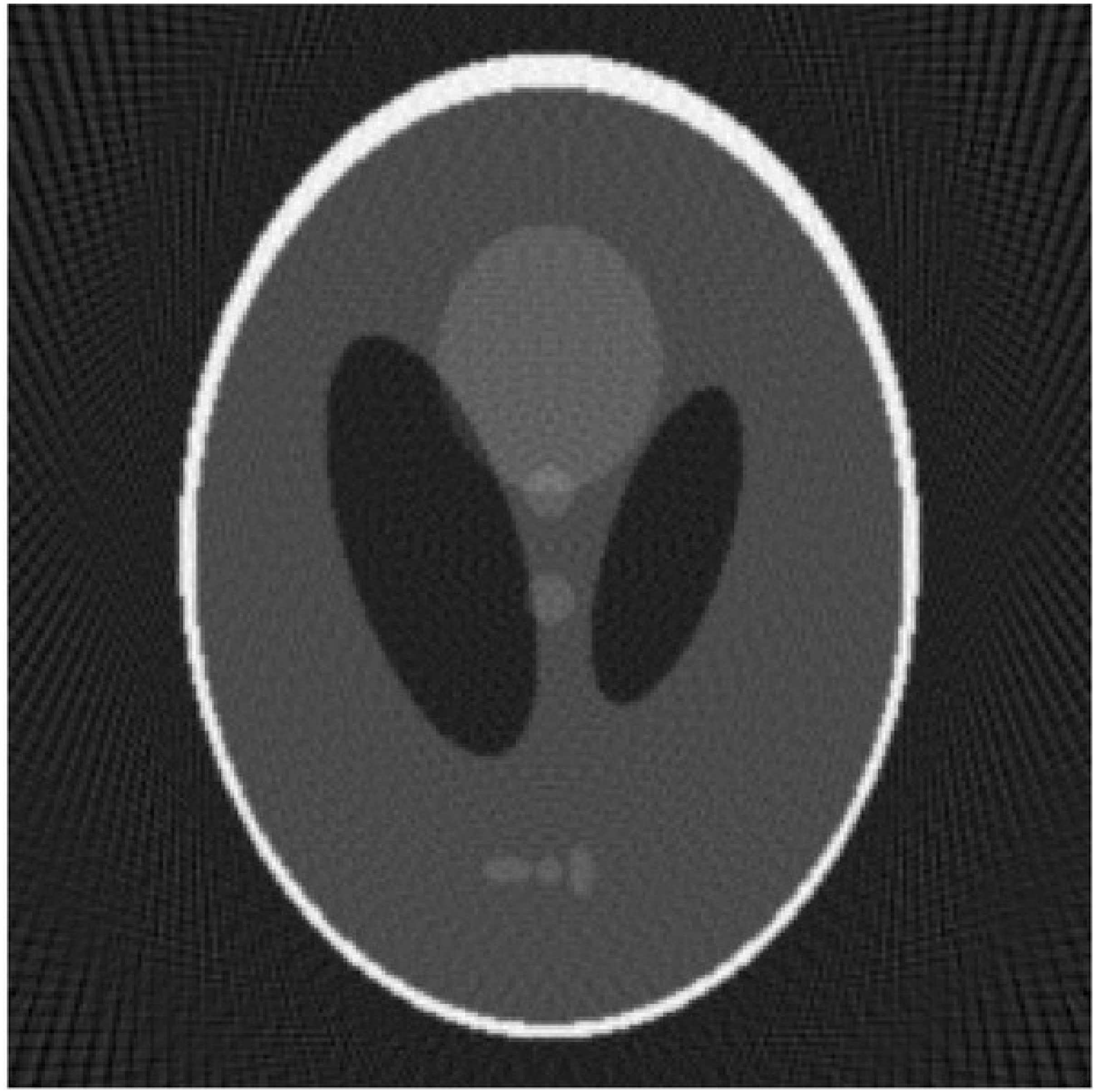
Inverse Radon Transform



?

Inverse Radon Transform





Detecting Lines using the Radon Transform

Detecting Lines Using the Radon Transform

The Radon transform is closely related to a common computer vision operation known as th
the Hough transform used to detect straight lines. The steps are

1. Compute a binary edge image using the `edge` function.

```
I = fitsread('solarspectra.fts');
I = mat2gray(I);
BW = edge(I);
imshow(I), figure, imshow(BW)
```

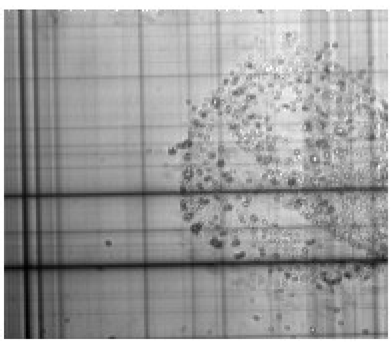
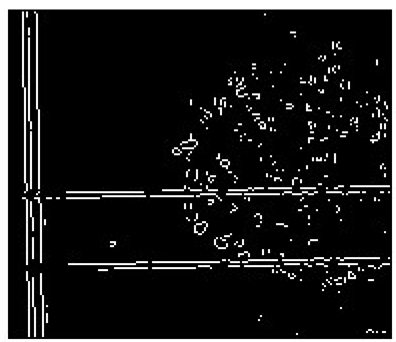


Image Courtesy of Ann Walker

Original Image



Edge Image

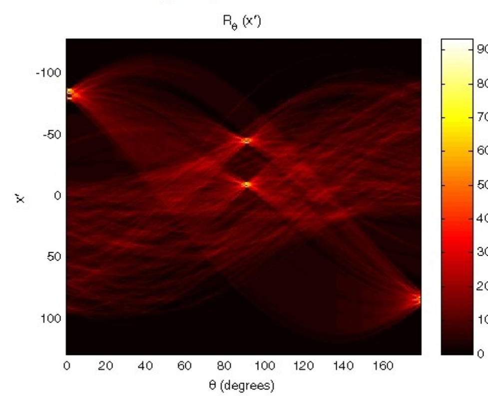
2. Compute the Radon transform of the edge image.

```
theta = 0:179;
[R,xp] = radon(BW,theta);
figure, imagesc(theta, xp, R); colormap(hot);
xlabel('\theta (degrees)'); ylabel('x\prime');
title('R_{\theta} (x\prime)');
colorbar
```

Radon Transform of an Edge Image

R (x')

Radon Transform of an Edge Image

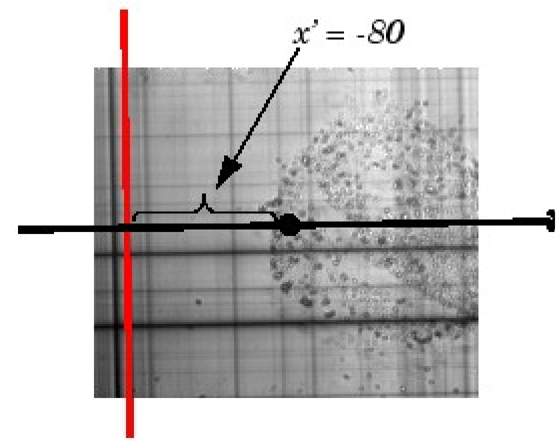


3. Find the locations of strong peaks in the Radon transform matrix. The locations of the

In the following figure, the strongest peaks in R correspond to $\theta = 1^\circ$ and $x' = -80$. The li
superimposed in red on the original image. The Radon transform geometry is shown in bla

peaks at $\theta = 1^\circ$ in the transform. Also, the lines perpendicular to this line appear as peaks at $\theta =$ ...

Radon Transform Geometry and the Strong



Back to Top

See also Hough transform