Classificação de textura:

Trabalhos do meus orientandos:

[T19] Mauricio Aandrés Barrera Acuña, "Rotation-Invariant Texture Classification Based on Graylevel Co-occurrence Matrices", dissertação de mestrado, 2013.

[Cn27] R.H. Ito, H.Y. Kim, W. J. Salcedo, "Classificação de Texturas Invariante a Rotação Usando Matriz de Co-ocorrência," *8th International Information and Telecommunication Technologies Symposium*, 2009.

GLCM (Graylevel co-occurrence matrix):

1) Cacula GLCM:

Let I be a grayscale image with $u \times v$ pixels and L gray-levels. The gray-level co-occurrence matrix $P_{(\Delta x,\Delta y)}$ is a $L \times L$ matrix parameterized by an offset $(\Delta x, \Delta y)$:

$$P_{(\Delta x,\Delta y)}(i,j) = \sum_{x=0}^{u-1}\sum_{y=0}^{v-1}\begin{cases}1, & if\ I(x,y) = i \text{ and } I(x+\Delta x, y+\Delta y) = j \\ 0, & otherwise\end{cases} \quad (1)$$

To avoid accessing pixels outside of the image domain, it is necessary either to use some kind of border interpolation or to restrict the range of (x, y).

2) Normaliza GLCM:

GLCM is usually normalized by dividing each element of the matrix by the sum of all elements. The normalized GLCM is denoted as $p(i,j)$.

3) Extrai features do GLCM normalizado (de acordo com [Acuña2013]):

$$f_{hom} = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1}\frac{1}{1+(1-j)^2}p(i,j) \quad (2)$$

$$f_{con} = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1}p(i,j)\cdot(i-j)^2 \quad (3)$$

$$f_{ent} = -\sum_{i=0}^{L-1}\sum_{j=0}^{L-1}p(i,j)\cdot \ln p(i,j) \quad (4)$$

$$f_{corr} = \sum_{i=0}^{L-1}\sum_{j=1}^{L-1}p(i,j)\cdot\frac{(i-\mu_x)(j-\mu_y)}{\sigma_x\sigma_y} \quad (5)$$

$$f_{enar} = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1}p^2(i,j) \quad (6)$$

$$f_{var} = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1}(i-\mu)^2\cdot p(i,j) \quad (7)$$

Nota: [Ito2009] apresenta definições diferentes para as seguintes características:

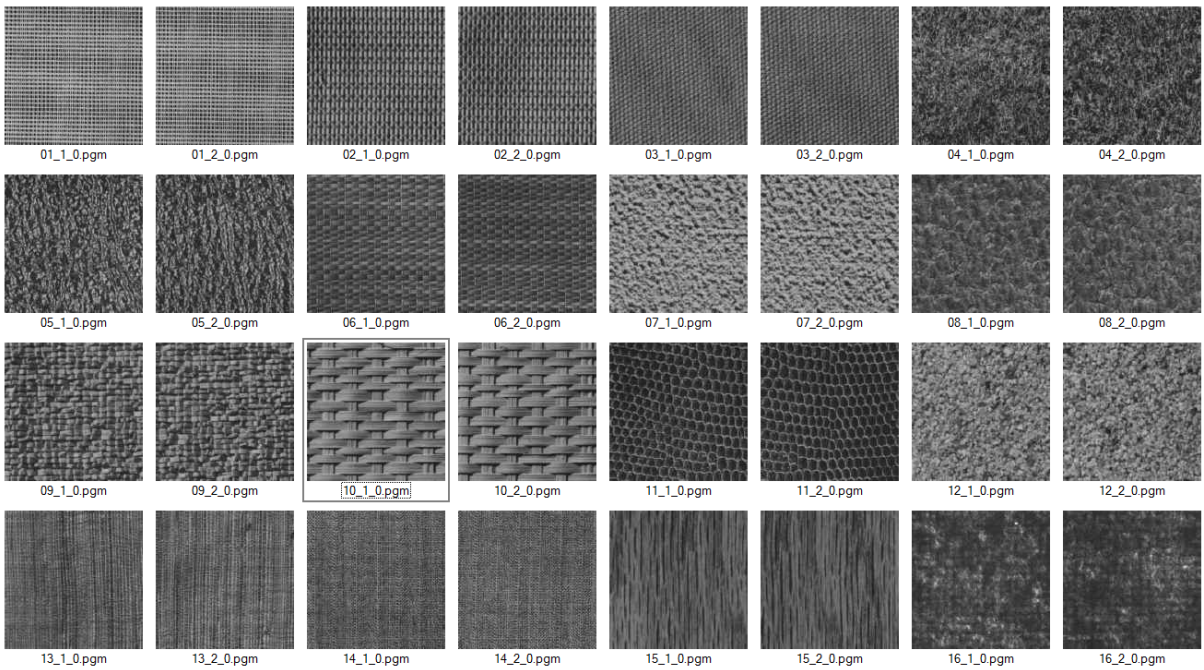
1) $f_{con} = \sum_{n=0}^{N_g-1} n^2 \left\{\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}p(i,j)\right\}_{|i-j|=n}$. Os resultados numéricos são iguais.

2) $f_{cor} = \frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}(ij)p(i,j) - \mu_x\mu_y}{\sigma_x\sigma_y}$. Os resultados numéricos são diferentes.

```
//glcm.cpp - pos2013
#include <cekeikon.h>

Mat_<int> calculaGLCM(Mat_<GRY> ent)
{ Mat_<int> MC(256,256,0);
  Vec2i delta(1,1);
  for (int l=0; l<ent.rows-1; l++)
    for (int c=0; c<ent.cols-1; c++)
      MC(ent(l,c),ent(l+delta[0],c+delta[1]))++;
  return MC;
}

Mat_<FLT> normalizaGLCM(Mat_<int> MC)
{ Mat_<FLT> mc(256,256,0.0);
  double soma=0.0;
  for (unsigned i=0; i<MC.total(); i++)
    soma += MC(i);
  for (unsigned i=0; i<MC.total(); i++)
    mc(i) = MC(i) / soma;
  return mc;
}

double fener(Mat_<FLT> mc)
{ double f=0.0;
  for (int l=0; l<mc.rows; l++)
    for (int c=0; c<mc.cols; c++)
      f += elev2(mc(l,c));
  return f;
}

double fent(Mat_<FLT> mc)
{ double f=0.0;
  for (int l=0; l<mc.rows; l++)
    for (int c=0; c<mc.cols; c++)
      if (mc(l,c)>0.0)
        f += mc(l,c) * log(mc(l,c));
  return -f;
}

double fcon(Mat_<FLT> mc)
{ double f=0.0;
  for (int l=0; l<mc.rows; l++)
    for (int c=0; c<mc.cols; c++)
      f += mc(l,c) * elev2(l-c);
  return f;
}

double mi_l(Mat_<FLT> mc)
{ double mi=0.0;
  for (int l=0; l<mc.rows; l++)
    for (int c=0; c<mc.cols; c++)
      mi += l * mc(l,c);
  return mi;
}

double mi_c(Mat_<FLT> mc)
{ double mi=0.0;
  for (int l=0; l<mc.rows; l++)
    for (int c=0; c<mc.cols; c++)
      mi += c * mc(l,c);
  return mi;
}
```

```
double var_l(Mat_<FLT> mc)
{ double mi=mi_l(mc);
  double f=0.0;
  for (int l=0; l<mc.rows; l++)
    for (int c=0; c<mc.cols; c++)
      f += elev2(l-mi) * mc(l,c);
  return f;
}

double var_c(Mat_<FLT> mc)
{ double mi=mi_c(mc);
  double f=0.0;
  for (int l=0; l<mc.rows; l++)
    for (int c=0; c<mc.cols; c++)
      f += elev2(c-mi) * mc(l,c);
  return f;
}

double fvar(Mat_<FLT> mc)
{ return var_l(mc); }

double fcorr(Mat_<FLT> mc)
{ double mil=mi_l(mc);
  double mic=mi_c(mc);
  double dl=sqrt(var_l(mc));
  double dc=sqrt(var_c(mc));
  double f=0.0;
  for (int l=0; l<mc.rows; l++)
    for (int c=0; c<mc.cols; c++)
      f += (mc(l,c)*(l-mil)*(c-mic))/(dl*dc);
  return f;
}

double fhom(Mat_<FLT> mc)
{ double f=0.0;
  for (int l=0; l<mc.rows; l++)
    for (int c=0; c<mc.cols; c++)
      f += mc(l,c) / (1 + elev2(l-c));
  return f;
}

int main(int argc, char** argv)
{ if (argc!=2)
    erro("Erro: GLCM ent*.pgm");

  vector<string> vs;
  vsWildCard(argv[1],vs);
  printf("%11s%11s%11s%11s%11s%11s%11s\n",
         "nomearq","fener","fent","fcon","fvar","fcorr","fhom");
  for (unsigned i=0; i<vs.size(); i++) {
    Mat_<GRY> ent; le(ent,vs[i]);
    Mat_<int> MC=calculaGLCM(ent);
    Mat_<FLT> mc=normalizaGLCM(MC);

    Mat_<FLT> f(1,6);
    f(0)=fener(mc);
    f(1)=fent(mc);
    f(2)=fcon(mc);
    f(3)=fvar(mc);
    f(4)=fcorr(mc);
    f(5)=fhom(mc);
    printf("%11s",vs[i].c_str());
    for (int i=0; i<f.cols; i++)
      printf("% 11.2e",f(i));
    printf("\n");
  }
}
```

Teste:



nomearq	fener	fent	fcon	fvar	fcorr	fhom
01_1_0.pgm	1.11e-004	9.35e+000	3.01e+003	1.80e+003	1.63e-001	2.57e-002
01_2_0.pgm	1.16e-004	9.31e+000	2.88e+003	1.69e+003	1.47e-001	2.64e-002
02_1_0.pgm	1.29e-004	9.17e+000	1.51e+003	1.50e+003	4.98e-001	3.15e-002
02_2_0.pgm	1.28e-004	9.17e+000	1.45e+003	1.40e+003	4.83e-001	3.04e-002
03_1_0.pgm	1.78e-004	8.78e+000	8.91e+002	5.14e+002	1.33e-001	4.06e-002
03_2_0.pgm	1.84e-004	8.75e+000	8.54e+002	5.08e+002	1.59e-001	3.99e-002
04_1_0.pgm	1.67e-004	8.89e+000	7.31e+002	6.93e+002	4.72e-001	4.99e-002
04_2_0.pgm	1.95e-004	8.78e+000	7.06e+002	6.79e+002	4.81e-001	5.71e-002
05_1_0.pgm	1.92e-004	8.99e+000	1.12e+003	1.11e+003	4.95e-001	5.80e-002
05_2_0.pgm	2.36e-004	8.89e+000	1.00e+003	1.05e+003	5.21e-001	6.41e-002
06_1_0.pgm	3.33e-004	8.37e+000	4.98e+002	4.53e+002	4.50e-001	6.93e-002
06_2_0.pgm	3.91e-004	8.23e+000	4.55e+002	4.10e+002	4.46e-001	7.62e-002
07_1_0.pgm	1.48e-004	9.06e+000	7.84e+002	1.15e+003	6.60e-001	5.06e-002
07_2_0.pgm	1.51e-004	9.05e+000	7.81e+002	1.15e+003	6.61e-001	5.22e-002
08_1_0.pgm	3.58e-004	8.23e+000	2.80e+002	3.67e+002	6.18e-001	8.29e-002
08_2_0.pgm	3.31e-004	8.29e+000	2.94e+002	3.74e+002	6.07e-001	8.02e-002
09_1_0.pgm	1.41e-004	9.09e+000	8.58e+002	1.13e+003	6.19e-001	5.37e-002
09_2_0.pgm	1.35e-004	9.12e+000	8.84e+002	1.15e+003	6.16e-001	5.37e-002
10_1_0.pgm	1.97e-004	8.84e+000	6.70e+002	9.81e+002	6.58e-001	6.80e-002
10_2_0.pgm	1.97e-004	8.83e+000	6.47e+002	1.01e+003	6.79e-001	6.87e-002
11_1_0.pgm	6.63e-004	8.18e+000	5.54e+002	6.35e+002	5.64e-001	1.16e-001
11_2_0.pgm	1.13e-003	7.94e+000	5.20e+002	6.38e+002	5.92e-001	1.33e-001
12_1_0.pgm	1.50e-004	9.01e+000	7.15e+002	8.19e+002	5.63e-001	5.42e-002
12_2_0.pgm	1.48e-004	9.03e+000	6.99e+002	8.56e+002	5.92e-001	5.31e-002
13_1_0.pgm	2.64e-004	8.49e+000	5.56e+002	3.64e+002	2.36e-001	5.87e-002
13_2_0.pgm	2.18e-004	8.65e+000	6.91e+002	4.37e+002	2.10e-001	4.93e-002
14_1_0.pgm	2.13e-004	8.61e+000	7.96e+002	4.12e+002	3.30e-002	4.20e-002
14_2_0.pgm	2.06e-004	8.64e+000	8.72e+002	4.31e+002	-1.11e-002	3.86e-002
15_1_0.pgm	3.42e-004	8.25e+000	3.02e+002	3.62e+002	5.85e-001	8.31e-002
15_2_0.pgm	3.29e-004	8.27e+000	3.15e+002	3.63e+002	5.67e-001	7.98e-002
16_1_0.pgm	7.31e-004	7.66e+000	1.71e+002	3.41e+002	7.50e-001	1.28e-001
16_2_0.pgm	9.39e-004	7.46e+000	1.31e+002	2.81e+002	7.67e-001	1.45e-001