TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

2018-06-27 (82 회차)

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 - 정유경

```
#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

void print(string str, Matx23f& A)
{
    cout << str << "[";
    for(int r=0; r<A.rows;r++)
    {
        for(int c=0; c<A.cols; c++)
        {
            cout << A(r,c);
            if(c < A.cols-1)
            {
                cout << ",";
                cout.width(4);
            }
        }

        if(r < A.rows-1)
        {
            cout << endl;
            cout.width(5);
        }
    }

    cout << "]" << endl;
}

int main(void)
{
    Matx23f A(1,2,3,
              4,5,6);
    Matx<float,2,3> K(1,2,3,4,5,6);
    print("K",K);
    cout << endl << "A =" << endl << (Mat)A << endl;
// 위의 두개 같은 결과


    Matx13f A0 = A.row(0);
    cout << endl << "A0 =" << endl << (Mat)A0 << endl;


    Matx21f A1 = A.col(0); // 1,4
    cout << endl << "A0 =" << endl << (Mat)A1 << endl;


    Matx22f A2 = A.get_minor<2,2>(0,1); // 2 행 2 열 0,1 을 기준점으로
서브행렬을 구하라
    cout << endl << "A2 =" << endl << (Mat)A2 << endl;


    Matx23f B = Matx23f::all(10.0f); // 전부 10
    cout << endl << "B =" << endl << (Mat)B << endl;


    Matx23f C,D,E,F;


    C = A+B;
    D = A-B;
    E = A*5;
    F = A.mul(B); // A 곱하기 10

}


#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

void print(string str, Matx23f& A)
{
    cout << str << "[";
    for(int r=0; r<A.rows;r++)
    {
        for(int c=0; c<A.cols; c++)
        {
            cout << A(r,c);
            if(c < A.cols-1)
            {
                cout << ",";
                cout.width(4);
            }
        }

        if(r < A.rows-1)
        {
            cout << endl;
            cout.width(5);
        }
    }

    cout << "]" << endl;
}

int main(void)
{
    Matx23f A(1,2,3,
              4,5,6);
```

```
    Matx<float,2,3> K(1,2,3,4,5,6);
    print("K",K);
    cout << endl << "A =" << endl << (Mat)A << endl;
// 위의 두개 같은 결과

    Matx13f A0 = A.row(0);
    cout << endl << "A0 =" << endl << (Mat)A0 << endl;

    Matx21f A1 = A.col(0);
    cout << endl << "A0 =" << endl << (Mat)A1 << endl;

    Matx22f A2 = A.get_minor<2,2>(0,1); // 0,1 을 기준점으로 서브행렬을 구하라
    cout << endl << "A2 =" << endl << (Mat)A2 << endl;

    Matx23f B = Matx23f::all(10.0f); // 전부 10
    cout << endl << "B =" << endl << (Mat)B << endl;

    Matx23f C,D,E,F;

    C = A+B;
    D = A-B;
    E = A*5;
    F = A.mul(B);
    // A * B.t() 앞행의 열의수 뒤행의 행의 수가 같아야 곱셈이 된다.
    // 만약 다르다면 전치행렬을 만들어서 3 바이 2 행렬을 만들어서 강제로 곱셈을 시켜주어 결과가 2 행 2 열이 나온다.

return 0;

}
```

```
#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

// 모두 0,1, 단위행렬 (all 은 모두 10.0)
// 행렬을 초기화 하는 방법, ones 는 거의 안쓴다

int main(void)
{
    Matx33f A = Matx33f::zeros();
    Matx33f B = Matx33f::ones();
    Matx33f C = Matx33f::eye();
    Matx33f D = Matx33f::all(10.0);

    cout << "A=" << endl << (Mat)A << endl;
    cout << "B=" << endl << (Mat)B << endl;
    cout << "C=" << endl << (Mat)C << endl;
    cout << "D=" << endl << (Mat)D << endl;

    return 0;
}

/*g++ -o opencv3 opencv3.cpp $(pkg-config opencv --libs) 패키지 컨피그 라는 명령어를 사용해서 헤더파일을 자동으로 설정*/

/*백색잠음은 가우시안잡음
 * 블러링 즉, 관심없는 영역은 깨진다. 이미지가 번진다.
 * 관심있는 영역만 분석할때 쓴다*/
```

```
#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(void)
{
    Matx16f A = Matx16f::randu(0.0,1.0);
    Matx16f B = Matx16f::randu(0.0,1.0);

    cout << "A=" << endl << (Mat)A << endl;
    cout << "B=" << endl << (Mat)B << endl;

    Matx23f C = A.reshape<2,3>();

    cout << "C=" << endl << (Mat)C << endl;

    return 0;
}
```

```
#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(void)
{
    Matx33d A(1,-1,-2,2,-3,-5,-1,3,5
            );
```

```
    Matx33d B = A.inv(DECOMP_CHOLESKY);
    Matx33d C = A.inv(DECOMP_LU);

    Matx33d D = A* B;
    Matx33d E = A*C;

    cout << "A="<< endl << (Mat)A << endl;
    cout << "B="<< endl << (Mat)B << endl;
    cout << "C="<< endl << (Mat)C << endl;
    cout << "D="<< endl << (Mat)D << endl;
    cout << "E="<< endl << (Mat)E << endl;

    return 0;

}
```

```
#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(void)
{
    Matx33d A(2,-1,1,3,3,9,3,3,5);

    Matx31d b(-1,0,4);
    Matx31d X = A.solve(b);

    cout << "X = " <<endl << (Mat)X << endl;

    Mat X2;

    solve((Mat)A, (Mat)b, X2);

    cout << "X2 = " << endl << (Mat)X << endl;

    return 0;
}
```

```
#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(void)
{
    Vec<float,3> X(1,0,0);
    Vec<float,3> Y(0,1,0);
    Vec3f Z = X.cross(Y);

    cout << "X = " << endl << (Mat)X << endl;
    cout << "Y = " << endl << (Mat)Y << endl;
    cout << "Z =X.cross(Y)= " << endl << (Mat)Z << endl;

    Point3f pt3 = X;

    cout << "pt3 = " << endl << pt3 << endl;

    X = Vec3f(1,2,3);
    Y = Vec3f(10,100,1000);

    Z = X.mul(Y);

    cout << "X = " << endl << X << endl;
    cout << "Y = " << endl << Y << endl;
    cout << "Z = X.mul(Y) = " << endl << Z << endl;

    cout << "sum(Z) = " << endl << sum(Z) << endl;
    cout << "douProsuct = " << endl << sum(Z)(0) << endl;

    X = Vec3f::all(0.0);

    cout << "X = " << endl << X << endl;

    return 0;
}
```

```
#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(void)
{
    Matx33f A(1,2,3,4,5,6,7,8,9);

    Mat B(A);

    cout << "B = " << endl << B << endl;
```

```
	cout << "B[0:1,0:3] = " << endl << B(Range(0,1), Range(0,3)) << endl;
	cout << "B[0:2,0:3] = " << endl << B(Range(0,2), Range(0,3)) << endl;
	cout << "B[1:2,0:3] = " << endl << B(Range(1,2), Range(0,3)) << endl;

	Mat C = B(Range(1,3),Range::all());

	cout << "C=" << endl << C << endl;

	Mat D = B(Range::all(), Range(1,3));

	cout << "D =" << endl << D << endl;

	B(Range(0,1), Range::all()).copyTo(B(Range(1,2),Range::all()));

	cout << "B= " << endl << B << endl;

	return 0;
}
```

```
#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(void)
{
	Ptr<IplImage> Image(cvLoadImage("index.jpeg", IMREAD_GRAYSCALE));
	if(Image.empty())
		return -1;

	cvSaveImage("1.bmp",Image);
	cvNamedWindow("Image",CV_WINDOW_AUTOSIZE);
	cvShowImage("Image",Image);
	cvWaitKey(0);
	cvDestroyAllWindows();

	return 0;
}
```

```
include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(void)
{
	Ptr<CvMat> matA(cvCreateMat(2,3,CV_32FC1));

	CV_MAT_ELEM(*matA, float, 0,0) = 1.0f;
	CV_MAT_ELEM(*matA, float, 0,1) = 2.0f;
	CV_MAT_ELEM(*matA, float, 0,2) = 3.0f;

	CV_MAT_ELEM(*matA, float, 1,0) = 4.0f;
	CV_MAT_ELEM(*matA, float, 1,1) = 5.0f;
	CV_MAT_ELEM(*matA, float, 1,2) = 6.0f;

	Mat A = cvarrToMat(matA);

	cout << "A = " << endl << A << endl;

	return 0;
}
```

```
#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(void)
{
	Mat A(2,3,CV_8UC1);
	Mat B(2,3,CV_8UC1,Scalar(0));
	Mat C(2,3,CV_8UC1,Scalar(1,2,3));

	float data[] = {1,2,3,4,5,6};

	Mat D(2,3,CV_32FC1,data);

	cout << "A = " << endl << A << endl;
	cout << "B = " << endl << B << endl;
	cout << "C = " << endl << C << endl;
	cout << "D = " << endl << D << endl;

	Mat A1(Size(3,2),CV_8UC1);
	Mat B1(Size(3,2),CV_8UC1,Scalar(0));
```

```
    Mat C1(Size(3,2),CV_8UC3,Scalar(1,2,3));
    Mat D1(Size(3,2),CV_8UC1,data);

    cout << "A1 = "<< endl << A1 << endl;
    cout << "B1 = "<< endl << B1 << endl;
    cout << "C1 = "<< endl << C1 << endl;
    cout << "D1 = "<< endl << D1 << endl;

    return 0;
}
```

```
#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(void)
{
    int sizes[] = {2,3,4};

    Mat A(3,sizes,CV_32FC1);
    Mat B(3,sizes,CV_32FC1,Scalar(0));

    cout << "B.dims = " << B.dims << endl;
    cout << "B.rows = " << B.rows << endl;
    cout << "B.cols = " << B.cols << endl;

    cout << "B.size[0] = " << B.size[0] << endl;
    cout << "B.size[1] = " << B.size[1] << endl;
    cout << "B.size[2] = " << B.size[2] << endl;

    for(int i = 0; i < B.size[0]; i++)
    {
        cout << "\nB[" << i << "]" << endl;

        for( int j =0; j< B.size[1]; j++)
        {
            for(int k =0; k < B.size[2]; k++)
            {
                cout << B.at<float>(i,j,k);

                if(k != B.size[2] -1)
                    cout << ",";
                else
                    cout << ";";
            }

            cout << endl;
        }
    }
    return 0;
}
```

```
#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(void)
{
    Mat A(2,3,CV_32FC1, Scalar(0));
    cout << "A = " << endl << A <<endl;

    A.create(2,3,CV_32FC1);
    cout << "A = " << endl << A << endl;

    A.create(3,3,CV_32FC1);
    cout << "A = " << endl << A << endl;

    A.create(Size(3,3),CV_8UC1);
    cout << "A =" << endl << A << endl;

    Mat B;
    int sizes[] = {3,3};
    B.create(2,sizes,CV_8UC1);

    cout << "B=" << endl << B << endl;

    return 0;
}
```

```
#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(void)
```

```
{
    Mat srcImage;
    srcImage.create(512,512,CV_8UC3);

    for(int i = 0; i < srcImage.rows; i++)
        for(int j = 0; j < srcImage.cols; j++)
            srcImage.at<Vec3b>(i,j) = Vec3b(255,255,255);

    imshow("srcImage",srcImage);
    waitKey();

    return 0;
}
```

```
#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;


int main(void)
{
     Mat A(4,5,CV_32FC3);
    cout << "A.rows = " << A.rows << endl;
    cout <<" A.cols = " << A.cols << endl;
    cout << "A.dims = " << A.dims << endl;

    Mat B = A;

    A.at<Vec3f>(0,0) = Vec3f(0.75,1.0,10.0);
    cout << "A.data = " << hex << (int *)A.data << endl;
    cout << "B.data = " << hex << (int *)B.data << endl;
    cout << "A.data[0] = " << *(float *)A.data << endl;
    cout << "A.data[4] = " << *(float *)(A.data + 4) << endl;
    cout << "A.data[8] = " << *(float *)(A.data + 8 ) << endl;
    cout << "B.data[0] = " << *(float *)B.data << endl;
    cout << "B.data[4] = " << *(float *)(B.data + 4) << endl;
    cout << "B.data[8] = " << *(float *)(B.data + 8 ) << endl;

    cout << "A.isContinuous() = " << A.isContinuous() << endl;
    cout << "A.total() = " << A.total() << endl;
    cout << "A.elemSize() =" << A.elemSize() << endl;
    cout << "A.elemSize1() = " << A.elemSize1() << endl;
    cout << "A.type() = "<< A.type() << endl;
    cout << "A.depth() = " << A.depth() << endl;
    cout << "A.channels() = " << A.channels() << endl;

    cout << "A.step=" << A.step << endl;
    cout << "A.step1() = " << A.step1() << endl;
    cout << "A.empty() = " << A.empty() << endl;
    cout << "A.size() = " << A.size() << endl;

    return 0;
}
```

```
#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(void)
{
   Mat A(3,3,CV_32F);
   int idx[2];

   for(int i=0; i< A.rows; i++)
     for(int j= 0; j< A.cols; j++)
     {
        A.at<float>(j,i)=i*A.cols +j;

     }

   cout << "A ="<<A<<endl;

   int nSum =0;
   for(int i=0; i<A.rows; i++)
      for(int j=0; j<A.cols; j++)
      {
            nSum += A.at<float>(i,j);
      }

    cout << "nSum = " << nSum << endl;
    return 0;

}
```

```
#include "opencv2/opencv.hpp"
#include <iostream>
```

```
using namespace std;
using namespace cv;

int main(void)
{
    Mat A(3,3,CV_64FC2);
    for(int i =0; i< A.rows; i++)
      for(int j = 0; j<A.cols; j++)
      {
          A.at<Vec2d>(i,j) = Vec2d(0,i * A.cols +j);

      }

    cout << A << endl;

    A.create(3,3,CV_64FC3);
    for(int i =0; i< A.rows; i++)
    {
      for(int j = 0; j<A.cols; j++)
      {
        A.at<Vec3d>(i,j) = Vec3d(0,i * A.cols +j);

      }
    }

    cout << A << endl;

    A.create(3,3,CV_64FC4);
    for(int i=0; i< A.rows; i++)
      for(int j =0; j< A.cols; j++)
      {
        A.at<Scalar>(i,j) = Scalar(0,0,0,i*A.cols +j);
      }

    cout << A << endl;
    return 0;

}
```

```
#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(void)
{
  Mat A(3,3,CV_32F);
  for(int i=0; i<A.rows; i++)
    for(int j=0; j<A.cols; j++)
      A.at<float>(i,j)= (float)(i*A.cols + j);

  Mat B = A.row(0);
  Mat C = A.col(0);
  Mat D = A.rowRange(0,2);
  Mat E = A.colRange(0,2);

  cout << " A = " << A << endl;
  cout << " B = " << B << endl;
  cout << " C = " << C << endl;
  cout << " D = " << D << endl;
  cout << " E = " << E << endl;

  A.row(0) = A.row(0) * 10;
  A.row(1) = A.row(1) * 100;
  A.row(2) = A.row(2) * 1000;
  cout << "A= " << A << endl;

  A.row(1) = A.row(2);
  cout << "A= " << A << endl;

  A.row(2).copyTo(A.row(1));

  cout<< "A = " << A << endl;
  cout<< "B = " << B << endl;
  cout<< "C = " << C << endl;
  cout<< "D = " << D << endl;
  cout<< "E = " << E << endl;

  return 0;
}
```

```
#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(void)
{
  Mat A(3,3,CV_32FC3);
```

```
    for(int i =0; i< A.rows; i++)
    {
        Vec3f* ptrA = A.ptr<Vec3f>(i);
        for(int j =0; j < A.cols; j++)
            ptrA[j] = Vec<float,3>(255,0,i*A.cols +j);

    }

    cout << "A = " << A << endl;

    Mat B(3,3,CV_32FC3);
    for(int i=0; i< A.rows; i++)
    {
        float* ptrB = B.ptr<float>(i);
        for(int j=0; j< A.cols; j++)
        {
            ptrB[j*3] = 255;
            ptrB[j*3+1] =0;
            ptrB[j*3+2] = i*B.cols+j;

        }
    }
    cout << "B =" << B << endl;

    return 0;


}


#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(void)
{
    Mat A(3,3,CV_32F);
    for(int i=0; i< A.rows; i++)
        for(int j=0; j < A.cols; j++)
            A.at<float>(i,j) = (float)(i*A.cols + j);

    cout << "A = " << A << endl;

    Mat B = A.clone();
    cout << "B = "<< B << endl;

    Mat C1;
    A.copyTo(C1);
    cout << "C1 = " << C1 << endl;

    Mat C2;
    A.copyTo(C2);
    cout << "C2 = " << C2 << endl;

    Mat mask(3,3,CV_8UC1,Scalar(0));
    mask.row(1).setTo(Scalar::all(1));
    cout << "mask = " << mask << endl;

    Mat C3;
    A.copyTo(C3,mask);
    cout << "C3 = " << C3 << endl;

    Mat D1;
    A.convertTo(D1, CV_8U);
    cout << "D1 = " << D1 << endl;

    Mat D2;
    A.convertTo(D2,CV_8U,10.0,1.0);
    cout << "D2 = " << D2 << endl;

    Mat E1;
    A.assignTo(E1);
    cout << "E1 = " << E1 << endl;

    Mat E2;
    A.assignTo(E2,CV_8U);
    cout << "E2 = " << E2 << endl;

    A.setTo(Scalar::all(0));
    cout << "A = " << A << endl;

    Mat F1 = A.reshape(0,1);
    cout << "F1 = " << F1.size() << " = " << F1 << endl;

    Mat F2 = A.reshape(0,9);
    cout << "F2 = " << F2.size() << " = " << F2 << endl;

    Mat F3 = A.reshape(3,3);
    cout << "F3 = " << F3.size() << " = " << F3 << endl;

    return 0;
```

```
}
```

```
#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(void)
{
    Mat A(3,3,CV_32F,Scalar::all(0));
    cout << "A = " << A.size() << " = " << A << endl;

    A.resize(2);
    cout << "A = " << A.size() << " = " << A << endl;

    A.resize(5,Scalar::all(1));
    cout << "A = " << A.size() << " = " << A << endl;

    A.resize(10);
    cout << "A = " << A.size() << " = " << A << endl;

    A.release();
    cout << "A = " << A.size() << " = " << A << endl;

    return 0;
}
```

```
#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(void)
{
    Mat A(10,10,CV_32F);
    for(int i=0; i<A.rows;i++)
        for(int j=0; j< A.cols; j++)
            A.at<float>(i,j) = (float)(i*A.cols +j);

    cout << "A = " << A.size() << " = " << A << endl;

    Mat B = A(Range(5,8),Range(3,6));
    cout << "B =" << B.size() << " =" << B << endl;

    Size wholeSize;
    Point ofs;
    B.locateROI(wholeSize,ofs);
    cout << "WholeSize = " << wholeSize << "ofs = " << ofs <<
endl;

    Mat C = B.adjustROI(1,1,1,1);

    cout << "B = " << B.size() << "B =" << B << endl;
    cout << "C = " << C.size() << "C = " << C << endl;

    return 0;
```

```
}
```

```
#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(void)
{
    Mat_<uchar> srcImage =
imread("joseph.jpeg",IMREAD_GRAYSCALE);

    float sum = 0;
    for(int i=0; i<srcImage.rows; i++)
        for(int j = 0; j< srcImage.cols; j++)
            sum += srcImage(i,j);

    cout << "Abg = sum / total = " << sum/ srcImage.total() <<
endl;

    waitKey();

    return 0;
}
```

과제 : 픽셀값 평균이하인 부분을 검정색으로 설정하여 이미지의 대비를 높여 선명하게 한다

```
#include "opencv2/opencv.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(void)
{
    Mat_<uchar> srcImage = imread("11.jpeg",IMREAD_GRAYSCALE);
    Mat_ <uchar> dstimage= srcImage;
    float sum = 0;
    for(int i=0; i<srcImage.rows; i++)
        for(int j = 0; j< srcImage.cols; j++)
            sum += srcImage(i,j);

    cout << "Abg = " << sum/ srcImage.total() << endl;

    for(int i=0; i<srcImage.rows; i++)
        for(int j = 0; j< srcImage.cols; j++)
        {
              if(srcImage(i,j) < sum/srcImage.total())
            {
                image(i,j) =0; // 검은색 pixel 값이 평균보다 작을때
            }
        }
        imshow("dstImage",image);
    waitKey();

    return 0;
}
```