Homework 1

Computer Vision 2017 Spring

2017.3.14

OpenCV

- Highly optimized C++ implementation of many CV algorithms
- Crossplatform
- Many bindings (Python, Matlab, Java . . .)

http://opencv.org

Install

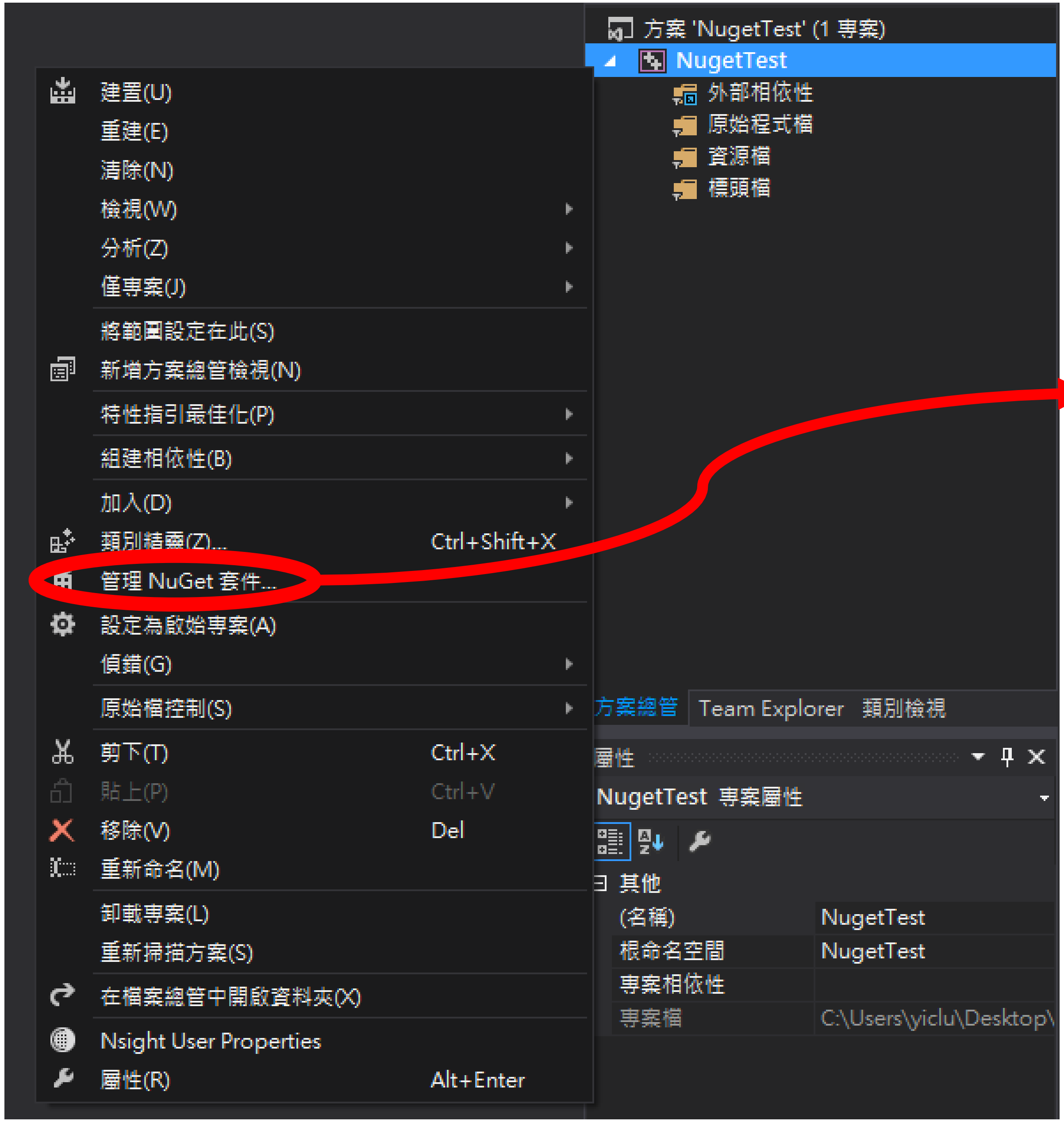
- VS 2013、VS 2015

1 , Open your visual Studio

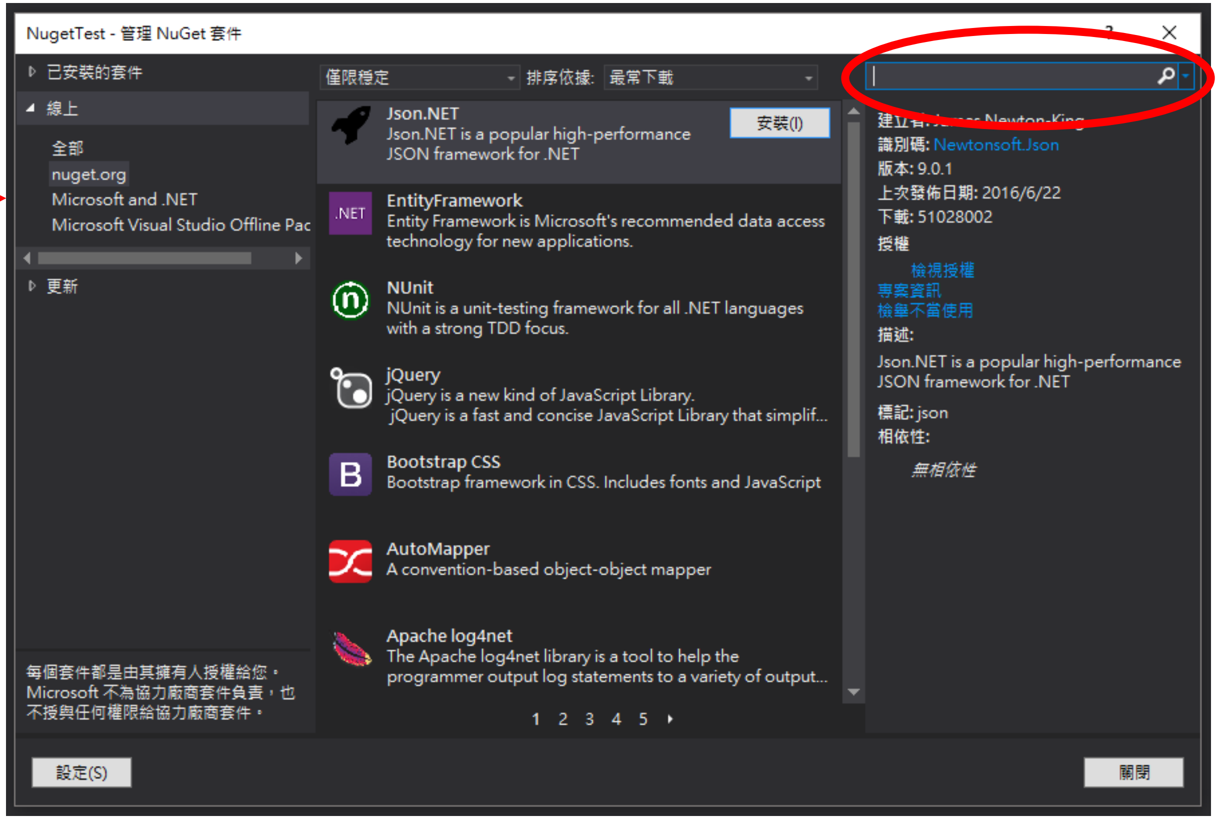
2 , New a Empty Project

3 , Install OpenCV

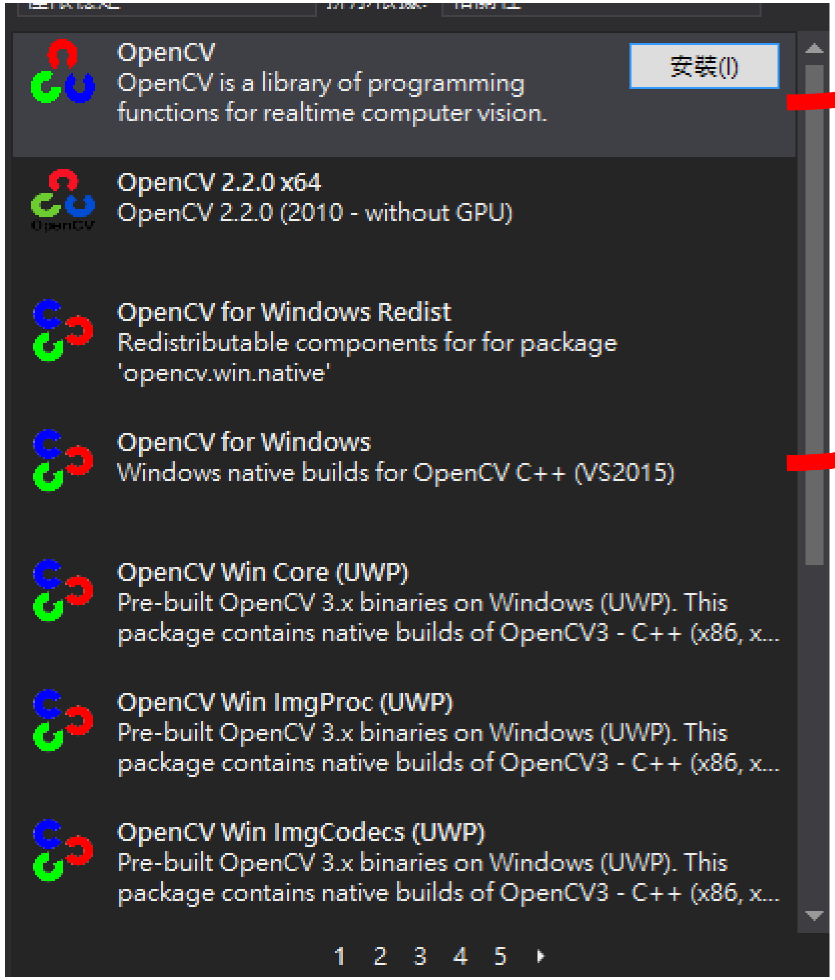
VS 2013



方案 'NugetTest' (1 專案)
NugetTest
外部相依性
原始程式檔
資源檔
標頭檔
建置(U)
重建(E)
清除(N)
檢視(W)
分析(Z)
僅專案(J)
將範圍設定在此(S)
新增方案總管檢視(N)
特性指引最佳化(P)
組建相依性(B)
加入(D)
類別精靈(Z)... Ctrl+Shift+X
管理 NuGet 套件...
設定為啟始專案(A)
偵錯(G)
原始檔控制(S)
剪下(T) Ctrl+X
貼上(P) Ctrl+V
移除(V) Del
重新命名(M)
卸載專案(L)
重新掃描方案(S)
在檔案總管中開啟資料夾(X)
Nsight User Properties
屬性(R) Alt+Enter
方案總管
Team Explorer
類別檢視
屬性
NugetTest 專案屬性
其他
(名稱) NugetTest
根命名空間 NugetTest
專案相依性
專案檔 C:\Users\yiclu\Desktop\

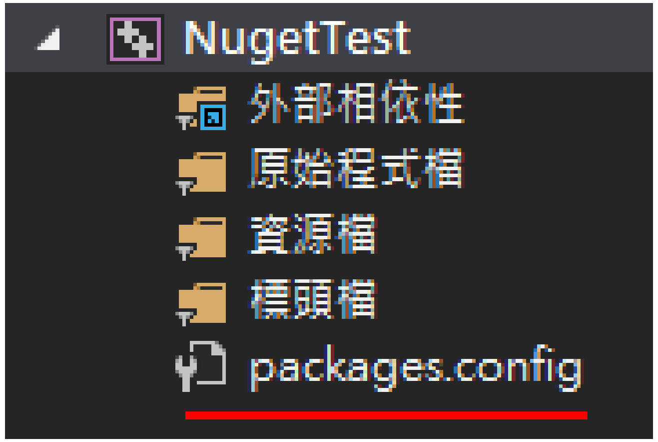


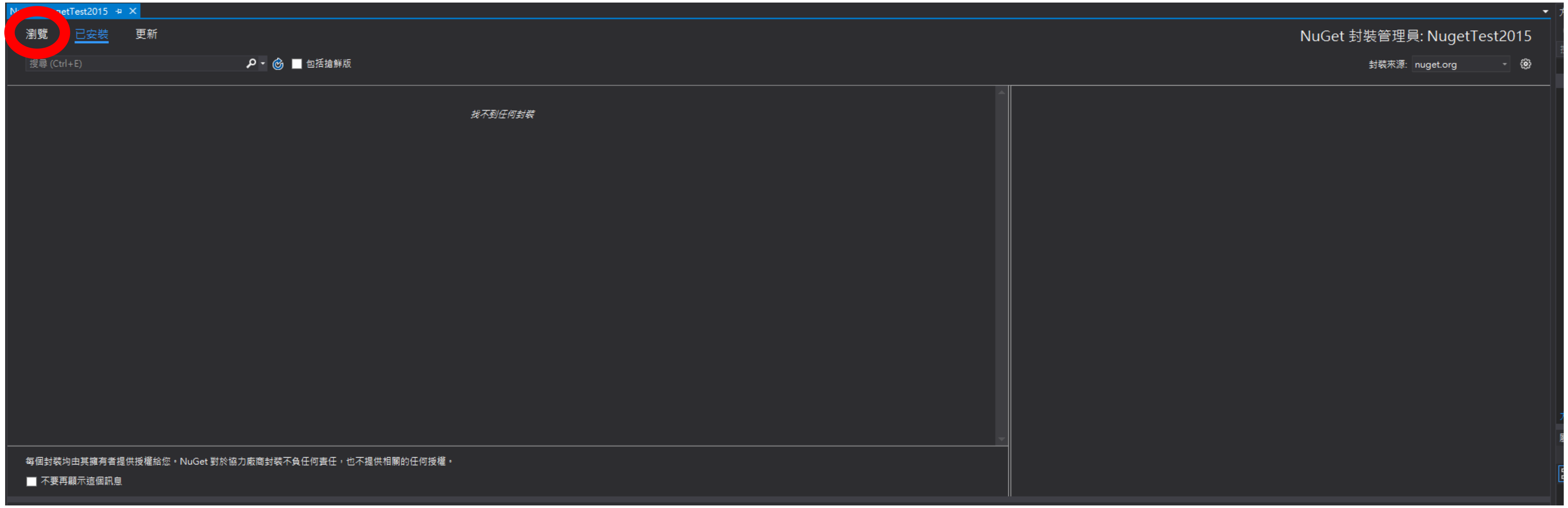
NugetTest - 管理 NuGet 套件
已安裝的套件
線上
全部
nuget.org
Microsoft and .NET
Microsoft Visual Studio Offline Pac
更新
僅限穩定
排序依據: 最常下載
Json.NET
Json.NET is a popular high-performance JSON framework for .NET
安裝(I)
EntityFramework
Entity Framework is Microsoft's recommended data access technology for new applications.
NUnit
NUnit is a unit-testing framework for all .NET languages with a strong TDD focus.
jQuery
jQuery is a new kind of JavaScript Library.
jQuery is a fast and concise JavaScript Library that simplif...
Bootstrap CSS
Bootstrap framework in CSS. Includes fonts and JavaScript
AutoMapper
A convention-based object-object mapper
Apache log4net
The Apache log4net library is a tool to help the programmer output log statements to a variety of output...
識別碼: Newtonsoft.Json
版本: 9.0.1
上次發佈日期: 2016/6/22
下載: 51028002
授權
檢視授權
專案資訊
檢舉不當使用
描述:
Json.NET is a popular high-performance JSON framework for .NET
標記: json
相依性:
無相依性
每個套件都是由其擁有人授權給您。Microsoft 不為協力廠商套件負責，也不授與任何權限給協力廠商套件。
1 2 3 4 5
設定(S)
關閉



VS 2013

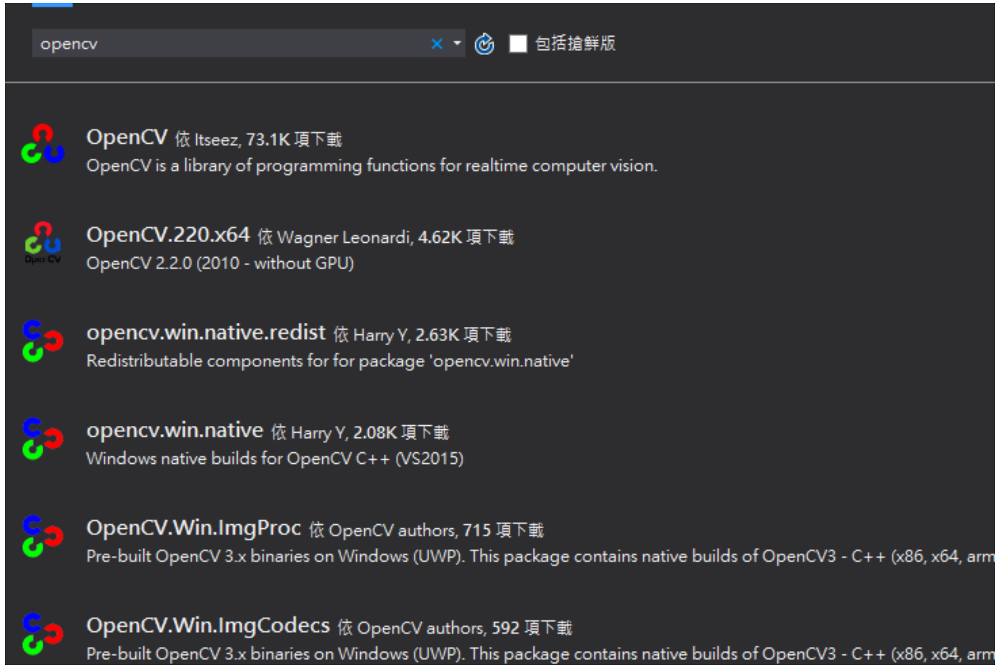
VS 2015




瀏覽
已安裝
更新
NuGet 封裝管理員: NugetTest2015
搜尋 (Ctrl+E)
包括搶鮮版
封裝來源:
nuget.org
找不到任何封裝
每個封裝均由其擁有者提供授權給您。NuGet 對於協力廠商封裝不負任何責任，也不提供相關的任何授權。
不要再顯示這個訊息

VS 2015

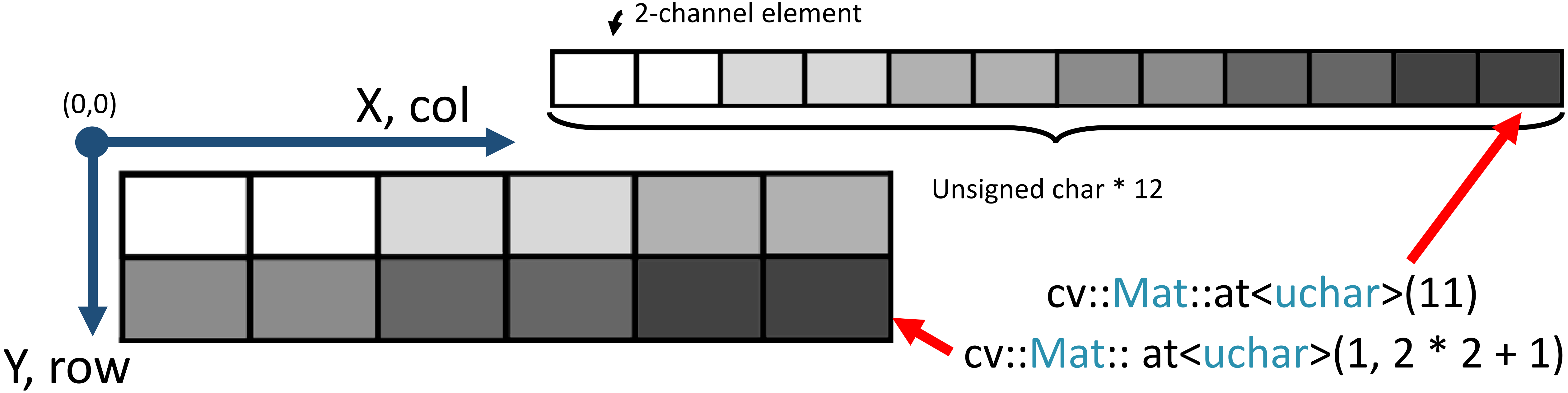

瀏覽
已安裝
更新
搜尋 (Ctrl+E)
包括搶鮮版
Newtonsoft.Json 依 James Newton-King, 50.9M 項下載
Json.NET is a popular high-performance JSON framework for .NET
.NET
EntityFramework 依 Microsoft, 27.6M 項下載
Entity Framework is Microsoft's recommended data access technology for new applications.
NUnit 依 Charlie Poole, 8.62M 項下載
NUnit is a unit-testing framework for all .NET languages with a strong TDD focus.
jQuery 依 jQuery Foundation, Inc., 32.5M 項下載
jQuery is a new kind of JavaScript Library.
jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, a
B
bootstrap 依 Twitter, Inc., 9.22M 項下載
Bootstrap framework in CSS. Includes fonts and JavaScript
AutoMapper 依 Jimmy Bogard, 7.48M 項下載
A convention-based object-object mapper


opencv
包括搶鮮版
OpenCV 依 Itseez, 73.1K 項下載
OpenCV is a library of programming functions for realtime computer vision.
OpenCV.220.x64 依 Wagner Leonardi, 4.62K 項下載
OpenCV 2.2.0 (2010 - without GPU)
opencv.win.native.redist 依 Harry Y, 2.63K 項下載
Redistributable components for for package 'opencv.win.native'
opencv.win.native 依 Harry Y, 2.08K 項下載
Windows native builds for OpenCV C++ (VS2015)
OpenCV.Win.ImgProc 依 OpenCV authors, 715 項下載
Pre-built OpenCV 3.x binaries on Windows (UWP). This package contains native builds of OpenCV3 - C++ (x86, x64, arm
OpenCV.Win.ImgCodecs 依 OpenCV authors, 592 項下載
Pre-built OpenCV 3.x binaries on Windows (UWP). This package contains native builds of OpenCV3 - C++ (x86, x64, arm

Mat

- Multi-dimensional array
- Handle the memory automatically

```
cv::Mat(2, 3, CV_8UC2);
```



Sample

- See OpenCV documentation and samples for more information

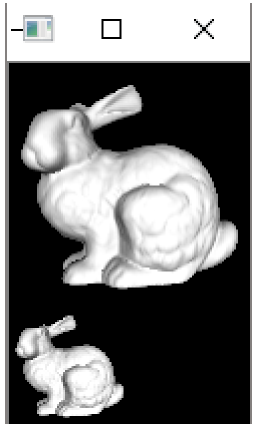
```
#include <opencv/highgui.h>
```

Include this in VS 2013

```
#include <opencv2/highgui/highgui.hpp>
```

Include this in VS 2015

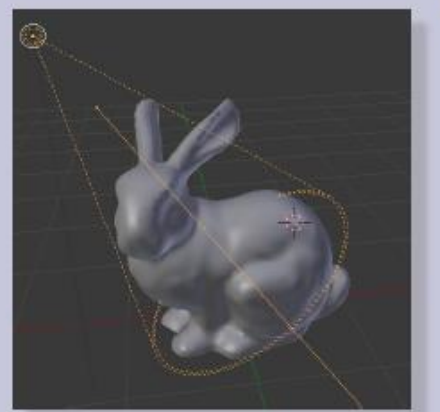
```
using namespace cv;

int main() {
    Mat Image = imread("test/bunny/pic1.bmp", IMREAD_GRAYSCALE);

    Mat tempImage = Image.clone();
    Mat smallImage(Image.rows / 2, Image.cols / 2, CV_8U);

    for (int rowIndex = 0; rowIndex < smallImage.rows; rowIndex++) {
        for (int colIndex = 0; colIndex < smallImage.cols; colIndex++) {
            smallImage.at<uchar>(rowIndex, colIndex) = tempImage.at<uchar>(rowIndex * 2, colIndex * 2);
        }
    }

    Mat result(tempImage.rows + smallImage.rows, tempImage.cols, CV_8U, Scalar(0));
    tempImage.copyTo(result(Rect(0, 0, tempImage.cols, tempImage.rows)));
    smallImage.copyTo(result(Rect(0, tempImage.rows, smallImage.cols, smallImage.rows)));

    imshow("CV", result);
    waitKey();

    return 0;
}
```



Photometric Stereo

grayscale

Lambertian Reflection



$$i = \mathrm{k_d}\,\mathrm{l}(\mathbf{s}^\intercal \mathbf{n}) \longrightarrow$$

$$\longleftarrow \mathrm{i} = \mathrm{k_d}\,\mathrm{l}(\mathbf{s}^\intercal \boldsymbol{n})$$

$i^{(m)}_{x,y}$	the intensity of the mth image at pixel (x, y)
$\mathrm{k_d}$	the color of the surface
l_m	the intensity of the mth incoming light
s_m	the **unit vector** pointing from the surface to the m incoming light
$\boldsymbol{n_{x,y}}$	the surface's normal vector (**unit vector**) at pixel (x, y)

For brevity, we omits the dependence of x, y and m.

Normal Estimation

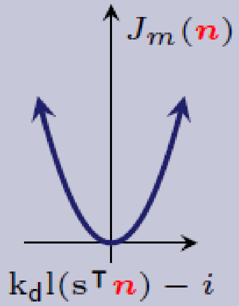
According to the reflection model, we suppose that the unknown normal vector and the intensity in the mth image at pixel (x,y) will satisfy

$$\mathrm{i}_{x,y}^{(m)} \overset{?}{=} \mathrm{k_d} \mathrm{l}_m (\mathbf{s}_m^\intercal \boldsymbol{n})$$

To estimate how bad a specific n is, we define the least squares loss function

$$J(\boldsymbol{n}) = \sum_m J_m(\boldsymbol{n}) = \sum_m \|\mathrm{k_d} \mathrm{l}_m (\mathbf{s}_m^\intercal \boldsymbol{n}) - \mathrm{i}^{(m)}\|^2$$

Normal Estimation



- The more loss $J(\boldsymbol{n})$ $\boldsymbol{n}$ has, the less chance $\boldsymbol{n}$ is the correct normal vector at pixel (x, y).
- Solve (`cv::Mat::inv`, `cv::Mat::t`) the following linear system to get the $\boldsymbol{n}$ with minimum loss:

$$S^\intercal S \boldsymbol{b} = S^\intercal \mathbf{i}, \quad \text{where } S = \begin{bmatrix} l_1 \mathbf{s}_1^\intercal \\ l_2 \mathbf{s}_2^\intercal \\ \vdots \\ l_m \mathbf{s}_m^\intercal \end{bmatrix}, \mathbf{i} = \begin{bmatrix} i^{(1)} \\ i^{(2)} \\ \vdots \\ i^{(m)} \end{bmatrix} \text{ and } \boldsymbol{b} = k_d \boldsymbol{n}$$

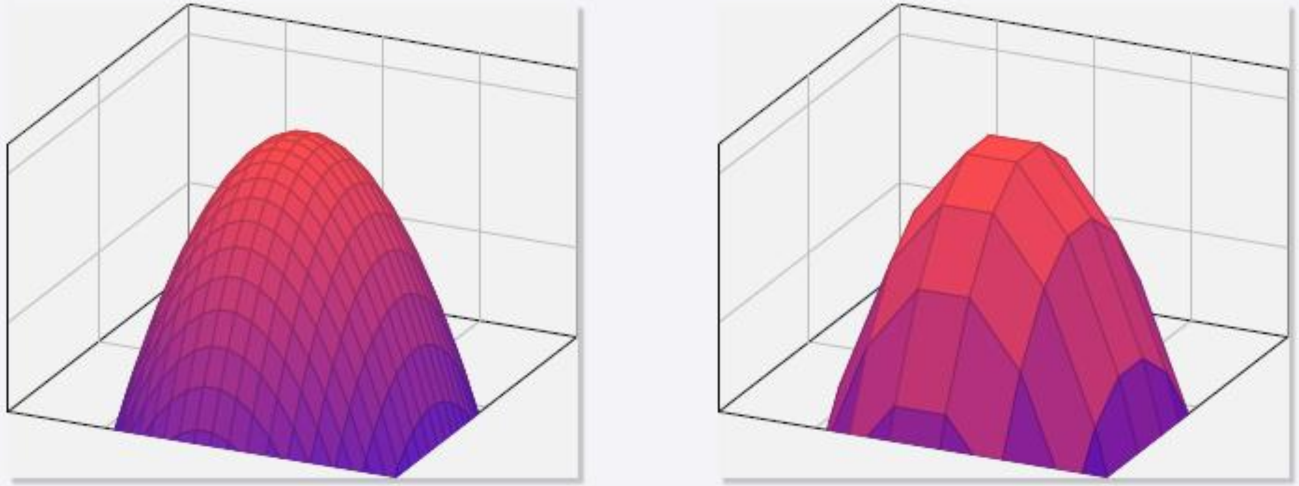
In practice we solve the problem with *QR* or *SVD*.

Surface Reconstruction

The surface $z(x, y)$ near pixel $(\mathrm{x}^*, \mathrm{y}^*)$ can be approximated by the tangent plane:

$$\mathrm{n}_1(x - \mathrm{x}^*) + \mathrm{n}_2(y - \mathrm{y}^*) + \mathrm{n}_3(z - z(\mathrm{x}^*, \mathrm{y}^*)) = 0 \qquad (1)$$

where $(\mathrm{n}_1, \mathrm{n}_2, \mathrm{n}_3)^\mathsf{T}$ is the normal vector at $(\mathrm{x}^*, \mathrm{y}^*)$.



Surface Reconstruction

The equation 1 can be rewritten as

$$z_{\text{approx}}(x, y) = \left(-\frac{\mathrm{n}_1}{\mathrm{n}_3}\right) x + \left(-\frac{\mathrm{n}_2}{\mathrm{n}_3}\right) y + \text{constant}$$

We can reconstruct the surface $\tilde{z}(x, y)$ as we know the gradient of z_{approx} at each pixel, for example, by

$$\tilde{z}(x, y) = \sum_{i=0}^{x-1} \left.\frac{\partial z_{\text{approx}}}{\partial x}\right|_{(i,0)} + \sum_{j=0}^{y-1} \left.\frac{\partial z_{\text{approx}}}{\partial y}\right|_{(x,j)}$$

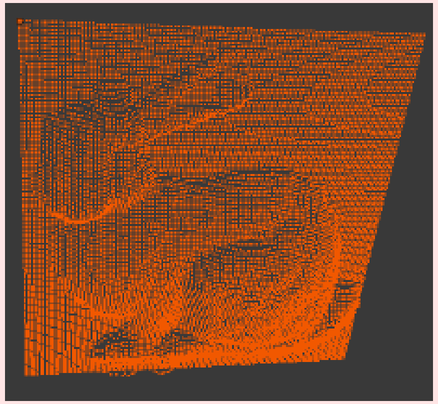
Surface Reconstruction

Note

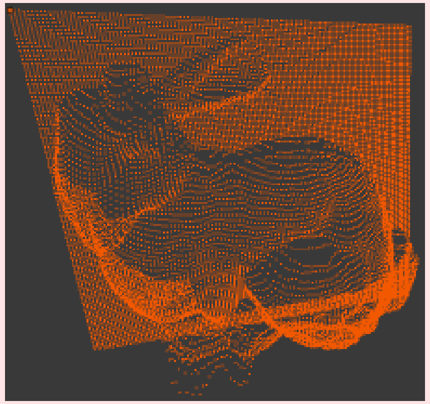
You will probably have to scale $\tilde{z}$ for visualizing the surface:

$$\tilde{z}_{\text{vis}}(x, y) = \alpha \tilde{z}(x, y)$$

$\alpha = 1$



$\alpha = 4$



Surface Reconstruction

- Other Tips
  - Weighted Least Squares measure loss terms with different weights

$$J_{\mathbf{W}}(\boldsymbol{n}) = \sum_m W_m J_m(\boldsymbol{n}) = \sum_m W_m \| \mathrm{k_d} \mathrm{l}_m (\mathrm{s_m}^\intercal \boldsymbol{n}) - \mathrm{i}^{(m)} \|^2$$

$$= \|\mathbf{W}\mathbf{S}b - \mathbf{W}\mathbf{l}\|^2, \quad \text{where } \mathbf{W} = \begin{bmatrix} \mathrm{w}_1 & & \\ & \mathrm{w}_2 & \\ & & \ddots \end{bmatrix}, \mathrm{w}_m = \sqrt{\mathrm{W}_m}$$

  - Sanity Check

$$\frac{\partial^2 z}{\partial x \partial y} = \frac{\partial^2 z}{\partial y \partial x}$$

Input & Output

- Input:
  - 6 .bmp image
  - LightSource.txt

- Output:

  .ply file
    - ply~end_header照打
    - comment alpha有改的話再改數字
    - end_header以下的點對應上面的x y z 顏色
  - http://potree.org/demo/plyViewer/plyViewer.html

Grading

70% Follow the instructions in page 9-14 To reconstruct surfaces

10% Experiment with tips mentioned in class or in page 15

10% reconstruct surfaces for some special input data

15% Take pictures of real objects as input data

15% Reconstruct surfaces by solving optimization problems
(see the Appendix in the course material)

Deadline

- 期限: 2017/03/30 (四) 11:59:59 pm
- 請將作業壓縮並以學號命名: ex 0987654-hw1.zip
    - 資料夾內包含:
        - 1. 學號-hw1-report.pdf
        - 2. bunny-surface.ply, venus-surface.ply…
        - 3. code 或 完整專案
- 上傳至e3
- Code需要加上對應流程註解
- Report包含程式流程說明 & 執行方式 & 自己多做了哪些功能&其他你想寫的…