|  |
| --- |
| **보간법 (Interpolation)** |

|  |  |
| --- | --- |
| 적용 분야 | S/W |
| 소속 부서 | 의료영상SW팀 |
| 작성 일자 | 2019. 03. 29. |
| 작성자 | 김재혁 |
| 보안 등급 |  |
| 기여율 | 100% |

목차

[1 개요 3](#_Toc497721509)

[2 연구(검토) 목적 4](#_Toc497721510)

[3 연구(검토) 세부 내용 4](#_Toc497721511)

[4 적용 사례 및 예시 7](#_Toc497721516)

[5 추가 및 보완(개선) 사항 8](#_Toc497721517)

[6 향후 계획 9](#_Toc497721518)

[7 참고 문헌 및 자료 10](#_Toc497721519)

# 개요

|  |  |  |  |
| --- | --- | --- | --- |
| 문서 구분 | ■ 기술 보고 □ 선행 조사 □ 경쟁사 분석 □ 기타 | | |
| 개발 구분 | ■ 개발용 □ 양산용 □ 선행 검토용 ■ 교육용 □ 기타 | | |
| 적용 분야 | ■ 공통 □ 의료기기 □ 산업기기 □ 신사업 □ 기타 | | |
| □ HW ■ SW □ FPGA/FW □ 광학 □ 기구 □ 기타 | | |
| 적용 모델 | QXLink 3 | | |
| 보고서 이름 | 보간법 (Interpolation) | | |
| 작성일 | 2019. 03. 29. | 작성 부서 | 의료영상SW팀 |
| 검토 기간 | 2019. 01. 02. ~ 2019. 03. 8. | | |
| 시험 기간 | 2019. 03. 08. ~ 2019. 03. 29. | | |
| 시험 시료 |  | | |
| 시험 장비  프로그램 | Visual studio 2015 | | |
| 키워드 | Interpolation, 보간법, Nearest interpolation, Linear interpolation, Cubic interpolation, Spline Interpolation, Bilinear, Bicubic, B-Spline, Lanczos | | |
| 요약 정리 | Gdiplus에서 제공하는 Interpolation을 사용하여 왔으나, 성능과 종류의 부족을 느껴 Interpolation에 대해 알아보고 이를 직접 구현해 보기로 한다. | | |

# 연구(검토) 목적

영상의 크기를 늘리는 Upsampling 시에 원본 Data가 없는 Pixel의 값을 보간 해주는 방법을 기존 Gdiplus를 이용하여 사용하고 있었으나, 성능 개선과 더 다양한 방식을 적용해 보고자 이 연구를 통해 Interpolation의 종류와 방법에 대해 알아보고 구현해 보자.

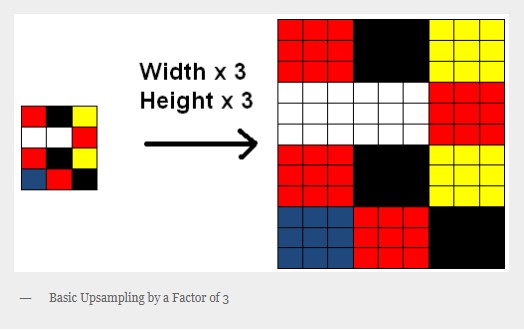
# 연구(검토) 세부 내용

## 개요

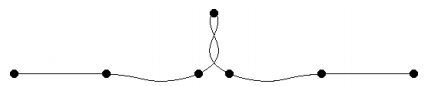
### Image resampling

* 디지털 영상은 같은 크기의 사각형 픽셀로 이루어 진다.
* Resampling은 픽셀에서 다른 너비와 높이를 갖는 영상의 새로운 버전을 만들기 위해 사용되는 수학적인 기술이다. 영상의 크기를 늘리는 것을 **Upsampling** 줄이는 것을 **Downsampling** 이라고 한다.
  + 예) Upsampling  
     🡪
* 위와 같이 영상이 Upsampling 되었을 때, Pixel의 수는 늘어나게 되는데 원본 샘플은 한정적이라 새롭게 만들어지는 영상의 모든 샘플을 만들어 낼 수 없다. 따라서, 원본 **샘플 사이의 추정 값을 얻기 위해 Interpolation이 필요하다.**
* Downsampling 역시 더 작은 영상으로 만들기 위해서는 원본 샘플의 몇몇을 무시해야 한다. 그리고 이는 원본 샘플의 **가중치 평균을 이용해 계산**하여, 새로운 픽셀에서 Overlay 되어야 한다.
* Resampling은 Resize transformation에 의해 수행된다.

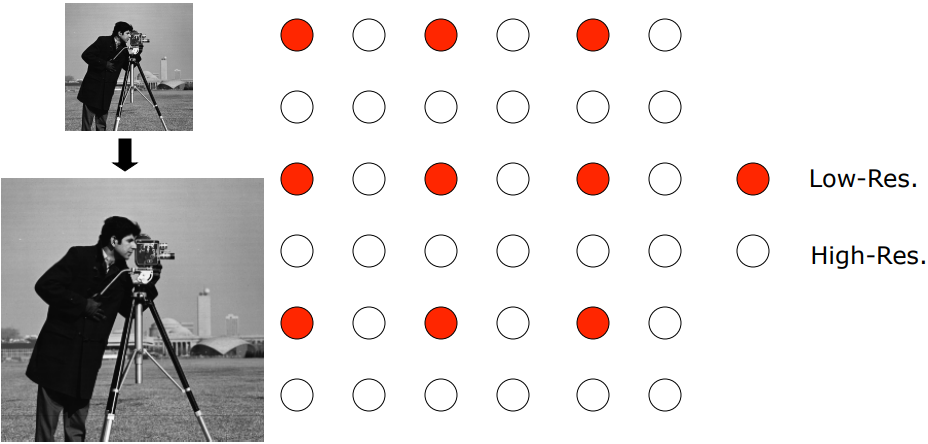
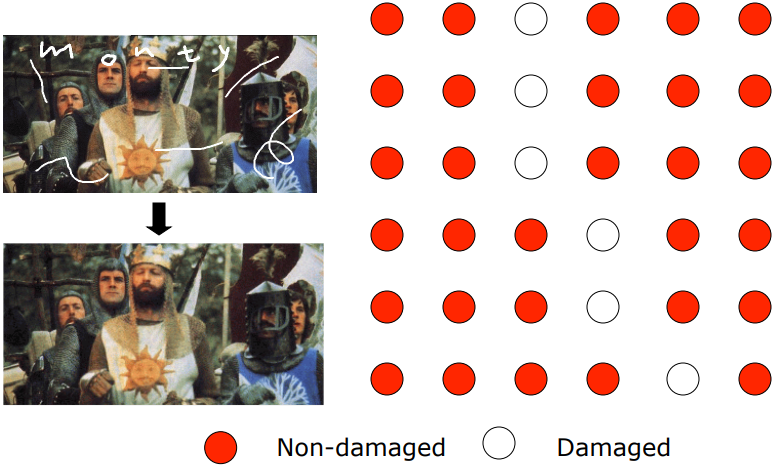
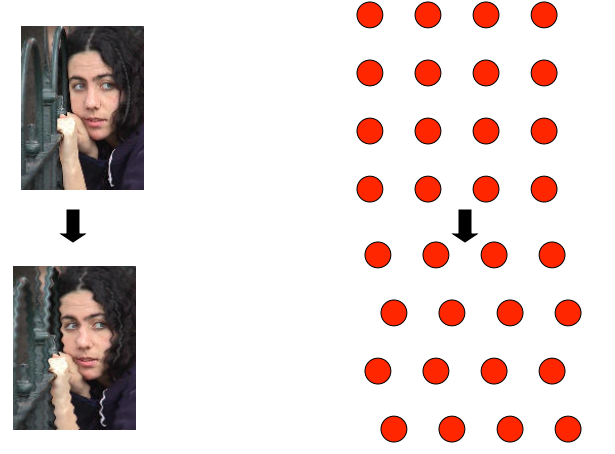
### Interpolation

* 영상을 Upsamling 할 때, 발생되는 Row와 Colume에는 실제 data가 제공되지 않는다. 이렇게 발생되는 Row와 Colume에 Data를 발생시켜 주기 위해 Interpolate을 적용하여 합리적인 Data로 채워준다.  
  

### Interpolation vs Approximation

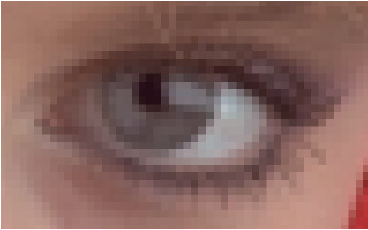
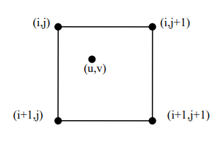
* Interpolation
  + 모든 특정 점들을 지나간다.
  + 더 정확하지만, 안정적이지 않다.  
    
* Approximation
  + 모든 점들을 지나가지 않는다.
  + 얻어 내기 편하다.  
    

### 사용분야

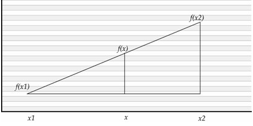
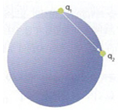
* 해상도 향상(Resolution Enhancement)  
  
* 영상 수리(Image Inpainting)  
  
* 이미지 뒤틀림(Image warping)  
  

## Interpolation 종류

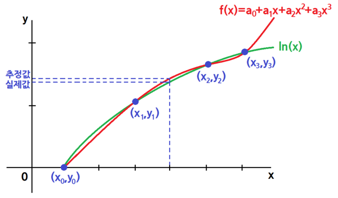
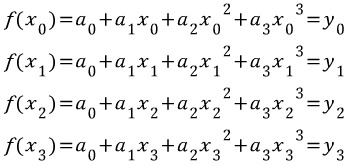
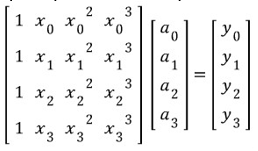
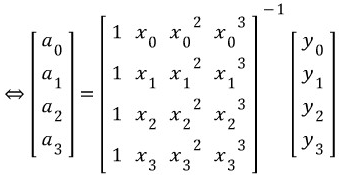
### Nearest Neighbor Interpolation

* 가장 간단한 보간법이다.
* 출력 픽셀 값은 입력 영상의 가장 가까운 샘플 값으로 할당된다.
* 해당 방법은 가장 간단하지만, 다른 이웃들을 고려하지 않으므로 품질이 떨어진다.  
  
* 아래와 같은 상황해서 (u, v)의 pixel 값은 (I, j)의 값으로 채워진다.  
  

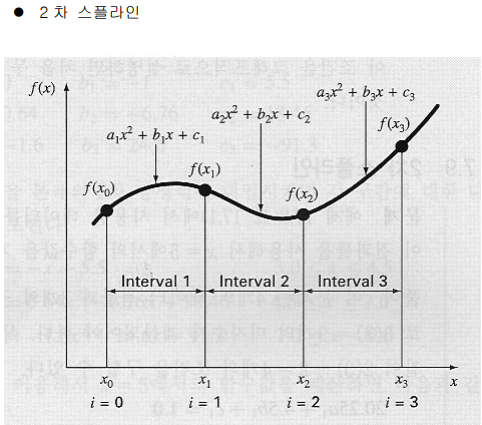
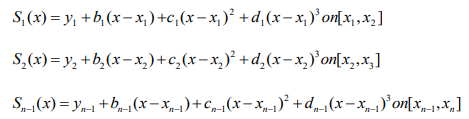
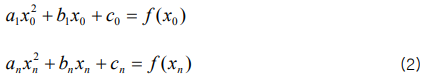
### Linear Interpolation

* 두 점의 값이 주어졌을 때, 그 사이에 위치한 값을 추정하기 위하여, 직선거리에 따라 선형적으로 계산하는 방법 이다.
* 두 점 (x1, f(x)), (x2, f(x2)) 사이에 위치한 (x,y)의 값을 추정하기 위해 다음과 같은 비례식을 구성한다  
    
    
  
* 허나, 다음과 같은 구면에서 q1, q2를 선형 보간 할 경우, 매끄럽지 않은 Linear interpolation이 이루어 진다.  
  

### Cubic Interpolation (= 3rd order polynomial interpolation)

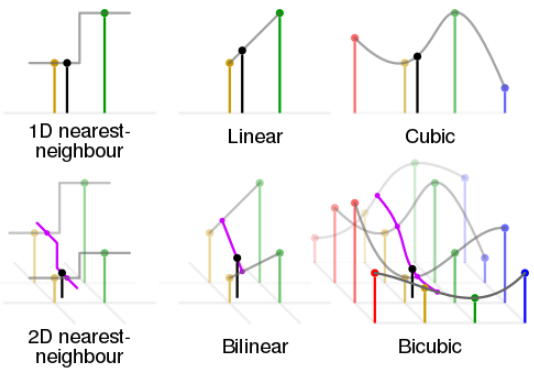
* 3rd order polynomial interpolation 이라고 불리며, 3차 다항식을 이용한 interpolatio이다.
* 4개의 Data가 주어졌을 때 사용 할 수 있다.
* 아래 그림은 Cubic interpolation을 요약한 그림이다.  
  
* 다음과 같은 3차 다항식의 일반식을 활용한다.  
  
* 일반식에 4점 (x0, y0), (x1, y1), (x2, y2), (x3, y3)을 지나도록 일반식에 대입한다.  
  
* 4개의 연립방정식을 행렬로 나타내면 다음과 같다.  
    
  

### Cubic spline interpolation

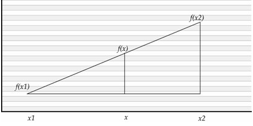
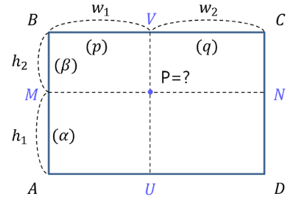
* 하나의 함수로 표현하는 Polynomial interpolation과 달리 각 구간마다의 추정함수를 구하는 것.
* Spline은 이웃한 구간 함수 경계점의 값뿐 아니라, 미분 값까지 일치하게 조절한다.
* Spline은 원본에 좀더 비슷한 결과를 만들어 내므로, Polynomial interpolation 보다 선호된다.
* 아래 그림은 Cubic spline interpolation을 요약한 그림이다.  
  
* 다음과 같은 일반식을 사용한다.  
  
* Spline의 조건은 다음과 같다.
* 내부 접하는 알려진 점에서 이웃하는 다항식들의 함수 값이 같아야 한다.  
  
* 첫 번째와 마지막 함수는 반드시 끝점을 통과해야 한다.  
  
* 내부 접하는 알려진 점에서 이웃한 다항식들의 도함수는 같아야 한다.  
  
* 2차 도함수가 첫 번째 데이터 점에서 0이라고 가정한다.  
  

## 2D interpolation

### 개요

* 1차원 공간에서의 Interpolation을 이용하여 2차원 공간에서 Interpolation하는 방법으로 다음의 그림을 보면 쉽게 이해 할 수 있다.   
  

### Bilinear Interpolation

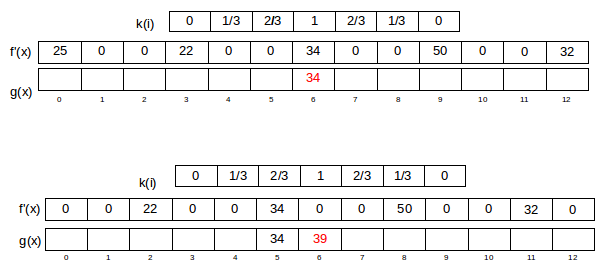
* Linear interpolation을 이용하여, 새로운 Pixel 값을 계산한다.
  + Linear interpolation 이란?
    - 두 점의 값이 주어졌을 때, 그 사이에 위치한 값을 추정하기 위하여, 직선거리에 따라 선형적으로 계산하는 방법 이다.
    - 두 점 (x1, f(x)), (x2, f(x2)) 사이에 위치한 (x,y)의 값을 추정하기 위해 다음과 같은 비례식을 구성한다  
        
        
      
* Upsampling 시, 새롭게 생성되는 Pixel을 둘러싸고 있는 2x2 Pixel data를 이용하여 계산한다.
* 1차원 Linear interpolation을 2차원으로 확장한 것으로 4개의 가까운 Pixel의 가중 평균값을 구한다.
* Pixel 당 Linear interpolation을 3번 수행하며, 가장 가까운 화소 4개에 가중치를 곱한 값을 합해서 얻게 된다.
* 다음과 같은 그림에서의 상세 동작은 다음과 같다.  
  
  + A와 B를 Linear interpolation 하여, M을 얻는다.
  + C와 D를 Linear interpolation 하여, N을 구한다.
  + M과 N을 Linear interpolation 하여, P를 구한다.

### Bicubic Interpolation

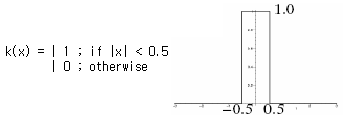
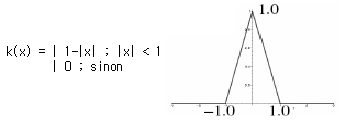
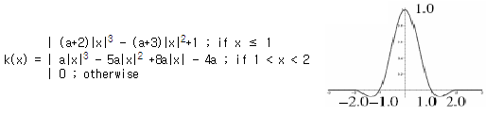
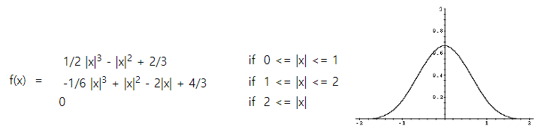
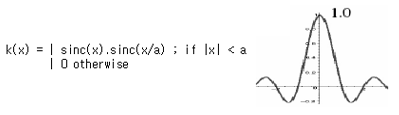
* Cubic interpolation혹은 Cubic spline interpolation을 이용하여, 새로운 Pixel 값을 계산한다.
* Upsampling 시, 새롭게 생성되는 Pixel을 둘러싸고 있는 4x4 Pixel data를 이용하여 계산한다.
* Bilinear interpolation 보다 더 선명하고 부드러운 영상을 생성한다.
* 추가 Parameter를 가지고, Resampling과정 동안 영상을 더 선명하게 할 수 있다.  
  

## Kernel & Convolution을 이용한 Interpolation

### 개요

* X축과 Y축이 존재하는 2D Interpolation에서 Convolution을 이용하여 Interpolation을 할 수 있다.  
  
* Convolution Kernel이란?
  + 입력영상에 비해 아주 작은 행렬로, 영상의 각 화소에 대해 커널의 중심을 처리 대상의 화소 위에 올려 놓고, 커널 행렬의 각 값과 대응 되는 화소의 값들을 각각 곱하고 이를 모두 더해 원하는 Pixel 값을 얻을 수 있다.
  + 위 함수에서 k(x)가 Convolution kernel에 해당 한다.
  + 예) Kernel k(x)가 [1/3, 2/3, 1, 2/3, 1/3] 일 때, Interpolation 된 값(g(x))를 다음과 같이 구할 수 있다.  
    

### 종류

* Nearest Neighbor Resampling  
  
* Bilinear Resampling  
  
* Bell Resampling   
  
* Hermite Resampling  
  
* Bicubic Resampling  
  
* B-Spline Resampling  
  
* Mitchell Resampling
  + B(B-spline curve)와 C(Cardinal curve)로 알려진 두 변수로 공식화 한 필터를 사용
  + 일반적으로 B= 1/3, C=1/3을 사용한다.  
    
* Catmull-Rom-Spline Resampling
  + Mitchell resampling과 같은 kernel을 사용하며, B=0, C=0.5를 사용한다.
* Lanczos Resampling
  + Parameter a=3이 일반적으로 사용된다.  
    

## 결과

: Gdi+로 Interpolation을 수행한 영상의 품질과 속도를 비교해보자.

### 품질 비교

* 자체 구현

|  |  |
| --- | --- |
| Bilinear |  |
| Bicubic | (a = -1.0) |

* Gdi+ 구현

|  |  |
| --- | --- |
| Bilinear |  |
| Bicubic | (Parameter 선택 없음) |

### 속도 비교

* 자체 구현

|  |  |
| --- | --- |
| Bilinear |  |
| Bicubic |  |

* Gdi+ 구현

|  |  |
| --- | --- |
| Bilinear |  |
| Bicubic |  |

# 참고 문헌 및 자료

## 수치해석

* https://blog.naver.com/mykepzzang/220546694965

## Kernel & Convolution을 이용한 Interpolation

* http://www.ijera.com/papers/Vol4\_issue10/Part%20-%201/K41007073.pdf
* <http://teaching.csse.uwa.edu.au/units/CITS4241/Handouts/Lecture04.html>
* https://clouard.users.greyc.fr/Pantheon/experiments/rescaling/index-en.html

Vieworks Co., Ltd

41-3, Burim-ro, 170beon-gil,

Dongan-gu, Anyang-si, Gyeonggi-do

431-060 Republic of Korea

<http://www.vieworks.com>