HTML5 Canvas 기반 이미지 편집 온라인 솔루션

2017103951 고성훈

2017104001 심건우

2017104016 이주형

2017104030 주광명

요약

HTML5에서 추가된 Canvas API를 활용해 웹에서 사용할 수 있는 이미지 에디터를 만들고자 한다. 그동안 웹에서는 성능상의 한계로 간단한 이미지 편집 기술만 사용할 수 있는 에디터만 존재했다. Thresholding과 Sharpening등을 활용하여 그림자 제거와 객체 보정 기능이 추가된 이미지 에디터를 만들고, 이를 다운로드 받지 않고 웹에서 바로 사용할 수 있도록 솔루션을 제공하고자 한다.

1. 서론

1.1 연구배경

이미지 에디터(Image Editor)는 다양한 이미지 편집 기능을 제공하는 프로그램이다. 이미지 에디터에는 Adobe Photoshop, Microsoft Expression, Pixlr, Paint.NET 등이 있으며 일반적으로 캔버스 사이즈 편집, 오브젝트 클릭 및 선택, 객체별 Layer 나누기, undo / redo 등의 기능을 제공한다.

HTML 5 등장 이전에는 웹에서 그래픽을 표현하기가 쉽지 않았다. 웹이 처음 등장했을 때는 문서 위주의 역할을 하였기 때문에 그래픽 표준이 정의되지 않았고 따라서 멀티미디어를 보여주는 데에는 한계가 있었다. 그래서 웹 이미지 에디터에서 사용할 수 있는 이미지 편집

기능이 다양하지 않았고 고도화되지 못했다. Fotor, Canva, PicMokey 등이 그 예시로, 이 웹 이미지 에디터들에서는 밝기 및 명암 조절, 텍스트 추가, 이미지 회전 및 자르기, 필터 효과 주기 등등의 기본적인 이미지 편집 기능만을 제공한다. 간단한 기능 이상의 이미지 편집을 하고 싶을 때는 다른 이미지 에디터 프로그램들을 다운로드 받아 사용해야 했고, 이는 사용자에게 번거로움을 불러왔다.

웹에서도 고급 그래픽을 표현하고 싶은 수요 증가에 따라 그래픽을 전문적으로 표현하는 HTML5의 Canvas API가 등장하고 상용화되었다. 덕분에 웹에서도 다양한 그래픽 기술을 처리할 수 있게 되었다.

Canvas API는 JavaScript 및 HTML <canvas> 요소를 통해 애니메이션, 게임 그래픽, 데이터 시각화, 사진 조작 및 실시간 비디오 처리등의 컴퓨터 그래픽을 그리는 수단을 제공한다. Canvas API는 주로 2D 그래픽에 사용이 되고, <canvas> 요소를 같이 사용하는 WebGI API는 3D 그래픽에 사용된다. Canvas API는 매우 강력하지만 사용하기 쉽기는 않기 때문에, 라이브러리를 사용하면 프로젝트를 더 빠르고 쉽게 만들 수 있다.

HTML5의 Canvas API가 등장함에 따라 웹에서 다양한 그래픽을 표현할 수 있게 되었지만, 그에 걸맞은 웹 이미지 에디터는 아직 나오지 않은 상황이다. 따라서 Canvas API를 응용해 기본적인 이미지 편집 기능을 제공하고 새로운 기능인 그림자 제거 기능과 보정 기능을 제공하는 웹 이미지 에디터를 만들고자 한다.

1.2 연구목표

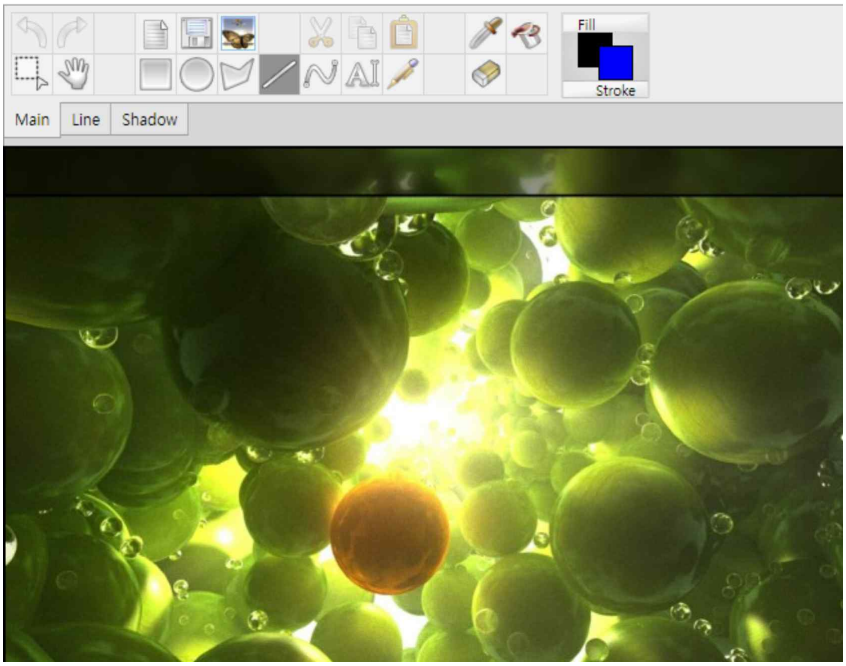
Canvas API를 기반으로 기존의 다른 이미지 에디터에 있는 기능을 모두 구현하면서도 필요로 하는 신규 기능을 개발하고자 한다. 또한 WYSIWYC(What You See Is What You Get) 방식을 통해 사용자가 이미지 에디터를 사용하기 편하게 하고, 에디터 편집을 통해 얻은 화면이 출력물과 동일하게 되도록 개발한다. 기존의 웹 이미지 에디터에서 제공하는 이미지 편집 기능을 모두 제공하는 동시에, 고유한 알고리즘을 적용시켜 본 프로젝트에서 만든 이

미지 에디터에서만 사용할 수 있는 차별화된 고유한 기능을 제공한다.

2. 관련연구

2.1 이미지 편집 웹사이트

2.1.1 PaintWeb



[그림 1] PaintWeb

윈도우의 그림판과 비슷한 기능을 하는 HTML5 기반의 Javascript Image Editor이다. 데모 버전으로 웹사이트(https://mihai.sucan.ro/paintweb/trunk/demos/demo1.html)에서 확인이 가능하며, 깃허브(https://github.com/gujc71/imageEditor)에서 소스를 다운 받아 사용할 수 있다.

2.1.2 Pixlr

별도의 소프트웨어 설치 없이 웹 브라우저에서 바로 사용할 수 있는 이미지 편집 그래픽 툴이다. 과거에는 Adobe Flash로 만들어졌으나, Adobe Flash가 서비스를 종료함에 따라

HTML5 기반의 후속 버전을 출시하였다. Adobe Photoshop과 유사한 기능을 제공하고, 다양한 언어 또한 지원한다.

2.2 라이브러리

Canvas API를 활용하는 라이브러리들에 대해 서술한다. Canvas API는 기본적으로 제공하는 추상화 단계가 낮기 때문에, 이를 직접 사용하기에는 어렵고 비용을 고려하면 비효율적이다. 따라서 Canvas API를 직접 사용하기보다는 라이브러리를 사용하는 것이 효율적이다.

2.2.1 EaselJS

EaselJS는 HTML5에서 고성능 인터랙티브 2D 콘텐츠를 구축하기 위한 라이브러리이다. 그래픽 조작 및 애니메이션 기능을 록을 제공한다. 또한 마우스 및 터치 상호 작용을 위한 강력한 인터랙티브 모델을 제공한다.

EaselJS는 게임, 생성 예술, 광고, 데이터 시각화 및 기타 고도의 그래픽 경험을 구축하는데 탁월하다. 외부 종속성이 없으며 사용하는 거의 모든 프레임워크와 호환된다.

2.2.2 Paper.js

Paper.js는 HTML5 Canvas 위에서 실행되는 오픈 소스 벡터 그래픽 스크립팅 프레임워크이다. 깔끔한 장면 그래프/문서 개체 모델과 벡터 그래픽 및 베지어 곡선을 만들고 작업할 수 있는 기능을 제공한다. 디자인이 일관성 있고, 프로그래밍 인터페이스가 깔끔하다.

2.2.3 Fabric.js

Fabric 라이브러리는 동일한 그래픽을 구현함에 있어 단순한 방법을 지원하며, 객체 집합이나 사용자 인터랙션을 제공한다. Fabric에서 객체는 각각이 독자적으로 존재하지 않고, 계층 구조를 형성하고 있기 때문에 메서드를 정의하고, 이를 모든 자식 클래스에서 공유할 수

있다.

3. 프로젝트 내용

3.1 알고리즘

3.1.1 Sharpening

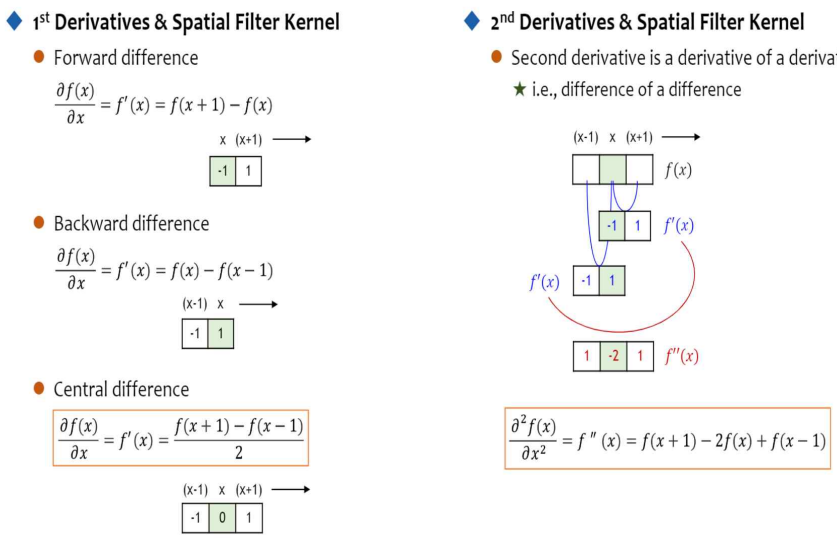
Sharpening이란 이미지의 초점이 잘 맞는 사진처럼 객체의 윤곽이 뚜렷하게 구분되도록 하는 것을 말한다. 객체를 구별할 수 있도록 보정해주는 과정에는 라플라시안 필터를 이용한다. 라플라시안 필터는 2개 미분으로 이루어진 마스크이다. 이미지에서의 엣지는 값이 급격하게 변하는 부분으로 고주파 성질을 가지고 있기 때문에 고주파 필터인 라플라시안 필터를 활용한다.

$$Laplace(f) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

[수식 1] 라플라시안 방정식

라플라시안 방정식은 각 x의 픽셀값의 2개 미분과 y의 픽셀값의 2개 미분의 합으로 구성된다. 특정 상황을 예시로 들어 1차 미분과 2차 미분의 차이를 설명하면 다음과 같다. 먼저 값이 변하지 않는 부분인 Constant intensity에서는 1차 미분 후에도 값이 변하지 않으므로 기울기 값은 0이고 2차 미분에서 기울기의 변화량이 0이므로 2차 미분 값 또한 0이다. 한편 값이 변하는 부분인 Intensity translation에서는 1차 미분에서는 값이 변하여 6에서 5로 즉, 기울기의 값은 –1이고 2차 미분에서 기울기의 변화량이 -1이므로 2차 미분 값 –1이다. 값이 일정하게 변하는 부분 Ramp에서는 1차 미분은 값이 일정하게 변하고 있음, 기울기의 값은 –1이고, 2차 미분에서 기울기의 값은 -1로 일정하기 때문에 2차 미분 값은 0이다. 값이 갑자기 확 튀는 부분인 Step에서 1차 미분에서는 값이 확 튀어서 기울기 값이 6까지 올라가고 2차

미분에서는 기울기 변화가 0에서 6까지 일어나기 때문에 2차 미분 값 또한 6이다. 마지막으로 Step 구간 다음에서는 1차 미분은 다음 값들이 일정한 기울기를 가지므로 1차 미분 값 0이고, 2차 미분은 기울기의 변화량 6에서 0으로 내려간 것이므로 2차 미분 값은 –6이다. 즉, 2차 미분인 경우 밝기의 변화가 큰 값에는 +에서 -로 가는 Zero crossing이 발생한다.



[그림 2] 1차 미분과 2차 미분

이것을 이용하면 밝기의 변화가 더 큰 부분들을 찾을 수 있다. 이렇게 엣지 부분 픽셀만 찾아 어떠한 값을 더해주면 엣지가 강조되는 효과를 얻는 방법으로 Sharpening 기술을 적용할 수 있다.

3.1.2 Lens Distortion

[그림 3] 굴곡을 적용시킨 이미지

x좌표, y좌표에 sin, cos 함수를 적용하여 새로운 사인파 코사인파 곡선으로 왜곡된 이미지를 만들 수 있다. 외곽 보정까지 해주어 외곽의 사라진 영역을 보정할 수 있다. exp는 이미지의 왜곡 지수를 나타내는 변수로, 1이면 원본과 동일하게 하고 1보다 작으면 오목 렌즈 효과를 내고 1보다 크면 볼록 렌즈 효과를 낸다. 위 그림 3은 exp=0.5로 설정한 결과이다. scale은 이미지에서 렌즈 효과를 주고 싶은 원 모양 영역의 크기를 비율로 나타낸 것이다. scale=1은 100%를 의미한다.

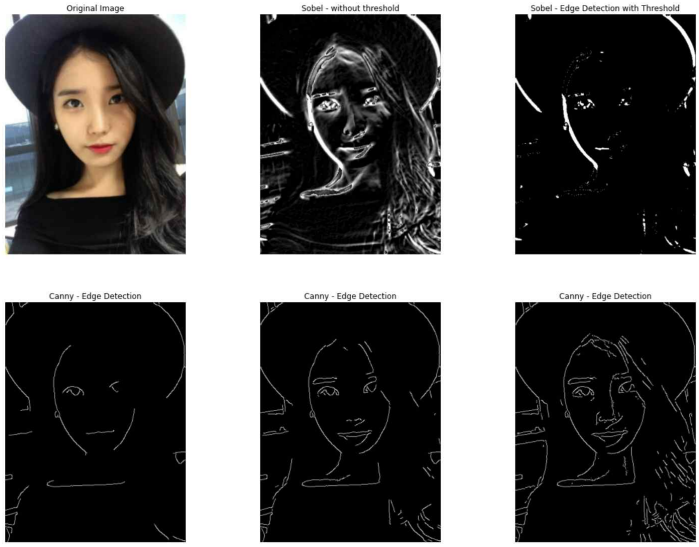
3.1.3 Edge Detecting

미분을 이용한 라플라시안 필터를 마찬가지로 이용한다. 라플라시안 모양이 다양한데, 1차 미분을 쓰냐 2차 미분을 쓰냐에 따라 나뉜다. 1차 미분은 edge가 시작하는 부분과 끝나는 부분이 굵게 나오는데 2차 미분은 edge가 시작하는 부분과 끝나는 부분이 더 얇기 때문에 엣지 또한 더 얇게 검출할 수 있다.

robert 필터는 대각선 검출에 특화되어 있고, 프리윅 필터는 y, x에 대한 엣지 검출에 특화되어 있다. 이 둘은 1차 미분의 값을 근거로 한다. 소벨 필터는 가운데 미분값에 더 가중치를 둔다. 소벨 필터는 더 부드럽게 엣지를 검출하는 효과가 있다. 프리윅 필터의 단점은 엣지가 아닌 부분까지 검출이 되기 때문에, 엣지 부분이 두꺼운 편이다. 로버트 필터의 단점은 일부 그림에서는 엣지가 잘 검출되는 것 같지만 인물과 같은 경우에서는 유실되는 정보가 많아 엣지가 잘 검출안되는 것을 볼 수 있다. 소벨은 프리윅과 마스크 모양 거의 같지만 업그레이드 형태이다. 가운데 값만 2배 선이 좀 두껍게 나타나고 검출 안 되어야 하는 부분까지 나타나기 때문에 쓰레쉬홀드를 사용한다. 이 때 유실되는 정보 많아지는 단점이 존재한다.

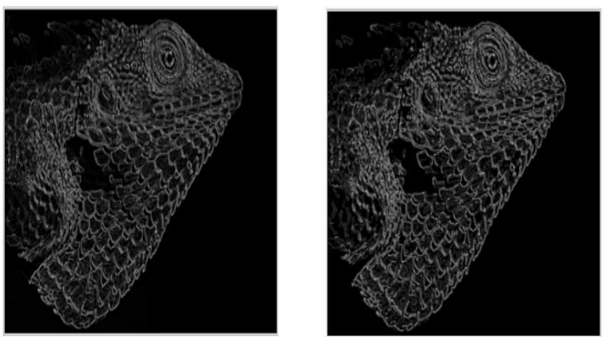
앞서 언급한 필터들을 업그레이드한게 캐니 엣지 디텍터이다. 캐니 엣지 디텍터는 2차 미분 값을 이용하는데, 이를 통해 magnitude값과 vector값을 구할 수 있다. 파란색 픽셀 기준

으로 2차 미분 시켜 구한 방향을 바탕으로 파란색이 속해있는 방향을 다 조사해 빨간색 픽셀이 다른 픽셀들 보다 크면 남겨두고, 어느 하나라도 작으면 0으로 엣지가 아니기 때문에 0으로 만들어버린다. 이 때 엣지는 주변 픽셀보다 크다.



[그림 4] 여러가지 필터들을 사용해 Edge Detecting을 수행한 모습

모든 픽셀을 처리해 준 후에, 더블 쓰레쉬홀딩사용한다. 두꺼비 가운데에 희미한 선들이 있는데 이것들을 더블 쓰레쉬홀등으로 없애줄 수 있다. 이 때 엣지를 두 개로 나누어주는 방법을 사용하는데, 값이 쎄면 strong, 애매하거나 약하면 weak로 분류한다. strong 엣지가 포함된 weak 엣지들은 엣지로 보고, 여기에 포함이 안 되면 바깥에 있는 것이므로 엣지로 보지 않는다. 캐니 엣지 디텍터에는 4, 8 두가지 값을 구현하였는데, 8은 주변 8픽셀, 4는 주변 4픽셀을 검출한다. 본 프로젝트에서 사용한 값은 8픽셀짜리임을 밝힌다.



[그림 5] 더블 쓰레쉬홀딩을 사용하기 전(좌)와 사용한 후(우)의 이미지 비교

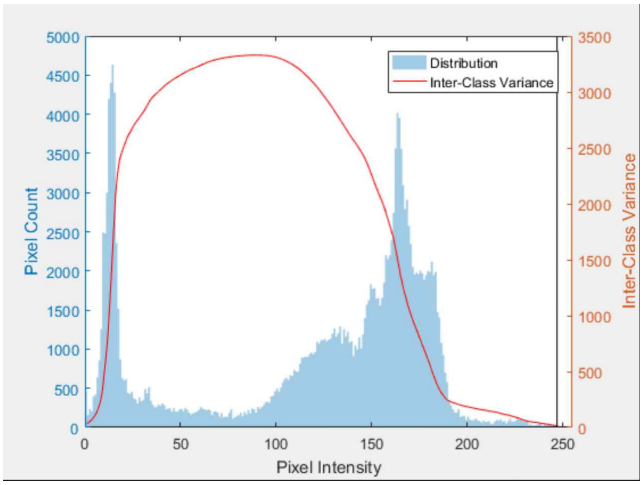
3.1.4 Otsu's global thresholding

Thresholding 어떠한 픽셀의 기준값으로 두 개의 클래스로 나눠서 한 클래스는 어떤 변화를 주고 다른 클래스는 다른 변화를 주어 두 클래스를 구분하게 만드는 방법이다[2]. 여기서 Thresholding의 기준값을 전체 픽셀의 평균값으로 정할 수 있지만 그렇게 되면 픽셀 분포가 몰려있을 때 문제가 발생하게 된다. Otsu's Method 알고리즘은 Between-Class variance의제곱을 이용한 방법이다. Between-Class variance란 히스토그램의 분포에서 어떤 상수 k를 기준으로 할 때 클래스 간에 떨어져 있는 거리의 제곱을 말한다. 클래스는 k라는 픽셀 기준으로 나누었을 때 나누어지는 두 개를 말한다. Within-class는 각 클래스들의 평균적으로 얼마나퍼져있느냐를 말하는 것이다. 이 클래스들을 이용한Between-Class variance의 수식은 다음과 같다.

$$\sigma_B^2(k) = P_1(k)(m_1(k) - m_G)^2 + P_2(k)(m_2(k) - m_G)^2$$
$$= \sigma_G^2 - \sigma_W^2(k)$$

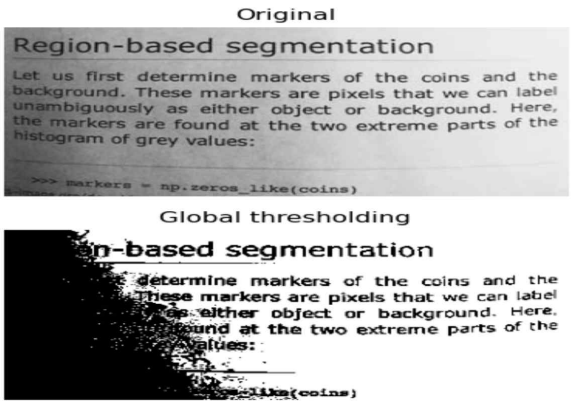
[수식 2] Between-Class variance

Between-Class variance는 전체 variance에서 within-class variance를 빼서 만들기 때문에 클래스 간 거리와 각 클래스의 퍼져있는 거리를 둘 다 고려하게 된다. 따라서 thresholding의 기준을 정할 때 전체 평균의 픽셀값으로 잡는 것보다 훨씬 정확하다.



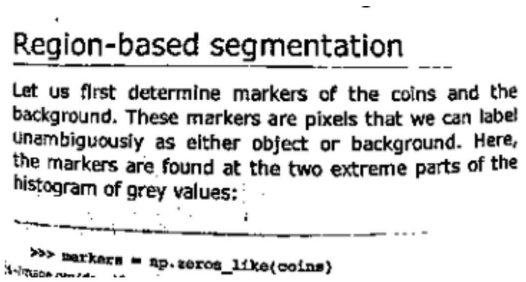
[그림 6] 각 픽셀에 대한 Between-Class variance

위와 같이 각 픽셀에 대해 Between-Class variance를 구해주면 빨간 선과 같은 모양이 그려지게 된다. Between-Class variance의 값이 최대가 된다는 것은 클래스 간 거리가 가장 큰 값이므로 위 그래프에서는 픽셀값 100 정도에서 가장 잘 분류가 된다고 볼 수 있다. 하지만 클래스를 잘 나누어도 가장 그림자가 진 부분은 배경이 워낙 어둡기 때문에 글자와 같은 클래스로 분류가 된다. 따라서 Otsu's global thresholding 만으로는 그림자를 제거할 수 없다는 문제점이 있어 Local thresholding을 적용해주어야 한다.



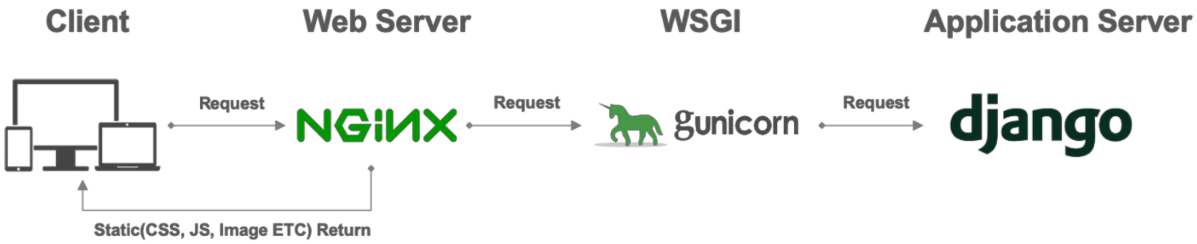
[그림 7] Otsu's global thresholding을 적용시킨 이미지

local thresholding이란 전체 이미지를 기준으로 thresholding을 하는 것이 아니라 전체 이미지를 나누어서 각 파티션 별로 thresholding을 해주는 것이다. 이때 임의의 범위인 x * y로 파티션으로 나누는 것을 윈도우 사이즈라고 한다. 윈도우 사이즈를 정할 때 서면 자료인 이미지의 경우에는 이를 구분하고자 하는 문자의 평균 획과 너비의 5배를 이상으로 해주는 것이 좋다. 이렇게 파티션 별로 나누어 그 파티션 안에서 Otsu's Method를 활용하여 정확한 기준을 구해 thresholding 한다면 그림자를 보다 명확하게 제거할 수 있다.



[그림 8] Local Thresholding을 적용시킨 이미지

3.2 프로젝트 전체 구조



[그림 9] 프로젝트 전체 구조 순서도

프로젝트 전체 구조는 다음과 같다. 클라이언트와 웹 서버가 통신하는 방식이며, 클라이언트에서 NGINX에 요청을 하면 CSS, JS Image를 건네주는 방식이다. NGINX는 다시 GUICORN에게 요청하고 django가 비전 알고리즘을 처리하는 API는 따로 서버를 만들었고, NGINX 없이 GUNICORN으로만 처리한다.

메인 서버의 경우 통합 파이썬 웹 프레임워크인(프론트엔드의 템플릿도 사용가능한) 장고를 선택했다. webserver는 정적파일 처리도 가능하고 동시접속 처리도 가능케 하기 위해 nginx를 사용했고, Web Server가 요청에 대한 정보를 Application에 전달하기 위한 위스기미들웨어로는 구니콘 라이브러리 사용했다.

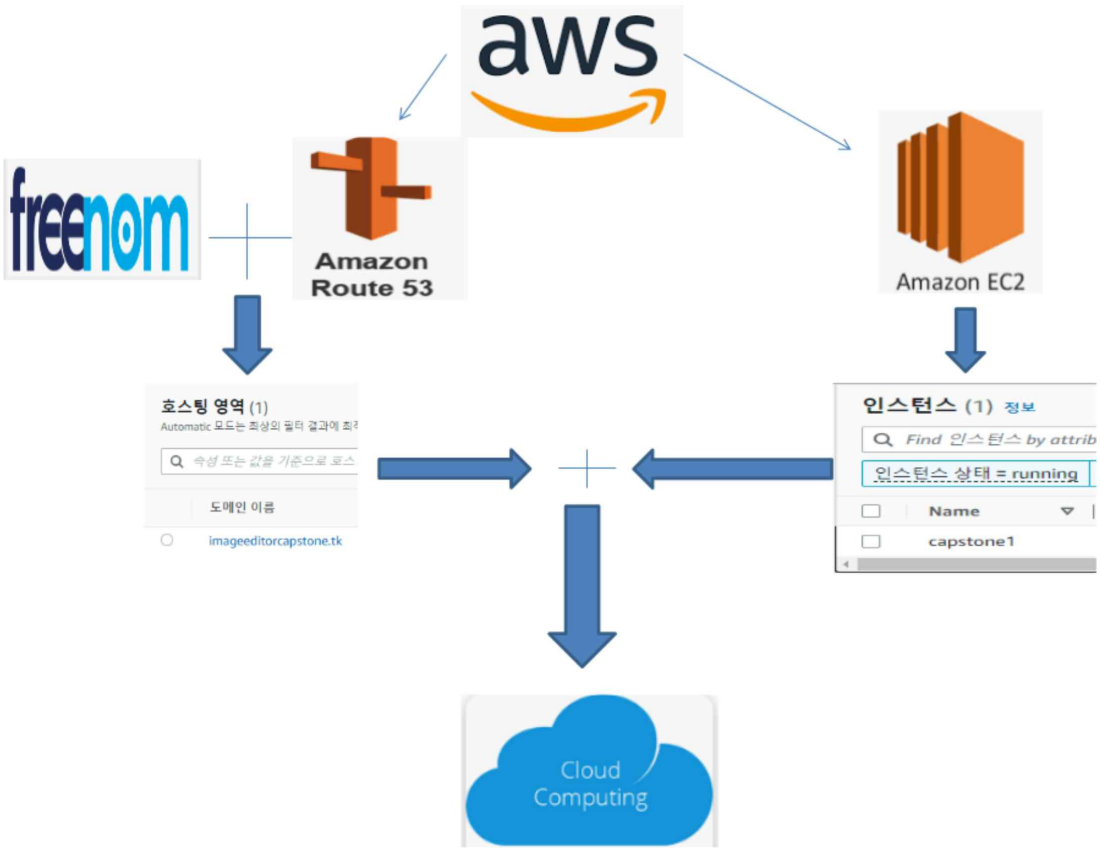
웹 서버는 아파치와 uwsgi등 여러 선택지가 있었으나 장고와 함께 가장 많이 쓰이는 웹 서버가 nginx, 구니콘이기 때문에 채택했다. 컴퓨터비전 알고리즘들을 계산하기 위한 API용 서버를 별도로 하나 더 두고 처리하기로 결정하였다.

API용 서버에서는 정적파일 처리가 필요 없기에 nginx 사용없이 gunicorn만으로 서버를 구축하였다.

프레임워크의 경우 fastAPI를 사용하기로 하였다. fastAPI는 스웨거 문서도 자동으로 작성해줄 뿐만 아니라 API구현에는 다른 대표적인 파이썬 프레임워크인 장고와 플라스크에 비해 훨씬 빠르기에 우선적으로 선택되었다. fastAPI스웨거 제일 위의 2개는 test용 api이다.

아래 5개의 api들은 post형식이고 호출하게 되면 각각의 알고리즘들이 이미지에 계산되어

서버에 남게되고 계산이 다 끝나게 되면 클라이언트에서 가장 아래의 get 메소드 api를 호출해 클라이언트가 서버에서 알고리즘이 적용된 이미지를 가져간다.

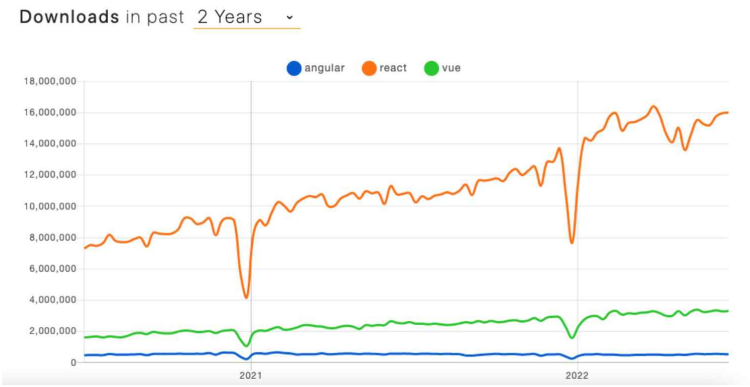


[그림 10] 프로젝트 라우드 상황

라우드 상황은 위 그림과 같다. AWS를 통해서 EC2 우분투 인스턴스 생성해서 클라우드 컴퓨팅을 통해 서버 컴퓨터를 만들었다. 사용자가 쉽게 접근할 수 있도록 freenom에서 도메인을 할당 받아 Route53을 통해 퍼블릭 ip와 연결해주어 도메인 이름으로 웹사이트 접근 가능하도록 하였다.

프론트엔드 라이브러리 선정과정은 다음과 같다. Angular는 Typescript를 지향하기 때문에 타입 안정성을 보장하지만, 사용자수가 가장 적기 때문에 관련 라이브러리가 많이 없고, 가장 배우기 어렵다. Vue는 가장 최근에 나온 프레임워크이기 때문에, Angular와 React의 장점

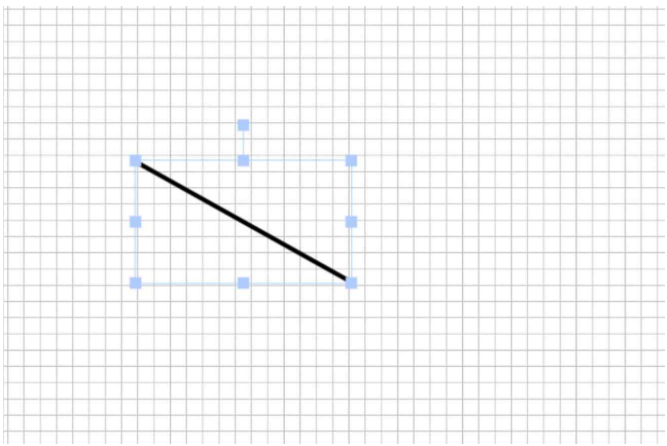
들을 어느 정도 적절히 가져왔다. 양방향 데이터 바인딩이 가능하며 개발 난이도가 다른 프레임워크에 비해 쉽다. 하지만 가장 늦게 배포되었기 때문에 관련 자료가 적다. React는 다른 프레임워크들에 비해 압도적인 사용량을 보여주고 있고, 그렇기 때문에 관련 자료도 가장 많고 또한 컴포넌트 기반이기 때문에 프로젝트에 가장 적절하다고 판단하여 적용하였다.



[그림 11] 프론트엔드 라이브러리 선정 과정

3.3 이미지 에디터

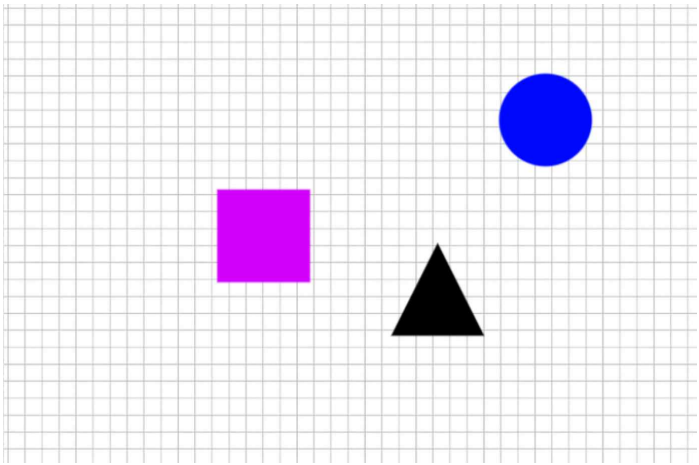
기본적인 이미지 편집 기능으로 그림 선 긋기, 도형, 반전 및 감마 효과를 줄 수 있게 리액트 환경에서 구현하였다.



[그림 12] 그림 선 긋기 기능 제공

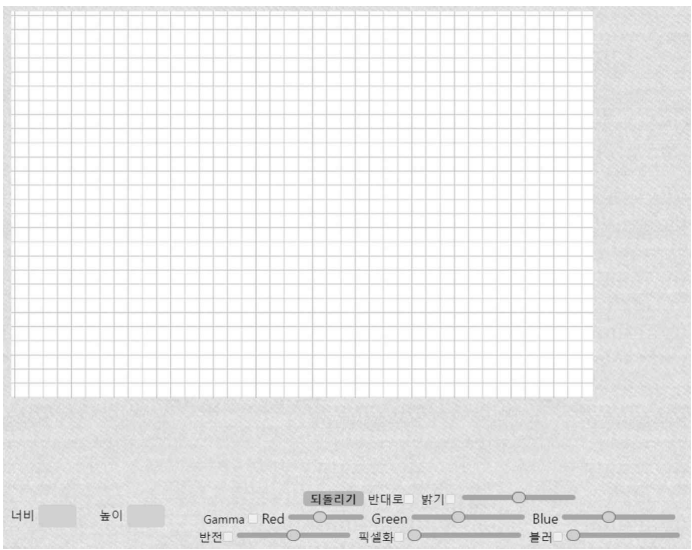
직선 및 곡선을 그을 수 있는 기본 기능을 제공한다. 선 굵기 뿐만 아니라 색깔을 수정할

수 있으며 이를 객체로 저장해 놓기 때문에 작업 취소 및 되돌리기, 추후에 다시 수정하기 등의 기능을 사용할 수 있다.



[그림 13] 다양한 도형을 그릴 수 있는 기능 제공

선과 마찬가지로 도형을 삽입하고 수정하는 기능 또한 제공한다. Circle, Triangle, Rectangle 등의 도형을 크기, 색깔에 맞춰 이미지 에디터에 넣을 수 있다.



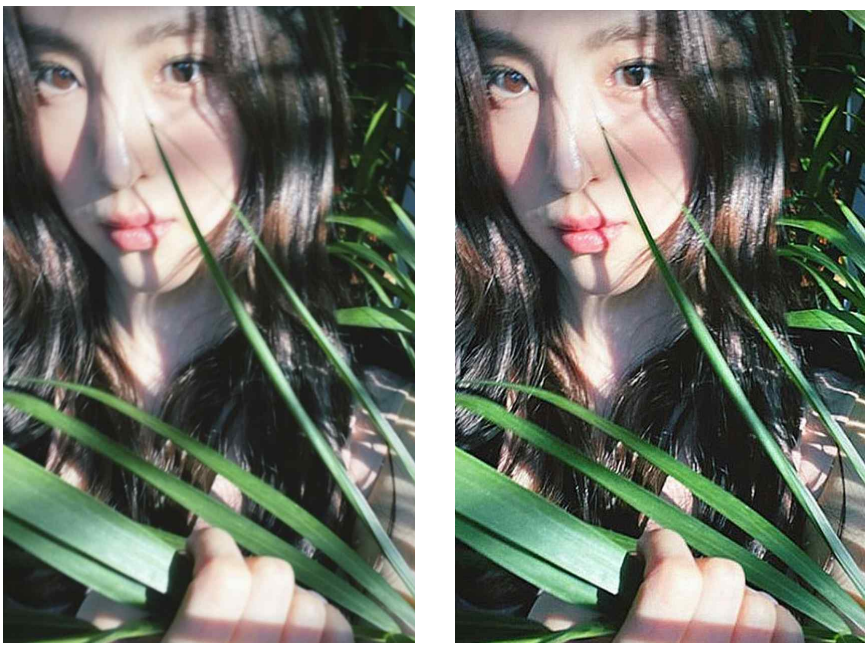
[그림 14] 반전, 픽셀화, 감마 효과를 줄 수 있는 탭

그 외에도 반전 효과, 픽셀화, 감마 효과의 정도를 파라미터로 조절하여 적용시킬 수 있는 탭을 구현하였다.

4. 프로젝트 결과

[그림 15] 블러링을 적용하기 전의 이미지(좌)와 적용한 후의 이미지(우)

풍경사진에 블러링을 적용시킨 모습이다. 컬러 사진에도 적용이 문제 없으며 화질이 더욱 선명해지고 깔끔해진 것을 볼 수 있다.



[그림 16] 블러링을 적용하기 전의 이미지(좌)와 적용한 후의 이미지(우)

또한 사람과 같은 이미지에도 잘 적용이 되는 것을 위의 비교에서 볼 수 있다.

[그림 17] 사인 효과를 적용한 이미지

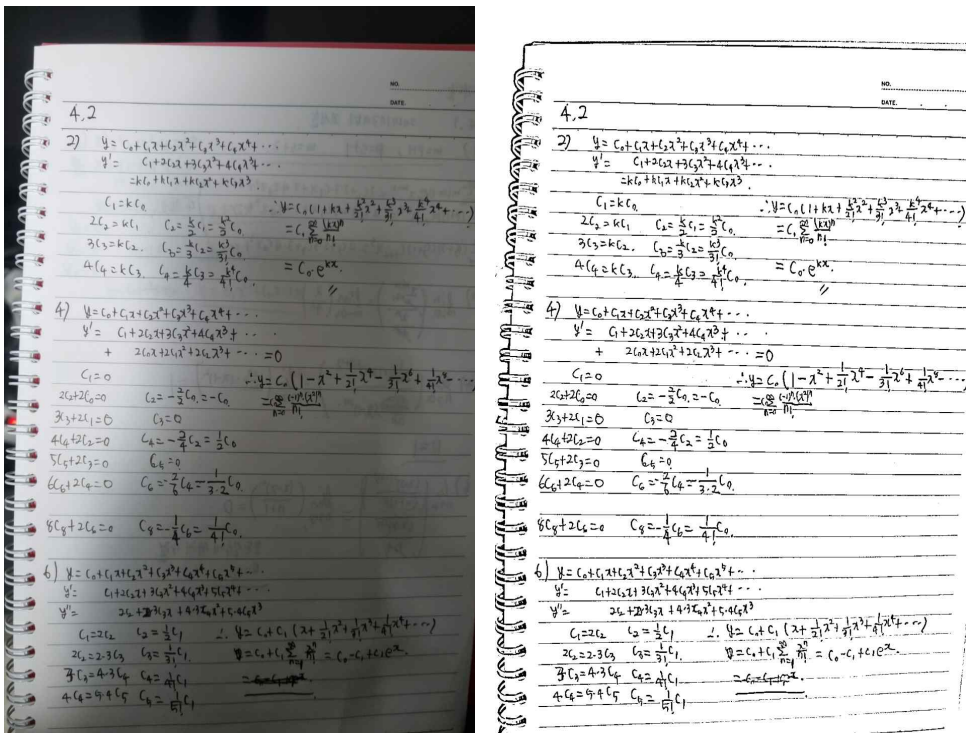
사인 효과를 인물 사진에 적용해 울렁이는 효과를 나타낸 결과를 보여주고 있다.

[그림 18] 렌즈 효과를 적용한 이미지

마찬가지로 인물 사진에 렌즈 효과를 적용해 우스꽝스러운 이미지를 도출할 수 있다.

[그림 19] Edge Detecting 적용 이미지

위 이미지는 엣지 디텍팅을 적용한 예시이다. 다소 모호할 수 있는 경계도 잘 구분하여 검은색의 오브젝트와 흰 선의 윤곽선으로 구별하여 다른 느낌의 이미지를 제공할 수 있다.



[그림 20] 그림자 제거 기능 적용

위 그림은 실제 미분방정식 과제의 제출 사진으로, 왼쪽 그림과 같이 방 안의 형광등, 조명 때문에 그림자가 지게 된다. 이 때 제출물에 따라 불이익을 얻을 수도 있다. 이를 해결하기 위해 thresholding 기능을 통해 그림자를 제거하는 알고리즘을 적용하였고 오른쪽 그림과 같이 글씨와 그림자를 구별하여 제출할 수 있도록 하였다.

5. 결론

5.1 기대효과

HTML의 Canvas API 라이브러리를 활용하여 이미지 자르기, 회전, 대칭 뿐만 아니라 다른 이미지 에디터에서는 볼 수 없는 고급 이미지 보정 기능을 제공하는 이미지 에디터 솔루션을 제공하였다. 단순한 이미지 에디팅 기능을 제공하는 다른 웹 이미지 편집 에디터와 비슷하게 기본적인 편집 기능을 제공하면서도 샤프닝, 블러링, 엣지 디텍팅, 렌즈 효과 등 다른 웹 이미지 편집 에디터는 물론 다운로드받아 사용하거나 유료 요금제를 채택하는 이미지 에디터에서도 보기 힘든 기능을 추가하였다. 따라서 본 프로젝트에서 제공하는 이미지 편집 솔루션은 고유한 기능을 가지고 있어 타 이미지 에디터와 차별점을 둘 수 있을 것으로 기대된다.

https://imageeditorcapstone.tk/ 왼쪽 링크에서 본 프로젝트의 이미지 에디터를 확인할 수 있다.

5.1 향후 연구 방향

현재 구현한 이미지 에디터의 기본 기능은 HTML의 Canvas API를 기반으로 한다. 다른 웹 이미지 에디터와 차별점을 둔 본 프로젝트만의 이미지 편집 기능 또한 코드는 직접 구현하였지만 알고리즘은 기존 연구에 기반하는 경향이 있었다. 따라서 향후에 연구를 지속할 수 있다면 본 프로젝트만의 고유한 Drawing 방식을 적용시키거나 새로운 알고리즘을 적용한

고유한 이미지 편집 기능을 추가로 제공해 더더욱 고유한 기능을 제공하는 차별화 둘 수 있는 웹 이미지 에디터로 활용할 수 있을 것이다.

6. 참고문헌

[1] Canvas_API : https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API

[2] fabricjs : http://fabricjs.com/docs/

[3] 정종윤, 박성배 "HTML5 Canvas 기반 오픈소스 이미지 에디터 라이브러리 개발." 한국소프트웨어종합학술대회 논문집 2021.12 (2021): 1318-1320.

[4] https://skylum.com/ko/creative /best-photo-editing-program

[5] Thresholding : https://scikit-image.org/docs/stable/auto_examples/applications /plot_thresholding.html

[6] Laplacian Filter : https://www.charlezz.com/?p=45203