OpenCL을 이용한 실시간 동영상 왜곡 구현*

박용진○ 박진홍 김진우 한탁돈
연세대학교 컴퓨터과학과
jini99@cs.yonsei.ac.kr jhpark@msl.yonsei.ac.kr jwkim@msl.yonsei.ac.kr hantack@msl.yonsei.ac.kr

Real-time Video Distortion Using OpenCL

Yong-Jin Park○ Jin-Hong Park Jin-Woo Kim Tack-Don Han
Dept of Computer Science, Yonsei University

1. 서론

영상 리터칭(Retouching) 기법은 영상에 다양한 효과를 주어 사용자에게 실제 영상과 다른 다양한 조작된 화면을 제공하는데 사용되고 있다. 영상 왜곡(Distortion) 기법은 영상 리터칭 기법의 하나로, 영상에 다양한 왜곡 효과를 주어 사용자가 다양한 영상을 얻을 수 있도록 한다. 영상 왜곡 중 가장 많이 사용되는 기법의 하나인 방사 왜곡(Radial distortion)은 크게 원통형 왜곡(Barrel distortion)과 바늘겨레 왜곡(Pincushion distortion)으로 나뉜다[1][2][3][4]. 영상 왜곡 기법은 현재 처리하는 화소(Pixel)의 색상값을 구하기 위해 다른 화소의 값을 필요로 하며, 이는 새로운 주소를 구해야 함을 의미한다. 모든 화소에 대해 이와 같이 새로운 주소를 구해야하기 때문에 영상의 크기가 증가됨에 따라 실시간 처리가 어려워지고 있다.

본 논문은 영상 왜곡 기법을 실시간으로 처리하기 위해 GPGPU(General Purpose Graphics Processing Unit)의 병렬성을 기반으로 OpenCL(Open Computing Language)을 이용하여 실시간으로 사용자가 원하는 형태의 동영상 왜곡을 수행하였다. 영상 왜곡 기법의 특성 상 각 화소의 처리는 다른 화소의 처리와 독립적으로 수행이 가능하기 때문에 이 병렬성을 활용하여 GPGPU에서 동영상을 실시간으로 영상 왜곡을 시킬 수 있다.

2. 본론

왜곡 기법은 다양한 방법으로 수행될 수 있다. 본 논문에서는 중심점(Center pixel)과의 거리에 비례하여 새로운 화소의 주소를 구하는 방법을 사용하였다. 현재 처리하는 화소 P(x,y)는 아래의 식과 같은 방법을 통해 새로운 주소 P'(x',y')를 구하게 되며, 이 새로운 주소의 화소값을 현재 화소인 P(x,y)에 그리게 된다. 영상의 모든 화소에 대해 이러한 과정을 거친 후에 최종 영상이 완성된다.

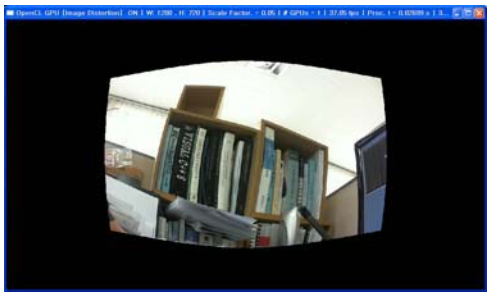
$$x' = Cx + (x - Cx) * ratio_x \qquad ratio_x = (1 - thres) + \frac{thres}{\sqrt{Cx}} * \sqrt{(x - Cx)^2 + (y - Cy)^2}$$

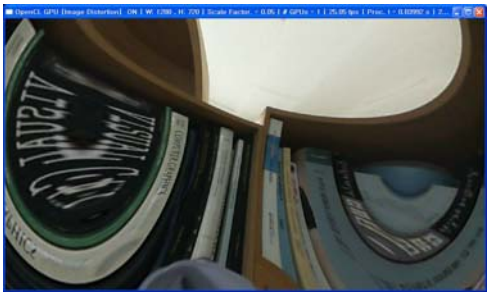
thres는 threshold값으로 이 값이 증가할수록 원통형 왜곡 현상이 발생되며, 이 값이 감소할수록 바늘겨레 왜곡 현상이 발생된다. 본 논문에서는 이 값을 키보드를 입력으로 받아 조절하여 실시간으로 화면을 변경할 수 있도록 하였다. 위 식의 결과는 소수점이 존재할 수 있으며, 이는 정확한 화소의 주소가 아닌 인접한 위치만을 나타내기 때문에 이 주소와 인접한 한 개의 화소만을 가져올 경우 영상 화질의 문제를 발생시킬 수 있다. 따라서 이진보간법 (Bilinear interpolation)을 적용시켜 화질을 개선하였다.

*본 연구는 삼성전자(주)의 협력을 받았음.

OpenCL은 여러 개의 쓰레드(Thread)를 이용하여 병렬적으로 동시에 여러 개의 화소를 처리할 수 있다. 커널 선언, 생성 및 초기값을 할당한 후 실행 큐에 커널을 삽입하는 과정을 거쳐 수행되며, GPU에 존재하는 여러 개의 쓰레드는 커널을 수행한다. 입력 영상의 주소를 이용하여 새로운 주소를 구하는 과정은 다른 쓰레드와 독립적으로 수행될 수 있으며, 입력 영상에서 새로운 주소의 화소값을 계산하여 결과 영상의 메모리에 할당을 하기 때문에 영상의 결과도 다른 쓰레드와 독립적으로 수행된다. 따라서 영상 왜곡 기법은 여러 개의 쓰레드를 독립적으로 동작시킬 수 있으며, 이는 많은 화소를 포함하는 크기가 큰 영상에서도 빠르게 영상 왜곡 기법이 적용될 수 있다.

CPU 실험 환경은 Intel Core2 Duo 3.0GHz를 사용하였으며, GPU 실험 환경은 Nvidia Geforce 8600 GT를 사용하였다. 또한 동영상은 1280*720 화소를 사용하였다. 실험 결과 thres가 0.05일 경우 CPU 환경에서는 6 FPS(Frame Per Second), GPU 환경에서는 38 FPS가 측정되었으며, thres가 -0.03일 경우 CPU 환경에서는 5.8 FPS, GPU 환경에서는 27 FPS가 측정되었다. thres값에 따라 성능의 차이가 나는 이유는 원통형 왜곡 현상일 경우 검은 영상이 나타나는 부분은 원본 영상에서 값을 가져오지 않고 검은색을 칠하게 되지만, 바늘겨레 왜곡 현상일 경우 화면의 모든 화소에 대해 원본 영상의 값을 가져오기 때문에 메모리에서 데이터를 가져오는데 필요한 시간이 더 소요되기 때문이다. 실험 결과에서 나타나듯 GPU를 이용한 방법이 약 4.7~6.3배의 성능 향상을 보여주며, 27 FPS 이상으로 실시간으로 처리가 가능함을 보여준다. 반면 CPU를 이용한 방법은 5~6 FPS로 실시간으로 처리가 어려움을 보여준다. 다음 그림은 thres가 0.05일 경우(좌) 및 -0.03일 경우(우)의 실험 결과 화면을 보여준다.





3. 결론

본 논문에서는 GPGPU의 병렬성을 활용하여 OpenCL을 이용하여 영상 왜곡 기법을 GPGPU 환경에서 구현하였다. 성능 검증을 위해 동영상을 입력 영상으로 받아 threshold 값을 실시간으로 조절하며 성능 평가를 수행하였다. 영상 왜곡 기법은 화소간의 독립성이 보장되기 때문에 각 처리 단계에서 병렬성을 충분히 활용할 수 있으며, CPU를 이용한 처리 방식에 비해 약 4.7~6.3배의 성능 향상을 보였다. 또한 27 FPS이상의 성능을 보이기 때문에 실시간으로 동영상 왜곡 기법 처리가 가능하다.

참고문헌

[1] F. Devernay and O. Faugeras, “Automatic calibration and removal of distortion from scenes of structured environments,” SPIE Conference on investigative and trial image processing,1995.

[2] H. Farid and A.C. Popescu, “Blind removal of Lens Distortion,” Journal of the Optical Society of America, 2001.

[3] R. Swaminatha and S.K. Nayer, “Non-metric calibration of wide angle lenses and poly-cameras,” IEEE Conference on computer Vision and pattern recognition, 1999.

[4] G. Taubin, “Camera model and triangulation,” Lecture notes EE-148, 3D Photography, Caltech, 2001.