임베디드 시스템 환경에서 효율적인 카메라 왜곡 보정 방법

천승환* · 유영호* · 장시웅*

*동의대학교

An Efficient Camera Calibration Method in Embedded System Environment

Seung-hwan Cheon* · Young-ho Yu* · Si-woong Jang*

*Dong-Eui University

E-mail : perari0@nate.com, yhyu@pusan.ac.kr, swjang@deu.ac.kr

요 약

자동차를 위한 전방향(omnidirectional) 감시 시스템 같은 감시 시스템, 로봇의 시각 역할 등 다양한 비젼 시스템에서 카메라가 장착되어 사용되고 있다. 광각 카메라에서 획득한 영상은 비선형적인 방사 왜곡을 가지고 있어서 정확한 영상을 획득하기 위해서는 왜곡 보정 작업을 수행하여야 한다. 카메라로부터 입력되는 왜곡 영상을 실시간으로 보정하기 위해서는 많은 연산량을 요구하므로 임베디드 환경에서는 왜곡 보정을 위해 별도의 시스템을 추가하거나 SOC 형태로 개발한 전용 H/W를 추가한다. 본 논문에서는 왜곡 보정 과정에 필요한 연산들을 분석하고, 연산량을 줄일 수 있는 개선된 왜곡 보정 방법을 제안한다. 또한, ARM11 기반 임베디드 환경에서 제안한 왜곡 보정 방법을 직접 구현하여 성능을 평가함으로써 임베디드 환경에서 추가의 시스템이나 추가의 비용없이 왜곡 보정을 수행할 수 있음을 보인다.

키워드

왜곡 보정, 임베디드 시스템, 광각 카메라

Ⅰ. 서 론

오늘날 비전 산업은 전방향 감시 시스템, 로봇의 시각 역할 등 카메라로부터 들어오는 영상을 분석하여 얻어지는 정보를 이용하는 시스템들이 주를 이루기 때문에 정확한 영상을 얻는 것이 무엇보다도 중요하다. 하지만 광각 렌즈가 부착된 카메라로 얻어진 영상들은 중심에서 벗어날수록 심한 방사 왜곡이 나타난다. 이는 다양한 광각 렌즈들의 특성과 초점거리로부터 나타나는 현상으로 가장자리 부분이 가운데 부분보다 심하게 휘어지는 현상이다[1]. 이러한 왜곡을 가진 영상은 패턴인식이나 검색과 같은 영상처리 시스템에 사용하면 심각한 오류를 일으킬 수 있다. 이러한 문제점들을 해결하기 위해 왜곡 보정 작업을 수행하여 정확한 영상을 획득하여야 한다[2].

기존의 연구에서는 카메라로부터 입력되는 왜곡 영상을 실시간으로 보정하기 위해서는 많은 연산량을 요구하므로 왜곡 보정을 위한 별도의 시스템을 추가하거나 SOC(System On Chip)형태의 전용 H/W를 추가했다. 이 연구들은 임베디드 환경에서 이미 충분한 성능을 입증했지만, 비용이 많이들고 시스템이 복잡해진다는 문제점이 있다. 이러한 문제점들을 개선하게 되면 비용감소와 품질 향상을 기대할 수 있다.

본 논문의 2장에서는 왜곡 보정에 관한 연구에 대해 설명하고 3장에서는 소프트웨어적인 개선방안을 살펴보며, 4장에서는 개발환경과 효율적인 왜곡 보정 처리 방법의 구현 및 성능평가를 하였으며, 5장에서 결론을 맺는다.

Ⅱ. 관련 연구

왜곡 현상이란 카메라를 이용해 보여지는 영상을 화면에 출력하게 되면 직선이 곡선처럼 보여지는 현상을 왜곡 현상이라 한다. 이러한 현상을 카메라 렌즈의 왜곡 현상이라고 한다. 카메라 렌즈에서 발생되는 왜곡 현상에는 방사 왜곡(radial distortion) 현상이 있다.

방사 왜곡 현상은 영상에서의 점 위치를 정상적인 위치에 맺히게 하지 못하고 렌즈 중심에서 방사방향으로 정상 위치의 안쪽 또는 바깥쪽으로 맺히는 현상이다. 이러한 방사 왜곡 현상의 원인은 카메라 렌즈의 중심보다 더 멀리 떨어진 곳을 지나가는 광선이 렌즈의 중심에 가까운 곳을 지나

가는 광선보다 휘어짐의 현상이 더 많이 일어나면서 발생한다. 방사 왜곡 현상을 나타낸 것은 그림 1과 같다.

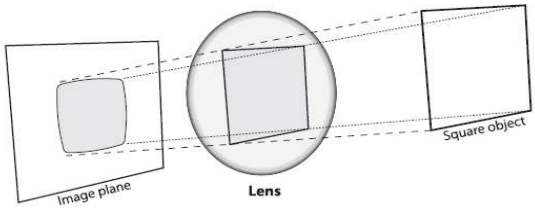

그림 1 방사 왜곡 현상의 구조

이러한 왜곡 현상을 보정하기 위한 여러 가지 방법 중 본 논문에서 사용하는 구(sphere)기반의 왜곡 영상 보정 방법을 그림 2에서 보여준다[3].

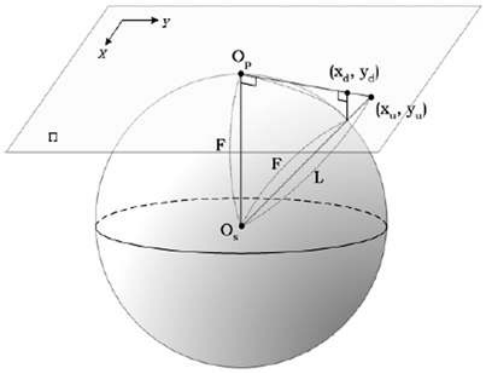

그림 2. 구 기반의 왜곡 보정

그림 2에서 F는 카메라의 초점거리, 구면에서 평면으로 투영된 임의의 점(x_d, y_d)는 왜곡된 영상의 좌표, 점(x_u, y_u)는 왜곡이 보정된 영상의 좌표이다. 그러므로 구의 중심에서 왜곡되지 않은 점까지의 거리 L과 초점거리 F를 이용하여 점(x_d, y_d)와 점(x_u, y_u)의 관계는 수식 (1)로 표현된다.

$$(x_d, y_d) = (x_u\frac{F}{L}, y_u\frac{F}{L}),\ L = \sqrt{x_u^2 + y_u^2 + F^2} \quad (1)$$

왜곡 보정을 위한 영상 처리는 많은 연산량을 필요로 한다. 기존 시스템들은 영상 처리의 성능을 위해 추가의 임베디드 시스템이나 SOC 형태로 영상 처리를 구현하였다. 본 논문에서는 앞서 살펴본 왜곡 현상을 보정하는 과정에 필요한 연산들을 분석하여 연산량을 줄여서 임베디드 시스템에 적합한 개선된 왜곡 보정 방법들을 제안함으로써 추가의 시스템이나 비용없이 왜곡 보정을 처리하는 방안을 제시한다.

III. 개선 방안

본 논문의 소프트웨어적인 개선 방안으로는 중복되는 소스를 제거하고 불필요한 소스를 수정 및 보완하는 방안을 제시한다. 본 논문에서는 2장에서 소개한 왜곡 보정 방법을 사용할 때 필요한 연산량을 효과적으로 줄이기 위한 방법을 제시한다.

연산량을 줄이기 위한 첫 번째 방법으로, 제곱근 연산 횟수를 줄이는 방법이다. 수식(1)을 이용해 왜곡 보정을 할 경우 L을 구하기 위해 사용되는 제곱근 연산을 픽셀마다 여러 번 수행하게 되는데 이를 한 번만 수행하도록 왜곡 보정 함수를 구성한다. 이를 통해 매번 제곱근 연산을 수행했을때보다 20%의 연산량 감소 효과를 예상할 수 있다.

두 번째 방법은 입력받는 (640 X 480)크기의 영상을 (320 X 240)영상으로 Resize하는 것으로 연산량의 75%가량을 줄일 수 있는 효과를 기대해 볼 수 있다. 이러한 효과를 기대할 수 있는 이유는 직접적으로 연산하는 픽셀수가 3/4만큼 줄어들기 때문이다.

세 번째 방법은 소스 내부의 for문을 사용할 때, 0에서부터 1씩 증가시켜 사용하는데, 비트맵의 특성상 마지막 픽셀부터 1씩 빼면서 연산하는 다운 카운팅 방식을 이용해 for문을 돌리게 되면 20%정도의 성능 향상을 기대할 수 있다[4].

IV. 구현 및 성능 평가

4장에서는 3장에서 제시한 세 가지 방법을 직접 구현하여 연산량 감소 효과를 실험을 통하여 확인한다. 그리고 제안하는 알고리즘이 임베디드 시스템에서 사용하기에 적합한 성능을 나타내는지 실험을 통하여 검증한다.

4.1 구현 환경

본 논문에서 구현하는 개선된 왜곡 보정 방법의 구현 환경은 다음의 표 1과 같다.

표 1. 시스템 사양

구성	사양
제품개발키트	I.MX35 ARM11 Applications Processor 512MB of NAND Flash Memory 128MB of 32bit DDR2 SDRAM memory 운영체제: WinCE 6.0
카메라	Model: FO-3000CN Pixels: 320K(656H×492V) Lens: 180°(D)×140°(H)×100°(V)
개발언어	컴파일러: Visual C++ 2005 플랫폼 : iMX35-3DS-PDK1_7-Mobillity

4.2 성능 평가

4장에서는 3장에서 제안한 개선 방안을 직접 구현하여 알고리즘의 성능을 평가한다. 먼저 (640 X 480)크기의 원 영상 그대로를 화면에 출력했을 때의 결과를 얻었다. 원 영상은 초당 31프레임의 영상을 받아오는 결과를 얻었다. 다음으로 2장에서 소개한 구기반의 왜곡 영상 보정 방법을 제안한 개선 방안을 적용하지 않

고 구현하여 성능을 평가한 결과 대략 7~8초당 1프레임을 받아오는 결과를 얻을 수 있었다. 이를 개선하기 위해 3장에서 제안한 각각의 개선 방안에 따라 얼마만큼의 성능 향상이 있는지를 테스트한다.

제곱근 연산 횟수를 줄이는 방안을 적용한 스퀘어루트 알고리즘을 구현하여 성능을 평가한다. 각 픽셀마다 여러 번 적용되는 제곱근 연산을 각 픽셀마다 한 번만 계산하도록 제곱근 연산을 미리 수행하여 테이블을 구성하여 사용한다. 스퀘어루트 알고리즘은 초당 0.5~0.8프레임을 받아올 수 있었다.

원 영상 이미지 크기인 (640 X 480)에서는 원하는 결과를 얻기가 어려우므로 이미지 크기를 사용자가 디스플레이에서 무리 없이 볼 수 있는 크기인 (320 X 240)으로 줄인 Resize 알고리즘을 구현하여 성능을 평가한다. Resize 알고리즘은 1~1.5 프레임을 받아올 수 있었다. 직접적으로 연산하는 픽셀수가 1/4로 줄었기 때문으로 분석된다.

for문을 다운카운팅 방식을 적용한 다운카운팅 알고리즘을 구현하여 성능을 평가한다. 다운카운팅을 했을 때, for문의 비교부분을 제거하여 테스팅 단계를 줄이도록 구현하였다. 그 결과 다운카운팅 알고리즘은 초당 0.5~0.8 프레임을 받아올 수 있었다.

그림 3은 위 세 가지 개선 방안을 각각 적용한 알고리즘의 성능을 매 1분마다 초당 평균 프레임수를 계산하여 나타낸 것이다.

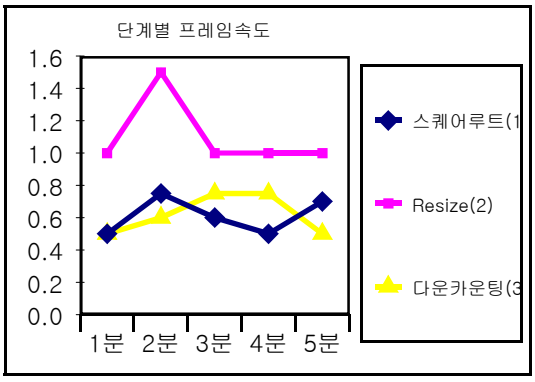

그림 3 개선 방안을 적용한 알고리즘

개선 방안을 각각 적용한 스퀘어루트 방식, Resize 방식, 다운카운팅 방식을 결합하였을 때 성능 향상 정도를 평가한다.

스퀘어루트 방식과 Resize 방식을 함께 적용하였을 경우, 초당 8프레임 정도의 속도가 나왔다. 그리고 Resize 방식과 다운카운팅 방식을 함께 적용하였을 경우, 초당 14프레임 정도의 속도가 나왔다. 위의 두가지 방법은 뚜렷한 성능 향상은 있었으나 사용자가 아무 무리없이 사용하기에는 부족한 수준의 영상 품질을 제공하는 수준이다.

스퀘어루트 방식, Resize 방식, 다운카운팅 방식을 모두 적용한 왜곡 보정 알고리즘은 초당 27~28프레임의 속도가 나왔다. 이는 사용자에게 초당 32프레임의 영상 품질과 거의 차이를 느낄 수 없을 정도의 영상 품질을 제공할 수 있는 수준이다.

그림 4는 개선 방안을 결합한 알고리즘들의 성능과 원영상을 그대로 보여줄 때의 성능을 매 1분마다 초당 평균 프레임수를 계산하여 나타낸 것이다.

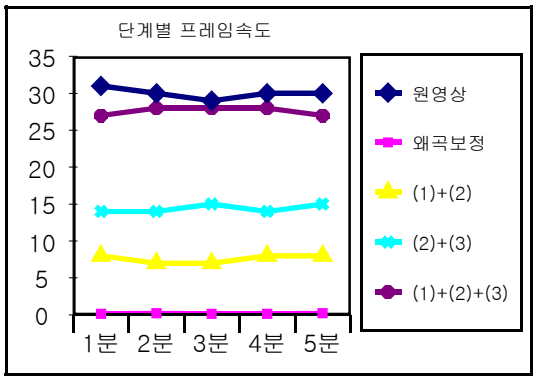

그림 4. 단계별 프레임 속도

그래프의 결과를 보면 원영상과 제안한 개선 방안을 모두 적용한 왜곡 보정 알고리즘은 초당 1~4프레임 정도의 차이를 보이고 있다. 이 차이는 사용자가 디스플레이 상에서 거의 차이를 느낄 수 없을 만큼의 수치이며, 원영상의 왜곡을 보정한 영상이므로 사물 분간 및 패턴인식, 검색등의 영상처리 시스템의 기본 영상으로 사용될 수 있다.

그림 5는 원영상과 제안한 개선 방안을 모두 적용한 알고리즘의 결과 영상을 보여준다.

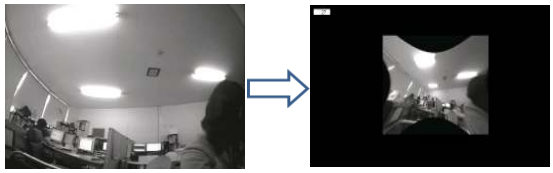

그림 5. 원영상(좌)과 왜곡보정 결과영상(우)

V. 결 론

본 논문에서는 임베디드 시스템 환경에서 적용할 수 있는 광각 카메라로부터 입력되는 왜곡 영상 보정 알고리즘을 제안하였다. 알고리즘의 성능 향상을 위해 왜곡 보정 과정에 필요한 연산들을 분석하고, 연산량을 줄일 수 있는 개선 방안을 제안하였다. 또한, 제안한 개선 방안을 적용한 알고리즘을 직접 설계 및 구현하여 성능을 평가하였다. 본 논문에서 제안하는 개선된 광각 카메라 왜곡 보정 방법은 감시 시스템, 로봇의 시각 역할, AVM(Around View Monitor) 시스템 등의 여러 비전 산업에서 별도의 시스템이나 SOC형태

의 전용 H/W의 추가 없이 적용할 수 있는 소프트웨어 중심적인 왜곡 보정 알고리즘이다.

본 논문에서는 임베디드 시스템 환경에서 사용자에게 실시간으로 왜곡 보정된 영상을 제공하는 것을 목표로 하였다. 현재의 결과는 사용자가 느끼기에는 거의 느끼지 못할 수준이지만, 초당 1~4 프레임 정도의 프레임을 처리하지 못하고 있다. 향후 연구에서 개선을 통하여 처리하지 못하는 프레임이 없도록 성능 향상 방안을 연구할 것이다. 또한, 비전 산업의 여러 시스템들에는 왜곡 보정뿐만 아니라 영상 정합, 호모그라피 변환 등 많은 영상 변환 방법이 있다. 이러한 영상 변환 방법들도 임베디드 시스템 환경에서 별도의 시스템이나 SOC형태의 전용 H/W없이 S/W적인 성능 향상 방법을 연구할 것이다. 나아가 제안한 알고리즘을 사용하여 차량용 네트워크를 통해 ARM11 보드 5개를 연결하여 4대의 카메라를 사용한 차량용 AVM 시스템을 구현할 것이다.

본 논문의 결과는 임베디드 시스템 환경에서 저비용의 영상처리 방법으로 활용될 수 있을 것으로 본다.

Acknowledgement

본 논문은 중소기업청에서 지원하는 2010년도 산학연공동기술개발사업(No.00042243)의 연구수행으로 인한 결과물임을 밝힙니다.

참고문헌

[1] G. Vass and T. Perlaki "Applying and removing lens distortion in post pro duction," Second Hungrian Conference on Computer Graphics and Geometry, 2003

[2] 김병익, 김대현, 배태욱, 김영춘, 심태은, 김덕규, “광각 카메라 영상의 보정을 위한 자동정합 좌표 추출 방법”, 멀티미디어학회 논문지, 제13권, 제3호, pp.410~416, 2010년 3월.

[3] 신주홍, 남동환, 권기준, 정순기, “Ellipsoid를 이용한 어안렌즈의 Non-metric 접근 왜곡 보정 기법”, HCI2005 학술대회논문집, 제14권, 제1호, pp.83-89, 2005년2월.

[4] 김유진, “임베디드 프로그래밍 C코드 최적화”, (주)한빛미디어, pp.215-223, 2010년 3월.

[5] 박민우, 장경호, 정순기, 윤팔주, “차량의 사각지대 제거를 위한 측/후방 카메라 영상 정합 시스템”, 2009 한국정보과학회논문집, 제36권, 제8호, pp.653~662, 2009년 8월.

[6] 안레센, 라림파, 인은규, 정정화, “자동차용 카메라의 실시간 탑뷰 시스템 구현”, 대한전자공학회, 제33권, 제1호, pp.1219~1222 , 2010년