평활한 완전 8방향 윤곽선과 이의 부호화 기법

조 성호○†, 김 인철‡, 이 상욱†

†서울대학교 전기공학부
‡한성대학교 정보전산학부

SMOOTHLY PERFECT 8-CONNECTED CONTOUR AND ITS CODING TECHNIQUE

Sungho Cho○†, *Rin Chul Kim*‡ *and Sang Uk Lee*†

†Signal Processing Lab., School of Electrical Eng., Seoul Nat'l Univ.
e-mail : shcho@Claudia.snu.ac.kr
‡School of Information and Computer Eng., Hansung Univ.
e-mail : rin@ws11.hansung.ac.kr

요약문

In this paper, we introduce the notion of the smoothly perfect 8-conntected (SP8C) contour and propose a coding technique for the SP8C contours. Based on the contour simplification using the majority filter propsed by Gu[6], SP8C contours are extracted on the contour lattice from the segmented image. By noting that, unlike the perfect 8-connected contours, the SP8C contours are restricted to travel in only 3 different directions along the contours, we also propose two techniques for encoding the SP8C contours. The one is the modified version of the neighbouring direction segment coding by Kaneko[2], while the other is to employ the notion of the entropy coding. From the comparison in terms of the entropy, it is shown that the proposed SP8C contours require less bits in encoding the diagonal contours than the 4-connected contours employed by Gu. And computer simulations reveal that the contours can be efficiently encoded by the proposed technique.

1. 서론

초 저전송률 영상 부호화를 위한 한 방법으로 객체지향 부호화 (object-oriented coding) 기법에 대한 연구가 활발히 진행되고 있다. 객체지향 부호화 기법에서는 영상 정보를 객체로 나누고, 각 객체에 대한 모양 혹은 윤곽선, 움직임, 색정보 등을 부호화하는 것이다. 전송 정보중 윤곽선은 저 전송률에서 가장 많은 정보량을 요구하는 것으로 이의 효율적인 부호화 기법에 대한 연구가 활발히 진행되고 있다[6, 7]. 지금까지 여러가지 윤곽선 부호화 기법이 알려져 있으나, 대표적인 방법은 윤곽선 방향을 따라 가면서 그 방향을 부호화하는 체인 부호화이다[1, 2, 3, 4, 5].

일반적으로 체인 부호화에서 사용하는 윤곽선은 다음의 두 방법으로 나타내어진다. 하나는 화소 영역(pixel domain)에서 윤곽선을 표시하는 방법이고, 다른 하나는 윤곽선 격자를 이용하여 표시하는 방법이다. 화소 영역에서 윤곽선을 표시하는 방법은 윤곽선을 처리함과 동시에 윤곽선상의 화소가 어느 영역에 속하는 지를 판단해야 하는 단점이 있다. 반면에 윤곽선 격자로 표시하는 방법은 화소와 화소 사이에 윤곽선 격자를 그림 1과 같이 정의하고, 윤곽선을 격자 1과 격자 2로 표시하는 방법으로 윤곽선 격자에 의해 화소들이 속한 영역이 명확하게 나뉘어 진다. 기존의 방법에서는 이렇게 하여 추출된 4-방향 윤곽선을 부호화한다. 그러나, 이 방법에서는 윤곽선 격자와 화소의 구조가 서로 다르다는 문제점이 있다.

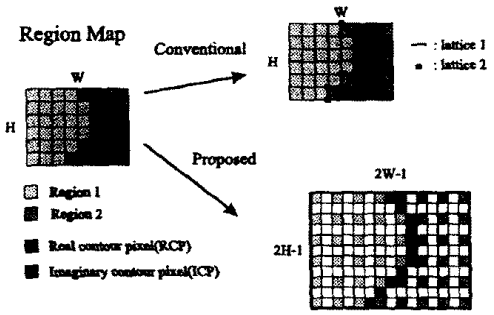


그림 1: Lattice Conversion

본 논문에서는 [6]에서 제안한 majority 필터링한 영역에 대해 새로운 윤곽선 격자를 이용하여 추출한 윤곽선을 평활한 완전 8방향(SP8C) 윤곽선으로 정의하고, SP8C 윤곽선 표현법은 윤곽선 격자 상에서 4-방향 윤곽선 표현법 보다 우수한 부호화 성능을 나타냄을 보이겠다.

2. 윤곽선 추출 기법

본 절에서는 영역화된 영상으로부터 SP8C 윤곽선을 추출하는 방법에 대하여 설명한다. 제안하는 윤곽선 추출 방법은 majority 필터링과 격자 변환 (lattice conversion)으로 이루어지는데, 각 과정에 대한 자세한 설명은 다음과 같다.

본 연구는 한국학술진흥재단의 학술연구 조성비의 지원으로 이루어졌습니다.

2.1. Majority filtering

일반적인 영역화된 영상은 잡음 등의 영향으로 영역의 경계에 고주파 성분이 다수 포함되어 있게된다. 특히, 그림 2에 보인 바와 같은 spark region 이 존재하면, 복호화된 영상의 주관적 화질이 저하될 뿐만아니라, 윤곽선 부호화에 많은 정보량이 요구된다는 단점이 있다. 이러한 단점을 해결하기위해, [6]에서는 일종의 non-linear 필터인 majority 필터를 제안하였다.

그림 2: Example of spark region

그림 3: Cross and square structuring element of size one

Majority 필터는 그림 3와 같은 stucturing element를 설정하여 필터링하고자하는 픽셀의 값을 주위에 가장 많은 영역의 값으로 바꾸어준다. 이와 같은 과정을 통해, 그림 2에 보인 spark region을 없앨 수 있을 뿐만 아니라, 윤곽선 방향 변화에서 제한을 두게 되어 효율적인 윤곽선 부호화를 할 수 있다. 본 논문에서는 크기가 1인 square structuring element를 사용한다.

2.2. 격자 변환

폭이 W이고 너비가 H인 영상에 윤곽선을 나타내기 위해서 폭과 너비가 각각 2W-1, 2H-1인 윤곽선 영상을 만들어 낸다. 이때, 그림 1에 보인 바와 같이 가로, 세로 모두 짝수번째에 있는 격자는 화소 영역을 나타내고, 가로, 세로 모두 홀수번째에 있는 화소는 가상의 윤곽선 픽셀(ICP)을 나타내고, 그렇지 않은 화소가 실제 윤곽선 픽셀(RCP)을 나타낸다. 그런데, ICP는 단지 윤곽선을 이어주기 위해 생성된 화소로서, 영역을 나누는 역할을 하지는 않는다. 따라서, 부호화에는 RCP만이 이용된다. 이때, 기존의 방법에서 4-방향 윤곽선으로 표시한 것과는 달리, 본 논문에서는 완전한 8-방향 윤곽선으로 윤곽선을 표시한다(그림 1에서 검은 격자로 표시). 참고로, Gu[6]등이 사용한 일반적인 4-방향 윤곽선은 그림 1에서 격자 1을 이용하여 이동을 하고, 격자 2는 시작점을 부호화하는데 이용된다. 이와 같이 격자 변환을 하면, 윤곽선 영상은 화소 영역에서 윤곽선을 표시한 것에 비해 2배 정도 많은 격자들로 윤곽선을 표시하게 된다.

이렇게 표시된 윤곽선은 일반적인 8-방향 윤곽선과는 달리 다음과 같은 특성을 보이게 된다.

- 크기가 1인 sqaure structuring element로 영역화 영상을 majority 필터링 하였기 때문에 윤곽선 영상은 대각선 방향 후에도 90° 방향 성분이 없어지게 되어 그림 4에 보인 바와 같이 수평/수직 방향 후는 물론 대각선 방향 후에도 3가지의 방향 변화 가능성만을 지니는 윤곽선이 된다.
- 따라서, 방향 변화는 항상 대각선 방향과 수직/수평 방향 변화가 번갈아 발생한다. 즉, 서로 다른 방향의 대각선 방향 변화, 혹은 서로 다른 방향의 수직/수평 방향 변화는 발생하지 않는다.
- 대각선 방향 변화에서의 윤곽선 픽셀의 갯수는 대각선 방향 변화 이전의 이동과 변화 이후의 이동에 관련되어 있다. 즉, 만약 대각선 방향 변화 이전과 이후의 방향이 같으면 대각선 방향 변화에서의 윤곽선 픽셀의 수는 짝수이고, 그렇지 않을 경우에는 픽셀의 수는 홀수가 된다.

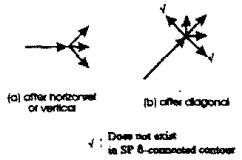


그림 4: Smoothly perfect 8-connected contour

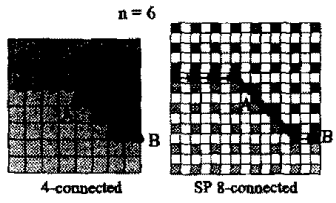
본 논문에서는 이와 같은 특성을 가지는 윤곽선을 평활한 완전 8방향(SP8C) 윤곽선이라고 부르겠다.

3. Entropy 비교

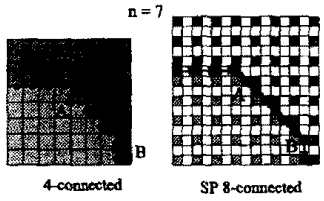
본 절에서는 전 절에서 설명한 majority 필터링 후 추출된 SP8C 윤곽선을 부호화하는 개념을 살펴보면서, 엔트로피 측면에서 4-방향 윤곽선과 부호화 효율을 비교하겠다.

4-방향 윤곽선과 SP8C 윤곽선은 수직 및 수평 방향 이동 후, 발생하는 방향은 모두 3방향으로 같다. 그리고, SP8C 윤곽선은 4-방향 윤곽선에 비해 2배의 격자들로 윤곽선이 표시되나, SP8C 윤곽선은 항상 2 격자 단위로 이동하므로, 실제 부호화하는 윤곽선의 갯수는 같아, 동일한 부호화 성능을 가진다.

한편, 대각선 방향 이동에서는 윤곽선 표현 기법에 따라 부호화 성능이 달라지게 된다. 본 논문에서는 majority 필터링 후 윤곽선을 얻기 때문에 대각선 방향 변화 후 또 다른 방향의 대각선 방향은 나타나지 않는다. 따라서, 대각선 방향 이동은 그림 5에 나타낸 바와 같이, 대각선 이동 전후의 윤곽선 방향이 동일한 경우와 상이한 경우의 두가지 경우로 나눌 수 있다. 만약, 대각선 방향 변화 윤곽선 픽셀의 수를 n이라고 하면, 그림 5에서 알 수 있듯이, 대각선 이동 전후의 윤곽선 방향이 동일한 경우에는 n이 짝수가 되고, 그렇지 않은 경우에는 n이 홀수가 된다.



(a) 짝수인 경우



(b) 홀수인 경우

그림 5: Comparison of the contour representation method

먼저, 4-방향 윤곽선에서는 직진 방향 후에 세 가지 방향 이동이 가능하고, 오른쪽 방향 이동이나 왼쪽 방향 이동 후에는 두

가지 방향 이동이 가능하다. 다시 말해서, entropy 측면에서 직진 방향 후에는 $\log 3$ 비트로써 부호화할 수 있고, 오른쪽/왼쪽 방향 후에는 $\log 2$ 비트로써 부호화할 수 있다.

그러나, SP8C 윤곽선에서는 수평/수직 방향 성분이나 대각선 방향 성분이나 모두 세 가지 방향 이동이 가능하다. 그리고, SP8C 윤곽선에서는 대각선 방향 이동 또한 수직/수평 방향 이동과 마찬가지로 2 격자 단위로 이동한다. 그러나, n 이 짝수인 경우와 홀수인 경우가 있으므로 이를 구별하기 위해 parity bit(PB)을 도입한다. 즉, n 이 짝수인 경우에는 PB=0 로 하고, 대각선 방향 이동 후, 추가적인 대각선 방향 이동이 없음을 나타냄과 동시에, 다음 이동이 대각선 방향 이동 이전의 방향과 같음을 나타낸다. 그리고, n 이 홀수인 경우에는 PB=1 로 하여, 대각선 이동 후 대각선 방향으로 윤곽선을 1 격자 만큼 추가적으로 이동시키고, 대각선 이동 전후의 윤곽선 방향이 다름을 나타낸다.

그림 5에 보인 바와 같이, A점에서 B점까지 대각선 방향으로 윤곽선이 이동한다고 할 때, 4-방향 윤곽선을 부호화를 하는데 요구되는 비트 수 $B_4(n)$과, SP8C 윤곽선을 부호화를 하는데 요구되는 비트 수 $B_{SP8}(n)$는 다음과 같이 주어진다.

$$\begin{aligned} B_4(n) &= n\log 2 \\ B_{SP8}(n) &= 1+(n-A)\log\sqrt{2} \\ \text{where, } A &= \begin{cases} 0, & \text{if } n \equiv 0 \pmod 2 \\ 1, & \text{otherwise} \end{cases} \end{aligned} \quad (1)$$

식 (1)를 참조하면, 제안하는 SP8C 윤곽선으로 표현하는 것이 기존의 4C 윤곽선 표현에 비해 적은 정보량으로 부호화할 수 있음을 알 수 있다. 수식적으로 대각선 방향 변화에서 약 50% 정도 비트가 적게 드는데, 이것은 SP8C 윤곽선의 장점을 이용하여 2 격자 단위로 윤곽선을 부호화하기 때문이다.

4. Entropy Code와 Modified NDSC

본 절에서는 SP8C 윤곽선을 부호화하는데 적합한 두가지 윤곽선 부호화 기법을 제시한다. 첫번째 방법은 entropy 부호화[4]에 근거한 방법이고, 두번째 방법은 NDSC 기법[2]을 수정한 것이다. 본 논문에서는 SP8C 윤곽선의 성능 검증에 주안점을 두었기 때문에 영상을 물체와 배경으로 분리한, foreground/backgroud(F/B) 영역화된 영상에 대한 적용으로 한정하였다. 따라서, 실제 윤곽선 부호화에 필요한 윤곽선 분기점에 대한 고려는 하지 않았다.

본 논문에서 고려한 두가지 부호화 방법에서 시작점은 다음과 같이 부호화 하였다. 입력 영상의 폭과 너비를 각각 W 및 H 라고 할 때, 시작점이 가질 수 있는 위치는 짝수 라인에서는$(0,2,4,\cdots,2H-1)$ 영역 화소의 상하에 있는 윤곽선 격자로 한정되므로 픽셀의 수가 $H-1$이 되고, 홀수 라인에서는$(1,3,5,\cdots,2H-2)$ 영역 화소의 좌우에 있는 윤곽선 격자로 제한되므로 픽셀의 수는 H가 된다. 따라서 원 영상이 QCIF 포맷일 경우($W=176, H=144$) 앞에서 설명한 점을 이용하면 시작점은 17비트로써 부호화된다. 또한 처음 시작 방향 부호(octal code)는 3비트로 부호화한다.

4.1. Entropy Code

Entropy 부호화 기법은 처음 시작 방향 부호(octal code)를 3비트로 부호화한 다음, SP8C 윤곽선의 특성에 따라 직선 방향 후와 대각선 방향 후의 두 가지 mode로 나누어진다. 현재 방향의 Chain 부호를 n라 하고, 다음 방향 Chain 부호를 n^*라 하자. 그러면 entropy 부호화는 다음과 같이 수행된다.

- 직선 방향 후 ($n \equiv 0 \pmod 2$)
  - $n^* \equiv n \pmod 8$: Code = 0
  - $n^* \equiv n+1 \pmod 8$: Code = 10 + PB
  - $n^* \equiv n-1 \pmod 8$: Code = 11 + PB
- 대각선 방향 후 ($n \equiv 1 \pmod 2$)
  - $n^* \equiv n \pmod 8$: Code = 0
  - $n^* \equiv n \pm 1 \pmod 8$: Code = 1

실험에서 부호화의 끝을 나타내는 부호는 삽입하지 않았다. 즉, 부호화의 끝은 영상의 가장자리로 윤곽선 픽셀이 이동을 하였거나, 아니면, 부호화를 한 픽셀로 다시 돌아올 경우에 부호화가 끝남을 확인한다고 가정을 하였다.

4.2. Modified NDSC

기존의 Kaneko가 제안한 NDSC[2] 알고리듬에 SP8C 윤곽선의 특성을 넣어서 부호화를 한다. 기존의 NDSC 알고리듬은 이웃하는 두 방향의 chain 부호를 가지는 윤곽선 픽셀들을 하나의 세그먼트로 분리하여 세그먼트 단위로 부호화하는 알고리듬이다. 한 세그먼트에서 다른 세그먼트로 이동을 할 때, Extended Octant Code라 해서 다음 세그먼트의 처음 방향을 정해주는 Code가 필요하다.

그런데, 제안하는 알고리듬에 의해 추출된 SP8C 윤곽선 영상은 항상 3가지 방향 만을 가지므로 이웃하는 2개의 방향으로 구성된 세그먼트 단위 부호화 알고리듬에서 다음 세그먼트의 방향이 정해진다는 점이 발생한다. 따라서, Extended Octant Code가 필요없다. 이전 세그먼트의 Octal Code가 n이고, 다음 세그먼트의 Octal Code를 n^*이라 하자. 그리고 세그먼트의 마지막 Chain Code를 d라 하면, Octal Code n에 따른 인접 Chain Code d는 n이거나 $n+1(\text{mod } 8)$이 된다. 그러면 다음 세그먼트의 Octal Code n^*는 다음 식 (2)와 같이 정해진다.

$$n^* = \begin{cases} n-1 \pmod 8 & \text{if } d \equiv n \pmod 8 \\ n+2 \pmod 8 & \text{if } d \equiv n+1 \pmod 8 \end{cases} \quad (2)$$

또한, 대각선 방향에서의 갯수를 예측할 수 있다.(즉, PB가 필요없다.) 이것은 대각선 방향에서 세그먼트가 끝났을 경우, 다음 세그먼트의 방향을 알 수 있다는 위 성질에 기인한다. 즉, 대각선 방향 변화 이전의 Chain Code를 d라 하고, 이후의 Chain Code를 n^*라 할때, 대각선 방향의 윤곽선 픽셀 수 L_d는 다음 식 (3)과 같이 정해진다.

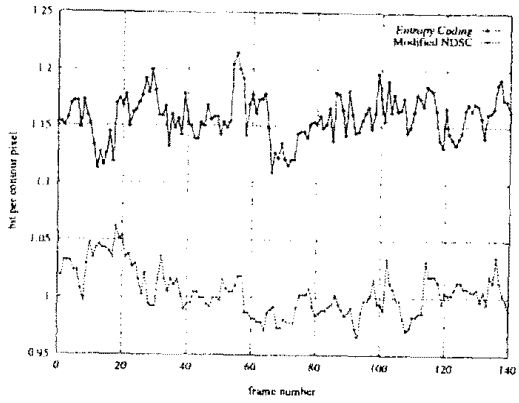
$$L_d \equiv \begin{cases} 0 \pmod 2 & \text{if } d = d^* \pmod 8 \\ 1 \pmod 2 & \text{otherwise} \end{cases} \quad (3)$$

즉, 대각선 방향의 윤곽선을 패러티 비트없이 항상 2 격자 단위로 부호화할 수 있다는 장점이 생긴다. 또한, 세그먼트 길이도 또한 윤곽선 2 격자 단위로 설정한다. 세그먼트 길이 부호화는 [2]의 방법을 따랐다.

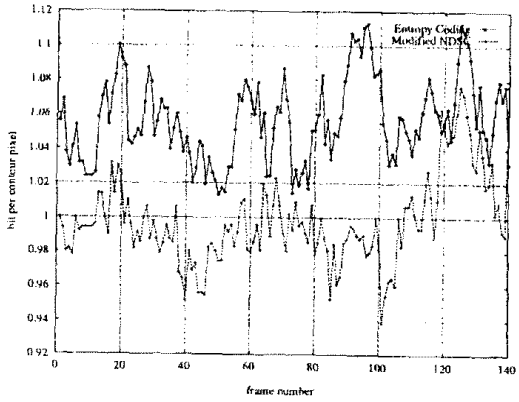
5. 실험 결과

Claire, Foreman, Mother and Daughter 영상을 B/F로 나누어 영역화한[9] 영상에 대해서 실험하였다. 제안하는 알고리듬에 의해 추출된 SP8C 윤곽선에 대해 Entropy Code, Modified NDSC 알고리듬의 성능을 비교하였다. 사용한 Criterion은 윤곽선 픽셀당 소요되는 평균 비트량을 이용하였으며, SP 8C 윤곽선 영상에서 실제 부호화해야 하는 윤곽선 픽셀인 RCP의 갯

수만을 세었다. 즉 (비트수/RCP 수)를 Criterion으로 이용하였다. 그림 6에 Claire, Foreman 동영상에 대한 결과를 비교하였다.



(a) Claire sequence



(b) Foreman sequence

그림 6: Performance of intra-frame coding of SP 8-connected contour.

다음 표 1에 첫 번째 프레임을 부호화하는데 드는 비트량과 140프레임을 8.33Hz로 하여 프레임간 상관성을 이용하지 않은 채 부호화하였을 때 드는 평균 윤곽선 픽셀당 비트 수를 나타내었다.

실험 결과에서 나타나듯이 부호화 비트 수가 기존의 알려져 있던 1.3bpp(bit per contour pixel)보다 낮은 0.97 ~ 1.16 bpp 정도로 나타나는데, 이것은 SP8C 윤곽선의 특성을 이용한 부호화 기법에 기인한다. 실험 결과에서 entropy 부호화보다 Modified NDSC 기법이 더 뛰어남을 알 수 있는데, Modified NDSC 기법에서 부가 비트인 새그먼트 길이 부호화가 대각선 방향을 시작할 때 드는 패러티 비트 부호화보다 더 효율적임을 알 수 있다.

6. 결론

본 논문에서는 기존의 윤곽선 추출 기법보다 효율적으로 윤곽선을 추출하고, 정보를 압축할 수 있는 기법을 제안하였다. 기존의 윤곽선 추출, 부호화 기법의 성능과 제안하는 기법을 이용하여 윤곽선을 추출하고, 윤곽선이 가지는 방향 제한의 성질을 이용하여 부호화를 하였을 때의 성능을 entropy 측면에서 비교하였다. 제안하는 기법을 이용하였을 때, 대각선 방향에서 효율적인 부호화를 수행하여 약 50%의 뛰어난 성능을 지님을 보였다. 본 논문에서 제안한 기법을 이용하여, 현재 연구가 활발히 진행되고 있는 객체 기반형 부호화기의 성능을 향상시킬 수 있을 것으로 기대된다.

표 1: Simulation result of intra-frame contour coding algorithms

Image	Claire	Foreman	M&D
1st frame CP No.	371	482	574
Entropy Coding(bits)	428	509	575
Modified NDSC(bits)	378	482	530
Entropy Coding(bpp)	1.158	1.057	1.087
Modified NDSC(bpp)	1.005	1.0	0.968

7. 참고문헌

[1] H. Freeman, "On the encoding of arbitrary geometric configurations," *IRE Trans. Electron. Comput.*, Vol. EC-10, pp. 260-268, June 1961.

[2] T. Kaneko and M. Okudaira, "Encoding of arbitrary curves based on chain code representation," *IEEE Trans. on Communications*, Vol. COM-33, pp. 697-707, July 1985.

[3] B. B. Chaudhuri and S. Chandrashekhar, "Neighboring direction runlength coding: An efficient contour coding scheme," *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 20, pp. 916-921, July 1990.

[4] C. C. Lu and J. G. Dunham, "Highly efficient coding schemes for contour lines based on chain code representations," *IEEE Trans. on Communications*, Vol. 39, No. 10, pp. 1511-1514, Oct. 1991.

[5] M. Eden and M. Kocher, "On the performance of contour coding algorithm in the context of image coding. part I: Contour segment coding," *Signal Processing*, Vol. 8, pp. 381-386, 1985.

[6] C. Gu and M. Kunt, "Contour simplification and motion compensated coding," *Special Issue of Signal Processing: Image Communication on Coding Techniques for Very Low Bitrate Video*, Vol. 7, No. 4-6, pp. 279-296, Nov. 1995.

[7] V.A. Christopoulos, C.A. Christopoulos, J. Cornelis, A.N. Skodras, "A new contour simplication filter for region-based coding," *EUSIPCO-96*, September, 1996.

[8] F. Marqués, J. Saueda and T. Gasull, "Shape and location coding for contour images," In *Proc. of PCS'93*, pp. 18.6.1-18.6.2, Lausanne, Switzerland, March 1993.

[9] P. Kauff, U. Gölz, S. Kruse and S. Rautenberg, "Improved image segmentation technique for hybrid waveform/object-oriented coding," in *Proc. of SPIE-VCIP '94*, Chicago, IL, pp. 1987-1998, September 1994.