이동 단말기에의 도로 네트워크 가시화†

Visualization of a road network on the mobile terminal

이정훈, 박경린
Junghoon Lee, Gyung-Leen Park
제주대학교 전산통계학과
{jhlee, glpark}@cheju.ac.kr

요 약

본 논문은 쉐이프 파일로 저장되어 있는 도로 네트워크를 분석하여 기억장소 요구량을 최소화하기 위해 다중인접 리스트 형태의 그래프로 변환하여 저장한 후 단말기 화면에 도시한다. 경위도 좌표를 단말기 화면 구성에 맞도록 사상하는 함수와 그 역함수를 구현하여 화면좌표계 상에서 스타일러스 입력 처리, Zoom in, Zoom out, Pan 등의 지도 연산을 수행한다. 또 A*에 기반한 경량급 경로 찾기 기능을 구현하여 단말기의 자원을 효율적으로 사용할 수 있도록 한다.

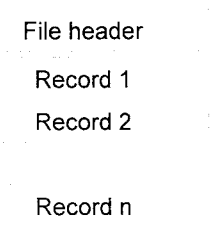
1. 서론

PDA, 텔레매틱스 단말기 등 모바일 장치들은 PC에 비해 CPU나 메모리 등의 자원이 한정적이며 화면의 크기 등 입출력 장치의 기능이 역시 제한적이다. 이러한 단말기에 정보를 탑재하기 위해서는 자료의 메모리 요구량을 최소화하고 단말기의 입출력 장치 특성에 맞추어 변환하여야 한다. 한편, 차량 텔레매틱스 장치의 보급에 따라 네비게이션 등 지리 정보에 대한 요구사항이 증대하였다[1]. 이러한 서비스를 효율적으로 제공하기 위해서는 도로 네트워크가 적은 메모리만으로 단말기에 탑재되어야 하며 지도의 정보가 작은 단말기 화면에 맞추어 제공되어야 한다.

본 논문에서는 첫째, 도로 네트워크를 포함하고 있는 쉐이프 파일을 분석하여 각 링크의 양 끝점만을 추출한 후 이를 다중인접 리스트에 기반한 그래프로 변환한다. 둘째, 단말기에 이 좌표점들에 따라 도로 네트워크를 도시하도록 하며 Zoom in, Zoom out, Pan 등의 기본적인 지도 연산들과 스타일러스 입력 처리를 구현한다. 셋째, 출발지와 목적지를 설정하고 A*에 기반한 경로를 계산한다[2].

2. 쉐이프 파일의 분석

도로 네트워크는 노드에 대한 정보와 링크에 대한 정보로 이루어진다. 노드는 각각의 교차로에 해당하여 하나의 점으로 표현되는 반면 링크는 교차로와 교차로의 길의 형태를 표현한 것으로 각 링크는 서로 다른 개수의 점들로 구성된다. 각 쉐이프 파일은 <그림 1>에서 보는 바와 같이 전체 파일의 정보를 나타내는 파일 헤더와 각각의 레코드들로 표현된다.



<그림 1> 쉐이프 파일 구조

† 본 연구는 지식경제부 및 정보통신진흥연구원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음 (IITA-2008-C1090-0040)

파일 헤더는 <그림 2>에서 보는바와 같이 100바이트로 구성되며 전체 파일에 포함된 도형 객체들에 있어서 경위도 좌표의 최대 최소값을 포함하고 있다. <그림 2>는 각 쉐이프 파일의 헤더를 읽어 분석하기 위한 자료구조를 정의한 것이다.

```
struct t_header {
    int fileCode
    int unused1, unused2, unused3, unused4;
    int unused5;
    int fileLength
    int version;
    int shapeType
    double xMin, yMin, xMax, yMax;
    double zMin, zMax, mMin, mMax
};
```

<그림 2> 쉐이프 파일 헤더의 구조

노드 파일에 있어서는 각 레코드는 하나의 점만을 포함하므로 길이는 모두 같다. 링크의 경우는 서로 다른 개수의 점들로 구성되므로 이에 대한 정보를 먼저 읽은 후 나머지 점들에 대한 좌표를 모두 읽어야 한다. <그림 3>은 이러한 과정을 보여주고 있으며 파일의 헤더를 모두 읽어들인 후 각 레코드 마다 레코드 헤더를 소진한 다음 <그림 3>의 코드를 수행시킨다. 여기서 쉐이프 파일은 파트의 개수, 점의 개수를 차례로 저장하므로 numPart와 numPoints라는 변수에 각각의 개수를 저장하고 이 수에 좌표의 길이를 곱하여 x, y좌표로 구성된 point 구조체에 각 점의 좌표를 읽어들인다.

```
ReadFile (fp, (unsigned char *) &numPart,
    sizeof(int), &cBytes, NULL);
ReadFile (fp, (unsigned char *) &numPoints,
    sizeof(int), &cBytes, NULL);
ReadFile(fp, dummy, numPart * sizeof (int),
    &cBytes, NULL);
ReadFile(fp, (unsigned char *) &(point[0]),
    numPoints * sizeof (struct t_point),
    &cBytes, NULL);
```

<그림 3> 좌표들의 처리

이와 같은 과정으로 쉐이프 파일에 대한 처리가 완료될 수 있으며 각 레코드에 있어서 양끝점만을 처리하여 다중인접 리스트를 구성할 수 있다. 단, 노드 쉐이프 파일의 경우 작성시의 오류로 인해 중복된 노드가 포함되거나 링크가 하나도 연결되지 않은 노드가 일부 포함될 수 있다. 따라서 링크 파일에서 끝점의 좌표를 얻어 노드 정보를 검색하여 제일 먼저 찾아지는 점을 링크의 노드로 설정한다. 이와 같은 과정은 중복저장된 노드를 미참조 노드로 만들며 이후 미참조 노드를 삭제함으로써 오류가 없는 그래프를 완성할 수 있다. 결국, 저장량도 많이 줄어들게 되며, 또 이 자료구조를 텍스트 파일로 저장하여 다양한 타입의 단말기 내의 프로그램에서 이용할 수 있도록 한다.

3. 단말기 프로그램

단말기 프로그램은 이렇게 구성된 텍스트 파일을 받아 각 좌표 마다 <그림 4>에서 보는 바와 같이 경위도 좌표를 화면 좌표계로 변환한다. 이 과정에서 화면 좌표계는 좌측 상단의 점이 원점이므로 x 좌표에 대해서는 최대점, 즉 xMax와 각 점과의 비례식에 의해 화면좌표를 구할 수 있는 반면 y축에 대해서는 최대좌표와 최소좌표가 반전되어 있으므로 현재 화면좌표계의 최대값을 갖는 Vsize에서 계산된 좌표를 빼준다. 또한 PDA의 화면에서는 상위부분에 메뉴가 위치할 수 있으므로 28픽셀을 뺀다. 현재의 화면 크기를 갖고 있는 Hsize와 Vsize 변수는 WM_SIZE 사건에서 변경된다.

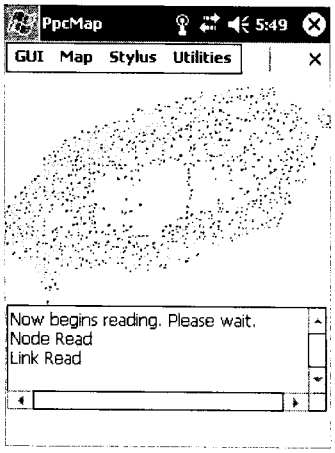
```
void TMtoDev (HWND hwnd,
        double x, double y, int *ix, int *iy) {
    double tmp1, tmp2, tmp3, tmp4;
    tmp1 = (x - xMin) / (xMax - xMin) ;
    tmp3 = tmp1 * ((double) Vsize);
    *ix = (int) tmp3;

    tmp2 = (y - yMin) / (yMax - yMin) ;
    tmp4 = tmp2 * ((double) Vsize);
    *iy = (int) tmp4;
    *iy = Vsize - *iy;
    *iy = *iy + 28;       // 상위 메뉴부분
}
```

<그림 4> 화면좌표계로의 변환

이상의 과정에 의해 제주도의 지도를 단말기의 화면에 도시하면 <그림 5>와 같다. 본 과정에서 사용된 쉐이프 파일은 경위도 좌표로서 WGS 좌표계를 채택하고

있다. 전체적인 화면 구조는 상단에 메뉴가 위치하고 이후 Canvas 윈도우에 지도 정보가 표현된다. 아래 프로그램이 사용자에게 주는 메시지를 위한 편집 윈도우가 생성되었다. 이 윈도우를 통해 사용자에게 프로그램의 현재 상태를 알려줄 수 있을 뿐 아니라 경로 배정 및 기타 기능에 대한 자세한 프로그램 수행 결과가 제공될 수 있다. 이는 사용자의 불필요한 추가 입력을 최소화할 수 있다.



<그림 5> 지도의 표현

이 프로그램에서는 사용자가 메뉴를 선택하거나 캔버스 윈도우에 스타일러스 입력을 할 수 있다. 이 경우 스타일러스에서 선택한 화면 좌표를 지도상에서의 경위도 좌표로 변환한 다음 해당 링크나 노드를 검색하여야 한다. 결국 화면 좌표계를 경위도 좌표로 변환하는 루틴이 필요하며 이는 <그림 4>의 역변환으로서 <그림 6>과 같이 구현된다.

```
void DevtoTM (int ix, int iy,
      double *dx, double *dy){
   double tmp;
   iy -= 28;          // 상위 메뉴부분
   tmp = ((double) ix)/((double) Hsize);
   *dx = tmp * (xMax - xMin) + xMin;
   tmp = ((double) (Vsize - iy))/
         ((double) Vsize);
   *dy = tmp * (yMax - yMin) + yMin;
}
```

<그림 6> 경위도좌표계로의 변환

<그림 4>와 <그림 6>의 좌표변환 함수는 모든 부분에서 이용된다. 이를 바탕으로 스타일러스가 선택한 좌표를 입력받아 출발지와 목적지를 결정하는 부분은 <그림 7>의 GuMarkSource와 GuMarkDest 명령 처리에 해당한다. 이는 현재 스타일러스의 선택을 경위도 좌표로 변환한 후 해당하는 전역변수에 저장한다.

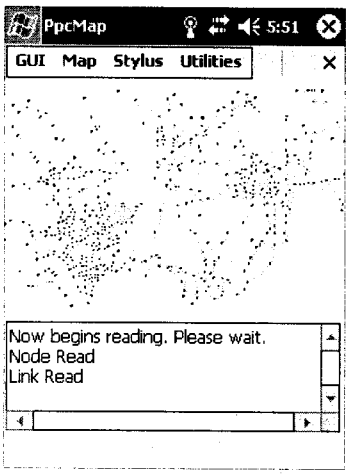
```
LRESULT DoLButUp(HWND hwnd,
   UINT iMsg, WPARAM wParam,
   LPARAM lParam) {
   int     x, y;
   double dx1, dx2, dy1, dy2 ;
   double dx, dy, res;
   x = LOWORD(lParam);
   y = HIWORD(lParam);

   DevtoTM(sx, sy, &dx1, &dy1);
   DevtoTM(x, y, &dx2, &dy2);
   switch (mode) {
     case MmShrink :
          xMin = dx1; yMax = dy1;
          xMax = dx2; yMin = dy2;
          InvalidateRect(hwnd, NULL, TRUE);
          break;
     case MmPan :
          dx = dx1 - dx2;
          dy = dy1 - dy2;
          xMin = xMin + dx;
          xMax = xMax + dx;
          yMin = yMin + dy;
          yMax = yMax + dy;
          InvalidateRect(hwnd, NULL, TRUE);
          break;
     case MmZoom :
          dx = xMax - xMin;
          dy = yMax - yMin;
          dx = dx /4.0;
          dy = dy /4.0;
          xMin -= dx; xMax += dx;
          yMin -= dy; yMax += dy;
          InvalidateRect(hwnd, NULL, TRUE);
          break;
     case GuMarkSource :
          DevtoTM(x, y, &dx, &dy);
          MarkSource.x = dx;
          MarkSource.y = dy;
          DisplayRect(MarkSource, BlueNode);
          break;
     case GuMarkDest :
          DevtoTM(x, y, &dx, &dy);
```

```
            MarkDest.x = dx;
            MarkDest.y = dy;
            DisplayRect(MarkDest, RedNode);
            break;
        default : break;
        }
        return 0;
}
```

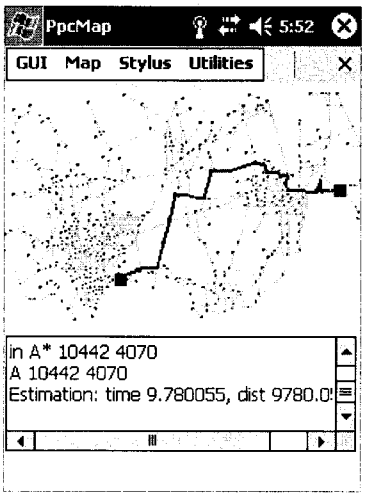
<그림 7> 좌표계 연산

<그림 7>에서는 또 화면상에서 특정 부분을 Zoom in 또는 Zoom out 하거나 View point를 옆으로 이동하는 Pan에 대한 처리 방법을 보이고 있다.



<그림 8> Zoom in의 결과

스타일러스로 사각형을 그리면 스타일러스가 모니터에 닿는 순간 MouseDown, 떼는 순간 MouseUp 사건이 발생하는데 이 사각형의 화면 좌표는 (sx, sy, x, y)라는 변수에 저장된다. 이를 처리하는 부분은 MnShrink, MnPan, MnZoom 등인데 공통적으로 xMax, xMin, yMax, yMin을 변경하고 있다. 한 예로 Zoom in의 경우 사각형의 화면좌표를 경위도 좌표로 바꾸어 최대최소값을 설정하고 화면을 재도시한다. 이 결과가 <그림 8>에 나타나있다. 화면 좌표의 변경 기능이외에 출발지와 목적지에서 제일 가까운 노드를 검색하고 이 노드들간의 최단 경로를 구하는 기능이 추가되었다. Dijkstra의 경우는 노드의 수가 많아지면 성능이 저하되어 CPU 성능이 낮은 단말기에서 응답시간이 문제가 된다. 따라서 항상 정답을 찾는 것은 아니지만 빠른 속도로 적당한 수준의 답을 발견할 수 있는 A* 알고리즘을 구현하였으며 이에 대한 경로 표시를 하도록 하였다. 그 결과는 <그림 9>에 보이고 있다.



<그림 9> A*에 의한 경로찾기

4. 결론

본 논문에서는 쉐이프 파일로 저장되어 있는 도로 네트워크를 분석하여 경량급의 다중인접 리스트로 변환하고 이를 단말기에서 읽어 화면에 지도 정보를 표시하였다. 이와 아울러 화면좌표와 경위도좌표간에 효율적인 변환 함수를 구현하여 화면좌표계 상에서 스타일러스 처리, Zoom in, Zoom out, Pan 등의 기능을 구현하였으며 A*에 기반한 경량급 경로 찾기 기능을 구현하여 단말기의 자원을 효율적으로 사용할 수 있도록 하였다.

참고문헌

[1] P. Green, "Driver Distraction, Telematics Design, and Workload Managers-Safety Issues and Solutions," Proc. Int'l Congress on Transportation Electronics pp. 165-180, 2004.

[2] A. Goldberg, H. Kaplan, and R. Werneck, "Reach for A*: Efficient point-to-point shortest path algorithms," MSR-TR-2005-132. Microsoft, 2005.