

임베디드 시스템을 위한 GPS/INS 결합 알고리즘의 실시간 동작 구현

Real-time implementation of GPS/INS integration algorithm for embedded systems

이재훈¹ · 김도윤² · 성상학² · 남동균¹ · 박주영¹

Jaehoon Lee, Do-Yoon Kim, Sanghak Sung, Dongkyun Nam and Jooyoung Park

¹고려대학교 제어계측공학과

E-mail: {white8704, skadas, parkj}@korea.ac.kr

²위드로봇

E-mail: {getcome, gotajji}@gmail.com

요 약

최근들어 MEMS 센서의 가격대비 성능 향상에 따라 INS와 GPS 센서 융합을 이용한 차량의 위치 및 자세를 결정하는 문제에 대한 관심이 증대되고 있다. 실제 시스템에 적용하기 위해서는 저가의 임베디드 시스템에서 실시간 동작이 보장되어야 하는 조건을 충족시켜야 한다. 본 논문에서는 GPS/INS 결합 문제를 정리한 후, 임베디드 시스템이 지원하는 여러 부가 기능들을 이용하여 해당 알고리즘이 의미 있는 서비스 시간 내에 동작할 수 있는지를 확인한다.

키워드 : Kalman Filter, GPS, INS, Embedded systems

1. 서 론

자신의 위치를 찾기 위한 위치 추적 방식은 크게 두 가지로 구분된다. 첫 번째는 초기 위치정보로부터 동체의 가속도 및 방향 측정치를 이용하여 현재의 위치를 계산하는 관성항법시스템(INS, Inertial Navigation System)이고 두 번째는 지구주위를 주기적으로 도는 위성을 통해 자신의 위치를 찾는 위성항법시스템(GPS, Global Positioning System)이다.

관성항법시스템(INS)은 짧은 기간의 오차는 작으나 센서에서 얻은 가속도 값을 적분하여 사용하기 때문에 시간이 지날수록 오차가 누적되는 특성이 있다. 하지만 위성항법시스템(GPS)은 순간적인 오차는 관성항법시스템(INS)에 비하여 크지만 시간이 지나도 오차가 누적되지 않는 특성을 지니고 있다. 따라서 관성항법시스템(INS)과 위성항법시스템(GPS)은 상호 보완적으로 적절히 결합해야 보다 정확한 동체(차량)의 위치를 찾을 수 있다.

GPS 신호는 1Hz로 갱신이 되기 때문에 INS 연산은 최소 이보다 10배 이상 빠른 주기로 계산이 되어야 의미가 있으며, 최근 차량 위치 기반 서비스에서는 100Hz 이상의 속도로 자신의 위치를 파악하는 요구가 늘어나고 있다. 본 논문에서는 INS/GPS 결합 문제를 정리한 후, 최근 들어 많은 기술적인 진보를 이룬 프로세서 설계 기술, MEMS 센서에 기반하여 INS와 GPS 센서융합을 이용한 차량의 위치 및 자세를 결정하는 하드웨어 및 소프트웨어

트웨어를 구현하고 시장에서 요구하는 실시간 동작을 확인한다.

2. 관성항법시스템(INS)

항법 알고리즘은 동체 좌표계의 관성측정장치(IMU, Inertial Measurement Unit)에서 얻은 센서 출력 값을 이용해 항체의 자세(Attitude), 속도, 위치의 정보를 계산하는 알고리즘이다. 본 절에서는 [1][5][6]의 내용에 기반을 둔 관성항법시스템 알고리즘에 대하여 정리한다.

먼저 동체 좌표계(Body frame)에 부착된 IMU는 3축 자이로와 3축 가속도계로 이루어진다. 항법 좌표계에서 본 동체 좌표계의 각도 변화율은 식 (1)과 같다[5][6].

$$\begin{aligned} \Delta\theta_{bb}^b &= (\Delta\theta_x \ \Delta\theta_y \ \Delta\theta_z)^T \\ &= \Delta\theta_{bb}^b - C_n^b (\alpha_n^b + \omega_n^b) \Delta t \end{aligned} \quad (1)$$

$$\Delta\theta = \sqrt{\Delta\theta_x^2 + \Delta\theta_y^2 + \Delta\theta_z^2} \quad (2)$$

각도 변화율의 크기는 식 (2)와 같이 계산되고, 여기서 얻어진 식 (1)과 식 (2)를 이용해 쿼터니언 자세계산식을 식 (3)과 같이 얻을 수 있다[5][6].

$$q_{k+1} = q_k + 0.5 \begin{pmatrix} c & s\Delta\theta_x & -s\Delta\theta_y & s\Delta\theta_z \\ -s\Delta\theta_x & c & s\Delta\theta_x & s\Delta\theta_y \\ s\Delta\theta_y & -s\Delta\theta_x & c & s\Delta\theta_z \\ -s\Delta\theta_x & -s\Delta\theta_y & -s\Delta\theta_z & c \end{pmatrix} q_k \quad (3)$$

여기서 s 와 c 는 아래와 같이 구해진다[5][6].

$$s = \frac{2}{\Delta\theta} \sin \frac{\Delta\theta}{2} = 1 - \frac{\Delta\theta^2}{24} + \frac{\Delta\theta^4}{1920} + \dots$$

$$c = 2 \left(\cos \frac{\Delta\theta}{2} - 1 \right) = -\frac{\Delta\theta^2}{4} + \frac{\Delta\theta^4}{192} + \dots \quad (4)$$

마지막으로 쿼터니언 자세계산식에서 얻은 쿼터니언으로 식 (5)을 이용해 방향코사인행렬(DCM)을 구한다[6].

$$C_b^n = \begin{pmatrix} (q_1^2 - q_2^2 - q_3^2 + q_4^2) & 2(q_1q_2 - q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 + q_3q_4) & q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 - q_1q_4) \\ 2(q_1q_3 - q_2q_4) & 2(q_2q_3 - q_1q_4) & q_1^2 - q_2^2 - q_3^2 + q_4^2 \end{pmatrix} \quad (5)$$

가속도계로 출력된 속도 변화량(Velocity increment)은 동체좌표계에 대한 값으로써 DCM을 통해 항법 좌표계로 변환시켜줘야 한다[5][6].

$$\Delta \underline{v}_y^n = C_b^n \begin{pmatrix} 1 & 0.5\Delta\theta_x & -0.5\Delta\theta_y \\ -0.5\Delta\theta_x & 1 & 0.5\Delta\theta_y \\ 0.5\Delta\theta_y & -0.5\Delta\theta_x & 1 \end{pmatrix} \Delta \underline{v}_y^b \quad (6)$$

여기서 1차 스컬링(Sculling) 보정이 이루어진다. 그리고 코리올리의 힘(Coriolis force)과 중력(Gravity)모델이 적용된 속도 변화량은 식 (7)과 같다[1][6].

$$\Delta \underline{v}^n = \Delta \underline{v}_y^n - (2\underline{\omega}_{ie}^n + \underline{\omega}_{en}^n) \times \underline{v}^n \Delta t + \underline{\gamma}^n \Delta t \quad (7)$$

여기서 $\underline{\gamma}^n = (0 \ 0 \ \gamma)^T$, γ 은 중력모델로 식 (8)과 같다[5].

$$\gamma = a_1(1 + a_2 \sin^2 \varphi + a_3 \sin^4 \varphi) + (a_4 + a_5 \sin^2 \varphi)h + a_6 h^2$$

$$\begin{aligned} a_1 &= 9.7803267715 & a_4 &= -0.0000030876910891 \\ a_2 &= 0.0052790414 & a_5 &= 0.0000000043977311 \\ a_3 &= 0.0000232718 & a_6 &= 0.0000000000007211 \end{aligned} \quad (8)$$

그리고 지구 자전 각속도 식과 코리올리의 힘(Coriolis force)에 의해 항법 좌표계(Navigation frame)가 편향되는 식은 각각 식 (9)과 (10)와 같다[1].

$$\underline{\omega}_{ie}^n = C_e^n \underline{\omega}_{ie}^e = (\omega_{ie} \cos \varphi \ 0 \ -\omega_{ie} \sin \varphi)^T \quad (9)$$

$$\underline{\omega}_{en}^n = (\dot{\lambda} \cos \varphi \ -\dot{\varphi} \ -\dot{\lambda} \sin \varphi)^T \quad (10)$$

따라서 속도와 위치는 각각 식 (11)과 식 (12)에 의해 구해진다[5][6][7].

$$\underline{v}_{k+1}^n = \underline{v}_k^n + \Delta \underline{v}_{k+1}^n \quad (11)$$

$$\underline{r}_{k+1}^n = \underline{r}_k^n + 0.5 \begin{pmatrix} 1 & 0 & 0 \\ R_N + h & 0 & 0 \\ 0 & \frac{1}{(R_E + h) \cos \varphi} & 0 \\ 0 & 0 & -1 \end{pmatrix} (\underline{v}_k^n + \underline{v}_{k+1}^n) \Delta t \quad (12)$$

여기서 위치는 $\underline{r}^n = (\varphi \ \lambda \ h)^T$ 로 각각 위도, 경도, 고도를 나타내고 R_N 과 R_E 는 각각 북쪽 자오선 방향과 동쪽 방향 곡률의 반경을 의미한다.

지금까지 설명한 INS(관성항법시스템)의 알고리즘 구

성도는 그림 1과 같다.[1]

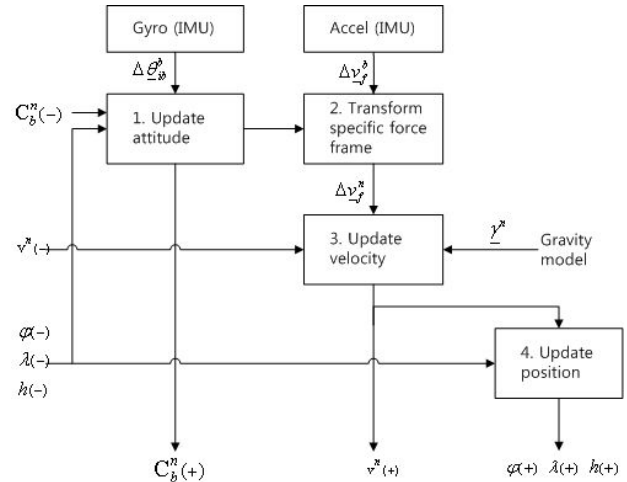


그림 1. 관성항법시스템의 알고리즘 구성도.

Fig 1. Block diagram of Inertial Navigation system.

3. 칼만 필터를 이용한 INS/GPS 융합

앞에서 살펴본 INS는 연속적인 항법 해를 제공해 동적 특성이 좋은 장점을 가지고 있다. 그러나 항법 계산 과정에서 적분에 의해 항법 오차가 누적되어 시간이 지날수록 오차가 증가하는 단점이 있다.

반면에 GPS는 인공위성에 의해 오차의 증가 없이 일정한 범위 내에서 비교적 정확한 항법 해를 제공하지만 계산주기가 1Hz정도로 INS에 비해 상대적으로 길고 지형적 특징에 따라 항법 해를 계산하지 못하는 단점이 있다. 따라서 상호 보완적으로 두 신호의 장점을 융합하게 되면 살려 전체적인 시스템 성능을 향상시킬 수 있다.

융합 알고리즘으로 본 논문에서는 칼만필터를 고려한다. 칼만필터는 상태변수(\underline{x})의 최적 추정치($\hat{\underline{x}}$)를 구하는 방법으로 상태변수의 오차분산 $E[(\underline{x} - \hat{\underline{x}})(\underline{x} - \hat{\underline{x}})^T]$ 을 최소화하는 필터이다. 비선형 시스템에 적용할 경우 보통 확장 칼만필터(Extended Kalman filter)를 사용한다.

관성항법시스템의 동특성을 기반으로 하여 위치, 속도 및 자세에 대한 오차를 상태벡터로 삼아 유도한 칼만 필터의 연속 시스템 모델(Continuous system model)은 식 (13)와 같다[2][3].

$$\dot{\underline{x}} = \underline{F}\underline{x} + \underline{G}\underline{u} \quad (13)$$

여기서 \underline{F} 는 시스템 행렬(Dynamic matrix), \underline{G} 는 디자인 행렬(Design matrix)이다[5].

$$F = \begin{pmatrix} F_{rr} & F_{rv} & 0 \\ F_{vr} & F_{vv} & (f^n \times) \\ F_{er} & F_{ev} & -(\underline{\omega}_n^n \times) \end{pmatrix} \quad \underline{x} = \begin{pmatrix} \delta r^n \\ \delta v^n \\ \underline{e}^n \end{pmatrix} \quad (14)$$

$$G = \begin{pmatrix} 0 & 0 \\ C_b^n & 0 \\ 0 & -C_b^n \end{pmatrix} \quad \underline{u} = \begin{pmatrix} \delta f_b^b \\ \delta \underline{\omega}_b^b \end{pmatrix}$$

여기서 위치 오차 역학식(Position error dynamics)관련 변수는 식 (15), (16)와 같다[4][5][7].

$$F_{rr} = \begin{pmatrix} \frac{\partial \phi}{\partial \phi} & \frac{\partial \phi}{\partial \lambda} & \frac{\partial \phi}{\partial h} \\ \frac{\partial \lambda}{\partial \phi} & \frac{\partial \lambda}{\partial \lambda} & \frac{\partial \lambda}{\partial h} \\ \frac{\partial h}{\partial \phi} & \frac{\partial h}{\partial \lambda} & \frac{\partial h}{\partial h} \end{pmatrix} = \begin{pmatrix} 0 & 0 & \frac{-v_N}{(R_E+h)^2} \\ \frac{v_E \sin \phi}{(R_E+h) \cos^2 \phi} & 0 & \frac{-v_E}{(R_E+h)^2 \cos^2 \phi} \\ 0 & 0 & 0 \end{pmatrix} \quad (15)$$

$$F_{rv} = \begin{pmatrix} \frac{\partial \phi}{\partial v_N} & \frac{\partial \phi}{\partial v_E} & \frac{\partial \phi}{\partial v_D} \\ \frac{\partial \lambda}{\partial v_N} & \frac{\partial \lambda}{\partial v_E} & \frac{\partial \lambda}{\partial v_D} \\ \frac{\partial h}{\partial v_N} & \frac{\partial h}{\partial v_E} & \frac{\partial h}{\partial v_D} \end{pmatrix} = \begin{pmatrix} \frac{1}{R_N+h} & 0 & 0 \\ 0 & \frac{1}{(R_E+h) \cos \phi} & 0 \\ 0 & 0 & -1 \end{pmatrix} \quad (16)$$

그리고 속도 오차 역학식(Velocity error dynamics)관련 변수는 식 (17), (18)과 같다[4][5][7].

$$F_{vr} = \begin{pmatrix} \frac{-2v_E \omega_{ie} \cos \phi}{\frac{v_E^2}{(R_E+h) \cos^2 \phi}} & 0 & \frac{-v_N v_D}{(R_N+h)^2} + \frac{v_E^2 \tan \phi}{(R_E+h)^2} \\ \frac{2\omega_{ie} (v_N \cos \phi - v_D \sin \phi)}{+ \frac{v_E v_N}{(R_E+h) \cos^2 \phi}} & 0 & \frac{-v_E v_D}{(R_E+h)^2} - \frac{v_N v_E \tan \phi}{(R_E+h)^2} \\ \frac{2v_E \omega_{ie} \sin \phi}{\frac{v_E^2}{(R_E+h)^2} + \frac{v_N^2}{(R_N+h)^2}} & 0 & \frac{v_E^2}{(R_E+h)^2} + \frac{v_N^2}{(R_N+h)^2} - 2\gamma/(R+h) \end{pmatrix} \quad (17)$$

$$F_{vv} = \begin{pmatrix} \frac{v_D}{R_N+h} & \frac{-2\omega_{ie} \sin \phi}{-2 \frac{v_E \tan \phi}{R_E+h}} & \frac{v_N}{R_N+h} \\ \frac{2\omega_{ie} \sin \phi}{+ \frac{v_E \tan \phi}{R_E+h}} & \frac{v_D + v_N \tan \phi}{R_E+h} & \frac{2\omega_{ie} \cos \phi + \frac{v_E}{R_E+h}} \\ \frac{-2v_N}{R_N+h} & \frac{-2\omega_{ie} \cos \phi - \frac{2v_E}{R_E+h}}{-2\omega_{ie} \cos \phi - \frac{2v_E}{R_E+h}} & 0 \end{pmatrix} \quad (18)$$

그리고 자세 오차 역학식(Attitude error dynamics)관련 변수는 식 (19), (20)와 같다[4][5][7].

$$F_{er} = \begin{pmatrix} -\omega_{ie} \sin \phi & 0 & \frac{-v_E}{(R_E+h)^2} \\ 0 & 0 & \frac{v_N}{(R_N+h)^2} \\ -\omega_{ie} \cos \phi - \frac{v_E}{(R_E+h) \cos^2 \phi} & 0 & \frac{v_E \tan \phi}{(R_E+h)^2} \end{pmatrix} \quad (19)$$

$$F_{ev} = \begin{pmatrix} 0 & \frac{1}{R_E+h} & 0 \\ \frac{-1}{R_N+h} & 0 & 0 \\ 0 & \frac{-\tan \phi}{R_E+h} & 0 \end{pmatrix} \quad (20)$$

칼만필터의 진행과정은 두 단계, 갱신(Update)과 예측(Prediction)으로 나뉜다. 먼저 갱신(Update) 과정에서 칼만이득(Kalman gain)은 식 (21)과 같이 정해진다. 그리고 추정치와 측정치에 의한 오차공분산 행렬은 각각 식 (22)과 식 (23)와 같이 갱신된다[2][3][7].

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \quad (21)$$

$$\hat{\underline{x}}_k = \hat{\underline{x}}_k^- + K_k (\underline{z}_k - H_k \hat{\underline{x}}_k^-) \quad (22)$$

$$P_k = (I - K_k H_k) P_k^- \quad (23)$$

예측(Prediction) 과정에서는 다음 스텝(Next step)에서 이용하게 될 추정치와 시간 전파된 오차공분산 행렬이 식 (24)과 식 (25)에서 구해진다[2][3][7].

$$\hat{\underline{x}}_{k+1}^- = \Phi_k \hat{\underline{x}}_k \quad (24)$$

$$P_{k+1}^- = \Phi_k P_k \Phi_k^T + Q_k \quad (25)$$

여기서 Φ_k 는 상태 변환 행렬(State transition matrix)으로 식 (26)과 같다[5][6].

$$\Phi_k = \exp(F \Delta t) \approx I + F \Delta t \quad (26)$$

GPS에서 받은 위치와 속도 정보는 측정벡터를 구하는데 사용된다. 측정벡터 \underline{z}_k 는 INS와 GPS의 위치, 속도 정보의 차이로 식 (27)과 같고, H_k 는 식 (28)과 같다.[5][6][7]

$$\underline{z}_k = \begin{pmatrix} (R_N+h)(\phi_{INS} - \phi_{GPS}) \\ (R_E+h) \cos \phi (\lambda_{INS} - \lambda_{GPS}) \\ h_{INS} - h_{GPS} \\ \underline{v}_{N,INS}^n - \underline{v}_{N,GPS}^n \\ \underline{v}_{E,INS}^n - \underline{v}_{E,GPS}^n \\ \underline{v}_{D,INS}^n - \underline{v}_{D,GPS}^n \end{pmatrix} \quad (27)$$

$$H_k = \begin{pmatrix} (R_N+h) & 0 & 0 & 0_{3 \times 3} & 0_{3 \times 3} \\ 0 & (R_E+h) \cos \phi & 0 & 0_{3 \times 3} & 0_{3 \times 3} \\ 0 & 0 & 1 & 0_{3 \times 3} & 0_{3 \times 3} \\ 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} & I_{3 \times 3} & 0_{3 \times 3} \end{pmatrix} \quad (28)$$

INS/GPS 결합 알고리즘의 전체 과정은 그림 2와 같다. 이는 되먹임(Feedback) 방식으로 칼만필터 추정치를 되먹임 함으로써 오차를 보정하여 준다.[1][4]

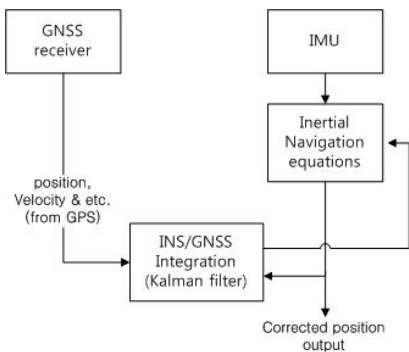


그림 2. 되먹임 방식을 이용한 INS/GPS 결합 알고리즘.

Fig 2. Block diagram of INS/GPS Integration Algorithm Based on Feedback strategy.

4. 실험 환경 및 임베디드 시스템을 고려한 구현과정에 대한 고찰

이 절에서는 앞에서 정리한 알고리즘이 UVS(유비쿼터스 차량센서)의 하드웨어 장치에서 실시간으로 동작할 수 있도록 알고리즘의 구성을 하드웨어 구성에 알맞게 작성한다. 본 실험환경에 사용된 가속도계, 자이로 및 GPS는 각각 LIS3LV02DQ, ADIS16100 그리고 UST-SNR-GPS이다. 그리고 시스템 구현 시 사용한 개발보드는 그림 3과 같다.

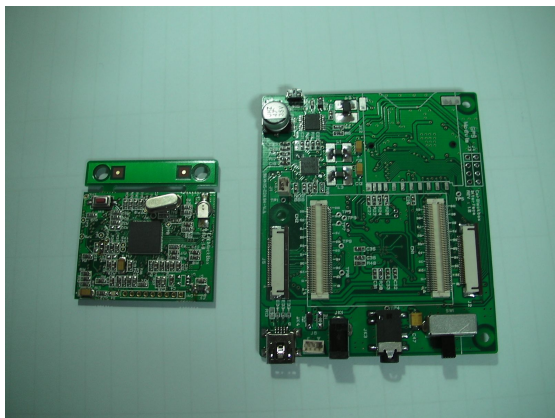


그림 3. 시스템 구현에 사용한 개발보드.

Fig 3. Target board for system implementation.

먼저 본 연구에서 사용하는 실험환경의 프로세서 특성은 아래 표 1과 같다[8].

- Cortex-A8은 향상된 SIMD(NEON) ARM V7 최신 명령어를 지원.
- 계층화된 캐시 메모리를 지원 (L2 캐시는 1MB까지 지원).
- 메모리 폭은 128비트까지 지원.
- 13단계의 파이프라인과 향상된 분기 예측 엔진.

표 1. 프로세서 특성.

Table 1. Features of Processor.

NEON 기능을 활용하여 하나의 명령어로 복합적인 기능을 수행하는 SIMD 명령어를 적극적으로 사용할수록

성능 향상이 극대화 됨을 볼 수 있었다. 여기서 C 컴파일러가 자동으로 NEON 명령어로 변경하지는 못하고, 개발자가 어셈블리 레벨에서 작업을 수행하거나 C 코드를 작성할 때 컴파일러가 Neon 명령어를 지원하게끔 compiler-friendly 형식으로 코드를 제작하여야 한다. 그리고 자주 실행하는 연산을 어셈블리 레벨에서 라이브러리로 제작하여 기능을 지원해야 한다.

Neon 명령어를 사용하여 제작한 라이브러리의 특징은 아래 표 2와 같다.

- 프로파일링 기법을 이용하여 연산을 분석하면 80% 이상이 행렬의 곱셈, 덧셈으로 구성되어 있음.
- 알고리즘에서 자주 사용되는 행렬 연산을 NEON 명령어로 구성하여 라이브러리화.
- ARMv6에서 MUL 명령어 1개, QADD 명령어 2개는 VQDMLAL 명령어 1개로 구현할 수 있음. 3 사이클의 수행 시간이 1 사이클로 3배 빨라지는 효과를 얻을 수 있음.

표 2. 라이브러리 특징.

Table 2. Features of Library.

NEON 명령어를 사용할 경우 단순 산술 연산에서는 3배 이상의 상승 효과를 볼 수 있었고, 정수 연산은 9배, 실수 연산은 5배의 성능 향상을 확인 하였다. 앞에서 정리한 알고리즘은 UIS-MPU에서 최적화를 하지 않은 경우 0.0197초 소요되며 NEON 명령어를 사용한 라이브러리를 이용한 경우 0.0021초 소요되었다. 연산량 측면에서 470Hz 속도로 동작이 가능하므로, 현재 100Hz의 속도 센서 신호를 획득하는 INS/GPS 결합 시스템에 적용이 가능하다. 전체 시스템을 위한 적용결과 예 중 하나가 그림 4에 보여 졌다.

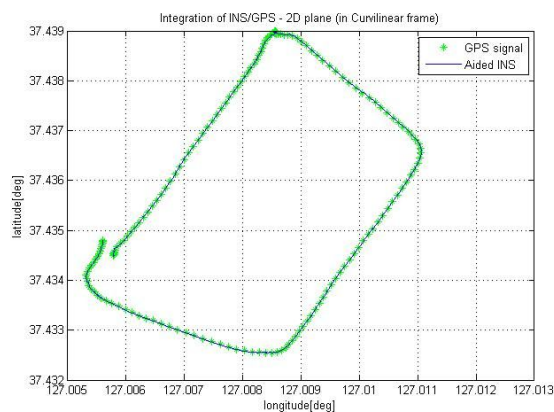


그림 4. 되먹임 방식을 이용한 INS/GPS 결합 결과.

Fig 4. Result of INS/GPS Integration Algorithm.

5. 결론 및 향후 계획

본 논문에서는 INS/GPS 결합 문제를 정리한 후 본 연구의 실험환경에 사용된 임베디드 시스템에 적용이 적

합한 형태로의 구현에 대하여 고찰해 보았다. 주요 관찰 대상은 INS/GPS 결합 알고리즘의 동작 속도이며 임베디드 시스템으로 구현하여 실시간(100Hz)으로 결합 알고리즘이 동작하는 것을 확인할 수 있었다. 본 연구 결과는 u-TSN 기반 기술로 사용될 것으로 기대되며, 하드웨어 리소스에 여5가 있어 향후 좀 더 향상된 알고리즘도 실시간으로 동작시킬 수 있을 것으로 예측된다.

감사의 글 : 본 연구는 2008년 u-Transportation 기반기술 개발, 과제번호: 06-교통핵심-A01-01에 의해 수행되었습니다, 연구비 지원에 감사드립니다.

참 고 문 헌

- [1] P. D. Groves, *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*, Artech house, 2007.
- [2] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, John Wiley & Sons, 1996.
- [3] J. A. Farrell, *Aided Navigation: GPS with High Rate Sensor*, McGraw Hill, 2007.
- [4] V. Kumar N., *Integration of Inertial Navigation System and Global Positioning System Using Kalman Filter*, M.Tech. Dissertation, Department of Aerospace Engineering Indian Institute of Technology, July 2004.
- [5] E. H. Shin, *Accuracy Improvement of Low Cost INS/GPS for Land Application*, MS Thesis, University of Calgary, December 2001.
- [6] A. Solimeno, *Low-Cost INS/GPS Data Fusion with Extended Kalman Filter for Airborne Applications*, MS Thesis, Instituto Superior Tecnico, July 2007.
- [7] V. A. Agarwal, H. Arya, B. Nayak and L. R. Saptarshi, "Extended Kalman Filter Based Loosely Coupled INS/GPS Integration Scheme using FPGA and DSP," *Int. J. Intelligent Defense Support Systems*, Vol. 1, No.1, pp.5-26, 2008.
- [8] ARM web, "ARM Cortex A8," http://www.arm.com/products/CPUs/ARM_Cortex-A8.html.