ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1994
Seria: AUTOMATYKA z.115 Nr kol. 1251

Damian DOBROCZYŃSKI
Antoni WOŹNIAK
Politechnika Poznańska
Katedra Automatyki, Robotyki i Informatyki

TRANSPUTEROWY SYSTEM TWORZENIA MAPY RASTROWEJ I KOMUNIKACJI Z ROBOTEM MOBILNYM

Streszczenie: W Katedrze stworzono podstawy transputerowego graficznego systemu interakcji Użytkownik-Robot oraz równoległy program tworzenia rastrowego modelu otoczenia robota mobilnego. Równoległy algorytm tworzenia mapy rastrowej oparty został o strukturę procesów zwaną *farmą procesorów* (ang. *processors farm*). System farmy podłączony jest do transputera zarządzającego interakcją Użytkownik-Robot, tak by w pełni równolegle zachodziły procesy tworzenia mapy i zarządzania ruchem robota. Sprawdzenie przydatności systemów transputerowych w zadaniach nawigacji robota mobilnego związane jest także z oceną pracy kart transputerowych firmy Quintek.

A TRANSPUTER SYSTEM OF GRID MAP COMPUTING AND COMUNICATION WITH A AMV

Summary: The basic transputer graphic-oriented system of AMV navigation and grid cartography has been created. The parallel algorythm of the grid map creation is based on a transputer structure called *processor farm*. The system is connected with a main transputer chip that control the software interface User-Vehicle module, so that the processes of grid map computing and the vehicle movement control could work simultanously.

TRANSPUTERISCHE REALISIERUNG DER ERSTELLUNG DER RASTRISCHE ARBEITSRAUMKARTE UND DES "USER-VEHICLE" KOMMUNIKATIONSSYSTEMS

Zusammenfassung: Die vorliegende Arbeit stellt die graphische, transputerische

Systeme für die Navigation eines autonomen mobilen Roboters und Weltmodellierung mit der Rasterkarte vor. Die Konzeption des Parallelalgoritmus für Erstellung der Rasterkarte des Arbeitsraumes wird präsentiert. Die Struktur *processor farm* wird verwendet.

1. Założenia systemu

Założono wstępnie, że system uruchomiony ma być na transputerowych kartach firmy Quintek. W posiadaniu katedry znajdują się trzy karty transputerowe: FAST9 XL z dziewięcioma transputerami, FAST1 z jednym transputerem i kolorowa karta graficzna Mosaiq z jednym transputerem i dwoma koprocesorami wektorowymi Zoran. Dla modułu tworzenia mapy rastrowej założono wstępnie:

- mapa tworzona będzie na podstawie danych z 24 sensorów ultradźwiękowych;
- transputerowa karta FAST9 wykorzystywać będzie co najmniej osiem ze swych dziewięciu procesorów do obliczania wartości rastrów mapy;
- wykonanie obliczeń dla jednej serii danych z sensorów będzie co najmniej porównywalne z czasem zebrania danych od 24 czujników.

Dla modułu graficznej interakcji z użytkownikiem założono wstępnie, że:

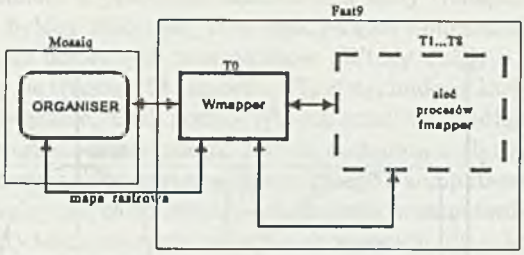
- użytkownik będzie miał możliwość nanoszenia na mapę otoczenia trajektorii ruchu robota składającej się z odcinków i łuków;
- każdemu z odcinków trajektorii będzie można przyporządkować parametry dynamiczne i kinematyczne;
- istnieć będzie możliwość nanoszenia na mapę wirtualnych obiektów;
- mapa otoczenia będzie aktualizowana za pomocą podsystemu farmy transputerów, aktualizacja będzie widoczna na ekranie monitora;
- użytkownik będzie miał możliwość posługiwania się myszką.

W trakcie realizacji projektu zaszły pewne zmiany, głównie w module obliczania mapy.

2. Algorytm tworzenia mapy

2.1. Mapa rastrowa

Mapa rastrowa powstaje poprzez wirtualne nałożenie na mapę otoczenia robota siatki, której oczka mają określone wymiary fizyczne. Powstałe w ten sposób komórki (rastry) mogą być łatwo zamodelowane w pamięci komputera jako dwuwymiarowa tablica. Zakłada się zwykle, że raster ma kształt kwadratowy. Komórki powstałej w ten sposób mapy mogą nieść ze sobą pewne wartości określające, czy jest ona zajęta, czy też nie. Wartości te będziemy dalej nazywać P_O i P_E. Przyjmuje się, że $P_O \in [-1; 0)$ a $P_E \in (0; 1]$.

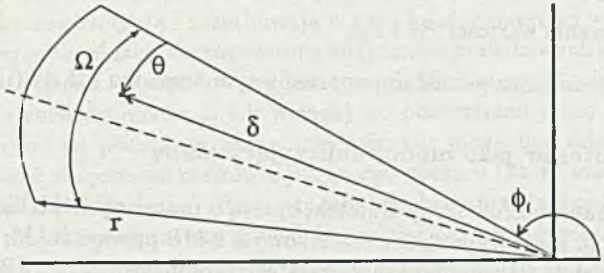


Rys. 1. Struktura transputerowego systemu współpracy z robotem mobilnym
Fig. 1. Structure of Transputer system cooperating with mobile robot

2.2. Algorytm Elfesa

A. Elfes [4][3] powiązał rozkład wiązki na płaszczyźnie mapy (rys. 2) ze sposobem przypisywania rastrom mapy wartości P_O i P_E. Elfes zakłada obarczenie odczytów sensorycznych dużą niepewnością, a przez to wyliczanie wartości rastrów mapy za pomocą funkcji probablistycznych (rys. 3). Założył, iż:

$$P_O = O_a(\theta, \Omega) O_r(\delta, \varepsilon, r)$$
$$P_E = E_a(\theta, \Omega) E_r(\delta, \varepsilon, r)$$

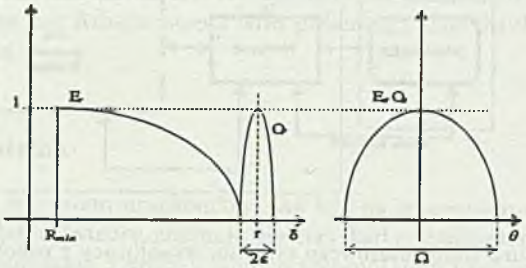


Rys. 2. Rozkład wiązki na płaszczyźnie mapy
Fig. 2. Beam split on the map plane

Poprzez znalezienie wszystkich komórek pokrytych przez rzut wiązki ultradźwiękowej na mapę można znaleźć odpowiadające im wartości O, E i wyliczyć odpowiednio oba prawdopodobieństwa. Procedura obliczająca rastry pod daną wiązką przyjmuje ostateczną postać:

(i) komórki mapy są nieznane (wszystkie równe 0);

(ii) odczyt danych z sensorów;



Rys. 3. Funkcje O i E
Fig. 3. Functions O and E

(iii) Obliczenie P_E:
superpozycja obszarów pustych:
$P_E = P_E(\text{komórka}) + P_E(\text{odczyt}) - P_E(\text{komórka}) \times P_E(\text{odczyt})$;

(iv) Obliczenia P_O:
poprawka dla P_O (wartość osłabiana jest przez P_E):
$P_O(\text{odczyt}) = P_O(\text{odczyt}) \times (1 - P_E(\text{komórka})$
superpozycja P_O:
$P_O(\text{komórka}) = P_O(\text{komórka}) + P_O(\text{odczyt}) - P_O(\text{komórka}) \times P_O(\text{odczyt})$;

(v) końcowe sumowanie wartości P_O i P_E;

(vi) jeżeli koniec,to zakończ procedurę, w przeciwnym wypadku idź do (ii),

3. Karta transputerów jako moduł obliczający mapę

Zastosowano kartę transputerów firmy Quintek, opartą o procesory INMOSa T805 z zegarem 25 MHz. Jest to karta wyposażona dodatkowo w 9 MB pamięci RAM; podłączana jest bezpośrednio do slotów komputera macierzystego (kompatybilnego z IBM). Nie posiada żadnych urządzeń I/O. Na oprogramowanie składa się kompilator języka Parallel C firmy 3L i wygodny, okienkowy debugger.

Jeden z dziewięciu transputerów jest podłączony bezpośrednio albo do procesora komputera macierzystego albo do innej karty transputerowej. Połączenia pomiędzy procesorami na płycie można swobodnie zmieniać programowo (za pomocą programu konfigurującego) i sprzętowo (jumper'ami).

W zadaniu napływające dane wejściowe składają się z 24 wartości uzyskanych z systemu sensorycznego. System sensoryczny składa się z 24 czujników ultradźwiękowych, rozłożonych wstępnie na łuku okręgu o pewnym promieniu. Równomierne objęcie całego zasięgu „widzialności" robota zapewnione jest przez ustawienie czujników co 15°.

Pierwszym rozwiązaniem w problemie zastosowania karty transputerów do obróbki danych sensorycznych byłoby założenie, iż w efektywnych obliczeniach powinno brać udział osiem z dziewięciu dostępnych transputerów. Wtedy każdy z „zatrudnionych" procesorów zajmowałby się trzema z 24 sensorów. Niestety, budowa karty transputerowej uniemożliwia takie rozwiązanie. Brak portów I/O uniemożliwia podłączenie nawet jednego transputera do systemu sensorycznego. Jedyną dostępną, najłatwiejszą drogą było połączenie poprzez modem radiowy systemu sensorycznego z komputerem macierzystym, a poprzez niego przesłanie danych sensorycznych do karty transputerów. Przepływ danych do procesorów karty także nie może odbywać się w sposób równoległy, ale najpierw przez: (i) tzw. transputer ROOT połączony bezpośrednio z procesorem intelowskim, (ii) a następnie poprzez pracujące transputery, które dzielą dane pomiędzy siebie. Droga przepływu informacji zależy od konfiguracji połączeń pomiędzy procesorami.

3.1. Wymagania zadania

Zadanie współbieżnego obliczania wartości mapy rastrowej wymaga przede wszystkim jednego: znajomości poprzednich wartości rastrów przez transputery obliczające wartości komórek spod kolejnych wiązek ultradźwiękowych (patrz punkt 2.2.). Wymaganie to okazało się najtrudniejszą przeszkodą do pokonania. Posiadana karta transputerowa wyposaża każdy z procesorów w 1 MB pamięci RAM. Jest to, oczywiście, dużo, nawet jak na wymagania pamięciowe tego zadania, ale nie zapewnia prawidłowego wykonania obliczeń. Dlaczego?

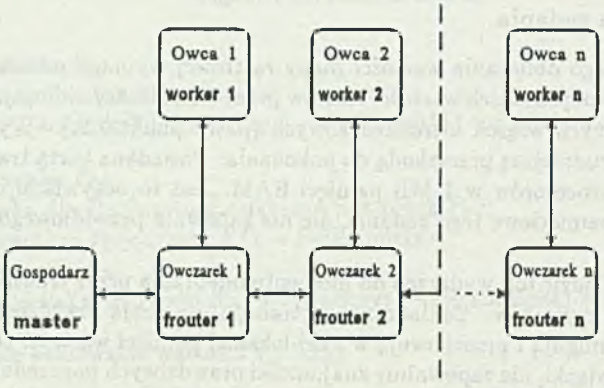
Spróbujmy prześledzić tok wydarzeń od momentu odebrania przez transputer ROOT zbioru danych z 24 czujników. Zakładając, że transputery mają przydzielone numery czujników, które obsługują i przechowują w swej lokalnej pamięci wartości rastrów spod przydzielonej sobie wiązki, nie zapewnimy znajomości prawdziwych poprzednich wartości mapy z tego regionu. Spowodowane jest to zmianami w orientacji i położeniu robota, co wywołuje ruch czujników (i ich wiązek) po powierzchni mapy. Tak więc wartości mapy obliczane na podstawie aktualnego odczytu mogą być fałszywe, gdyż dane te mogą wymagać znajomości rastrów z już innego obszaru (który mógłby być zajmowany poprzednio przez inną wiązkę). Rastry te mogłyby być uprzednio obliczone przez sąsiedni transputer, ale jak w najprostszy sposób dowiedzieć się, co on przechowuje w swej pamięci, i w jaki sposób wyznaczyć, z których transputerów potrzebne są dane?

Prostszym rozwiązaniem, które narzuciło się prawie natychmiast, było zastosowanie mechanizmu pracy w systemach wieloprocesorowych, zwanym *farmą procesorów*.

3.2. Farma procesorów

Zastosowanie farmy procesorów wymaga założenia, że każdy z procesorów z farmy wykonuje jeden, ten sam algorytm, natomiast dane, na podstawie których pracuje, są różne. Ponadto, przepływ danych i decyzja, który z procesorów „przechwyci" napływające dane, zależy tylko od stopnia „zapracowania" procesorów. Innymi słowy mówiąc, pracujące w farmie procesory przechwytują przepływające przez sieć informacje wtedy, gdy aktualnie nie są zajęte.

Dostarczone przez producenta kompilatora Parallel C biblioteki zapewniają bardzo prosty sposób tworzenia farmy [1]. Schemat takiej sieci pokazany jest na rysunku 4. W sieci wyróżnimy Gospodarza (Master), mającego za zadanie wysyłkę w sieć (a właściwie w farmę) danych, Owce (Slaves) pracujące na podstawie danych według tego samego algorytmu (można by to przyrównać do przeżuwania sieczki) oraz tzw. *frouters*, czyli procesy pomocnicze, które można by nazwać, zachowując terminologię pasterską, Owczarkami. Mają one za zadanie (i) przyjmować napływające dane, (ii) sprawdzać, czy oddana im pod opiekę owca może te dane przyjąć, (iii) odsyłać niewykorzystane przez jego owcę dane dalej w sieć, (iv) przyjmować dane od innych owczarków i przesyłać je do Gospodarza oraz (v) zapewnić powrót obrobionych przez jego owcę danych do Gospodarza. Można więc ustalić, że transputer pracujący „na" farmie (i) odbiera tylko z wejścia



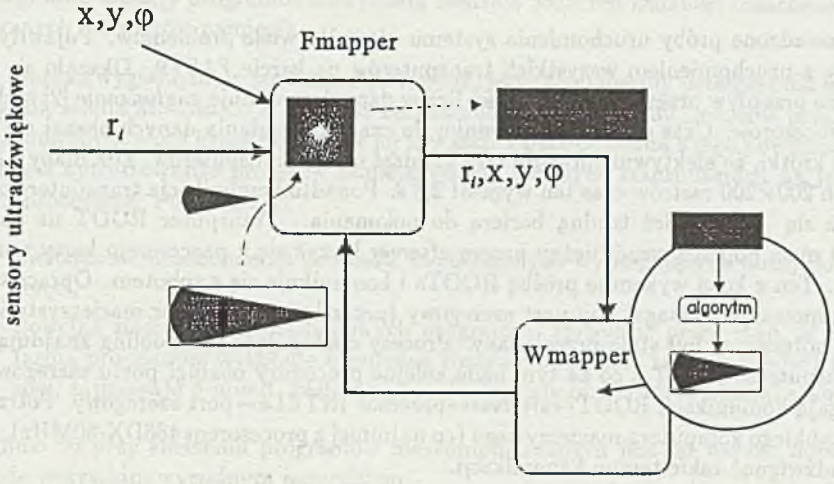
Rys. 4. Struktura farmy procesorów
Fig. 4. Structure of processors farm

daną sensoryczną, (ii) oblicza na podstawie algorytmu z punktu 2.2. wartości rastrów, (iii) wraca do punktu (i).

Ten schemat należałoby uzupełnić o pewne szczegóły:

- na wejściu oprócz wartości odczytu z sensora pojawiają się także: (i) orientacja czujnika na mapie, (ii) położenie czujnika na mapie, (iii) blok mapy zawierający te komórki, które mieszczą się pod aktualną wiązką;
- na wyjściu pojawia się blok mapy o wymiarach identycznych z blokiem wejściowym, ale o zaktualizowanych wartościach rastrów.

Transputer grający rolę Gospodarza (i) odbiera zestaw danych sensorycznych, (ii) oblicza pozycję i wymiary bloków odpowiadających kolejnym wiązkom, (iii) wysyła w sieć dane sensoryczne, orientacje i pozycje czujników, (iv) odbiera z sieci zaktualizowane bloki i składa z nich globalną mapę rastrową. W naszym zadaniu proces Gospodarza został nazwany Fmapper'em,a proces owcy Wmapper'em (rys. 5).



Rys. 5. Przepływ danych w farmie
Fig. 5. Farm data flow

4. ORGANISER

Drugim celem prac było stworzenie podstaw graficznego systemu interakcji Użytkownik–Robot. Wynikiem tego był program ORGANISER, który miałby za zadanie skupiać w sobie moduł tworzący mapę rastrową, jak i wszystkie procedury obsługujące komunikację z robotem mobilnym. Procedury komunikacyjne zostały wcześniej opracowane wraz z przygotowywaniem studentów Automatyki i Robotyki do pracy z robotem mobilnym LABMATE amerykańskiej firmy TRC. Robot został też wyposażony w system sensoryczny Proximity Subsystem, na którego złożył się system 24 sensorów ultradźwiękowych i 24 podczerwonych czujników obecności przeszkody. Do tych systemów także przygotowano procedury komunikacji. Wszystkie wyżej wspomniane procedury napisano w języku C.

W zamyśle przeprowadzonych prac było połączenie karty graficznej Mosaiq (wyposażonej w jeden transputer) z kartą FAST9. Mosaiq zajmowałby się operacjami graficznymi, a FAST9 obliczeniami mapy. FAST9 poprzez procesor Mosaiqa otrzymywałby dane sensoryczne, a zaktualizowana mapa byłaby zobrazowana na monitorze za pomocą wyspecjalizowanych bibliotek graficznych. Użytkownik miałby możliwość zaprojektowania wstępnie trajektorii przejazdu, w czasie którego odbierane byłyby dane z systemu sensorycznego i budowana byłaby mapa rastrowa.

Udało się zrealizować system ORGANISER; jak dotychczas, nie połączono ze sobą obu kart.

5. Eksperymenty

Przeprowadzone próby uruchomienia systemu ujawniły wiele problemów. Pojawiły się kłopoty z uruchomieniem wszystkich transputerów na karcie FAST9. Okazało się bowiem, że przepływ przez sieć tak wielkiej liczby danych powoduje *zagłodzenie* [2] niektórych procesorów. Czas obliczeń w stosunku do czasu przesyłania danych okazał się być na tyle krótki, że efektywnie pracują trzy zamiast ośmiu transputerów. Dla mapy o wymiarach 200×200 rastrów czas ten wynosił 2,8 s. Ponadto komunikacja transputer-robot okazała się również trudną barierą do pokonania. Transputer ROOT na karcie FAST9 musi poprzez współbieżny proces afserver łączyć się z procesorem karty macierzystej. Ten z kolei wykonuje prośbę ROOTa i komunikuje się z robotem. Opracowane oprogramowanie wymaga, aby port szeregowy (przez który komputer macierzysty i robot są połączone) był stale przeglądany. Procesy obsługujące ten pooling znajdują się na transputerze ROOT,a co za tym idzie,kolejne procedury obsługi portu szeregowego wymagają komunikacji ROOT↔afserver↔procesor INTELa↔port szeregowy. Potrzeba było szybkiego komputera macierzystego (co najmniej z procesorem 486DX-50MHz), aby mógł udźwignąć takie tempo komunikacji.

Wobec zaistniałych problemów postanowiono przeprowadzić szereg symulacji, które wykryły problemy z zagłodzeniem procesorów. Program ORGANISER spełnił większość założeń i przy połączeniu z modułem obliczającym mapę mógłby stanowić wygodne narzędzie przy eksperymentach z mapą rastrową.

6. Podsumowanie

■ Farma procesorów założona na karcie FAST9 nie zdała w pełni swego egzaminu. Jest to nie tyle problem złego (lub niedoskonałego) oprogramowania farmy, lecz dopasowania specyfiki obliczeń do posiadanego sprzętu. Wymagana w obliczeniach znajomość wartości rastrów z poprzedniego odczytu zmusiła do przesyłania bardzo dużych bloków informacji przez sieć transputerową. Karta FAST9 jest systemem rozproszonym, a więc do pewnych celów nie jest odpowiednia. Wieloprocesorowe obliczanie mapy bitowej jest zadaniem wymagającym od pewnej liczby procesorów znajomości wspólnego dla nich obszaru pamięci lub zawartości pamięci jednego procesora przechowującego kopię mapy. Można więc pójść drogą rozwijania większej szybkości przesyłania danych lub drogą implementowania tego algorytmu na szybkich systemach z pamięcią dzieloną przez procesory. To drugie rozwiązanie wydaje się najodpowiedniejsze.

■ Niestety, nie da się powiedzieć, że posiadane oprogramowanie jest spolegliwe. Kompilator Parallel C posiada pewne wady:

- brak kontroli zgodności liczby argumentów w funkcjach,
- wolne przetwarzanie tekstów programów źródłowych (związane z wadami samej karty opisanymi poniżej).

Oprogramowanie wspomagające wymaga od użytkownika „zagłębiania się" w bardzo szczegółowe aspekty programowania (ścisła kontrola połączeń kanałów, oszacowania wymaganych rozmiarów pamięci).

■ Bardzo wygodnym, w porównaniu z innymi, jest proponowany debugger dla języków programowania 3L. Jest on w zasadzie bardzo zbliżony do systemu śledzenia programów proponowanym przez firmę Borland w jej językach TURBO. Jedną z jego poważniejszych wad jest symulowanie procesów istniejących na osobnych transputerach na jednym tylko procesorze. Powoduje to:

- niemożność uruchomienia procesów współbieżnych wymagających dużej ilości pamięci,
- powolne śledzenie wielozadaniowych programów (przejście przez etap, w którym farma procesorów wykonuje niezbędne wstępne obliczenia tablic sinusów i cosinusów, zajmowało 5 minut, podczas gdy w rzeczywistości trwa to ułamki sekund)

Pomimo to przy śledzeniu programów nieskomplikowanych jest on bardzo dobrym, a przede wszystkim wygodnym narzędziem.

■ Karty FAST są przeznaczone głównie do celów dydaktycznych. Wskazuje na to już sama postać tego systemu transputerowego: dodatkowa karta dla komputera IBM bez urządzeń I/O. Producenci postanowili jak najbardziej uprościć system wykorzystując do kontaktu z użytkownikiem komputer IBM. Niestety, niesie to za sobą pewne konsekwencje:

- wolna współpraca kart z monitorem; komunikaty wypisywane są niezwykle wolno i zwykle bardziej opłaca się wykorzystywać polecenia przerwań dla procesora IBM niż korzystać ze standardowych bibliotek kompilatorów,
- podobne kłopoty z innymi peryferiami — brak bezpośredniego dostępu do portów szeregowych i równoległych komputera macierzystego. Istniejąca pośrednia łączność poprzez komputer macierzysty bardzo wolna.

Najbardziej efektywnym wykorzystaniem tych kart jest implementowanie programów obliczeniowych nie wymagających przesyłania dużych bloków informacji.

■ Brak bezpośredniego dostępu do peryferii uniemożliwia wykorzystywanie karty do sterowania szybkich procesów. To właśnie stało się przyczyną kłopotów z uruchomieniem środowiska ORGANISER. Wymagana była bardzo ścisła komunikacja z robotem LABMATE i jego systemem sensorycznym. Odbiór informacji z robota nie jest w żaden sposób synchronizowany — należy śledzić bajty wysyłane przez jego sterownik do komputera macierzystego. Niestety, jest to uniemożliwione przy współpracy kart z powolnym komputerem macierzystym typu IBM/AT/286. Okazało się, że wymagana jest obecność przynajmniej komputera z procesorem 486-50MHz. Komunikacja w tym wypadku też nie jest pewna.

Rozwiązaniem problemu szybkiej komunikacji robot-transputer mogłoby być napisanie nowego procesu afservera, który w swej części „intelowskiej" zawierałby gotowe procedury komunikacyjne. Część transputerowa wysyłałaby tylko komunikaty komend.

■ Obliczanie mapy rastrowej zostało dodatkowo opóźnione przez wolny system komunikacji systemu sensorycznego z komputerem macierzystym. Jeżeli średni czas obliczania mapy od momentu przyjęcia danych sensorycznych wynosi ok. 0.5 s, to licząc go wraz z czasem wysłania sygnału prośby z systemu odczytu danych i przesłania ich do komputera macierzystego, wydłuża się on do 2 s.

■ Karta Mosaiq jest najbardziej spolegliwa spośród wszystkich posiadanych kart. Bogata biblioteka procedur graficznych pozwala na wygodne pisanie środowisk opartych o graficzną interakcję z użytkownikiem. Co prawda obecność współbieżnego procesu m_engine, odpowiedzialnego za wykonywanie wszystkich procedur graficznych, może być przyczyną spowolnienia programu użytkownika, lecz łatwo można temu zaradzić dołączając do Mosaiq'a którąś z kart FAST.

LITERATURA

[1] 3L. *Parallel C. User Guide*, 1990.

[2] M. Ben-Ari. *Podstawy programowania współbieżnego*. WNT, Warszawa 1989.

[3] A. Woźniak, D. Dobroczyński, T. Jedwabny. Tworzenie mapy bitowej otoczenia robota mobilnego z wykorzystaniem przetwarzania równoległego. W: *IV Krajowa konferencja robotyki*. Politechnika Wrocławska, 1993.

[4] A. Elfes. Sonar-based real-world mapping and navigation. W: *Autonomous Vehicle Robots*, s. 233–240. Verlag & Co., 1988.

Recenzent: Dr inż. Henryk Palus

Wpłynęło do Redakcji do 30.04.1994r.

Abstract

Building our system we decided to use a transputer board FAST9 and Mosaiq made by Quintek LTD. The lack of I/O ports caused some problems with vehicle-user communication. The algorithm of grid map computing is based on A. Elfes' one [4]. Four probabilistic functions were used to compute values of cells covered by an ultrasonic sensor beam. FAST9 board was decided to control several parallel processes each of them

handling the computing cells procedure. The task requires shared data base of the map cells. FAST9 is a system of distributed memory, so it was decided to implement a pattern of multi-processor work called processor farm. This scheme provided the knowledge of the map cells values to all of the transputers.

Some problems appeared during several simulation of the process. The serial links of the transputers does not provided fast enough communication that can handle the required amount of transferred bytes. Only three from among eight transputers took part in computations. Five processors were starved. It was caused by improper value of computing time vs transfer time.

The graphic-oriented module of interface between User and AMV Labmate was created. ORGANISER runs on Mosaiq transputer graphic board. It provides several tools for trajectory planning and environment map creation. The whole system is thought to be composed of two modules: (i) ORGANISER on Mosaiq and (ii) Fmapper that runs on FAST9 which is connected to Mosaiq. The main purpose of such a configuration was to make the map creation processes run simultanously with the robot trajectory executing.