Object Sorting Using a Distributed Manipulator Array

Declan Walsh and Piotr Dudek

School of Electrical and Electronic Engineering
The University of Manchester
Manchester, UK
declan.walsh@postgrad.manchester.ac.uk; p.dudek@manchester.ac.uk

***Abstract* — A cellular array algorithm is presented which allows the manipulation of objects across a distributed manipulator array. Objects are placed on the manipulator array surface and the algorithm differentiates between two different types of objects depending on their physical properties. The different objects are then controlled by the algorithm and the manipulator array so that the objects end up grouped together in different regions of the manipulator array. The algorithm has been verified using a physics-based 3D simulation engine.**

I. Introduction

DISTRIBUTED manipulator arrays are physical systems which can manipulate objects placed on them by interacting with the object and applying a force through actuation. Different types of actuation mechanisms exist in physical actuator arrays including, but not limited to, pins, wheels, air-jet valves, and electromagnetic actuators. The manipulator arrays are comprised of a cellular array lattice of surface actuators with a coherent action of many units needed to achieve the desired manipulation. These physical manipulator arrays can either be controlled by a central processing unit or distributed processors. This research investigates distributed control of manipulators where each cell in the array only interacts with local neighbouring manipulators.

A wheel-based manipulator system described in [1] uses rotating wheels to actuate larger objects on a plane. Electromagnetic actuation has been used in an interactive workbench [2] where objects are manipulated across a tabletop and 'Madgets' [3], a system that uses magnetic objects such as sliders and buttons which allows users to interact with a tabletop surface. Air-jet actuation [4] offers a contactless, distributed manipulation of objects where objects are maneuvered to defined region of the tabletop. While not specifically aimed towards object manipulation, pin actuation is demonstrated in 'Relief' [5] where a two-dimensional array of pins is used to animate a three-dimensional surface for displaying purposes. Furthermore, 'inForm' [6] uses an array of pins demonstrating object actuation as well as being able to provide a user interface for human interaction. Such distributed manipulator arrays have shown potential for the autonomous manipulation of objects with the direction of motion of the objects conditional upon particular properties of the respective objects such as size, shape, and colour.

A smart surface physics simulator ('SmartSurf') of a distributed manipulator array is presented in [7] and displayed in Fig 1. It operates in a closed-loop system – sense, compute and. This physics simulator uses real world physical properties of objects and incorporates them into an environment. The simulator is configurable and includes sensors, objects, and pin actuators.

Using this simulator, an algorithm is presented in which multiple objects are autonomously manipulated on a surface in such a way that objects of one particular size are moved to one side of the surface while objects of a different size are moved to the opposite side of the surface, demonstrating the distributed manipulation of physical objects in a 3D physical environment.

II. Manipulator Array Setup

The SmartSurf simulation environment is configured so that a 2D array of 64 × 64 pin actuators is used as the simulation surface for object manipulation. All dimensions referred to are in arbitrary units. Actuators are rectangular pins arranged in a 2D array in the *xy*-plane with one degree of freedom in the *z*-direction. The face of the pin, which forms the surface, is 0.8 × 0.8 units with a distance from the centre of each pin to the centre of its neighbouring pins of 1 unit, leaving a space of 0.2 units between each pin. Each actuator includes a proximity sensor which can detect the presence of an object above the pin. The surface is controlled by APRON array processing software [8] which receives a 2D array of sensor values from the simulator, computes the algorithm and sends a 2D array of values to the actuators to move them to a defined location. APRON runs the algorithm on each sensor-pin combination in parallel, effectively simulating a SIMD cellular processor array. In this setup, an actuator value of 4 units indicates the actuators is an 'up' to their maximum position while a value of 0 units indicates actuators being 'down' at their lowest position. Border pixels are permanently set to a value of 6 units to produce a wall of pins around the surface, preventing objects falling off the surface.

This work has been supported by EPSRC; Grant No.: EP/H023623/1

978-1-4799-6007-1/14/$31.00 ©2014 IEEE

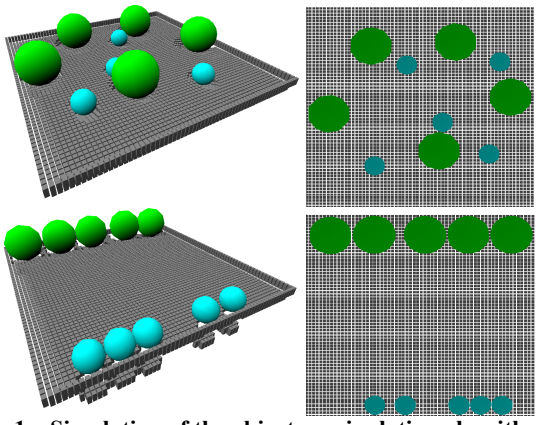

Fig 1 – Simulation of the object manipulation algorithm; initial positions at top, final positions at bottom

III. Object Manipulation Algorithm

Spheres of two different sizes are placed on the manipulator array and the algorithm differentiates the objects from each other based on their size and instructs the manipulator array to manipulate the objects into two separate groups on the table (Fig 1) by moving pins up or down. The APRON software processes data in a 2D array of pixels with each pixel corresponding to on sensor/pin combination and each pixel being able to communicate with its four nearest direct neighbours. The processing steps involved in the algorithm are shown in Fig 2. When an object is located on top of a pin, the sensors are activated. Neighbouring activated sensors indicate a continuous object (Fig 2a). Objects in close proximity will appear as one continuous object and to avoid this, the algorithm erodes the sensor footprint (Fig 2b). Once the objects are separated, the size of each object can be determined by counting the number of steps from the edge of each object, to the centre of the object. To implement this, an initial value of '1' is assigned to all border pixels, with each unassigned nearest neighbours being assigned an incremental value until all pixels are assigned a value (Fig 2c). The largest assigned value for each object is then used to indicate the minimum number of steps from the border to the centre of the object. By propagating this value to all connected pixels belonging to the object, a size identifier for that particular object is formed (Fig 2d). The object pixels are then dilated by one pixel so that the pixels cover the entire are of the object again (Fig 2e). A threshold value between the size identifier of the larger and smaller objects is then defined to differentiate between the objects and move the objects in their respective directions. Any pixels with a size identifier smaller than the threshold (threshold=2 in this example) are moved to the west while pixels with a size identifier larger than the threshold are moved to the east (Fig 2f). The actuator pins are then moved up and down to manipulate the objects to their desired direction. To move an object in a particular direction, the pins create a well for the objects to fall into so that it is confined by the sides of the pins. By moving this well in the desired direction, the manipulator array provides a constricted and controlled movement of objects (Fig 2g). Object collisions are simply resolved by the physics of the system, with no additional processing required.

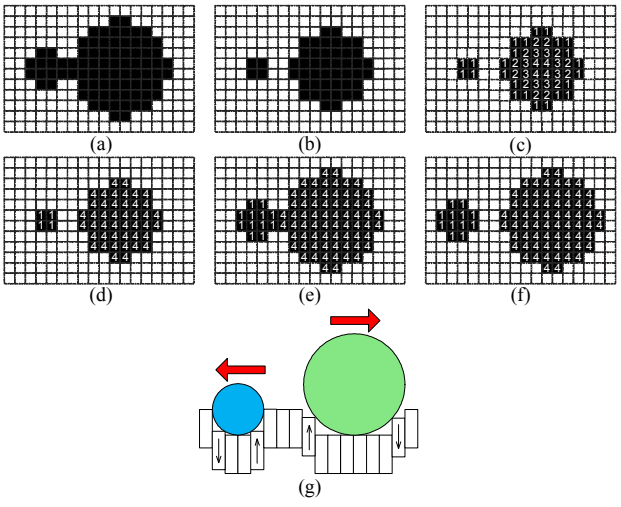

Fig 2 – Processing Steps involved in object manipulation algorithm; (a) sensor data; (b) erosion; (c) count steps to centre of object; (d) maximum number of steps in object; (e) dilation; (f) move west/east; (g) actuator state

As the objects are moved to their new locations, the system is re-evaluated and the algorithm continues to run, separating the objects in different directions until they reach the border pins as the edge of the surface which prevents the objects falling off the surface and gathers the objects together.

IV. Conclusion

The presented algorithm demonstrates the autonomous manipulation of objects using a manipulator array of pin actuators. Each actuator is controlled by a single processor with only a sensor input, actuator output, and communication with its four direct neighbours and no communication to other processors or actuators in the array. Processing is distributed and each cell runs the same algorithm demonstrating the scalability of such distributed manipulator arrays.

References

[1] J. E. Luntz, W. Messner, and H. Choset. "Distributed manipulation using discrete actuator arrays". The Intl Journal of Robotics Research, July 2001, pp. 553-583.

[2] G. Pangaro, D. Maynes-Aminzade and H. Ishii, "The actuated workbench: Computer-controlled actuation in tabletop tangible interfaces", in Proceedings of the 15th annual ACM symposium on User interface software and technology (UIST'02), Paris, France, 2002, pp. 181-190.

[3] M. Weiss, F. Schwarz, S. Jakubowski, and J. Borchers. "Madgets: Actuating Widgets on Interactive Tabletops", UIST '10, pages 293-302, New York, NY, USA, 2010.

[4] K. Boutoustous, G. J. Laurent, E. Dedu, L. Matignon, J. Bourgeois, and N. Le Fort-Piat, "Distributed control architecture for smart surfaces", IEEE Int. Conf. on Intelligent Robots and Systems, 2010.

[5] D. Leithinger and H. Ishii, "Relief: A scalable actuated shape display", in Proc. of the fourth international conference on Tangible, embedded, and embodied interaction (TEI '10), Cambridge, MA, 2010, pp. 221-222.

[6] S. Follmer, D. Leithinger, A. Olwal, A. Hogge and H Ishii, "inForm: Dynamic physical affordances and constraints through shape and object actuation", Proc. of the 26th ACM Symp on User interface software and technology (UIST '13). ACM, New York, NY, 2013, pp. 417-426.

[7] D. R. W. Barr, D. Walsh and P. Dudek, "A smart surface simulation environment", in IEEE International Conference on Systems, Man and Cybernetics (SMC 2013), Manchester, UK, 2013, pp. 4456-4461.

[8] D. R. W. Barr and P. Dudek, "Apron: A cellular processor array simulation and hardware design tool", EURASIP Journal on Advances in Signal Processing, vol. 2009, p. 3, 2009.