Real-time Hand Pose Recognition Using Low-Resolution Depth Images

Zhenyao Mo, Ulrich Neumann
University of Southern California
Computer Graphics and Immersive Technologies Lab
{zmo, uneuman}@graphics.usc.edu

Abstract

Gesture recognition methods based on intensity or color images often suffer from low efficiency and lack of robustness. In this paper, we employ a new laser-based camera that produces reliable low-resolution depth images at video rates. By decomposing and recognizing hand poses as finger states (finger poses and finger inter-relations), we achieve robust hand pose recognition in real-time (30 frames/second).

1. Introduction

As an alternative to traditional input devices (keyboards, mice, joysticks, etc), gestures have long been recognized and pursued as a desirable mechanism for natural human computer interaction. However, gesture recognition remains a challenging problem. Digital gloves or marker-based methods provide feasible solutions, but wearing extra equipment on the hand(s) lessens the natural and general character of gesture interfaces.

Vision-based methods are non-invasive, and many algorithms have been proposed using intensity or color images from one or more video cameras. The difficulties of gesture recognition using regular cameras include the following:

- Noisy segmentation of hand(s) from complex backgrounds and changing illumination;
- Simple features extracted from images produce ambiguities, whereas sophisticated feature extraction can greatly increase processing time;
- Hand tracking methods suffer from initialization and tracking failures.

Thus, low computational-efficiency and lack of robustness has impeded the adoption of vision methods for practical gesture applications.

Infrared cameras are used in [7] to accurately extraction a user's hand from background, resulting in a binarization of an input image. Hand profiles are used to recognize hand poses, which also suffers from ambiguities. Only a limited set of hand poses can be recognized, which disqualifies infrared cameras from being effective input devices for general gesture interfaces.

Although many argue that the development of faster computing hardware and more sophisticated algorithms will lead to robust recognition, we show that real-time robust recognition is obtained by employing a new vision-sensor technology.

Laser-based cameras from Canesta can provide reliable depth images at video rate (30 frames/second). These cameras are compact and self-contained, providing both range and intensity images at video rates. The resolution of current Canesta camera is low (64x64 pixels) to reduce the cost of the cameras and qualify them as potential input devices for personal or portable computers.

Instead of representing and recovering hand poses in a high-dimentional joint angle space, in this paper we propose a finger spelling mechanism by decomposing hand poses into finger states, including finger poses and finger inter-relations. The number of distinguishable finger states is highly limited, and finger states can be identified robustly and efficiently from depth images.

After reviewing related work, section 3 introduces a finger spelling schema to represent hand poses in a discrete finger state space; section 4 describes our hand pose recognition algorithms in detail; section 5 discusses the limitations; we conclude the paper in section 6.

2. Related work

Prior vision-based hand pose estimation techniques fall into two categories: appearance-based and model-based. A review of early work in gesture recognition is provided in [8].

Model-based methods match a hand model to features extracted from input images. In [9], a 27 degree-offreedom (DOF) hand-model is tracked in real time using two cameras. In [13], a model is built from truncated quadrics, and techniques from projective geometry are applied. Visual feature sets can be ambiguous, and the work in [15], [12], and [4] obtain correct poses by integrating constraints and statistical tracking methods.

Appearance-based methods [5] are appealing for recognizing a small set of gestures. However, for larger gesture sets, such methods lack robustness and require extensive training data. In [14], the training data problem is addressed by combining a few labeled images with a large set of unlabeled images.

"Estimation by Synthesis" is a combination of model-based and appearance-based methods. Graphical hand models are controlled to create images, which are then compared with input images to estimate hand poses. These methods often need to produce a large number of generated images to account for the variety of possible hand poses and views. In [11] an adjacency map is used to reduce the search space, achieving real-time performance with a cluster of six personal computers. In [2], a hierarchical retrieval based on combined measures is used to reduce the processing time for one frame to a few seconds.

3. Hand pose representation

A human hand is an articulate object with stable structure. Hand poses are often represented in a high dimensional space of joint angles. However, in this paper, we introduce a discrete hand pose space based on the following two observations:

- Using low-resolution images from a Canesta camera, recovered joint angles are quantitatively inaccurate;
- Hand pose vocabulary used in a gesture interface is limited because the gestures must be easy to perform and easy to distinguish from each other.

Thus, instead of using a continuous high-dimensional joint angle space, we propose a finger-spelling schema to represent hand poses. We decompose a hand pose into finger states, and define the hand pose as finger poses and finger inter-relations.

Each finger can be in one of the seven basic poses shown in Fig. 1. Due to their similarity, our current implementation does not distinguish between "bend" and "half-bend" poses. Thus, only six distinguishable finger poses are recognized.

Each pair of fingers can be in one of the four inter-relations shown in Fig. 2.

This simple finger-spelling schema can be used to represent a large set of hand poses for gesture interfaces. For example, SignWriting [1] is recognized as the most expressive notion system for sign languages [10], and all 41 basic hand poses in SignWriting can be distinguishingly encoded using this schema.

For a more detailed description of encoding a hand pose by finger spelling, see Appendix A and B.

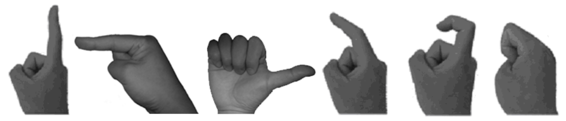
Figure 1. basic finger poses: up, forward, side, half-bend/bend, half-closed, and closed (from left to right).

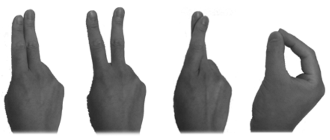
Figure 2. finger inter-relations: group, separate, cross, and loop (from left to right).

4. Method

This section describes in detail a hand pose recognition algorithm that uses a Canesta camera.

At video rate, a Canesta camera outputs depth images at a resolution of 64x64 pixels. We use Model DP-205 with 55-degree FOV. Mounted on desk, with a user's hand approximately one meter away, the camera provides hand images of reasonable size without over-constraining the space for hand motion. The 16-bit depth data are delivered in units of one millimeter resolution. For detailed specification, see [3].

The algorithm assumes we constrain a user's palm to face the camera within a +/-30degree rotation. This is needed because the low-resolution images may not provide enough visible information for identifying hand poses from side views. In practice, this constraint is rarely exceeded since it is quite natural to direct gestures toward a viewer or camera.

Fig.3 shows the algorithm framework.

4.1. A hand model

Hands can be modeled at different levels of detail, according to the application needs. Our system uses a representation of finger poses, thus, our model is a right hand with a palm and 14 phalanxes, as shown in Fig. 4.

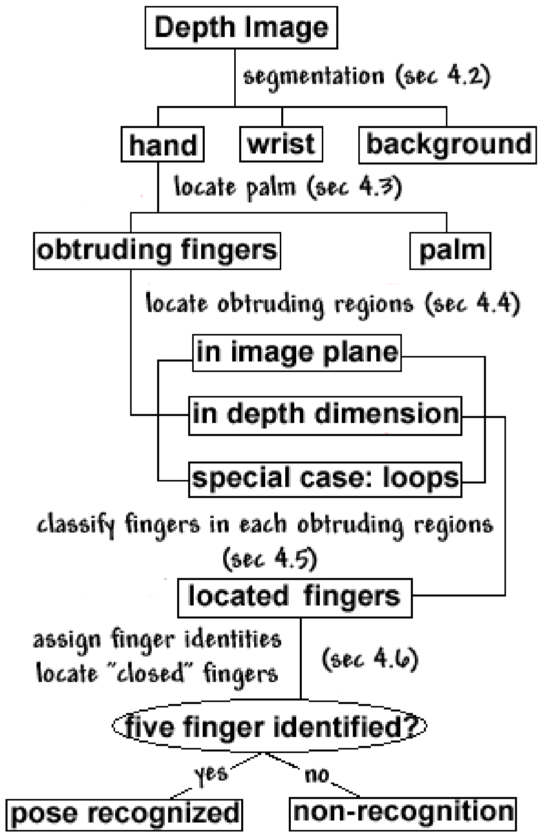

Figure 3. a general framework of the hand pose recognition algorithm.

A phalanx's length is approximated in proportion to palm size. For example, the lengths of three index finger phalanxes are defined as $(1.0s, 1.0s, 2.0s)$, where s is a factor related to palm size. By measuring the palm size, we obtain an estimation of each phalanx's length. The estimation is sufficient for our algorithm because we only recognize hand poses qualitatively.

A new user is required to initialize the system by producing the pose of a "flat" hand (Fig. 12, #12) facing the camera, so the palm size can be estimated (see section 4.3). Thus, the algorithm easily adapts to varied user hand sizes by resetting the palm size.

4.2. Segmenting hand, wrist, and background

We assume that the user's right hand is the closest object to the camera. Thus, the segmentation of hand and non-hand region in a depth image is trivial. A depth value $\hat{z} = z_0 + \delta z$ separates the foreground hand (F) from the

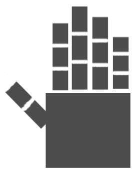

Figure 4. a hand model.

background:

$$z_0 = min(z(r,c))$$

$$F = \{(r,c) \mid z(r,c) < \hat{z}\}$$

where $(r, c, z(r,c))$ is the row index, column index, and depth value of a point in the depth image, and δz is a range threshold that is related to palm size (see section 4.3).

The segmented foreground F contains the hand and part of the wrist. The boundary B_w (see Fig. 5) that divides the wrist into foreground and background parts is identified as

$$B_w = \{(r,c) \mid (r,c) \in B \ \& \ |z(r,c) - \hat{z}| < \epsilon\}$$

where B is the boundary that encloses the foreground F.

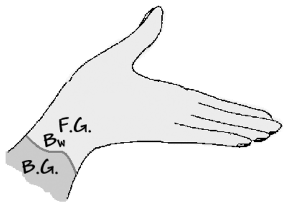

Figure 5. hand segmentation: the curve B_w divides the wrist into background (B.G.) and foreground (F.G.).

4.3. Locating palm center

The palm center C_0 is defined as the point in the hand region that maximizes its distance (d_2 as the distance in the image plane) to the closest hand boundary:

$$C_0 = \arg\min_{P \in F}\{\min_{Q \in B}(d_2(P,Q))\}$$

C_0 is located by applying a morphological erosion operation. Note that inner boundaries (section 4.4.3) must be excluded as starting points of the morphological erosion operation.

The palm size R is defined as the distance between C_0 and the closest boundary point in the world coordinate system (d_3):

$$R = \min_{Q \in B}\{d_3(C_0, Q)\}$$

A point (r, c) with depth value z can be mapped to the world coordinate (x, y, z) with a given focal length of the sensor lens (8mm for Canesta DP205).

4.4. Locating obtruding fingers

In this subsection we describe three processes that are applied to locate obtruding fingers. Other fingers are considered as in "closed"-poses.

For an obtruding fingertip t, it is apparent that either

$$d_2(t, C_0) > k \cdot R$$

or

$$z(C_0) - z(t) > \tau$$

where $k > 1$ is a scaling factor, and τ is a threshold that is related to the palm size.

4.4.1 Profile-based

Extended fingers can be located from the hand region profile.

An obtruding finger region is located using the following operation:

1. compute the curvature of the boundary curve;
2. local curvature extreme can be classified as "peak" points (for example, t_0 in Fig. 6) and "valley" points (for example, l_n in Fig. 6); "peak" points are potential fingertips;
3. a potential fingertip t_0 divides the boundary B into B_l and B_r;
4. starting from t_0, for each point l_i on B_l, a corresponding point r_i on B_r is computed as the nearest point to l_i; and vice versa a point l_j for each r_j on B_r:

$$r_i = \arg\min_{p \in B}\{d_2(l_i, p)\}$$

$$l_j = \arg\min_{q \in B}\{d_2(r_j, q)\}$$

5. terminate step 4 when a) a "valley" point is encountered, or b) distance $d_i = d_3(l_i, r_i) > \delta$ (see Fig. 6); δ is a threshold related to the palm size.

4.4.2 Depth-based

An obtruding finger merged with the palm region may not be identified in the hand profile, as the thumb shown in Fig. 7-left.

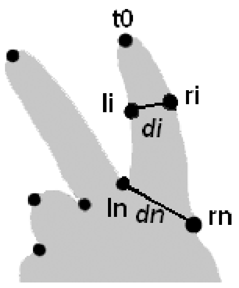

Figure 6. a hand region profile.

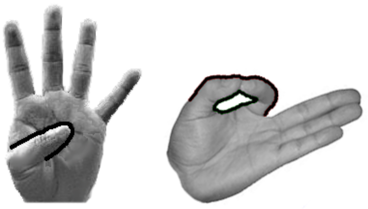

Figure 7. left: depth boundaries; right, an inner loop.

Depth boundaries D are located where a noticeable depth difference exists between neighboring points:

$$D = \{p \mid p \in F \,\&\, p \notin B \,\&\, \triangle\}$$

$$\triangle := \; (\exists q \in F, d_2(p, q) = 1 \,\&\, z(q) - z(p) > \delta)$$

A similar operation as in section 4.4.1 is performed on depth boundaries to locate a finger region. However, if no potential fingertips can be identified in a depth boundary, the pair points will be located on a parallel depth boundary. A depth boundary with no fingertips and no parallel depth boundaries is discarded.

4.4.3 Dealing with inner profile boundaries

An inner boundary exists when two fingers touch to form a loop, as shown in Fig. 7-right. A similar operation as in section 4.4.1 is performed on inner boundaries; however, for each point on an inner boundary , its pair is located on the outer boundary.

4.5. Classify finger poses and inter-relations

Each located finger region is defined by its boundary $\{li, ri\}(i = 1 - n)$. The number of fingers that may occupy one region is decided by the average width of the region:

$$\overline{w} = \frac{1}{n}\sum_i d_2(l_i, r_i)$$

For a single-finger region (i.e., $\overline{w} < k \cdot R$), the finger pose can be classified according to its depth values along the "middle curve" as shown in Fig. 8.

Fig. 9 shows the generalized patterns of depth values along the "middle curve" for different finger poses:

$$d_i = z(m_i)\ (i = 1 - n)$$

Such pattern difference is employed to recognize a finger pose.

For a multi-band region, if inner boundaries are involved, a "loop" relation between the thumb and another finger is recognized. A "cross" relation is distinguished from a "group" relation when a depth discontinuity is detected in the region.

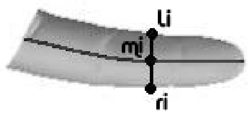

Figure 8. the "middle curve" of a singer-finger region is defined as points set $\{m_i = \frac{l_i+r_i}{2}\}$

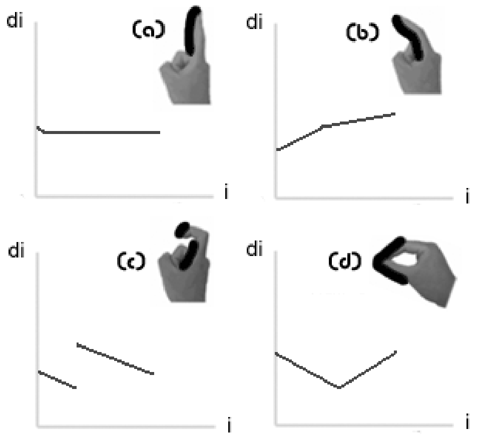

Figure 9. distinguishable depth value patterns along the finger region "middle line" for finger poses: (a) "up/side"; (b) "bend/half-bend"; (c) "half-closed"; (d) "loop".

4.6. Assign finger identities

The last step is to assign identities to each recognized finger, i.e., label each finger as one of the thumb, index finger, middle finger, ring finger, and small finger. As we constrain our poses to frontal views, we can simply assign fingers based on their ordering around the palm center. If there are gaps between two identified fingers, by default we assume one or more "closed"-pose fingers are in between depending on the size of a gap, as shown in Fig. 10.

No hand pose is recognized unless exactly five fingers are identified.

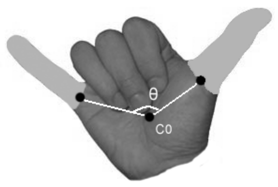

Figure 10. the number of "closed"-pose fingers between two identified fingers is determined by the angle θ; C_0 is the palm center.

4.7. Hand pose recognition

Assume n template hand poses $\{P_i \mid i = 1-n\}$ are to be recognized for an interface. Each hand pose is decomposed as finger poses and finger inter-relations. Using the encoding mechanism as described in Appendix A, we define the n template hand poses as n vectors $\{\vec{T_i} \mid i = 1 - n\}$.

A hand pose recognized from the camera is also encoded to a vector $\vec{T}$. Comparisons are performed between vector $\vec{T}$ and each $\vec{T_i}$, and the template hand pose P_k is recognized when two vectors $\vec{T}$ and $\vec{T_k}$ match.

The algorithm can be easily adapted to recognize different sets of hand poses by providing a corresponding set of encoding vectors for each hand poses. This feature is useful for building different gesture interfaces [6].

5. Discussion

There are a few hand poses and cases our algorithm fails to recognize:

1. Non-frontal view poses;
2. Hand poses that cannot be represented by our finger spelling schema, as shown in Fig. 11-left;
3. Hand poses in depth images with motion blur, as shown in Fig. 11-right.
4. Hand poses that are difficult to distinguish in low-resolution images; for example, our algorithm recognizes pose #01, #03, #23, #24, and #25 in Fig. 12 as the same pose.

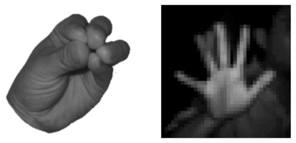

Figure 11. left, a recognition failure for a pose that cannot be encoded; right, a recognition failure because of motion blur.

However, most hand poses that are plausible candidates for a gesture interface can be represented in our finger-spelling schema, and can be recognized by our algorithm. The recognizable hand pose vocabulary is sufficient to build a sophisticated gesture interface. For example, 37 out of total 41 basic hand poses in SignWriting can be correctly recognized.

The benefit of representing hand poses using a finger spelling encoding schema is that both the recognized poses and the candidates are encoded as 12-digit vectors, to which efficient comparison can be performed. Thus, incorporating a large number of hand poses in one interface will not significantly impact the recognition efficiency.

6. Conclusion

In this paper, we present an algorithm for hand pose recognition using low-resolution depth images from a real-time laser camera. Reliable depth images provide robust hand region segmentation and remove ambiguities that many intensity or color image based algorithms suffer. Due to the constraints of low-resolution input, it is difficult to pursue a quantitatively plausible estimation of joint angles. Instead, we propose a finger-spelling schema to represent hand poses in a discrete space.

The algorithm is simple, efficient, and robust, recognizing a wide range of hand poses that are plausible for gesture interfaces from a frontal view (within +/- 30 FOV). The algorithm estimates an approximate hand model based on the measurement of palm size, thereby adapted to different users with one simple parameter. The implementation runs in real-time, consuming only 34% of the CPU time in average on a personal computer with a Pentium IV 1.0GHz processor. This hand pose recognition algorithm, together with a Canesta camera, provides a compelling and robust solution for gesture interfaces.

Our next steps are to extend the algorithm to recognize hand poses in non-frontal views and to extend the algorithm and recognition system to dynamic gestures.

References

[1] *Http://www.signwriting.org/.*

[2] V. Athitsos and S. Sclaroff. An appearance-based framework for 3d hand shape classification and camera viewpoint estimation. In *Proc. 5th Intl Conf. On Automatic Face and Gesture Recognition*, pages 40–45, 2002.

[3] Canesta. *Canesta electronic perception SDK reference manual, version 2.0.*

[4] S. Lee and I. Cohen. 3d hand reconstruction from a monocular view. In *Proc. of the 7th Intl Conf. On Pattern Recognition (ICPR 04)*, 2004.

[5] J. Martin and J. L. Crowley. An appearance-based approach to gesture-recognition. In *ICIAP (2)*, pages 340–347, 1997.

[6] Z. Mo and U. Neumann. Lexical gesture interface. In *Proc. IEEE ICVS 06*, 2006.

[7] K. Oka, Y. Sato, and H. Koike. Real-time tracking of multiple fingertips and gesture recognition for augmented desk interface systems. In *Proc. Face and Gesture 98*, 1998.

[8] V. Pavlovic, R. Sharma, and T. Huang. Visual interpretation of hand gestures for human-computer interaction: a review. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 19(7), July 1997.

[9] J. Rehg and T. Kanade. Visual tracking of high dof articulated structures: an application to human hand tracking. In *3rd European Conference on Computer Vision (ECCV 94)*, pages 35–46. Springer-Verlag, 1994.

[10] A. Rosenberg. *Writing signed languages: in support of adopting an ASL writing system (Matser thesis).* University of Kansas, Department of Linguistics, 1999.

[11] N. Shimada, K. Kimura, and Y. Shirai. Real-time 3-d hand estimation on 2-d appearance retrieval using monocular camera. In *Proc. of the IEEE ICCV Workshop on RATFG-RTS 01*, pages 23–30, 2001.

[12] N. Shimada, Y. Shirai, Y. Kuno, and J. Miura. Hand gesture estimation and model refinement using monocular camera - ambiguity limitation by inequality constraints. In *Proc. of 3rd Intl Conf. On Automatic Face and Gesture Recognition*, pages 268–273, 1998.

[13] B. Stenger, P. R. S. Mendonca, and R. Cipolla. Model-based 3d tracking of an articulated hand. In *Proc. Conference on Computer Vision and Pattern Recognition*, volume II, pages 310–315, 2001.

[14] Y. Wu and T. Huang. View-independent recognition of hand postures. In *Proc. of CVPR 2000*, volume II, pages 88–94, 2000.

[15] Y. Wu, L. J. Y., and T. Huang. Capturing natural hand articulation. In *Proc. 8th Int'l Conf. on Computer Vision*, volume II, pages 426–432, 2001.

Appendix A: encoding a hand pose using finger-spelling schema

f_1	f_2	f_3	f_4	f_5	f_{12}	f_{13}	f_{14}	f_{15}	f_{23}	f_{34}	f_{45}

We use a 12-digit integer vector (however, 36 bits are sufficient using binary encoding) to encode a hand pose: five digits $\{f_i\}$ (i=1~5), f_i encoding finger i's pose (f_1 for thumb, f_2 for index finger, etc.); seven digits $\{f_{12}, f_{13}, f_{14}, f_{15}, f_{23}, f_{34}, f_{45}\}$, f_{ij} encoding the inter-relation between finger i and j.

f_i = 0 : unspecified	f_{ij} = 0 : unspecified
1 : up	1 : separate
2 : forward	2 : group
3 : side	3 : cross (f_i in front of f_j)
4 : half-bend	4 : cross (f_j in front of f_i)
5 : bend	5 : loop
6 : half-closed	
7 : closed	

Appendix B: Encoding of 41 basic hand poses in SignWriting

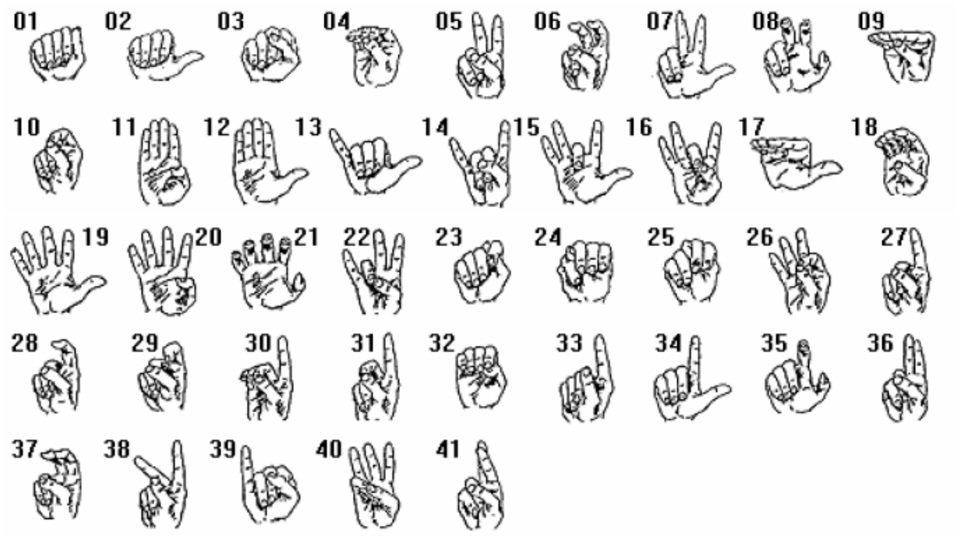

Figure 12: 41 basic hand poses in SignWriting (copied from [1])

01 : [1 7 7 7 7; 2 1 1 1 2 2 2]	15 : [3 1 7 1 1; 1 1 1 1 1 1 1]	29 : [7 6 7 7 7; 1 1 1 1 1 2 2]
02 : [3 7 7 7 7; 1 1 1 1 2 2 2]	16 : [5 1 5 1 1; 1 5 1 1 1 1 1]	30 : [5 1 5 5 5; 1 5 1 1 1 2 2]
03 : [7 7 7 7 7; 1 1 1 1 2 2 2]	17 : [3 2 2 2 2; 1 1 1 1 2 2 2]	31 : [5 1 5 7 7; 1 5 1 1 1 1 2]
04 : [5 5 5 5 5; 5 1 1 1 2 2 2]	18 : [5 4 4 4 4; 1 1 1 1 2 2 2]	32 : [7 6 6 6 6; 1 1 1 1 2 2 2]
05 : [7 1 1 7 7; 1 1 1 1 1 1 2]	19 : [3 1 1 1 1; 1 1 1 1 1 1 1]	33 : [7 1 7 7 7; 1 1 1 1 1 2 2]
06 : [7 4 4 7 7; 1 1 1 1 1 1 2]	20 : [7 1 1 1 1; 1 1 1 1 1 1 1]	34 : [3 1 7 7 7; 1 1 1 1 1 2 2]
07 : [3 1 1 7 7; 1 1 1 1 1 1 2]	21 : [4 4 4 4 4; 1 1 1 1 1 1 1]	35 : [4 4 7 7 7; 1 1 1 1 1 2 2]
08 : [4 4 4 7 7; 1 1 1 1 1 1 2]	22 : [5 1 1 5 1; 1 1 5 1 1 1 1]	36 : [7 1 1 7 7; 1 1 1 1 2 1 2]
09 : [2 2 2 2 2; 5 1 1 1 2 2 2]	23 : [7 7 7 7 7; 1 1 1 1 1 2 2]	37 : [7 4 4 7 7; 1 1 1 1 2 1 2]
10 : [5 5 7 7 7; 5 1 1 1 1 2 2]	24 : [7 7 7 7 7; 1 1 1 1 2 2 1]	38 : [7 1 2 7 7; 1 1 1 1 1 1 2]
11 : [7 1 1 1 1; 1 1 1 1 2 2 2]	25 : [7 7 7 7 7; 1 1 1 1 2 1 2]	39 : [7 7 7 7 1; 1 1 1 1 2 2 1]
12 : [3 1 1 1 1; 1 1 1 1 2 2 2]	26 : [5 5 1 1 1; 5 1 1 1 1 1 1]	40 : [7 1 1 1 7; 1 1 1 1 1 1 1]
13 : [3 7 7 7 1; 1 1 1 1 2 2 1]	27 : [7 1 7 7 7; 1 1 1 1 1 2 2]	41 : [7 1 1 7 7; 1 1 1 1 3 1 2]
14 : [7 1 7 7 1; 1 1 1 1 1 2 1]	28 : [7 4 7 7 7; 1 1 1 1 1 2 2]	