EXHIBIT 4.19

computing a scaling velocity from a change in a distance between the innermost and outermost fingers ***(Fig. 4c and Col. 5, 41-58 and Col. 6, lines 6-19)***;

supplementing the computed scaling velocity with a measure of scaling velocity selective for symmetric scaling about a fixed point between a thumb and other fingers of the given hand ***(Col. 8, line 65—Col. 9, line 36)***; and

transmitting the computed, supplemented scaling velocity as a control signal to an electronic or electromechanical device ***(Col. 6, lines 6-19)***.

As to **Claim 21**, Yasutake teaches computing a rotational velocity from a change in angle between the innermost and outermost fingers ***(Fig. 4b and Col. 4, line 4-23, Col. 5, lines 5-23 and Col. 5, line 59—Col. 6, line 5)***;

supplementing the computed rotational velocity with a measure of rotational velocity selective for symmetric rotation about a fixed point between the thumb and other fingers of the given hand ***(Col. 8, line 65—Col. 9, line 36)***; and

transmitting the computed rotational velocity as a control signal to an electronic or electromechanical device ***(Fig. 4b and Col. 4, line 4-23, Col. 5, lines 5-23 and Col. 5, line 59—Col. 6, line 5)***.

As to **Claims 22**, Yasutake teaches computing a translation weighting for each finger ***(Fig. 4a, note that the translation weighting is the same for each contact)***;

computing a translational velocity for each finger ***(Fig. 4a and Col. 4, lines 4-23 and Col. 5, lines 5-25 and 41-58)***;

computing a translational velocity average from the computed translational velocity components and the computed translation weightings ***(Fig. 4a and Col. 4, lines 4-23 and Col. 5, lines 5-25 and 41-58)***; and

transmitting the computed, supplemented translational velocity average as a control signal to an electronic or electromechanical device ***(Fig. 4a and Col. 4, lines 4-23 and Col. 5, lines 5-25 and 41-58)***.

As to **Claims 26**, Yasutake teaches computing a translation weighting for each finger ***(Fig. 4a, note that the translation weighting is the same for each contact)***;

computing a translational velocity for each finger ***(Fig. 4a and Col. 4, lines 4-23 and Col. 5, lines 5-25 and 41-58)***;

computing a translational velocity average from the computed translational velocities and the computed translation weightings ***(Fig. 4a and Col. 4, lines 4-23 and Col. 5, lines 5-25 and 41-58)***; and

transmitting the computed, supplemented translational velocity average as a control signal to an electronic or electromechanical device ***(Fig. 4a and Col. 4, lines 4-23 and Col. 5, lines 5-25 and 41-58)***.

As to **Claim 30**, Yasutake teaches method for extracting multiple degrees of freedom of hand motion from successive proximity images representing successive scans of a plurality of proximity sensors of a multi-touch surface, the method comprising:

tracking, through successive proximity images, a plurality of contacts associated with a plurality of fingers ***(Col. 2, lines 15-45, Col. 3, lines 40-53 and Col. 4, lines 4-23)***;

finding an innermost finger and an outermost finger for a given hand ***(See Figs. 1, 2 and 4a-4c)***;

computing a rotational velocity from a change in angle between the innermost and outermost fingers ***(Fig. 4b and Col. 4, line 4-23, Col. 5, lines 5-23 and Col. 5, line 59—Col. 6, line 5)***;

supplementing the computed rotational velocity with a measure of rotational velocity selective for symmetric rotation about a fixed point between a thumb and other fingers of the given hand ***(Col. 8, line 65—Col. 9, line 36)***;

transmitting the computed, supplemented rotational velocity as a control signal to an electronic or electromechanical device ***(Fig. 4b and Col. 4, line 4-23, Col. 5, lines 5-23 and Col. 5, line 59—Col. 6, line 5)***.

As to **Claim 31**, Yasutake teaches computing a translation weighting for each finger ***(Fig. 4a, note that the translation weighting is the same for each contact)***;

computing a translational velocity for each finger ***(Fig. 4a and Col. 4, lines 4-23 and Col. 5, lines 5-25 and 41-58)***;

computing a translational velocity average from the computed translational velocities and the computed translation weightings ***(Fig. 4a and Col. 4, lines 4-23 and Col. 5, lines 5-25 and 41-58)***; and

transmitting the computed, supplemental translational velocity average as a control signal to an electronic or electromechanical device ***(Fig. 4a and Col. 4, lines 4-23 and Col. 5, lines 5-25 and 41-58)***.

As to **Claim 35**, Yasutake teaches a method for extracting multiple degrees of freedom of hand motion from successive proximity images representing successive scans of a plurality of proximity sensors of a multi-touch surface, the method comprising:

tracking, through successive proximity images, a plurality of contacts associated with a plurality of fingers ***(Col. 2, lines 15-45, Col. 3, lines 40-53 and Col. 4, lines 4-23)***;

computing a translation weighting for each finger ***(Fig. 4a, note that the translation weighting is the same for each contact)***;

computing a translational velocity for each finger ***(Fig. 4a and Col. 4, lines 4-23 and Col. 5, lines 5-25 and 41-58)***;

computing a translational velocity average from the computed translational velocities and the computed translation weightings ***(Fig. 4a and Col. 4, lines 4-23 and Col. 5, lines 5-25 and 41-58)***; and

transmitting the computed, supplemented translational velocity average as a control signal to an electronic or electromechanical device ***(Fig. 4a and Col. 4, lines 4-23 and Col. 5, lines 5-25 and 41-58)***.

Allowable Subject Matter

3. Claims 4-6, 8-10, 13-15, 17-19, 23-25, 27-29, 32-34 and 36-38 would be allowable if rewritten to overcome the rejection(s) under 35 U.S.C. 112, 2nd paragraph, set forth in this Office action and to include all of the limitations of the base claim and any intervening claims.

4. The following is a statement of reasons for the indication of allowable subject matter: As to Claims 4, 8, 13, 17, 23, 27, 32 and 36, the prior art fails to teach or suggest, either alone or in combination that "the computed translation weightings of innermost and outermost fingers are constant and computed translation weightings of central fingers are inversely related to polar component speeds so as to prevent vertical translation bias while performing hand scaling and rotation but otherwise include all available fingers in the computed translational velocity average."

Inquiries

Any inquiry concerning this communication or earlier communications from the examiner should be directed to RODNEY AMADIZ whose telephone number is (571)272-7762. The examiner can normally be reached on M-F 8:30-5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Sumati Lefkowitz can be reached on (571) 272-3638. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Sumati Lefkowitz/
Supervisory Patent Examiner, Art Unit 2629

/R. A./
Examiner, Art Unit 2629
9/28/09

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/559,736	11/14/2006	WAYNE WESTERMAN	10684-20086.08	4583

69753 7590 09/02/2009

APPLE C/O MORRISON AND FOERSTER ,LLP
LOS ANGELES
555 WEST FIFTH STREET SUITE 3500
LOS ANGELES, CA 90013-1024

EXAMINER	
AMADIZ, RODNEY	
ART UNIT	**PAPER NUMBER**
2629	

MAIL DATE	DELIVERY MODE
09/02/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	11/559,736	WESTERMAN ET AL.	
	Examiner	Art Unit	
	RODNEY AMADIZ	2629	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *14 November 2006*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-23* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-23* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *14 November 2006* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *12/12/08, 3/27/09, 7/08/09*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Specification

1. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

Claim Objections

2. The numbering of claims is not in accordance with 37 CFR 1.126.

Misnumbered claims 20-24 have been renumbered 19-23.

Claim Rejections - 35 USC § 101

3. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 1-17 are rejected under 35 U.S.C. 101 as not falling within one of the four statutory categories of invention. Based on Supreme Court precedent and recent Federal Circuit decisions, a statutory "process" under 35 U.S.C. 101 must (1) be tied to another statutory category (such as a particular apparatus), or (2) transform underlying subject matter (such as an article or material) to a different state or thing. While the instant claim recites a series of steps or acts to be performed, the claim neither transforms underlying subject matter nor is positively tied to another statutory category that accomplishes the claimed method steps, and therefore does not qualify as a

statutory process.

Claims 1 and 2 do not explicitly recite a structural tie to perform the steps claimed. The Applicant has provided no explicit and deliberate definitions of “generating”, “finding”, “paring” and “predicting” to limit the steps to the electronic form and the claim language itself is sufficiently broad to read on a user simply mentally generating one or more predicted paths, mentally finding the closest predicted path, mentally finding the closest surface contact having a centroid closest to the predicted path and mentally pairing each surface contact with its closest predicted path if the surface contact is also the closest surface contact to the predicted path. Claim 2 follows the same logic.

Claim 18 is **not** rejected under 35 U.S.C. 101 because the claim requires that w the paths be used to generate user interface interactions thereby tying the method step to a machine of some sort.

Claim Rejections - 35 USC § 102

4. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

5. Claims 1-9 and 18-23 are rejected under 35 U.S.C. 102(b) as being anticipated by Yasutake (U.S. Patent 5,483,261—hereinafter "Yasutake").

As to **Claim 1**, Yasutake teaches a method for associating into paths one or more surface contacts from successive proximity images, the successive proximity images including a current proximity image and one or more prior proximity images ***(Figs. 5 and 7a-d)***, the method comprising:

generating one or more predicted paths by predicting from the one or more prior proximity images current positions of the one or more surface contacts ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***;

finding for each surface contact in the current proximity image a closest predicted path ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***;

finding for each predicted path a closest surface contact, the closest surface contact having a centroid closest to the predicted path ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***; and

pairing each surface contact with its closest predicted path if the surface contact is also the closest surface contact to the predicted path ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

As to **Claim 2**, Yasutake teaches a method for associating into paths one or more surface contacts from successive proximity images, the successive proximity images including a current proximity image and one or more prior proximity images ***(Figs. 5 and 7a-d)***, the method comprising:

generating one or more predicted paths by predicting from the one or more prior proximity images current positions of the one or more surface contacts***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***;

finding for each surface contact in the current proximity image a closest predicted path ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***;

finding for each predicted path a closest surface contact, the closest surface contact having a centroid closest to the predicted path and within a path-dependent tracking radius of the predicted path ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***; and

pairing each surface contact with its closest predicted path if the surface contact is also the closest surface contact to the predicted path and if the centroid of the closest surface contact is within the path-dependent tracking radius of the predicted path ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

As to **Claims 3/1 and 3/2**, Yasutake teaches starting new paths for unpaired surface contacts ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

As to **Claim 4**, Yasutake teaches deactivating unpaired predicted paths ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

As to **Claims 5/1 and 5/2**, Yasutake teaches deactivating unpaired predicted paths ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

As to **Claims 6/1 and 6/2**, Yasutake teaches updating path parameters for each of the one or more predicted paths from one or more measured parameters of the surface contact paired with each predicted path ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

As to **Claims 7**, Yasutake teaches starting new paths for unpaired surface contacts ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

As to **Claims 8**, Yasutake teaches deactivating unpaired predicted paths ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

As to **Claims 9**, Yasutake teaches deactivating unpaired predicted paths ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

As to **Claim 18**, Yasutake teaches a method for associating into paths one or more groups of pixels from successive proximity images, each group of pixels corresponding to a distinguishable hand part or other touch object on or near the surface of a multi-touch apparatus and each proximity image representing a scan of a plurality of proximity sensors of the multi-touch apparatus, the successive proximity

images including a current proximity image and one or more prior proximity images, the method comprising ***(Figs. 5 and 7a-d)***:

predicting paths for each of the one or more groups of pixels from the one or more prior proximity images ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***;

pairing each group of pixels with its predicted path ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***;

whereby the paths may be used to generate user interface interactions in response to motion of the distinguishable hand parts or other touch objects through the successive proximity images ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

As to **Claim 19 (submitted as claim 20 on 11/14/06)**, Yasutake teaches starting new paths for unpaired groups of pixels ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

As to **Claim 20 (submitted as claim 21 on 11/14/06)**, Yasutake teaches deactivating unpaired predicted paths ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

As to **Claim 21 (submitted as claim 22 on 11/14/06)**, Yasutake teaches deactivating unpaired predicted paths ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

As to **Claim 22 (submitted as claim 23 on 11/14/06)**, Yasutake teaches that at least one of the groups of pixels corresponds to a hand part near but not on the multi-

touch surface ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

As to **Claim 23 (submitted as claim 24 on 11/14/06)**, Yasutake teaches starting new paths for unpaired groups of pixels, wherein at least one unpaired group of pixels corresponds to a hand part near but not on the multi-touch surface ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 10-17 are rejected under 35 U.S.C. 103(a) as being unpatentable over Yasutake in view of McAvinney (U.S. Patent 4,746,770—hereinafter "McAvinney").

8. As to **Claim 10**, Yasutake fails to teach determining a velocity of the one or more surface contacts along one or more corresponding existing paths. Examiner cites McAvinney to teach determining a velocity of the one or more surface contacts along one or more corresponding existing paths ***(See Figs. 19, 22, 39 and 40 and Col. 1, line 40-46, Col. 6, lines 49-54, Col. 9, lines 55-63, Col. 11, line 46—Col. 12, line 14 and Col. 15, line 40—Col. 17, line 33)***. At the time the invention was made, it would have been obvious to a person of ordinary skill in the art to determine the velocity of the one or more surface contacts along one or more corresponding existing paths as taught by

McAvinney in the method taught by Yasutake in order to permit the identification and the alteration of graphical objects ***(McAvinney – Abstract)***.

As to **Claim 11**, Yasutake teaches starting new paths for unpaired surface contacts ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

As to **Claim 12**, Yasutake teaches deactivating unpaired predicted paths ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

As to **Claim 13**, Yasutake teaches deactivating unpaired predicted paths ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

As to **Claim 14**, Yasutake teaches updating path parameters for each of the one or more predicted paths from one or more measured parameters of the surface contact paired with each predicted path ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

As to **Claim 15**, Yasutake teaches starting new paths for unpaired surface contacts ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

As to **Claim 16**, Yasutake teaches deactivating unpaired predicted paths ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

As to **Claim 17**, Yasutake teaches deactivating unpaired predicted paths ***(Figs. 5 and 7a-7d, Col. 6, line 20—Col. 7, line 18, Col. 7, lines 60-67 and Col. 8, lines 48-64)***.

Inquiries

Any inquiry concerning this communication or earlier communications from the examiner should be directed to RODNEY AMADIZ whose telephone number is (571)272-7762. The examiner can normally be reached on M-F 8:30-5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Sumati Lefkowitz can be reached on (571) 272-3638. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Sumati Lefkowitz/
Supervisory Patent Examiner, Art Unit 2629

/R. A./
Examiner, Art Unit 2629
8/27/09

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/428,522	07/03/2006	WAYNE WESTERMAN	10684-20086.06	4146

69753 7590 08/25/2009

APPLE C/O MORRISON AND FOERSTER ,LLP
LOS ANGELES
555 WEST FIFTH STREET SUITE 3500
LOS ANGELES, CA 90013-1024

EXAMINER	
SHAPIRO, LEONID	
ART UNIT	PAPER NUMBER
2629	

MAIL DATE	DELIVERY MODE
08/25/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	11/428,522	WESTERMAN ET AL.	
	Examiner	Art Unit	
	Leonid Shapiro	2629	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *11 June 2009*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,3-10 and 16-22* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,3-10 and 16-22* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

1. The drawings are objected to under 37 CFR 1.83(a). The drawings must show every feature of the invention specifics disclosed in the claims. Therefore, the newly introduced limitations of independent claims 1,16-17,20: ": form a proximity image of multiple ***touch contacts*** on the touch surface; segmenting the proximity image into a plurality of groups of electrodes, ***each group representing a distinguishable contact***..." must be shown or the feature(s) canceled from the claim(s). No new matter should be entered.

Corrected drawing sheets in compliance with 37 CFR 1.121(d) are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. The figure or figure number of an amended drawing should not be labeled as "amended." If a drawing figure is to be canceled, the appropriate figure must be removed from the replacement sheet, and where necessary, the remaining figures must be renumbered and appropriate changes made to the brief description of the several views of the drawings for consistency. Additional replacement sheets may be necessary to show the renumbering of the remaining figures. Each drawing sheet submitted after the filing date of an application must be labeled in the top margin as either "Replacement Sheet" or "New Sheet" pursuant to 37 CFR 1.121(d). If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

Specification

2. The specification is objected to as failing to provide proper antecedent basis for the claimed subject matter. See 37 CFR 1.75(d)(1) and MPEP § 608.01(o). Correction of the following is required:

The newly introduced limitations of independent claims 1,16-17,20: ": form a proximity image of multiple ***touch contacts*** on the touch surface; segmenting the proximity image into a plurality of groups of electrodes, ***each group representing a distinguishable contact***…" are not described in the Specification.

Notice that proximity images are scanned, segmented in order to find a (single) distinguishable contact, and only after that multiple touch contacts could be found.

Claim Rejections - 35 USC § 112

The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

3. Claims 1,3-10, 16-22 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention.

The newly introduced limitations of independent claims 1,16-17,20: ": form a proximity image of multiple ***touch contacts*** on the touch surface; segmenting the

proximity image into a plurality of groups of electrodes, ***each group representing a distinguishable contact***…" are not described in the Specification or shown in Figures.

Notice that proximity images are scanned, segmented in order to find a (single) distinguishable contact, and only after that multiple touch contacts could be found.

Claims 3-10,18-19,21-22 depend on claims 1,17,20.

4. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

Claim 1-10,16-22 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

It not clear how to form a proximity image of multiple touch contacts on the touch surface ***before*** segmenting the proximity image into a plurality of groups of electrodes, each group representing a distinguishable contact, as recited in independent claims 1,16-17,20?

Notice, that proximity images are images of shadow of the object near to contact surface (hovering) and single or multiple contacts could be define only after analyzing proximity images.

Claims 2-10,18-19,21-22 depend on claims 1,17,20.

Rejection on merits could be only done after resolving enablement and indefinites problems.

Telephone inquire

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Leonid Shapiro whose telephone number is 571-272-7683. The examiner can normally be reached on 8 a.m. to 5 p.m..

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Richard Hjerpe can be reached on 571-272-7691. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

08/16/09
/L. S./
Examiner, Art Unit 2629

/Richard Hjerpe/
Supervisory Patent Examiner, Art Unit 2629

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/428,515	07/03/2006	WAYNE WESTERMAN	10684-20086.05	4133

69753 7590 11/19/2009

APPLE C/O MORRISON AND FOERSTER ,LLP
LOS ANGELES
555 WEST FIFTH STREET SUITE 3500
LOS ANGELES, CA 90013-1024

EXAMINER	
PERVAN, MICHAEL	
ART UNIT	PAPER NUMBER
2629	

MAIL DATE	DELIVERY MODE
11/19/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	11/428,515	WESTERMAN ET AL.	
	Examiner	Art Unit	
	Michael Pervan	2629	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *14 August 2009*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-24* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-24* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *03 July 2006* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *3/20/09 6/18/09 8/4/09 8/19/09 9/23/09 10/27/09*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

9DETAILED ACTION

Response to Arguments

1. Applicant's arguments with respect to claims 1-24 have been considered but are moot in view of the new ground(s) of rejection.

Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1, 5, 15, 17, 19, 21 and 23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Yasutake in view of Shieh (US 5,748,184; previously presented) in further view of Siddiqui et al.

In regards to claims 1, 17 and 21, Yasutake discloses a method for mapping gestures performed on a multi- touch surface to graphical user interface commands, the method comprising:

detecting a plurality of contacts on the multi-touch surface (col. 3, lines 49-61); and

determining a whole hand translation across the multi-touch surface from movement of the whole hand contacts (col. 4, lines 4-23).

Yasutake does not disclose matching the detected plurality of contacts to distinct parts of a hand and generating a pan command in response to the whole hand translation on the multi-touch surface.

Shieh discloses matching the detected plurality of contacts to distinct parts of a hand (Fig. 2 and col. 3, lines 46-61).

It would have been obvious at the time of invention to modify Yasutake with the teachings of Shieh, detecting a handprint, because it would allow the system to distinguish between users, allowing the user to have their own custom commands.

Yasutake and Shieh do generating a pan command in response to the whole hand translation on the multi-touch surface.

Siddiqui discloses generating a pan command in response to the whole hand translation (col. 21, lines 25-44). By incorporating the motion of moving a mouse to generate a pan command as shown in Siddiqui into the device of Yasutake, one would achieve the ability to generate a pan command by mimicking the motion of moving a mouse.

It would have been obvious at the time of invention to modify Yasutake and Shieh with the teachings of Siddiqui, generating pan command by moving mouse, because it allows the user to freely move what is being displayed on the display without having to use scroll bars.

In regards to claim 5, Yasutake does not disclose the method of claim 1 further comprising generating a zoom command in response to detecting hand scaling by uniformly flexing or extending fingers on the multi-touch surface.

Siddiqui discloses generating a zoom command in response to detecting hand scaling by uniformly flexing or extending fingers (col. 18, lines 20-35). By incorporating the motion of moving a scroll wheel to generate a zoom command as shown in Siddiqui into the device of Yasutake, one would achieve the ability to generate a zoom command by mimicking the motion of moving a scroll wheel.

It would have been obvious at the time of invention to modify Yasutake with the teachings of Siddiqui, generating zoom command by moving scroll wheel, because it allows the user to freely zoom what is being displayed on the display without having to continuously touch a zoom in or zoom out button.

In regards to claims 15, 19 and 23, Yasutake discloses a method for mapping gestures performed on a multi- touch surface to graphical user interface commands, the method comprising:

detecting a plurality of contacts on the multi-touch surface (col. 3, lines 49-61).

Yasutake does not disclose matching the detected plurality of contacts to distinct parts of a hand and detecting hand scaling resulting from uniformly flexing or extending the fingers on the multi-touch surface and generating a zoom command in response to the detected hand scaling.

Shieh discloses matching the detected plurality of contacts to distinct parts of a hand (Fig. 2 and col. 3, lines 46-61).

It would have been obvious at the time of invention to modify Yasutake with the teachings of Shieh, detecting a handprint, because it would allow the system to distinguish between users, allowing the user to have their own custom commands.

Yasutake and Shieh do not disclose detecting hand scaling resulting from uniformly flexing or extending the fingers on the multi-touch surface and generating a zoom command in response to the detected hand scaling.

Siddiqui discloses detecting hand scaling resulting from uniformly flexing or extending the fingers on the multi-touch surface and generating a zoom command in response to the detected hand scaling (col. 18, lines 20-35). By incorporating the motion of moving a scroll wheel to generate a zoom command as shown in Siddiqui into the device of Yasutake, one would achieve the ability to generate a zoom command by mimicking the motion of moving a scroll wheel.

It would have been obvious at the time of invention to modify Yasutake and Shieh with the teachings of Siddiqui, generating zoom command by moving scroll wheel, because it allows the user to freely zoom what is being displayed on the display without having to continuously touch a zoom in or zoom out button.

4. Claims 2-4 and 6-8 are rejected under 35 U.S.C. 103(a) as being unpatentable over Yasutake in view of Shieh in view of Siddiqui et al in further view of Redlich.

In regards to claim 2, Yasutake, Shieh and Siddiqui do not disclose the method of claim 1 further comprising generating a rotate command in response to detecting a hand rotation on the multi-touch surface.

Redlich discloses generating a rotate command in response to detecting a hand rotation (col. 10, lines 29-58). By incorporating the motion of rotating a mouse to generate a rotate command as shown in Redlich into the device of Yasutake, one would achieve the ability to generate a rotate command by mimicking the motion of rotating a mouse.

It would have been obvious at the time of invention to modify Yasutake, Shieh and Siddiqui with the teachings of Redlich, generating rotate command by rotating a mouse, because it would allow the user to rotate what is being displayed on the display without having to touch a rotate left or rotate right button.

In regards to claim 3, Yasutake, Shieh and Siddiqui do not disclose the method of claim 2 wherein the hand rotation is rotation about a wrist.

Redlich discloses wherein the hand rotation is rotation about a wrist (col. 10, lines 29-58).

It would have been obvious at the time of invention to modify Yasutake, Shieh and Siddiqui with the teachings of Redlich, generating rotate command by rotating a mouse, because it would allow the user to rotate what is being displayed on the display without having to touch a rotate left or rotate right button.

In regards to claim 4, Yasutake, Shieh, Siddiqui and Redlich do not disclose the method of claim 2 wherein the hand rotation is rotation between fingers.

Redlich discloses wherein the hand rotation is rotation about a wrist (col. 10, lines 29-58).

Since there is no benefit or advantage described in the specification for choosing hand rotation about a wrist or between fingers, it would have been obvious to one of ordinary skill in the art at the time of invention to choose either having hand rotation between fingers or about a wrist based on a design choice.

In regards to claims 6-8, Yasutake and Shieh do not disclose the method of claim 2 further comprising generating a zoom command in response to detecting hand scaling by uniformly flexing or extending fingers on the multi-touch surface.

Siddiqui discloses generating a zoom command in response to detecting hand scaling by uniformly flexing or extending fingers (col. 18, lines 20-35). By incorporating the motion of moving a scroll wheel to generate a zoom command as shown in Siddiqui into the device of Yasutake, one would achieve the ability to generate a zoom command by mimicking the motion of moving a scroll wheel.

It would have been obvious at the time of invention to modify Yasutake and Shieh with the teachings of Siddiqui, generating zoom command by moving scroll wheel, because it allows the user to freely zoom what is being displayed on the display without having to continuously touch a zoom in or zoom out button.

5. Claims 9-11, 18 and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Yasutake in view of Shieh in further view of Redlich.

In regards to claims 9, 18 and 22, Yasutake discloses a method for mapping gestures performed on a multi- touch surface to graphical user interface commands, the method comprising:

detecting a plurality of contacts on the multi-touch surface (col. 3, lines 49-61).

Yasutake does not disclose matching the detected plurality of contacts to distinct parts of a hand and determining a hand rotation from movement of the hand contacts and generating a rotate command in response to detecting a hand rotation on the multi-touch surface.

Shieh discloses matching the detected plurality of contacts to distinct parts of a hand (Fig. 2 and col. 3, lines 46-61).

It would have been obvious at the time of invention to modify Yasutake with the teachings of Shieh, detecting a handprint, because it would allow the system to distinguish between users, allowing the user to have their own custom commands.

Yasutake and Shieh do not disclose determining a hand rotation from movement of the hand contacts and generating a rotate command in response to detecting a hand rotation on the multi-touch surface.

Redlich discloses determining a hand rotation from movement of the hand and generating a rotate command in response to detecting a hand rotation (col. 10, lines 29-58). By incorporating the motion of rotating a mouse to generate a rotate command as

shown in Redlich into the device of Yasutake, one would achieve the ability to generate a rotate command by mimicking the motion of rotating a mouse.

It would have been obvious at the time of invention to modify Yasutake and Shieh with the teachings of Redlich, generating rotate command by rotating a mouse, because it would allow the user to rotate what is being displayed on the display without having to touch a rotate left or rotate right button.

In regards to claim 10, Yasutake and Shieh do not disclose the method of claim 2 wherein the hand rotation is rotation about a wrist.

Redlich discloses wherein the hand rotation is rotation about a wrist (col. 10, lines 29-58).

It would have been obvious at the time of invention to modify Yasutake and Shieh with the teachings of Redlich, generating rotate command by rotating a mouse, because it would allow the user to rotate what is being displayed on the display without having to touch a rotate left or rotate right button.

In regards to claim 11, Yasutake, Shieh and Redlich do not disclose the method of claim 2 wherein the hand rotation is rotation between fingers.

Redlich discloses wherein the hand rotation is rotation about a wrist (col. 10, lines 29-58).

Since there is no benefit or advantage described in the specification for choosing hand rotation about a wrist or between fingers, it would have been obvious to one of

ordinary skill in the art at the time of invention to choose either having hand rotation between fingers or about a wrist based on a design choice.

6. Claims 12-14 are rejected under 35 U.S.C. 103(a) as being unpatentable over Yasutake in view of Shieh in view of Redlich in further view of Siddiqui et al.

In regards to claims 12-14, Yasutake, Shieh and Redlich do not disclose the method of claim 2 further comprising generating a zoom command in response to detecting hand scaling by uniformly flexing or extending fingers on the multi-touch surface.

Siddiqui discloses generating a zoom command in response to detecting hand scaling by uniformly flexing or extending fingers (col. 18, lines 20-35). By incorporating the motion of moving a scroll wheel to generate a zoom command as shown in Siddiqui into the device of Yasutake, one would achieve the ability to generate a zoom command by mimicking the motion of moving a scroll wheel.

It would have been obvious at the time of invention to modify Yasutake, Shieh and Redlich with the teachings of Siddiqui, generating zoom command by moving scroll wheel, because it allows the user to freely zoom what is being displayed on the display without having to continuously touch a zoom in or zoom out button.

7. Claims 16, 20 and 24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Yasutake in view of Shieh in further view of Cutler et al.

In regards to claims 16, 20 and 24, Yasutake discloses a method for mapping gestures performed on a multi- touch surface to graphical user interface commands, the method comprising:

detecting a plurality of contacts on the multi-touch surface (col. 3, lines 49-61).

Yasutake does not disclose matching the detected plurality of contacts to distinct parts of a hand and detecting a first gesture being performed by the first hand and a second gesture being performed by the second hand and manipulating a foreground object in accordance with the first gesture and manipulating a background object with the second gesture.

Shieh discloses matching the detected plurality of contacts to distinct parts of a hand (Fig. 2 and col. 3, lines 46-61).

It would have been obvious at the time of invention to modify Yasutake with the teachings of Shieh, detecting a handprint, because it would allow the system to distinguish between users, allowing the user to have their own custom commands.

Yasutake does not disclose detecting a first gesture being performed by the first hand and a second gesture being performed by the second hand and manipulating a foreground object in accordance with the first gesture and manipulating a background object with the second gesture.

Cutler discloses detecting a first gesture (moving hand to position model) being performed by the first hand and a second gesture (moving hand to zoom) being performed by the second hand and manipulating (position model) a foreground object in accordance with the first gesture and manipulating a background object (zooming) with

the second gesture (Table 1 and page 5, Coordinated Asymmetric Interaction, paragraph 2; two-handed zooming).

It would have been obvious at the time of invention to modify Yasutake and Shieh with the teachings of Cutler, manipulating objects and backgrounds, because it gives the user better control over because they are using their arms and hands to manipulate an object or background rather using another object.

Conclusion

8. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

9. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael Pervan whose telephone number is (571) 272-0910. The examiner can normally be reached on Monday - Friday between 8am - 5pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Amr Awad can be reached on (571) 272-7764. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

MVP

/Amr Awad/
Supervisory Patent Examiner, Art Unit 2629

Nov. 3, 2009

The Automatic Recognition of Gestures

Dean Harris Rubine

December, 1991

CMU–CS–91–202

Submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Computer Science at Carnegie Mellon University.

Thesis Committee:

Roger B. Dannenberg, Advisor
Dario Giuse
Brad Myers
William A. S. Buxton, University of Toronto

Abstract

Gesture-based interfaces, in which the user specifies commands by simple freehand drawings, offer an alternative to traditional keyboard, menu, and direct manipulation interfaces. The ability to specify objects, an operation, and additional parameters with a single intuitive gesture makes gesture-based systems appealing to both novice and experienced users.

Unfortunately, the difficulty in building gesture-based systems has prevented such systems from being adequately explored. This dissertation presents work that attempts to alleviate two of the major difficulties: the construction of gesture classifiers and the integration of gestures into direct-manipulation interfaces. Three example gesture-based applications were built to demonstrate this work.

Gesture-based systems require classifiers to distinguish between the possible gestures a user may enter. In the past, classifiers have often been hand-coded for each new application, making them difficult to build, change, and maintain. This dissertation applies elementary statistical pattern recognition techniques to produce gesture classifiers that are trained by example, greatly simplifying their creation and maintenance. Both single-path gestures (drawn with a mouse or stylus) and multiple-path gestures (consisting of the simultaneous paths of multiple fingers) may be classified. On a 1 MIPS workstation, a 30-class single-path recognizer takes 175 milliseconds to train (once the examples have been entered), and classification takes 9 milliseconds, typically achieving 97% accuracy. A method for classifying a gesture as soon as it is unambiguous is also presented.

This dissertation also describes GRANDMA, a toolkit for building gesture-based applications based on Smalltalk's Model/View/Controller paradigm. Using GRANDMA, one associates sets of gesture classes with individual views or entire view classes. A gesture class can be specified at runtime by entering a few examples of the class, typically 15. The semantics of a gesture class can be specified at runtime via a simple programming interface. Besides allowing for easy experimentation with gesture-based interfaces, GRANDMA sports a novel input architecture, capable of supporting multiple input devices and multi-threaded dialogues. The notion of virtual tools and semantic feedback are shown to arise naturally from GRANDMA's approach.

Acknowledgments

First and foremost, I wish to express my enormous gratitude to my advisor, Roger Dannenberg. Roger was always there when I needed him, never failing to come up with a fresh idea. In retrospect, I should have availed myself more than I did. In his own work, Roger always addresses fundamental problems, and his solutions are always simple and elegant. I try to follow Roger's example in my own work, usually falling far short. Roger, thank you for your insight and your example. Sorry for taking so long.

I was incredibly lucky that Brad Myers showed up at CMU while I was working on this research. His seminar on user interface software gave me the knowledge and breadth I needed to approach the problem of software architectures for gesture-based systems. Furthermore, his extensive comments on drafts of this document improved it immensely. Much of the merit in this work is due to him. Thank you, Brad. I am also grateful to Bill Buxton and Dario Giuse, both of whom provided valuable criticism and excellent suggestions during the course of this work.

It was Paul McAvinney's influence that led me to my thesis topic; had I never met him, mine would have been a dissertation on compiler technology. Paul is an inexhaustible source of ideas, and this thesis is really the *second* idea of Paul's that I've spent multiple years pursuing. Exploring Paul's ideas could easily be the life's work of hundreds of researchers. Thanks, Paul, you madman you.

My wife Ruth Sample deserves much of the credit for the existence of this dissertation. She supported me immeasurably, fed me and clothed me, made me laugh, motivated me to finish, and lovingly tolerated me the whole time. Honey, I love you. Thanks for everything.

I could not have done it with the love and support of my parents, Shirley and Stanley, my brother Scott, my uncle Donald, and my grandma Bertha. For years they encouraged me to be a doctor, and they were not the least bit dismayed when they found out the kind of doctor I wanted to be. They hardly even balked when "just another year" turned out to be six. Thanks, folks, you're the best. I love you all very much.

My friends Dale Amon, Josh Bloch, Blaine Burks, Paul Crumley, Ken Goldberg, Klaus Gross, Gary Keim, Charlie Krueger, Kenny Nail, Eric Nyberg, Barak Pearlmutter, Todd Rockoff, Tom Neuendorffer, Marie-Helene Serra, Ellen Siegal, Kathy Swedlow, Paul Vranesevic, Peter Velikonja, and Brad White all helped me in innumerable ways, from technical assistance to making life worth living. Peter and Klaus deserve special thanks for all the time and aid they've given me over the years. Also, Mark Maimone and John Howard provided valuable criticism which helped me prepare for my oral examination. I am grateful to you all.

I wish to also thank my dog Dismal, who was present at my feet during much of the design, implementation, and writing efforts, and who concurs on all opinions. Dismal, however, strongly objects to this dissertation's focus on *human* gesture.

I also wish to acknowledge the excellent environment that CMU Computer Science provides; none of this work would have been possible without their support. In particular, I'd like to thank Nico Habermann and the faculty for supporting my work for so long, and my dear friends Sharon Burks, Sylvia Berry, Edith Colmer, and Cathy Copetas.

Contents

List of Figures

List of Tables

to Grandma Bertha

Chapter 1

Introduction

People naturally use hand motions to communicate with other people. This dissertation explores the use of human gestures to communicate with computers.

Random House [122] defines "gesture" as "the movement of the body, head, arms, hands, or face that is expressive of an idea, opinion, emotion, etc." This is a rather general definition, which characterizes well what is generally thought of as gesture. It might eventually be possible through computer vision for machines to interpret gestures, as defined above, in real time. Currently such an approach is well beyond the state of the art in computer science.

Because of this, the term "gesture" usually has a restricted connotation when used in the context of human-computer interaction. There, gesture refers to hand markings, entered with a stylus or mouse, which function to indicate scope and commands [109]. Buxton [14] gives a fine example, reproduced here as figure 1.1. In this dissertation, such gestures are referred to as *single-path* gestures.

Recently, input devices able to track the paths of multiple fingers have come into use. The Sensor Frame [84] and the DataGlove [32, 130] are two examples. The human-computer interaction community has naturally extended the use of the term "gesture" to refer to hand motions used to indicate commands and scope, entered via such multiple finger input devices. These are referred to here as *multi-path* gestures.

Rather than defining gesture more precisely at this point, the following section describes an

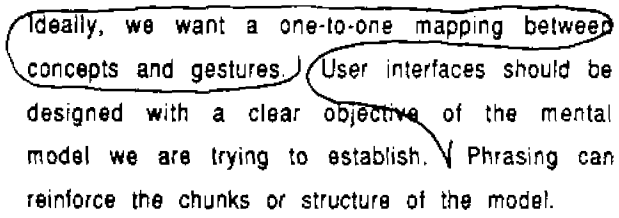

Figure 1.1: Proofreader's Gesture (from Buxton [15])

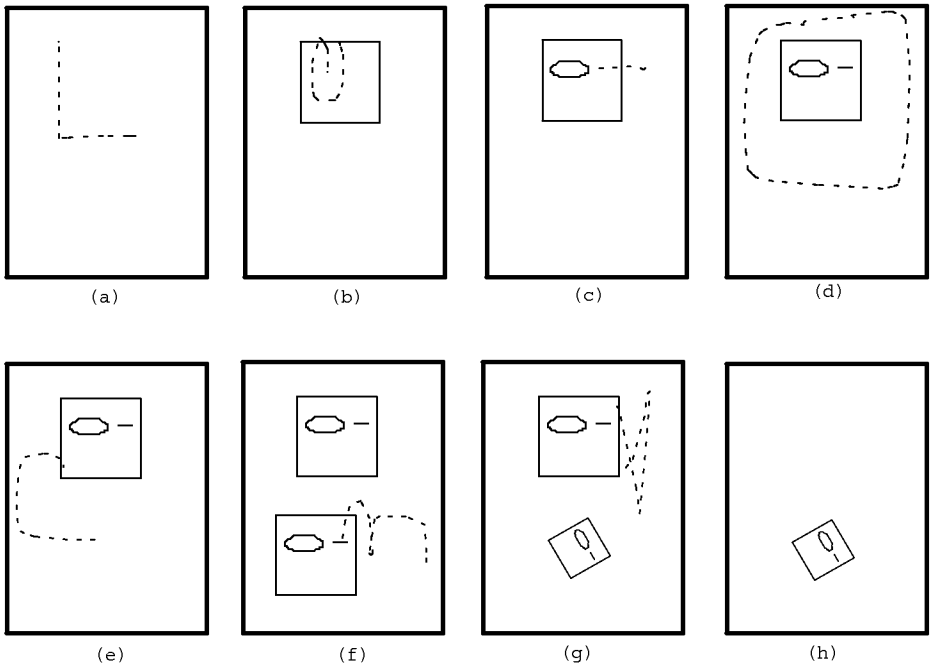

Figure 1.2: GDP, a gesture-based drawing program

example application with a gestural interface. A more technical definition of gesture will be presented in section 1.6.

1.1 An Example Gesture-based Application

GRANDMA is a toolkit used to create gesture-based systems. It was built by the author and is described in detail in the pages that follow. GRANDMA was used to create GDP, a gesture-based drawing editor loosely based on DP [42]. GDP provides for the creation and manipulation of lines, rectangles, ellipses, and text. In this section, GDP is used as an example gesture-based system. GDP's operation is presented first, followed by a description of how GRANDMA was used to create GDP's gestural interface.

1.1.1 GDP from the user's perspective

GDP's operation from a user's point of view will now be described. (GDP's design and implementation is presented in detail in Section 8.1.) The intent is to give the reader a concrete example of a gesture-based system before embarking on a general discussion of such systems. Furthermore, the description of GDP serves to illustrates many of GRANDMA's capabilities. A new interaction technique, which combines gesture and direct manipulation in a single interaction, is also introduced in the description.

Figure 1.2 shows some snapshots of GDP in action. When first started, GDP presents the user with a blank window. Panel (a) shows the rectangle gesture being entered. This gesture is drawn like an "L."[1] The user begins the gesture by positioning the mouse cursor and pressing a mouse button. The user then draws the gesture by moving the mouse.

The gesture is shown on the screen as is being entered. This technique is called *inking* [109], and provides valuable feedback to the user. In the figure, inking is shown with dotted lines so that the gesture may be distinguished from the objects in the drawing. In GDP, the inking is done with solid lines, and disappears as soon as the gesture has been recognized.

The end of the rectangle gesture is indicated in one of two ways. If the user simply releases the mouse button immediately after drawing "L" a rectangle is created, one corner of which is at the start of the gesture (where the button was first pressed), with the opposite corner at the end of the gesture (where the button was released). Another way to end the gesture is to stop moving the mouse for a given amount of time (0.2 seconds works well), while still pressing the mouse button. In this case, a rectangle is created with one corner at the start of the gesture, and the opposite corner at the current mouse location. As long as the button is held, that corner is dragged by the mouse, enabling the size and shape of the rectangle to be determined interactively.

Panel (b) of figure 1.2 shows the rectangle that has been created and the ellipse gesture. This gesture creates an ellipse with its center at the start of the gesture. A point on the ellipse tracks the mouse after the gesture has been recognized; this gives the user interactive control over the size and eccentricity of the ellipse.

Panel (c) shows the created ellipse, and a line gesture. Similar to the rectangle and the ellipse, the start of the gesture determines one endpoint of the newly created line, and the mouse position after the gesture has been recognized determines the other endpoint, allowing the line to be rubberbanded.

Panel (d) shows all three shapes being encircled by a pack gesture. This gesture packs (groups) all the objects which it encloses into a single composite object, which can then be manipulated as a unit. Panel (e) shows a copy gesture being made; the composite object is copied and the copy is dragged by the mouse.

Panel (f) shows the rotate-and-scale gesture. The object is made to rotate around the starting point of the gesture; a point on the object is dragged by the mouse, allowing the user to interactively determine the size and orientation of the object.

Panel (g) shows the delete gesture, essentially an "X" drawn with a single stroke. The object at the gesture start is deleted, as shown in panel (h).

This brief description of GDP illustrates a number of features of gesture-based systems. Perhaps the most striking feature is that each gesture corresponds to a high-level operation. The class of the gesture determines the operation; attributes of the gesture determine its scope (the operands) and any additional parameters. For example, the delete gesture specifies the object to be deleted, the pack gesture specifies the objects to be combined, and the line gesture specifies the endpoints of the line.

[1] It is often convenient to describe single-path gestures as if they were handwritten letters. This is not meant to imply that gesture-based systems can only recognize alphabetic symbols, or even that they usually recognize alphabetic symbols. The many ways in which gesture-based systems are distinct from handwriting-recognition systems will be enumerated in section 1.8.

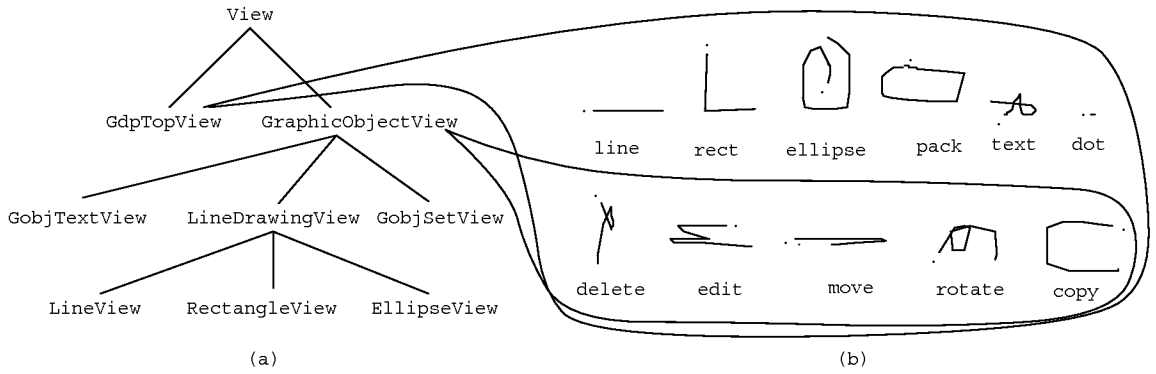

Figure 1.3: GDP's `View` class hierarchy and associated gestures
A period indicates the first point of each gesture.

It is possible to control more than positional parameters with gestural attributes. For example, one version of GDP uses the length (in pixels) of the line gesture to control the thickness of the new line.

Note how gesturing and direct manipulation are combined in a new two-phase interaction technique. The first phase, the collection of the gesture, ends when the user stops moving the mouse while holding the button. At that time, the gesture is recognized and a number of parameters to the application command are determined. After recognition, a manipulation phase is entered during which the user can control additional parameters interactively.

In addition to its gestural interface, GDP provides a more traditional click-and-drag interface. This is mainly used to compare the two styles of interface, and is further discussed in Section 8.1. The gestural interface is grafted on top of the click-and-drag interface, as will be explained next.

1.1.2 Using GRANDMA to Design GDP's Gestures

In the current work, the *gesture designer* creates a gestural interface to an application out of an existing click-and-drag interface to the application. Both the click-and-drag interface and the application are built using the object-oriented toolkit GRANDMA. The gesture designer only modifies the way input is handled, leaving the output mechanisms untouched.

A system built using GRANDMA utilizes the object-oriented programming paradigm to represent windows and the graphics objects displayed in windows. For example, figure 1.3a shows GDP's `View` class hierarchy.[2] This hierarchy shows the relationship of the classes concerned with output. The task of the gesture designer is to determine which of these classes are to have associated gestures, and for each such view class, to design a set of gestures that intuitively expresses the allowable operations on the view. Figure 1.2b shows the sets of gestures associated with GDP's `GraphicObjectView` and `GdpTopView` classes. The `GraphicObjectView` collectively

[2]For expositional purposes, the hierarchy shown is a simplified version of the actual hierarchy. Some of the details that follow have also been simplified. Section 8.1 tells the truth in gory detail.

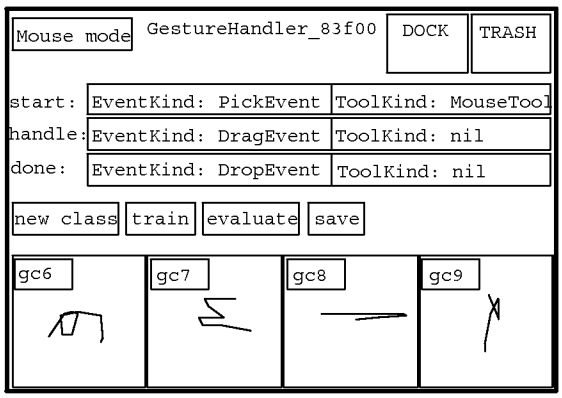

Figure 1.4: Manipulating gesture handlers at runtime

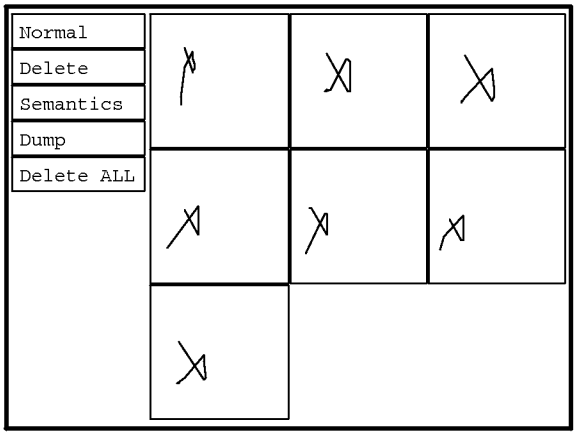

Figure 1.5: Adding examples of the delete gesture

refers to the line, rectangle, and ellipse shapes, while `GdpTopView` represents the window in which GDP runs.

GRANDMA is a Model/View/Controller-like system [70]. In GRANDMA, an input event handler (a "controller" in MVC terms) may be associated with a view class, and thus shared between all instances of the class (including instances of subclasses). This adds flexibility while eliminating a major overhead of Smalltalk MVC, where one or more controller objects are associated with each view object that expects input.

The gesture designer adds gestures to GDP's initial click-and-drag interface at runtime. The first step is to create a new gesture handler and associate it the `GraphicObjectView` class, easily done using GRANDMA. Figure 1.4 shows the gesture handler window after a number of gestures have been created (using the "new class" button), and figure 1.5 shows the window in which examples of the delete gesture have been entered. Fifteen examples of each gesture class typically suffice. If a gesture is to vary in size and/or orientation, the examples should reflect that.

Clicking on the "Semantics" button brings up a window that the designer uses to specify the semantics of each gesture in the handler's set. The window is a structured editing and browsing interface to a simple Objective-C [28] interpreter, and the designer enters three expressions: `recog`, evaluated when the gesture is first recognized; `manip`, evaluated on subsequent mouse points; and `done`, evaluated when the mouse button is released. In this case, the delete semantics simply change the mouse cursor to a delete cursor, providing feedback to the user, and then delete the view at which the gesture was aimed. The expressions entered are[3]

[3]Objective C syntax is used throughout. `[view delete]` sends the `delete` message to the object referred to by the variable `view`. `[handler mousetool:DeleteCursor]` sends the `mousetool:` message to the object referred to by the variable `handler` passing the value of the variable `DeleteCursor` as an argument. See Section 6.3 for more information on Objective C notation.

```
recog = [_Seq :[handler mousetool:DeleteCursor]
              :[view delete]];
manip = nil;
done = nil;
```

The designer may now immediately try out the delete gesture, as in figure 1.2g.

The designer repeats the process to create a gesture handler for the set of gestures associated with class GdpTopView, the view that refers to the window in which GDP runs. This handler deals with the gestures that create graphic objects, the pack gesture (which creates a set out of the enclosed graphic objects), the dot gesture (which repeats the last command), and the gestures also handled by GraphicObjectView's gesture handler (which when made at a GdpTopView change the cursor without operating directly on a graphic object).

The attributes of the gesture are directly available for use in the gesture semantics. For example, the semantics of the line gesture are:

```
recog = [Seq :[handler mousetool:LineCursor]
             :[[view createLine]
                     setEndpoint:0 x:<startX> y:<startY>]];
manip = [recog setEndpoint:1 x:<currentX> y:<currentY>];
done = nil;
```

The semantic expressions execute in a rich environment in which, for example, view is bound to the view at which the gesture was directed (in this case a GdpTopView) and handler is bound to the current gesture handler. Note that Seq executes its arguments sequentially, returning the value of the last, in this case the newly created line. This is bound to recog for later use in the manip expression.

The example shows how the gesture attributes, shown in angle brackets, are useful in the semantic expressions. The attributes <startX> and <startY>, the coordinates of the first point in the gesture, are used to determine one endpoint of the line, while <currentX> and <currentY>, the mouse coordinates, determine the other endpoint.

Many other gesture attributes are useful in semantics. The line semantics could be augmented to control the thickness of the line from the maximum speed or total path length of the gesture. The rectangle semantics could use the initial angle of the rectangle gesture to determine the orientation of the rectangle. The attribute <enclosed> is especially noteworthy: it contains a list of views enclosed by the gesture and is used, for example, by the pack gesture (figure 1.2d). When convenient, the semantics can simulate input to the click-and-drag interface, rather than communicating directly with application objects or their views, as shown above.

When the first point of a gesture is over more than one gesture-handling view, the union of the set of gestures recognized by each handler is used, with priority given to the foremost views. For example, any gesture made at a GDP GraphicObjectView is necessarily made over the GdpTopView. A delete gesture made at a graphic object would be handled by the GraphicObjectView while a line gesture at the same place would be handled by the GdpTopView. Set union also occurs when gestures are (conceptually) inherited via the view class hierarchy. For example, the gesture designer might create a new gesture handler for the GobjSetView class containing an unpack gesture. The set of gestures recognized by

`GobjSetViews` would then consist of the unpack gesture as well as the five gestures handled by `GraphicObjectView`.

1.2 Glossary

This section defines and clarifies some terms that will be used throughout the dissertation. It may safely be skipped and referred back to as needed. Some of the terms (click, drag) have their common usage in the human-computer interaction community, while others (pick, move, drop) are given technical definitions solely for use here.

class In this dissertation, "class" is used in two ways. "Gesture class" refers to a set of gestures all of which are intended to be treated the same, for example, the class of delete gestures. (In this dissertation, the names of gesture classes will be shown in sans serif typeface.) The job of a gesture recognizer is, given an example gesture, to determine its class (see also "gesture"). "Class" is also used in the object-oriented sense, referring to the type (loosely speaking) of a software object. It should be clear from context which of these meanings is intended.

click A click consists of positioning the mouse cursor and then pressing and releasing a mouse button, with no intervening mouse motion. In the Macintosh, a click is generally used to select an object on the screen.

click-and-drag A click-and-drag interface is a direct-manipulation interface in which objects on the screen are operated upon using mouse clicks, drags, and sometimes double-clicks.

direct manipulation A direct-manipulation interface is one in which the user manipulates a graphic representation of the underlying data by pointing at and/or moving them with an appropriate device, such as a mouse with buttons.

double-click A double-click is two clicks in rapid succession.

drag A drag consists of locating the mouse cursor and pressing the mouse button, moving the mouse cursor while holding the mouse button, and then releasing the mouse button. Drag interactions are used in click-and-drag interfaces to, for example, move objects around on the screen.

drop The final part of a drag (or click) interaction in which the mouse button is released.

eager recognition A kind of gesture recognition in which gestures are often recognized without the end of the gesture having to be explicitly signaled. Ideally, an eager recognizer will recognize a gesture as soon as enough of it has been seen to determine its class unambiguously.

gesture Essentially a freehand drawing used to indicate a command and all its parameters. Depending on context, the term maybe used to refer to an example gesture or a class of gestures, *e.g.* "a delete gesture" means an example gesture belonging to the class of delete gestures. Usually "gesture" refers to the part of the interaction up until the input is recognized as one

of a number of possible gesture classes, but sometimes the entire interaction (which includes a manipulation phase after recognition) is referred to as a gesture.

move The component of drag interaction during which the mouse is moved while a mouse button is held down. It is the presence of a move that distinguishes a click from a drag.

multi-path A multi-path gesture is one made with an input device that allows more than one position to be indicated simultaneously (multiple pointers). One may make multi-path gestures with a Sensor Frame, a multiple-finger touch pad, or a DataGlove, to name a few such devices.

off-line Considering an algorithm to be a sequence of operations, an off-line algorithm is one which examines subsequent operations before producing output for the current operation.

on-line An on-line algorithm is one in which the output of an operation is produced before any subsequent operations are read.

pick The initial part of a drag (or click) interaction consisting of positioning the mouse cursor at the desired location and pressing a mouse button.

press refers to the pressing of a mouse button.

real-time A real-time algorithm is an on-line algorithm in which each operation is processed in time bounded by a constant.

release refers to the releasing of a mouse button.

segment A segment is an approximately linear portion of a stroke. For example, the letter "L" is two segments, one vertical and one horizontal.

single-path A single-path gesture is one drawn by an input device, such as a mouse or stylus, capable of specifying only a single point over time. A single-path gesture may consist of multiple strokes (like the character "X").

single-stroke A single-stroke gesture is a single-path gesture that is one stroke. Thus drawing "L" is a single-stroke gesture, while "X" is not. In this dissertation the only single-path gestures considered are single-stroke gestures.

stroke A stroke is an unbroken curve made by a single movement of a pen, stylus, mouse, or other instrument. Generally, strokes begin and end with explicit user actions (*e.g.* , pen down/pen up, mouse button down/mouse button up).

1.3 Summary of Contributions

This dissertation makes contributions in four areas: new interaction techniques, new algorithms for gesture recognition, a new way of integrating gestures into user interfaces, and a new architecture for input in object-oriented toolkits.

The first new interaction technique is the two-phase combination of single-stroke gesture collection followed by direct manipulation, mentioned previously. In the GDP example discussed above, the boundary between the two phases is an interval of motionlessness. Eager recognition, the second new interaction technique, eliminates this interval by recognizing the single-stroke gesture and entering the manipulation phase as soon as enough of the gesture has been seen to do so unambiguously, making the entire interaction very smooth. A third new interaction technique is the two-phase interaction applied to multi-path gestures: after a multi-path gesture has been recognized, individual paths (*i.e.* fingers, possibly including additional fingers not involved in making the recognized gesture) may be assigned to manipulate independent application parameters simultaneously.

The second contribution is a new trainable, single-stroke recognition algorithm tailored for recognizing gestures. The classification is based on meaningful features, which in addition to being useful for recognition are also suitable for passing to application routines. The particular set of features used has been shown to be suitable for many different gesture sets, and is easily extensible. When restricted to features that can be updated incrementally in constant time per input point, arbitrarily large gestures may be handled. The single-stroke recognition algorithm has been extended to do eager recognition (eager recognizers are automatically generated from example gestures), and also to multi-path gesture recognition.

Third, a new paradigm for creating gestural interfaces is also propounded. As seen in the example, starting from a click-and-drag implementation of an interface, gestures are associated with classes of views (display objects), with the set of gestures recognized at a particular screen location dynamically determined by the set of overlapping views at the location, and by inheritance up the class hierarchy of each such view. The classification and attributes of gestures map directly to application operations and parameters. The creation, deletion, and manipulation of gesture handlers, gesture classes, gesture examples, and gesture semantics all occur at runtime, enabling quick and easy experimentation with gestural interfaces.

Fourth, GRANDMA, as an object-oriented user interface toolkit, makes some contributions to the area of input handling. Event handler objects are associated with particular views or entire view classes. A single event handler may be shared between many different objects, eliminating a major overhead of MVC systems. Multiple event handlers may be associated with a single object, enabling the object to support multiple interaction techniques simultaneously, including the use of multiple input devices. Furthermore, a single mechanism handles both mouse tools (*e.g.* a delete cursor that deletes clicked-upon objects) and virtual tools (*e.g.* a delete icon that is dragged around and dropped upon objects to delete them). Additionally, GRANDMA provides support for semantic feedback, and enables the runtime creation and manipulation of event handlers.

1.4 Motivation for Gestures

At this point, the reader should have a good idea of the scope of the work to be presented in this dissertation. Stepping back, this section begins a general discussion of gestures by examining the motivation for using and studying gesture-based interfaces. Much of the discussion is based on that of Buxton [14].

Computers get faster, bitmapped displays produce ever increasing information rates, speech and

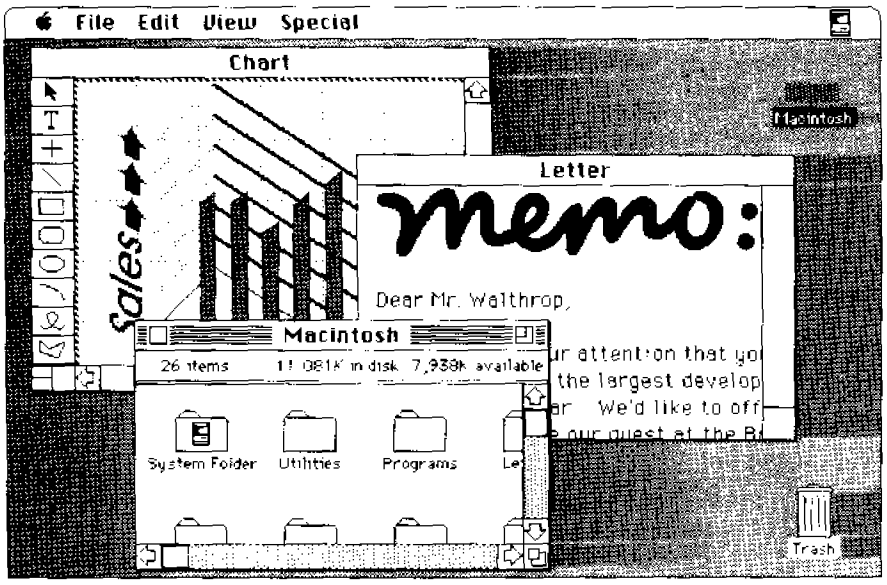

Figure 1.6: Macintosh Finder, MacDraw, and MacWrite (from Apple [2])

music can be generated in real-time, yet input just seems to plod along with little or no improvement. This is regrettable because, in Paul McAvinney's words [84], most of the useful information in the world resides in humans, not computers. Most people who interact with computers spend most of their time entering information [22]. Due to this input bottleneck, the total time to do many tasks would hardly improve even if computers became infinitely fast. Thus, improvements in input technology are a major factor in improving the productivity of computer users in general.

Of course, progress has been made. Input has progressed from batch data entry, to interactive line editors, to two-dimensional screen editors, to mouse-based systems with bitmapped displays. Pointing with a mouse has proved a useful interaction technique in many applications. "Click and drag" interfaces, where the user directly manipulates graphic objects on the screen with a mouse, are often very intuitive to use. Because of this, direct manipulation interfaces have become commonplace, despite being rather difficult to build.

Consider the Macintosh [2], generally regarded as having a good direct-manipulation interface. As shown in figure 1.6, the screen has on it a number of graphic objects, including file icons, folder icons, sliders, buttons, and pull-down menu names. Each one is generally a rectangular region, which may be clicked, sometimes double-clicked, and sometimes dragged. The Macintosh Finder, which may be used to access all Macintosh applications and documents, is almost entirely controlled via these three interaction techniques.[4]

The click and double-click interactions have a single object (or location) as parameter. The drag

[4]Obviously this discussion ignores keyboard entry of text and commands.

interaction has two parameters: an object or location where the mouse button is first pressed, and another object or location at the release point. Having only these three interaction techniques is one reason the Macintosh is simple to operate. There is, however, a cost: both the application and the user must express all operations in terms of these three interaction techniques.

An application that provides more than three operations on any given object (as many do) has several design alternatives. The first, exemplified by the Finder, relies heavily on *selection*. In the Finder, a click interaction selects an object, a double-click opens an object (the meaning of which depends upon the object's type), and a drag moves an object (the meaning of which is also object-type specific). Opening an object by a double-click is a means for invoking the most common operation on the object, *e.g.* opening a MacWrite document starts the MacWrite application on the document. Dragging is used for adjusting sliders (such as those which scroll windows), changing window size or position, moving files between folders, and selecting menu items.

All other operations are done in at least two steps: first the object to be operated upon is selected, and then the desired operation is chosen from a menu. For example, to print an object, one selects it (click) then chooses "Print" from the appropriate menu (drag); to move some text, one selects it (drag), chooses "Cut" (drag), selects an insertion point (click), and chooses "Paste" (drag). The cost of only having three interaction techniques is that some operations are necessarily performed via a sequence of interactions. The user must adjust her mental model so that she thinks in terms of the component operations.

An alternative to the selection-based click-and-drag approach is one based on modes. Consider MacDraw [2], a drawing program. The user is presented with a palette offering choices such as line, text, rectangles, circles, and so on. Clicking on the "line" icon puts the program into line-drawing mode. The next drag operation in the drawing window cause lines to be drawn. In MacDraw, after the drag operation the program reverts back to selection mode. DP, the program upon which GDP is based, is similar except that it remains in its current mode until it is explicitly changed. Mistakes occur when the user believes he is in one mode but is actually in another. The claim that direct manipulation interfaces derive their power from being modeless is not really true. Good direct manipulation interfaces simply make the modes very visible, which helps to alleviate the problems of modal interfaces.

By mandating the sole use of click, double-click, and drag interactions, the Macintosh interface paradigm necessarily causes conceptually primitive tasks to be divided into a sequence of primitive interactions. The intent of gestural interfaces is to avoid this division, by packing the basic interaction with all the parameters necessary to complete the entire transaction. Ideally, each primitive task in the user's model of the application is executed with a single gesture. Such interfaces would have less modeness than the current so-called modeless interfaces.

The Macintosh discussion in the previous section is somewhat oversimplified. Many applications allow variations on the basic interaction techniques; for example "shift-click" (holding the shift key while clicking the mouse) adds an object to the current set of selected objects. Other computer systems allow different mouse buttons to indicated different operations. There is a tradeoff between having a small number and a large number of (consistently applied) interaction techniques. The former results in a system whose primitive operations are easy to learn, perform, and recall, but a single natural chunk may be divided into a sequence of operations. In the latter case, the primitive

operations are harder to learn (because there are more of them), but each one can potentially implement an entire natural chunk.

The motivation for gestural interfaces may also apply to interfaces which combine modalities (*e.g.* speech and pointing). As with gestures, one potential benefit of multi-modal interfaces is that different modalities allow many parameters to be specified simultaneously, thus eliminating the need for modes. The "Put-That-There" system is one example [12].

1.5 Primitive Interactions

The discussion thus far has been vague as to what exactly may be considered a "primitive" interaction technique. The Macintosh has three: click, double-click, and drag. It is interesting to ask what criteria can be used for judging the "primitiveness" of proposed interaction techniques.

Buxton [14] suggests physical tension as a criterion. The user, starting from a relaxed state, begins a primitive interaction by tensing some muscles. The interaction is over when the user again relaxes those muscles. Buxton cites evidence that "such periods of tension are accompanied by a heightened state of attentiveness and improved performance." The three Macintosh interaction techniques all satisfy this concept of primitive interaction. (Presumably the user remains tense during a double-click because the time between clicks is short.)

Buxton likens the primitive interaction to a musical phrase. Each consists of a period of tension followed by a return to a state where a new phrase may be introduced. In human-computer interaction, such a phrase is used to accomplish a chunk of a task. The goal is to make each of these chunks a primitive task in the user's model of the application domain. This is what a gesture-based interface attempts to do.

1.6 The Anatomy of a Gesture

In this section a technical definition of gesture is developed, and the syntactic and semantic properties of gestures are then discussed. The dictionary definition of gesture, "expressive motion," has already been seen. How can the notion of gesture in a form suitable for sensing and processing by machine be captured?

1.6.1 Gestural motion

The motion aspect of gesture is formalized as follows: a gesture consists of the paths of multiple points over time. The points in question are (conceptually) affixed to the parts of the body which perform the gesture. For hand gestures, the points tracked might include the fingertips, knuckles, palm, and wrist of each hand. Over the course of a gesture, each point traces a path in space. Assuming enough points (attached to the body in appropriate places), these paths contain the essence of the gestural motion. A computer with appropriate hardware can rapidly sample positions along the paths, thus conveniently capturing the gesture.

The idea of gesture as the motion of multiple points over time is a generalization of pointing. Pointing may be considered the simplest gesture: it specifies a single position at an instance of

time. This is generalized to allow for the movement of the point over time, *i.e.* a path. A further generalization admits multiple paths, *i.e.* the movement of multiple points over time.[5]

Current gesture-sensing hardware limits both the number of points which may be tracked simultaneously and the dimensionality of the space in which the points travel. Gestures limited to the motion of a single point are referred to here as *single-path* gestures. Most previous gestural research has focused upon gestures made with a stylus and tablet, mouse, or single-finger touch pad. The gestures which may be made with such devices are two-dimensional, single-path gestures.

An additional feature of existing hardware is that the points are not tracked at all times. For example, a touch pad can only determine finger position when the finger is touching the pad. Thus, the path of the point will have a beginning (when the finger first makes contact) and an end (when the finger is lifted). This apparent limitation of certain gesture-sensing hardware may be used to delineate the start and possibly the end of each gesture, a necessary function in gesture-based systems. Mouse buttons may be used to similar effect.[6]

In all the work reported here, a gesture (including the manipulation phase after recognition) is always a primitive interaction. A gesture begins with the user going from a relaxed state to one of muscular tension, and ends when the user again relaxes. It is further assumed that the tension or relaxation of the user is directly indicated by some aspect of the sensing hardware. For mouse gestures, the user is considered in a state of tension if and only if a mouse button is pressed. Thus, in the current work a double-click is not considered a gesture. This is certainly a limitation, but one that could be removed, for example by having a minimum time that the button needs to be released before the user is considered to have relaxed. This added complication has not been explored here.

The space in which the points of the gesture move is typically physical space, and thus a path is represented by a set of points (x, y, z, t) consisting of three spatial Cartesian coordinates and time. However, there are devices which measure non-spatial gestural parameters; hence, gestures consisting of paths through a space where at least some of the coordinates are not lengths are possible. For example, some touch pads can sense force, and for this hardware a gesture path might consist of a set of points (x, y, f, t), f being the force measurement at time t.

The formalization of gesture as multiple paths is just one among many possible representations. It is a good representation because it coincides nicely with most of the existing gesture-sensing hardware, and it is a useful form for efficient processing. The multiple-snapshot representation, in which each snapshot gives the position of multiple points at a single instant, is another possibility, and in some sense may be considered the dual of multiple paths. Such a representation might be more suitable for gestural data derived from hardware (such as video cameras) which are not considered in this dissertation.

1.6.2 Gestural meaning

In addition to the physical aspect of a gesture, there is the content or meaning of the gesture to consider. Generally speaking, a gesture contains two kinds of information: *categorical* and

[5] A configuration of multiple points at a single instance of time may be termed *posture*. Posture recognition is commonly used with the DataGlove.

[6] Buxton [17] presents a model of the discrete signaling capabilities of various pointing devices and a list of the signaling requirements for common interaction techniques.

parametric. Consider the different motions between people meaning "come here" (beckoning gestures), "stop" (prohibiting gestures), and "keep going" (encouragement gestures). These are different categories, or *classes*, of gestures. Within each class, a gesture also can indicate parametric data. For example, a parameter of the beckoning gesture is the urgency of the request: "hurry up" or "take your time." In general, the category of the gesture must be determined before the parameters can be interpreted.

Parametric information itself comes in two forms. The first is the kind of information that can be culled at the time the gesture is classified. For example, the position, size and orientation of the gesture fall into this category. The second kind of parametric gestural information is manipulation information. After the gesture is recognized, the user can use this kind of parametric information to continuously communicate information. An example would be the directional information communicated by the gestures of a person helping a driver to back up a truck. An example from GDP (see Section 1.1) is the rubberbanding of a line after it is created, where the user continuously manipulates one endpoint.

The term "gesture" as used here does not exactly correspond to what is normally thought of as gesture. Many gestures cannot currently be processed by machine due to limitations of existing gesture-sensing hardware. Also, consider what might be referred to as "direct-manipulation gestures." A person turning a knob would not normally be considered to be gesturing. However, a similar motion used to manipulate the graphic image of a knob drawn on a computer display is considered to be a gesture. Actually, the difference here is more illusory than real: a person might make the knob-turning gesture at another person, in effect asking the latter to turn the knob. The intent here is simply to point out the very broad class of motions considered herein to be gesture.

While the notion of gesture developed here is very general (multiple paths), in practice, machine gestures have hitherto almost always been limited to finger and/or hand motions. Furthermore, the paths have largely been restricted to two dimensions. The concentration on two-dimensional hand gesturing is a result of the available gesture-sensing hardware. Of course, such hardware was built because it was believed that hand and fingers are capable of accurate and diverse gesturing, yet more amenable to practical detection than facial or other body motions. With the appearance of new input devices, three (or more) dimensional gesturing, as well as the use of parts of the body other than the hand, are becoming possible. Nonetheless, this dissertation concentrates largely on two-dimensional hand gestures, assuming that by viewing gesture simply as multiple paths, the work described may be applied to non-hand gestures, or generalized to apply to gestures in three or more dimensions.

1.7 Gesture-based systems

A gesture-based interface, as the term is used here, is one in which the user specifies commands by gesturing. Typically, gesturing consists of drawing or other freehand motions. Excluded from the class of gesture-based interfaces are those in which input is done solely via keyboard, menu, or click-and-drag interactions. In other words, while pointing is in some sense the most basic gesture, those interfaces in which pointing is the only form of gesture are not considered here to be gesture-based interfaces. A gesture-based system is a program (or set of programs) with which the user interacts via a gesture-based interface.

In all but the simplest gesture-based systems, the user may enter a gesture belonging to one of several different gesture categories or classes; the different classes refer to different commands to the system. An important component of gesture-based systems is the gesture *recognizer* or *classifier*, the module whose job is to classify the user's gesture as the first step toward inferring its meaning. This dissertation addresses the implementation of gesture recognizers, and their incorporation into gesture-based systems.

1.7.1 The four states of interaction

User interaction with the gesture-based systems considered in this dissertation may be described using the following four state model. The states–WAIT, COLLECT, MANIPULATE, EXECUTE–usually occur in sequence for each interaction.

- The WAIT state is the quiescent state of the system. The system is waiting for the user to initiate a gesture.
- The COLLECT state is entered when the user begins to gesture. While in this state, the system collects gestural data from the input hardware in anticipation of classifying the gesture. For most gesturing hardware, an explicit start action (such as pressing a mouse button) indicates the beginning of each gesture, and thus causes the system to enter this state.
- The MANIPULATE state is entered once the gesture is classified. This occurs in one of three ways:

  1. The end of the gesture is indicated explicitly, *e.g.* by releasing the mouse button;
  2. the end of the gesture is indicated implicitly, *e.g.* by a timeout which indicates the user has not moved the mouse for, say, 200 milliseconds; or
  3. the system initiates classification because it believes it has now seen enough of the gesture to classify it unambiguously (eager recognition).

  When the MANIPULATE state is entered, the system should provide feedback to the user as to the classification of the gesture and update any screen objects accordingly. While in this state, the user can further manipulate the screen objects with his motions.
- The EXECUTE state is entered when the user has completed his role in the interaction, and has indicated such (*e.g.* by releasing the mouse button). At this point the system performs any final actions as implied by the user's gesture. Ideally, this state lasts only a very short time, after which the display is updated to reflect the current state of the system, and the system reverts back to the WAIT state.

This model is sufficient to describe most current systems which use pointing devices. (For simplicity, keyboard input is ignored.) Depending on the system, the COLLECT or MANIPULATE state may be omitted from the cycle. A handwriting interface will usually omit the MANIPULATE state, classifying the collected characters and executing the resulting command. Conversely, a

direct-manipulation system will omit the COLLECT state (and the attendant classification). The GDP example described above has both COLLECT and MANIPULATE phases. The result is the new two-phase interaction technique mentioned earlier.

1.8 A Comparison with Handwriting Systems

In this section, the frequently asked question, "how do gesture-based systems differ from handwriting systems?" is addressed.

Handwriting systems may broadly be grouped into two classes: on-line and off-line. On-line handwriting recognition simply means characters are recognized as they are drawn. Usually, the characters are drawn with a stylus on a tablet, thus the recognition process takes as input a list of successive points or line segments. The problem is thus considerably different than off-line handwriting recognition, in which the characters are first drawn on paper, and then optically scanned and represented as two-dimensional rasters. Suen, Berthod, and Mori review the literature of both on-line and off-line handwriting systems [125], while Tappert, Suen, and Wakaha [129] give a recent review of on-line handwriting systems. The intention here is to contrast gesture-based systems with on-line handwriting recognition systems, as these are the most closely related.

Gesture-based systems have much in common with systems which employ on-line handwriting recognition for input. Both use freehand drawing as the primary means of user input, and both depend on recognizers to interpret that input. However, there are some important differences between the two classes of systems, differences that illustrate the merits of gesture-based systems:

- Gestures may be motions in two, three, or more dimensions, whereas handwriting systems are necessarily two-dimensional. Similarly, single-path and multiple-path gestures are both possible, whereas handwriting is always a single path.

- The alphabet used in a handwriting system is generally well-known and fixed, and users will generally have lifelong experience writing that alphabet. With gestures, it is less likely that users will have preconceptions or extensive experience.

- In addition to the command itself, a single gesture can specify parameters to the command. The proofreader's gesture (figure 1.1) discussed above, is an excellent example. Another example, also due to Buxton [21], and used in GSCORE (Section 8.2), is a musical score editor, in which a single stroke indicates the location, pitch, and duration of a note to be added to the score.

- As stated, a command and all its parameters may be specified with a single gesture. The physical relaxation of the user when she completes a gesture reinforces the conceptual completion of a command [14].

- Gestures of a given class may vary in both size and orientation. Typical handwriting recognizers expect the characters to be of a particular size and oriented in the usual manner (though successful systems will necessarily be able to cope with at least small variations in size and orientation). However, some gesture commands may use the size and orientation to specify

parameters; gesture recognizers must be able to recognize such gestures in whatever size and orientation they occur. Kim [67] discusses augmenting a handwriting recognition system so as to allow it to recognize some gestures independently of their size and orientation. Chapter 3 discusses the approach taken here toward the same end.

- Gestures can have a dynamic component. Handwriting systems usually view the input character as a static picture. In a gesture-based system, the same stroke may have different meanings if drawn left-to-right, right-to-left, quickly, or slowly. Gesture recognizers may use such directional and temporal information in the recognition process.

In summary, gestures may potentially deal in dimensions other than the two commonly used in handwriting, be drawn from unusual alphabets, specify entire commands, vary in size and orientation, and have a dynamic component. Thus, while ideas from on-line handwriting recognition algorithms may be used for gesture recognition, handwriting recognizers generally rely on assumptions that make them inadequate for gesture recognition. The ideal gesture recognition algorithm should be adaptable to new gestures, dimensions, additional features, and variations in size and orientation, and should produce parametric information in addition to a classification. Unfortunately, the price for this generality is the likelyhood that a gesture recognizer, when used for handwriting recognition, will be less accurate than a recognizer built and tuned specifically for handwriting recognition.

1.9 Motivation for this Research

In spite of the potential advantages of gesture-based systems, only a handful have been built. Examples include Button Box [86], editing using proofreader's symbols [25], the Char-rec note-input tool [21], and a spreadsheet application built at IBM [109]. These and other gesture-based systems are discussed in section 2.2. Gesture recognition in most existing systems has been done by writing code to recognize the particular set of gestures used by the system. This code is usually complicated, making the systems (and the set of gestures accepted) difficult to create, maintain, and modify. These difficulties are the reasons more gesture-based systems have not been built.

One goal of the present work is to eliminate hand-coding as the way to create gesture recognizers. Instead, gesture classes are specified by giving examples of gestures in the class. From these examples, recognizers are automatically constructed. If a particular gesture class is to be recognized in any size or orientation, its examples of the class should reflect that. Similarly, by making all of the examples of a given class the same size or orientation, the system learns that gestures in this class must appear in the same size or orientation as the examples. The first half of this dissertation is concerned with the automatic construction of gesture recognizers.

Even given gesture recognition, it is still difficult to build direct-manipulation systems which incorporate gestures. This is the motivation for the second half of this dissertation, which describes GRANDMA–Gesture Recognizers Automated in a Novel Direct Manipulation Architecture.

1.10 Criteria for Gesture-based Systems

The goal of this research was to produce tools which aid in the construction of gesture-based systems. The efficacy of the tools may be judged by how well the tools and resulting gesture-based systems satisfy the following criteria.

1.10.1 Meaningful gestures must be specifiable

A meaningful gesture may be rather complex, involving simultaneous motions of a number of points. These complex gestures must be easily specifiable. Two methods of specification are possible: specification by example, and specification by description. In the former, each application has a training session in which examples of the different gestures are submitted to the system. The result of the training is a representation for all gestures that the system must recognize, and this representation is used to drive the actual gesture recognizer that will run as part of the application. In the latter method of specification, a description of each gesture is written in a gesture description language, which is a formal language in which the "syntax" of each gesture is specified. For example, a set of gestures may be specified by a context-free grammar, in which the terminals represent primitive motions (e.g. "straight line segment") and gestures are non-terminals composed of terminals and other non-terminals.

All else being equal, the author considers specification by example to be superior to specification by description. In order to specify gestures by description, it will be necessary for the specifier to learn a description language. Conversely, in order to specify by example, the specifier need only be able to gesture. Given a system in which gestures are specified by example, the possibility arises for end users to train the system directly, either to replace the existing gestures with ones more to their liking, or to have the system improve its recognition accuracy by adapting to the particular idiosyncrasies of a given user's gestures.

One potential drawback of specification by example is the difficulty in specifying the allowable variations between gestures of a given class. In a description language, it can be made straightforward to declare that gestures of a given class may be of any size or of any orientation. The same information might be conveyed to a specify-by-example system by having multiple examples of a single class vary in size or orientation. The system would then have to *infer* that the size or orientation of a given gesture class was irrelevant to the classification of the gesture. Also, training classifiers may take longer, and recognition may be less accurate, when using examples as specifications, though this is by no means necessarily so. Similar issues arise in demonstrational interfaces [97].

1.10.2 Accurate recognition

An important characterization of a gesture recognition system will be the frequency with which gestures fail to be recognized or are recognized incorrectly. Obviously it is desirable that these numbers be made as small as possible. Questions pertaining to the amount of inaccuracy acceptable to people are difficult to answer objectively. There will likely be tradeoffs between the complexity of gestures, the number of different gestures to be disambiguated, the time needed for recognition, and the accuracy of recognition.

In speech recognition there is the problem that the accuracy of recognition decreases as the user population grows. However the analogous problem in gesture recognition is not as easy to gauge. Different people speak the same words differently due to inevitable differences in anatomy and upbringing. The way a person says a word is largely determined before she encounters a speech recognition system. By contrast, most people have few preconceptions of the way to gesture at a machine. People will most likely be able to adapt themselves to gesturing in ways the machine understands. The recognition system may similarly adapt to each user's gestures. It would be interesting, though outside the scope of this dissertation, to study the fraction of incorrectly recognized gestures as a function of a person's experience with the system.

1.10.3 Evaluation of accuracy

It should be possible for a gesture-based system to monitor its own performance with respect to accuracy of recognition. This is not necessarily easy, since in general it is impossible to know which gesture the user had intended to make. A good gesture-based system should incorporate some method by which the user can easily inform the system when a gesture has been classified incorrectly. Ideally, this method should be integrated with the undo or abort features of the systems. (Lerner [78] gives an alternative in which subsequent user actions are monitored to determine when the user is satisfied with the results of system heuristics.)

1.10.4 Efficient recognition

The goal of this work is to enable the construction of applications that use gestures as input, the idea being that gesture input will enhance human/computer interaction. Speed of recognition is very important–a slow system would be frustrating to use and hinder rather than enhance interaction

Speed is a very important factor in the success or failure of user interfaces in general. Baecker and Buxton [5] state that one of the chief determinants of user satisfaction with interactive computer systems is response time. Poor performance in a direct-manipulation system is particularly bad, as any noticeable delay destroys the feeling of directness. Rapid recognition is essential to the success of gesture as a medium for human-computer interaction, even if achieving it means sacrificing certain features or, perhaps, a limited amount of recognition accuracy.

1.10.5 On-line/real-time recognition

When possible, the recognition system should attempt to match partial inputs with possible gestures. It may also be desirable to inform the user as soon as possible when the input does not seem to match any possible gesture. An on-line/real-time matching algorithm has these desirable properties. The gesture recognition algorithms discussed in Chapters 3, 4, and 5 all do a small, bounded amount of work given each new input point, and are thus all on-line/real-time algorithms.

1.10.6 General quantitative application interface

An application must specify what happens when a gesture is recognized. This will often take the form of a callback to an application-specific routine. There is an opportunity here to relay the

parametric data contained in the gesture to the application. This includes the parametric data which can be derived when the gesture is first recognized, as well as any manipulation data which follows.

1.10.7 Immediate feedback

In certain applications, it is desirable that the application be informed immediately once a gesture is recognized but before it is completed. An example is the turning of a knob: once the system recognizes that the user is gesturing to turn a knob it can monitor the exact details of a gesture, relaying quantitative data to the application. The application can respond by immediately and continuously varying the parameter which the knob controls (for example the volume of a musical instrument).

1.10.8 Context restrictions

A gesture sensing system should be able, within a single application, to sense different sets of gestures in different contexts. An example of a context is a particular area of the display screen. Different areas could respond to different sets of gestures. The set of gestures to which the application responds should also be variable over time–the application program entering a new mode could potentially cause a different set of gestures to be sensed.

The idea of contexts is closely related to the idea of using gestures to manipulate graphic objects. Associated with each picture of an object on the screen will be an area of the screen within which gestures refer to the object. A good gesture recognition system should allow the application program to make this association explicit.

1.10.9 Efficient training

An ideal system would allow the user to experiment with different gesture classes, and also adapt to the user's gestures to improve recognition accuracy. It would be desirable if the system responded immediately to any changes in the gesture specifications; a system that took several hours to retrain itself would not be a good platform for experimentation.

1.10.10 Good handling of misclassifications

Misclassifications of gestures are a fact of life in gesture-based systems. A typical system might have a recognition rate of 95% or 99%. This means one out of twenty or one out of one hundred gestures will be misunderstood. A gesture-based system should be prepared to deal with the possibility of misclassification, typically by providing easy access to abort and undo facilities.

1.10.11 Device independence

Certain assumptions about the form of the input data are necessary if gesture systems are to be built. As previously stated, the assumption made here is that the input device will supply position as a function of time for each input "path" (or supply data from which it is convenient to calculate such positions). (A path may be thought of as a continuous curve drawn by a single finger.) This form of

data is supplied by the Sensor Frame, and (at least for the single finger case) a mouse and a clock can be made to supply similar data. The recognition systems should do their recognition based on the position versus time data; in this way other input devices may also benefit from this research.

1.10.12 Device utilization

Each particular brand of input hardware used for gesture sensing will have characteristics that other brands of hardware will not have. It would be unfortunate not to take advantage of all the special features of the hardware. For example, the Sensor Frame can compute finger angle and finger velocity.[7] While for device independence it may be desirable that the gesture matching not depend on the value of these inputs, there should be some facility for passing these parameters to the application specific code, if the application so desires. Baecker [4] states the case strongly: "Although portability is facilitated by device-independence, interactivity and usability are enhanced by *device dependence*."

1.11 Outline

The following chapter describes previous related work in gesture-based systems. This is divided into four sections: Section 2.1 discusses various hardware devices suitable for gestural input. Section 2.2 discusses existing gesture-based systems. Section 2.3 reviews the various approaches to pattern recognition in order to determine their potential for gesture recognition. Section 2.4 examines existing software systems and toolkits that are used to build direct-manipulation interfaces. Ideas from such systems will be generalized in order to incorporate gesture recognition into such systems.

Everything after Chapter 2 focuses on various aspects of the gesture-based interface creation tool built by the author. Such a tool makes it easy to 1) specify and create classifiers, and 2) associate gestures classes and their meanings with graphic objects. The former goal is addressed in Chapters 3, 4, and 5, the latter in 6 and 7.

The discussion of the implementation of gesture recognition begins in Chapter 3. Here the problem of classifying single-path, two-dimensional gestures is tackled. This chapter assumes that the start and end of the gesture are known, and uses statistical pattern recognition to derive efficient gesture classifiers. The training of such classifiers from example gestures is also covered.

Chapter 3 shows how to classify single-path gestures; Chapter 4 shows *when*. This chapter addresses the problem of recognizing gestures while they are being made, without any explicit indication of the end of the gesture. The approach taken is to define and construct another classifier. This classifier is intended solely to discriminate between ambiguous and unambiguous subgestures.

Chapter 5 extends the statistical approach to the recognition of multiple-path gestures. This is useful for utilizing devices that can sense the positions of multiple fingers simultaneously, in particular the Sensor Frame.

Chapter 6 presents the architecture of an object-oriented toolkit for the construction of direct-manipulation systems. Like many other systems, this architecture is based on the Model-View-

[7] This describes the Sensor Frame as originally envisioned. The hardware is capable of producing a few bits of finger velocity and angle information, although to date this has not been attempted.

Controller paradigm. Compared to previous toolkits, the input model is considerably generalized in preparation for the incorporation of gesture recognition into a direct-manipulation system. The notion of virtual tools, through which input may be generated by software objects in the same manner as by hardware input devices, is introduced. Semantic feedback will be shown to arise naturally from this approach.

Chapter 7 shows how gesture recognizers are incorporated into the direct-manipulation architecture presented in Chapter 6. A gesture handler may be associated with a particular view of an object on the screen, or at any level in the view hierarchy. In this manner, different objects will respond to different sets of gestures. The communication of parametric data from gesture handler to application is also examined.

Chapter 8 discusses three gesture-based systems built using these techniques: GDP, GSCORE, and MDP. The first two, GDP and GSCORE, use mouse gestures. GDP, as already mentioned, is the drawing editor based on DP. GSCORE is a musical score editor, based on Buxton's SSSP work [21]. MDP is also a drawing editor, but it operates using multi-path gestures made with a Sensor Frame. The design and implementation of each system is discussed, and the gestures for each shown.

Chapter 9 evaluates a number of aspects of this work. The particular recognition algorithms are tested for recognition accuracy. Measurements of the performance of the gesture classifiers used in the applications is presented. Then, an informal user study assessing the utility of gesture-based systems is discussed.

Finally, Chapter 10 concludes this dissertation. The contributions of this dissertation are discussed, as are the directions for future work.

1.12 What Is Not Covered

This dissertation attempts to cover many topics relevant to gesture-based systems, though by no means all of them. In particular, the issues involved in the ergonomics and suitability of gesture-based systems applied to various task domains have not been studied. It is the opinion of the author that such issues can only be studied after the tools have been made available which allow easy creation of and experimentation with such systems. The intent of the current work is to provide such tools. Future research is needed to determine how to use the tools to create the most usable gesture-based systems possible.

Of course, choices have had to be made in the implementation of such tools. By avoiding the problem of determining which kind of gesture-based systems are best, the work opens itself to charges of possibly "throwing the baby out with the bath-water." The claim is that the general system produced is capable of implementing systems comparable to many existing gesture-based systems; the example applications implemented (see Chapter 8) support this claim. Furthermore, the places where restrictive choices have been made (*e.g.* two-dimensional gestures) have been indicated, and extensible and scalable methods (*e.g.* linear discrimination) have been used wherever possible.

There are two major limitations of the current work. The first is that single-path multi-stroke gestures (*e.g.* handwritten characters) are not handled. Most existing gesture-based systems use single-path multi-stroke gestures. The second limitation is that the start of a gesture must be

explicitly indicated. This rules out (at least at first glance) using devices such as the DataGlove which lack buttons or other explicit signaling hardware. However, one result of the current work is that these apparent limitations give rise to certain advantages in gestural interfaces. For example, the limitations enforce Buxton's notion of tension and release mentioned above.

Gestural output, *i.e.* generating a gesture in response to a query, is also not covered. For an example of gestural output, ask the author why he has taken so long to complete this dissertation.

Chapter 2

Related Work

This chapter discusses previous work relevant to gesture recognition. This includes hardware devices suitable for gestural input, existing gesture-based systems, pattern recognition techniques, and software systems for building user interfaces.

Before delving into details, it is worth mentioning some general work that attempts to define gesture as a technique for interacting with computers. Morrel-Samuels [87] examines the distinction between gestural and lexical commands, and then further discusses problems and advantages of gestural commands. Wolf and Rhyne [140] integrate gesture into a general taxonomy of direct manipulation interactions. Rhyne and Wolf [109] discuss in general terms human-factors concerns of gestural interfaces, as well as hardware and software issues.

The use of gesture as an interaction technique is justified in a number of studies. Wolf [139] performed two experiments that showed gestural interfaces compare favorably to keyboard interfaces. Wolf [141] showed that many different people naturally use the same gestures in a text-editing context. Hauptmann [49] demonstrated a similar result for an image manipulation task, further showing that people prefer to combine gesture and speech rather than use either modality alone.

2.1 Input Devices

A number of input devices are suitable for providing input to a gesture recognizer. This section concentrates on those devices which provide the position of one or more points over time, or whose data is easily converted into that representation. The intention is to list the types of devices which can potentially be used for gesturing. The techniques developed in this dissertation can be applied, directly or with some generalization, to the devices mentioned.

A large variety of devices may be used as two-dimensional, single-path gesturing devices. Some graphical input devices, such as mice [33], tablets and styli, light pens, joysticks, trackballs, touch tablets, thumb-wheels, and single-finger touch screens [107, 124], have been in common use for years. Less common are foot controllers, knee controllers, eye trackers [12], and tongue-activated joysticks. Each may potentially be used for gestural input, though ergonomically some are better suited for gesturing than others. Baecker and Buxton [5], Buxton [14], and Buxton, Hill and Rowley [18] discuss the suitability of many of the above devices for various tasks. Buxton further points out

that two different joysticks, for example, may have very different properties that must be considered with respect to the task.

For gesturing, as with pointing, it is useful for a device to have some signaling capability in addition to the pointer. For example, a mouse usually has one or more buttons, the pressing of which can be used to indicate the start of a gesture. Similarly, tablets usually indicate when the stylus makes or breaks contact with the tablet (though with a tablet it is not possible to carefully position the screen cursor before contact). If a device does not have this signaling capacity, it will be necessary to simulate it somehow. Exactly how this is done can have a large impact on whether or not the device will be suitable for gesturing.

The 3SPACE Isotrack system, developed by Polhemus Navigation Sciences Division of McDonnel Douglas Electronics Company [32], is a device which measures the position and orientation of a stylus or a one-inch cube using magnetic fields. The Polhemus sensor, as it is often called, is a full six-degree-of-freedom sensor, returning x, y, and z rectangular coordinates, as well as azimuth, altitude, and roll angles. It is potentially useful for single path gesturing in three positional dimensions. By considering the angular dimensions, 4, 5, or 6 dimensional gestures may be entered. It is also possible to use one of the angular dimensions for signaling purposes.

Bell Laboratories has produced prototypes of a clear plate capable of detecting the position and pressure of many fingers [10, 99]. The position information is two-dimensional, and there is a third dimension as well: finger pressure. The author has seen the device reliably track 10 fingers simultaneously. The pressure detection may be used for signaling purposes, or as a third dimension for gesturing. The inventor of the multi-finger touch plate has invented another device, the Radio Drum [11], which can sense the position of multiple antennae in three dimensions. To date, the antennae have been embedded in the tips of drum sticks (thus the name), but it would also be possible to make a glove containing the antenna which would make the device more suitable for detecting hand gestures.

The Sensor Frame [84] is a frame mounted on a workstation screen (figure 2.1). It consists of a light source (which frames the screen) and four optical sensors (one in each corner). The Sensor Frame computes the two-dimensional positions of up to three fingertips in a plane parallel to, and slightly above the screen. The net result is similar to a multi-finger touch screen. The author has used the Sensor Frame to verify the multi-finger recognition algorithm described in Chapter 5. The Sensor Cube [85] is a device similar to the Sensor Frame but capable of sensing finger positions in three dimensions. It is currently under construction. The VideoHarp [112, 111] is a musical instrument based on the same sensing technology, and is designed to capture parametric gestural data.

The DataGlove [32, 130] is a glove worn on the hand able to produce the positions of multiple fingers as well as other points on the hand in three dimensions. By itself it can only output relative positions. However, in combination with the Polhemus sensor, absolute finger positions can be computed. Such a device can translate gestures as complex as American Sign Language [123] into a multi-path form suitable for processing. The DataGlove, the similar Dexterous Hand Master from Exos, and the Power Glove from Mattel, are shown in figure 2.2.

The DataGlove comes with hardware which may be trained to recognize certain static configurations of the glove. For example, the DataGlove hardware might be trained to recognize a fist,

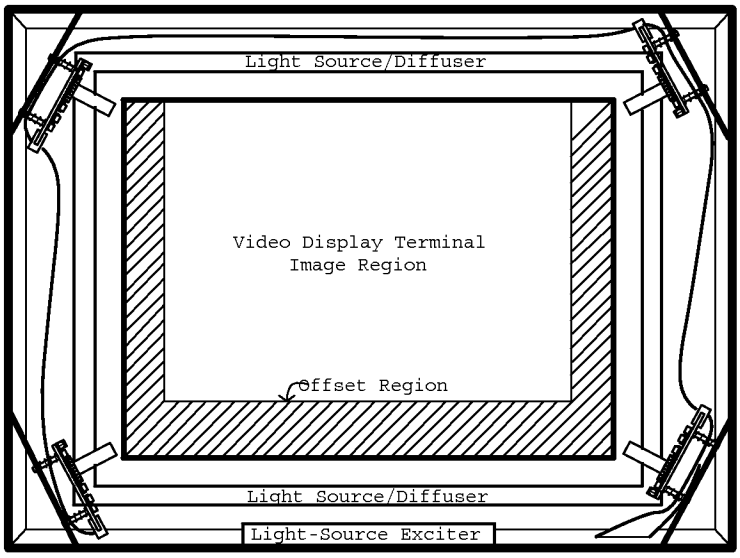

Figure 2.1: The Sensor Frame

The Sensor Frame is a frame mounted on a computer display consisting of a rectangular light source and four sensors, one in each corner. It is capable of detecting up to three fingers its field of view. (Drawing by Paul McAvinney.)

Figure 2.2: The DataGlove, Dexterous Hand Master, and PowerGlove (from Eglowstein [32])

The DataGlove, Dexterous Hand Master, and PowerGlove are three glove-like input devices capable of measuring the angles of various hand and finger joints.

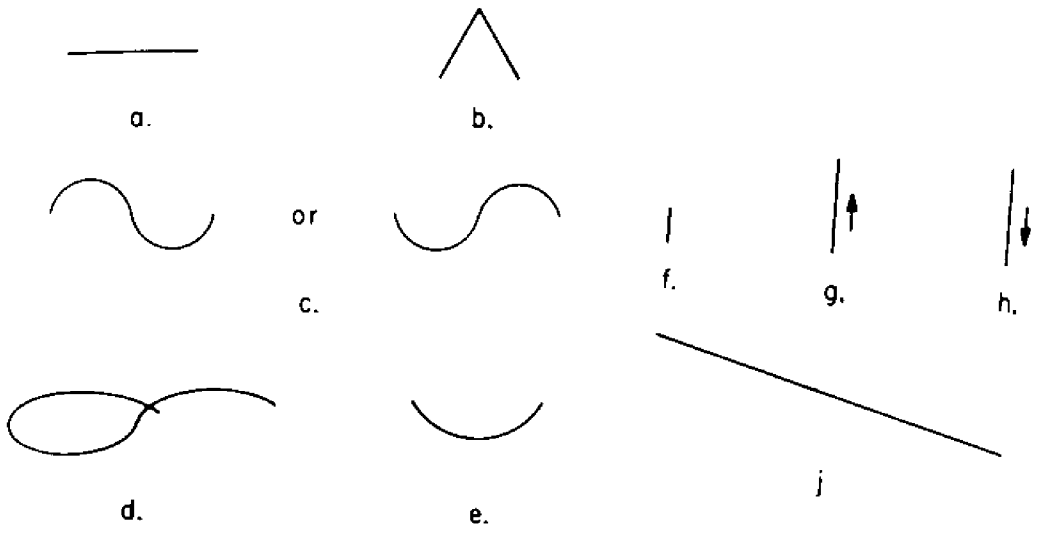

Figure 2.3: Proofreading symbols (from Coleman [25])
The operations intended by each are as follows: a) delete text (from a single line), b) insert text, c) swap text, d) move text, e) join (delete space), f) insert space, g) scroll up, h) scroll down, and i) delete multiple lines of text. Many of the marks convey additional parameters to the operation, e.g. *the text to be moved or deleted.*

signaling the host computer whenever a fist is made. These static hand positions are not considered to be gestures, since they do not involve motion. The glove hardware recognizes "posture" rather than gesture, the distinction being that posture is a static snapshot (a pose), while gesture involves motion over time. Nonetheless, it is a rather elegant way to add signaling capability to a device without buttons or switches.

The Videodesk [71, 72] is an input device based on a constrained form of video input. The Videodesk consists of a translucent tablecloth over a glass top. Under the desk is a light source, over the desk a video camera. The user's hands are placed over the desk. The tablecloth diffuses the light, the net effect being that the camera receives an image of the silhouette of the hands. Additional hardware is used to detect and track the user's fingertips.

Some researchers have investigated the attachment of point light sources to various points on the body or hand to get position information as a function of time. The output of a camera (or pair of cameras for three dimensional input) can be used as input to a gesture sensor.

2.2 Example Gesture-based Systems

This section describes a number of existing gesture-based systems that have been described in the literature. A system must both classify its gestural input and use information other than the class (*i.e.* parametric information) to be included in this survey. The order is roughly chronological.

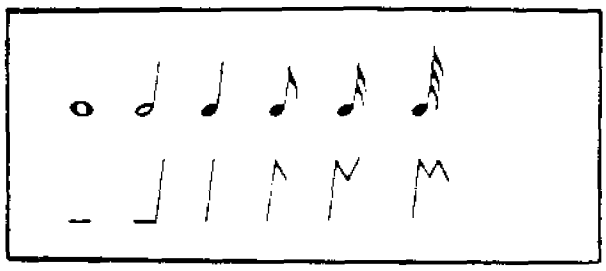

Figure 2.4: Note gestures (from Buxton [21])
A single gesture indicates note duration (from the shape of the stroke as shown) as well as pitch and starting time, both of which are determined from the position of the start of the gesture.

Coleman [25] has created a text editor which used hand-drawn proofreader's symbols to specify editing commands (figure 2.3). For example, a sideways "S" indicated that two sets of characters should be interchanged, the characters themselves being delimited by the two halves of the "S." The input device was a touch tablet, and the gesture classification was done by a hand-coded discrimination net (*i.e.* a loop-free flowchart).[1]

Buxton [21] has built a musical score editor with a small amount of gesture input using a mouse (figure 2.4). His system used simple gestures to indicate note durations and scoping operations. Buxton considered this system to be more a character recognition system than a gesture-based system, the characters being taken from an alphabet of musical symbols. Since information was derived not only from the classification of the characters, but their positions as well, the author considers this to be a gesture-based system in the true sense. Buxton's technique was later incorporated into Notewriter II, a commercial music scoring program. Lamb and Buckley [76] describe a gesture-based music editor usable by children.

Margaret Minsky [86] implemented a system called Button Box, which uses gestures for selection, movement, and path specification to provide a complete Logo programming environment (figure 2.5). Her input device was a clear plate mounted in front of a display. The device sensed the position and shear forces of a single finger touching the plate. Minksy proposed the use of multiple fingers for gesture input, but never experimented with an actual multiple-finger input device.

In Minsky's system, buttons for each Logo operation were displayed on the screen. Tapping a button caused it to execute; touching a button and dragging it caused it to be moved. The classification needed to distinguish between a touch and a tap was programmed by hand. There were buttons used for copying other buttons and for grouping sets of buttons together. A path could be drawn through a series of buttons–touching the end of a path caused its constituent buttons to execute sequentially.

VIDEOPLACE [72] is a system based on the Videodesk. As stated above, the silhouette of the user's hands are monitored. When a hand is placed in a pointing posture, the tip of the index finger

[1]Curiously, this research was done while Coleman was a graduate student at Carnegie Mellon. Coleman apparently never received a Ph.D. from CMU, and it would be twenty years before another CMU graduate student (me) would go near the topic of gesture recognition.

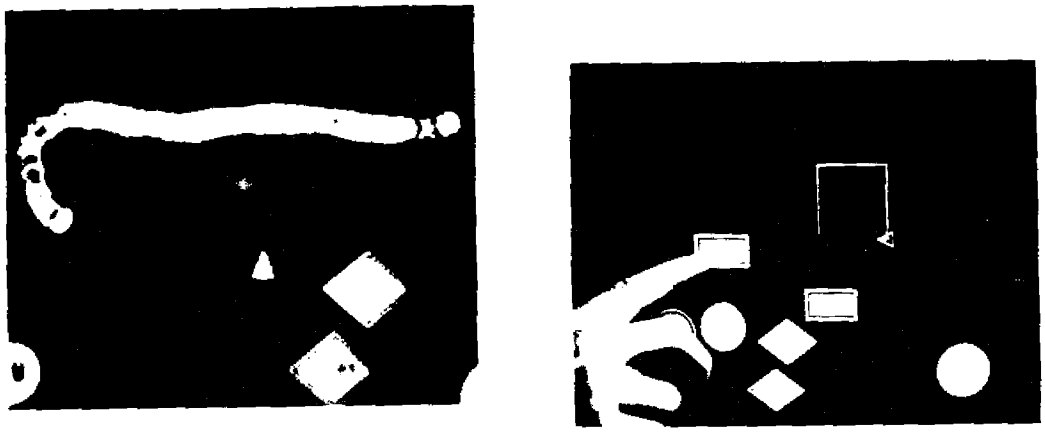

Figure 2.5: Button Box (from Minksy [86])

Tapping a displayed button causes it to execute its assigned function while touching a button and dragging it causes it to be moved.

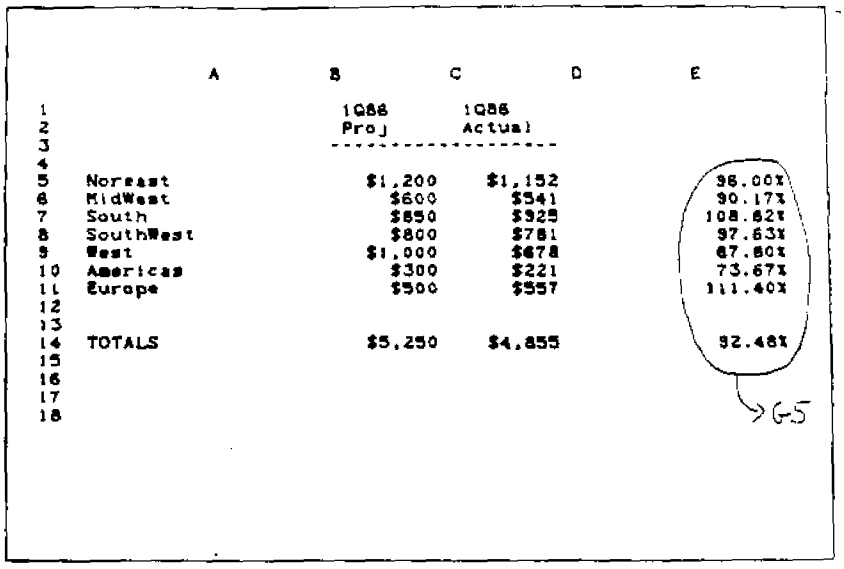

Figure 2.6: A gesture-based spreadsheet (from Rhyne and Wolf [109])

The Paper-Like Interface project produces systems which combine gesture and handwriting. The input shown here selects a group of cells and requests they be moved to the cell beginning at location "G5."

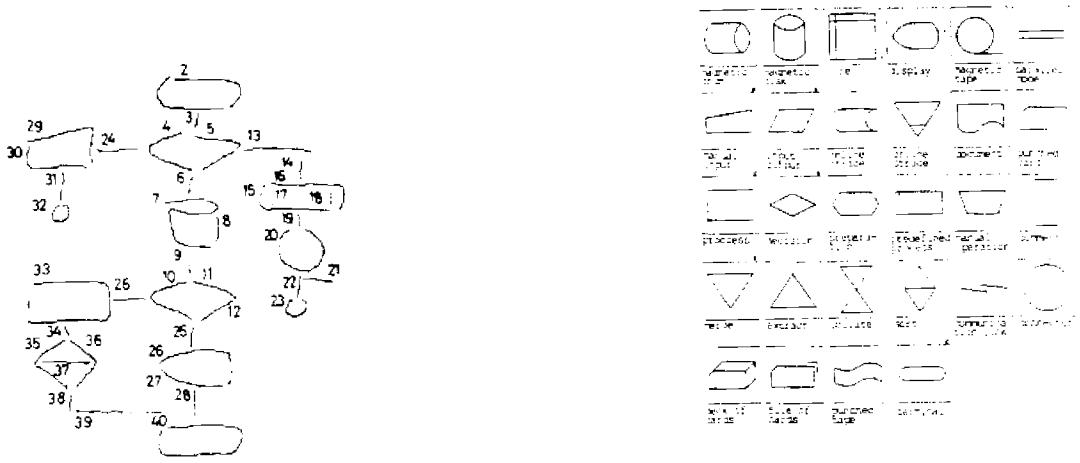

Figure 2.7: Recognizing flowchart symbols

Recognizing flowchart symbols (from Murase and Wakahara [89]). The system takes an entire freehand drawing of a flowchart (left) and recognizes the individual flowchart symbols (right), producing an internal representation of the flowchart (as nodes and edges) and a flowchart picture in which the freehand symbols are replaced by machine generated line-drawings drawn from the alphabet of symbols. This system shows a style of interface in which pattern recognition is used for something other than the detection of gestures or characters.

may be used for menu selection. After selection, the fingertips may be used to manipulated graphic objects, such as the controlling points of a spline curve.

A group at IBM doing research into gestural human-computer systems has produced a gesture-based spreadsheet application [109]. Somewhat similar to Coleman's editor, the user manipulates the spreadsheet by gesturing with a stylus on a tablet (figure 2.6). For example, deletion is done by drawing an "X" over a cell, selection by an "0", and moving selected cells by an arrow, the tip of which indicates the destination of the move. The application is interesting in that it combines handwriting recognition (isolated letters and numbers) with gesturing. For example, by using handwriting the user can enter numbers or text into a cell without using a keyboard. The portion of the recognizer which classifies letters, numbers, and gestures of a fixed size and orientation has (presumably) been trained by example using standard handwriting recognition techniques. However, the recognition of gestures which vary in size or orientation requires hand coding [67].

Murase and Wakahara [89] describe a system in which freehand-drawn flowcharts symbols are recognized by machine (figure 2.7). Tamura and Kawasaki [128] have a system which recognizes sign-language gestures from video input (figure 2.8).

HITS from MCC [55] and Artkit from the University of Arizona [52] are both systems that may be used to construct gesture-based interfaces. The author has seen a system built with HITS similar

Figure 2.8: Sign language recognition (from Tamura [128])
This system processes an image from a video camera in order to recognize a form of Japanese sign language.

to that of Murase; in it an entire control panel is drawn freehand, and then the freehand symbols are segmented, classified and replaced by icons. (Similar work is discussed by Martin, *et. al* [82], also from MCC.) Artkit has much in common with the GRANDMA system described in this dissertation, and will be mentioned again later (Sections 4.1 and 6.8). Artkit systems tend to be similar to those created using GRANDMA, in that gesture commands are executed as soon as they are entered.

Kurtenbach and Buxton [75] have implemented a drawing program based on single-stroke gestures (figure 2.9). They have used the program to study, among other things, issues of scope in gestural systems. To the present author, GEdit's most interesting attribute is the use of compound gestures, as shown in the figure. GEdit's gesture recognizer is hand-coded.

The Glove-talk system [34] uses a DataGlove to control a speech synthesizer (figure 2.10). Like Artkit and the work described in Chapter 4, Glove-talk performs eager recognition: a gesture is recognized and acted upon without its end being indicated explicitly. Weimer and Ganapathy [136] describe a system combining DataGlove gesture and speech recognition.

The use of the circling gesture as an alternative means of selection is considered in Jackson and Roske-Hofstrand [61]. In their system, the start of the circling gesture is detected automatically, *i.e.* the mouse buttons are not used. Circling is also used for selection in the JUNO system from Xerox Corporation [142].

A number of computer products offer a stylus and tablet as their sole or primary input device. These systems include GRID Systems Corp.'s GRIDPad [50], Active Book Company's new portable [43], Pencept Inc.'s computer [59], Scenario's DynaWriter, Toshiba's PenPC, Sony's Palmtop, Mometa's laptop, MicroSlate's Datalite, DFM System's TraveLite, Agilis Corp.'s system, and Go Corp.'s PenPoint system [81, 24]. While details of the interface of many of these systems are hard to find (many of these systems have not yet been released), the author suspects that many use gestures. For further reading, please see [16, 106, 31].

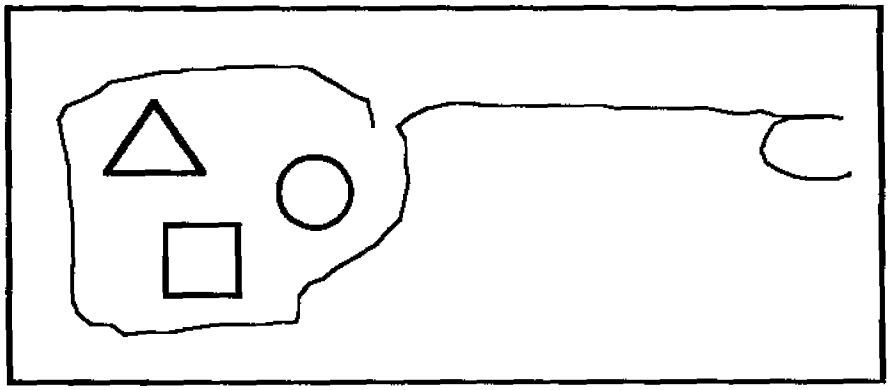

Figure 2.9: Copying a group of objects in GEdit (from Kurtenbach and Buxton [75])
Note the compound gesture: the initial closed curve does selection, and the final "C" indicates the data should be copied rather than moved.

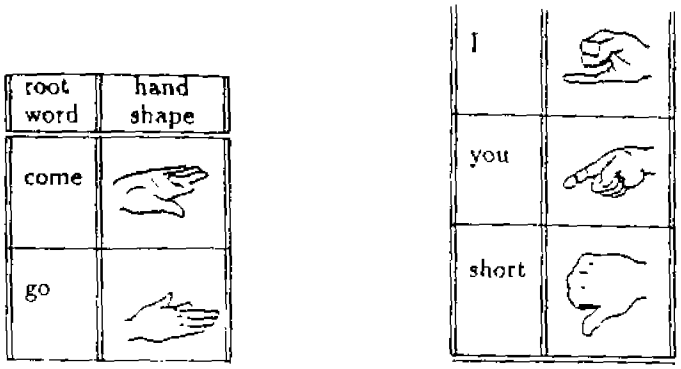

Figure 2.10: GloveTalk (from Fels and Hinton [34])
GloveTalk connects a DataGlove to a speech synthesizer through several neural networks. Gestures indicate root words (shown) and modifiers. Reversing the direction of the hand motion causes a word to be emitted from the synthesizer as well as indicating the start of the next gesture.

Tap	·	Select/Invoke
Press-hold	●	Initiate drag (move, wipe-through)
Tap-hold	· ●	Initiate drag (copy)
Flick (four directions)	|	Scroll/Browse
Cross out	X	Delete
Scratch out	≡	Delete
Circle	○	Edit
Check	✓	Options
Caret	∧	Insert
Brackets	[ ]	Select object, adjust selection
Pigtail (vertical)	℘	Delete character
Down-right	L	Insert space

Figure 2.11: Basic PenPoint gestures (from Carr [24])

Recently, prototypes of Go Corporation's PenPoint system have been demonstrated. Each consists of a notebook-sized computer with a flat display. The sole input device is a stylus, which is used for gestures and handwriting on the display itself. Figure 2.11 shows the basic gestures recognized; depending on the context, additional gestures and handwriting can also be recognized. As can be seen, PenPoint gestures may consist of multiple strokes. Although it seems that trainable recognition algorithms are used internally, at the present time the user cannot add any new gestures to the existing set. The hardware is able to sense pen *proximity* (how near the stylus is to the tablet), which is used to help detect the end of multi-stroke gestures and characters. PenPoint applications include a drawing program, a word processor, and a form-based data entry system.

Many of the above systems combine gesture and direct manipulation in the same interface. GEdit, for example, appears to treat mouse input as gestural when begun on the background window, but drags objects when mouse input begins on the object. Almost none combine gesture and direct manipulation in the same interaction. One exception, PenPoint, uses the dot gesture (touching the stylus to the tablet and then not moving until recognition has been indicated) to drag graphic objects. Button Box does something similar for dragging objects. Artkit [52] uses eager recognition, more or less crediting the idea to me.

2.3 Approaches for Gesture Classification

Fu [40] states that "the problem of pattern recognition usually denotes a discrimination or classification of a set of processes or events." Clearly gesture recognition, in which the input is considered to be an event to be classified as one of a particular set of gestures, is a problem of pattern recognition.

In this dissertation, known techniques of pattern recognition are applied to the problem of sensing gestures.

The general pattern recognition problem consists of two subproblems: pattern representation and decision making [40]. This implies that the architecture of the general pattern recognition consists of two main parts. First, the *representer* takes the raw pattern as input and outputs the internal representation of the pattern. Then, the *decider* takes as input the output of the representer, and outputs a classification (and/or a description) of the pattern.

This section reviews the pattern recognition work relevant to gesture recognition. In particular, the on-line recognition of handwritten characters is discussed whenever possible, since that is the closest solved problem to gesture recognition. For a good overview of handwriting systems in general, see Suen *et. al.* [125] or Tappert *et. al.* [129].

The review is divided into two parts: alternatives for representers and alternatives for deciders. Each alternative is briefly explained, usually by reference to an existing system which uses the approach. The advantages and disadvantages of the alternative are then discussed, particularly as they apply to single-path gesture recognition.

2.3.1 Alternatives for Representers

The representer module takes the raw data from the input device and transforms it into a form suitable for classification by the decider. In the case of single-path gestures, as with on-line handprint, the raw data consists of a sequence of points. The representer outputs *features* of the input pattern.

Representers may be grouped in terms of the kinds of features which they output. The major kinds of features are: templates, global transformations, zones, and geometric features. While a single representer may combine different kinds of features, representers are discussed here as if each only outputs one kind of feature. This will make clearer the differences between the kinds of features. Also, in practice most representers do depend largely on a single kind of feature.

Templates.

Templates are the simplest features to compute: they are simply the input data in its raw form. For a path, a template would simply consist of the sequence of points which make up the path. Recognition systems based on templates require the decider to do the difficult work; namely, matching the template of the input pattern to stored example templates for each class.

Templates have the obvious advantage that the features are simple to compute. One disadvantage is that the size of the feature data grows with the size of the input, making the features unsuitable as input to certain kinds of deciders. Also, template features are very sensitive to changes in the size, location, or orientation of the input, complicating classifiers which attempt to allow for variations of these within a given class. Examples of template systems are mentioned in the discussion of template matching below.

Global Transformations.

Some of the problems of template features are addressed by global transformations of the input data. The transformations are often mathematically defined so as to be invariant under *e.g.* rotation, translation, or scaling of the input data. For example, the Fourier transform will result in features invariant with respect to rotation of the input pattern [46]. Global transformations generally output a fixed number of features, often smaller than the input data.

A set of fixed features allows for a greater variety in the choice of deciders, and obviously the invariance properties allow for variations within a class. Unfortunately, there is no way to "turn off" these invariances in order to disallow intra-class variation. Also, the global transformations generally take as input a two-dimensional raster, making the technique awkward to use for path data (it would have to first be transformed into raster data). Furthermore, the computation of the transformation may be expensive, and the resulting features do not usually have a useful parametric interpretations (in the sense of Section 1.6.2), requiring a separate pass over the data to gather parametric information.

Zones.

Zoning is a simple way of deriving features from a path. Space is divided into a number of zones, and an input path is transformed into the sequence of zones which the path traverses [57]. One variation on this scheme incorporates the direction each zone is entered into the encoding [101]. As with templates, the number of features are not fixed; thus only certain deciders may be used. The major advantage of zoning schemes are their simplicity and efficiency.

If the recognition is to be size invariant, zoning schemes generally require the input to be normalized ahead of time. Making a zoning scheme rotationally invariant is more difficult. Such normalizations make it impossible to compute zones incrementally as the input data is received. Also, small changes to a pattern might cause zones to be missed entirely, resulting in misclassification. And again, the features do not usually hold any useful parametric information.

Geometric Features.

Geometric features are the most commonly used in handwriting recognition [125]. Some geometric features of a path (such as its total length, total angle, number of times it crosses itself, *etc.*) represent global properties of the path. Local properties, such as the sequence of basic strokes, may also be represented.

It is possible to use combinations of geometric features, each invariant under some transformations of the input pattern but not others. For example, the initial angle of a path may be a feature, and all other features might be invariant with respect to rotation of the input. In this fashion, classifiers may potentially be created which allow different variations on a per-class basis.

Geometric features often carry useful parametric information, *e.g.* the total path length, a geometric feature, is potentially a useful parameter. Also, geometric features can be fed to deciders which expect a fixed number of features (if only global geometric features are used), or to deciders which expect a sequence of features (if local features are used).

Geometric features tend to be more complex to compute than the other types of features listed. With care, however, the computation can be made efficient and incremental. For all these reasons, the current work concentrates on the use of global geometric features for the single-path gesture recognition in this dissertation (see Chapter 3).

2.3.2 Alternatives for Deciders

Given a vector or sequence of features output by a representer, it is the job of the decider to determine the class of the input pattern with those features. Seven general methods for deciders may be enumerated: template-matching, dictionary lookup, a discrimination net, statistical matching, linguistic matching, connectionism, and *ad hoc*. Some of the methods are suitable to only one kind of representer, while others are more generally applicable.

Template-matching.

A template-matching decider compares a given input template to one or more prototypical templates of each expected class. Typically, the decider is based on a function which measures the similarity (or dissimilarity) between pairs of templates. The input is classified as being a member of the same class as the prototype to which it is most similar. Usually there is a similarity threshold, below which the input will be rejected as belonging to none of the possible classes.

The similarity metric may be computed as a correlation function between the input and the prototype [69]. Dynamic programming techniques may be used to efficiently warp the input in order to better match up points in the input template to those in the prototype [133, 60, 9].

Template systems have the advantage that the prototypes are simply example templates, making the system easy to train. In order to accommodate large variations, for example in the orientation of a given gesture, a number of different prototypes of various orientation must be specified. Unfortunately, a large number of prototypes can make the use of template matching prohibitively expensive, since the input pattern must be compared to every template.

Lipscomb [80] presents a variation on template matching used for recognizing gestures. In his scheme, each training example is considered at different resolutions, giving rise to multiple templates per example. (The algorithm is thus similar to multiscale algorithms used in image processing [138].) Lipscomb has applied the multiscale technique to stroke data by using an angle filter, in which different resolutions correspond to different thresholds applied to the angles in the gestures. To represent a gesture at a given resolution, points are discarded so that the remaining angles are all below the threshold. To classify an input gesture, first its highest resolution representation is (conceptually) compared to each template (at every resolution). Successively lower resolutions of the input are tried in turn, until an exact match is found. Multiple matches are decided in favor of the template whose resolution is closest to the current resolution of the input.

Dictionary lookup.

When the input features are a sequence of tokens taken from a small alphabet, lookup techniques may be used. This is often how zoning features are classified [101]. The advantage is efficient

recognition, since binary search (or similar algorithms) may be used to lookup patterns in the dictionary. Often some allowance is made for non-exact matches, since otherwise classification is sensitive to small changes in the input. Even with such allowances, dictionary systems are often brittle, due to the features employed (*e.g.* sequences of zones). Of course, a dictionary is initially created from example training input. It is also a simple matter to add new entries for rejected patterns; thus the dictionary system can adapt to a given user.

Discrimination nets.

A discrimination net (also called a decision tree) is basically a flowchart without loops. Each interior node contains a boolean condition on the features, and is connected to two other nodes (a "true" branch and a "false" branch). Each leaf node is labeled with a class name. A given feature set is classified by starting at the root note, evaluating each condition encountered and taking the appropriate branch, stopping and outputting the classification when a leaf node is reached.

Discrimination nets may be created by hand [25], or derived from example inputs [8]. They are more appropriate to classifying fixed-length feature vectors, rather than sequences of arbitrary length, and often result in accurate and efficient classifiers. However, discrimination nets trained by example tend to become unwieldy as the number of examples grows.

Statistical matching.

In statistical matching, the statistics of example feature vectors are used to derive classifiers. Typically, statistical matchers operate only on feature vectors, not sequences. Some typical statistics used are: average feature vector per class, per-class variances of the individual features, and per-class correlations within features. One method of statistical matching is to compute the distance of the input feature vector to the average feature vector of each class, choosing the class which is the closest. Another method uses the statistics to derive per-class discrimination functions over the features. Discrimination functions are like evaluation functions: each discrimination function is applied to the input feature vector, the class being determined by the largest result. Fisher [35] showed how to create discrimination functions which are simply linear combinations of the input features, and thus particularly efficient. Arakawa *et. al.*[3] used statistical classification of Fourier features for on-line handwriting recognition; Chapter 3 of the present work uses statistical classification of geometric features.

Some statistical classifiers, such as the Fisher classifier, make assumptions about the distributions of features within a class (such as multivariate normality); those tend to perform poorly when the assumptions are violated. Other classifiers [48] make no such assumptions, but instead attempt to estimate the form of the distribution from the training examples. Such classifiers tend to require many training examples before they function adequately. The former approach is adopted in the current work, with the feature set carefully chosen so as to not violate assumptions about the underlying distribution too drastically.

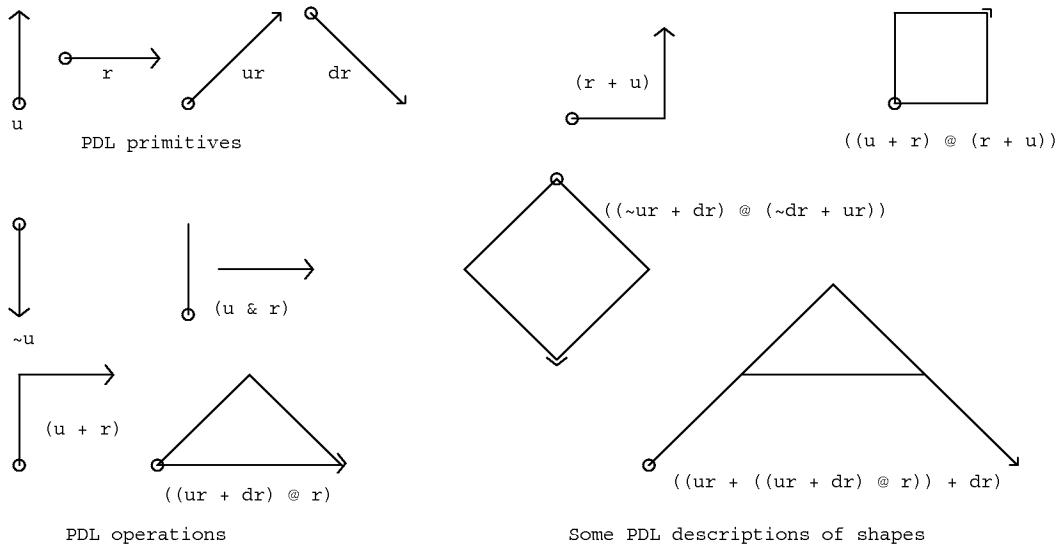

Figure 2.12: Shaw's Picture Description Language

PDL enables line drawings to be coded in string form, making it possible to apply textual parsing algorithms to the recognition of line drawings. The component line segments and combining operations are shown on the left; the right shows how the letter "A" can be described using these primitives.

Linguistic matching.

The linguistic approach attempts to apply automata and formal language theory to the problem of pattern recognition [37]. The representer outputs a sequence of tokens which is composed of a set of pattern primitives and composition operators representing the relation between the primitives. The decider has a grammar for each possible pattern class. It takes as input the sentence and attempts to parse it with respect to each pattern class grammar. Ideally, exactly one of the parses is successful and the pattern is classified thus. A useful side effect of the syntax analysis is the parse tree (or other parse trace) which reveals the internal structure of the pattern.

Linguistic recognizers may be classified based on the form of the representer output. If the output is a string then standard language recognition technology, such as regular expressions and context-free grammars, may be used to parse the input. An error-correcting parser may be used in order to robustly deal with errors in the input. Alternatively, the output of the representer may be a tree or graph, in which case the decider could use graph matching algorithms to do the parse.

The token sequence could come from a zoning representer, a representer based on local geometric properties, or from the output of a lower-level classifier. The latter is a hybrid approach–where, for example, statistical recognition is used to classify paths (or path segments), and linguistic recognition is used to classify based on the relationships between paths. This approach is similar to that taken by Fu in a number of applications [40, 39, 38].

Shaw's picture description language (PDL, see figure 2.12) has been used successfully to describe and classify line drawings [116, 40]. In another system, Stallings [120, 37] uses the composition