Der letzte Teil dieser Artikelreihe beschreibt eine komplette Fallstudie, die auf zahlreiche der im Rahmen dieser Serie vorgestellten Ansätze Bezug nimmt. Genauer gesagt geht es um das Problem des Pattern-Matching in Bildern.

Dr. Lothar Wenzel

Digitale Signalverarbeitung ist keine Hexerei

Teil 13: Abschließende Fallstudie

Allein schon ein geeignetes Beispiel für diesen Schlußpart der Serie zu finden, ist keine einfache Aufgabe. Naturgemäß möchte man möglichst viele Teile der Signalverarbeitung zu Wort kommen lassen, aber ungerecht ist man dabei in jedem Fall. Das Problem des Pattern-Matching in Bildern ist vielleicht ein ganz akzeptabler Kompromiß, unter anderem schon deshalb, weil man zwischen ein- und zweidimensionaler Signalverarbeitung hin und her pendeln muß. Zahlreiche weitere Tools kommen zur Anwendung. Die wichtigste Botschaft des letzten Teils der Serie ist sicherlich die, daß einigermaßen anspruchsvolle praktische Anwendungen fast stets die konzertierte Aktion zahlreicher Einzelkomponenten erfordern.

Auf den folgenden Seiten werden im einzelnen behandelt:

1. Problemstellung
2. Entwurf einer Lösungsstrategie
3. Kantendetektion
4. Codierung der Kantenzüge
5. Normalisierung und Vergleich
6. Akzeptanz von Kandidaten
7. Genauigkeit der Lösung
8. Implementierungsfragen
9. Alternative Strategien

Problemstellung

Wohl kaum eine zweite Problemstellung im Gebiet des maschinellen Sehens kommt in der Bedeutung dem Pattern-Matching gleich. Ausgangspunkt ist ein digitalisiertes Graustufenbild (typische Größe 512 × 512 Pixel) und ein zweites kleineres Template (ebenfalls digitalisiertes Muster-Bild mit einer typischen Größe von 64 × 64 Pixel). Das Template ist vor der eigentlichen Erkennungsroutine bekannt, d.h., man hat gewöhnlich ausreichend Zeit, die Gegebenheiten einer existierenden Template-Vorlage genauestens zu studieren. Gemeint sind dabei vielleicht zehn Sekunden, die in der Signalverarbeitung allerdings wirklich einer Ewigkeit gleichkommen.

Die Aufgabe besteht nun darin, das Template im originalen Bild genauestens zu lokalisieren, d.h., eine oder mehrere Kopien des Templates sollten sich im Original befinden. Man denke etwa an ein Computerboard als Bildvorgabe, wobei das Template einen speziellen Chip darstellt. Die Frage ist nun, ob – und wenn ja, an welcher Stelle des Boards – der Chip oder die Chips zu finden sind (*Bild 1*).

Das alles ist nur eine sehr oberflächliche Beschreibung der Gesamtproblematik. Zahlreiche Störfaktoren und andere Erschwernisse treten in der Praxis auf. So wird die Qualität des Templates im Regelfalle exzellent sein, gleiches kann nicht immer von den Bildern gesagt werden, bei deren Auswertung man zusätzlich unter starkem Zeitdruck steht (einige 100 ms). Ferner können Templates durchaus im Bild gedreht vorliegen, und nicht selten sind sie auch mit einem Vergrößerungs- bzw. Verkleinerungsfaktor behaftet. Die Liste dieser Unannehmlichkeiten ist leider noch länger.

Entwurf einer Lösungsstrategie

Noch vor etwa zehn Jahren gab es eigentlich nur eine einzige Methode, das Problem des Pattern-Matching zu bewältigen. Gemeint ist die normalisierte Kreuzkorrelation, eine Methode, die man zuweilen noch heute im Einsatz hat. Die Grundidee ist im wesentlichen ein pixelweiser Vergleich zwischen Template und allen möglichen Konstellationen auf dem Bild. Der Rechenaufwand hierfür ist enorm und nimmt dramatische Züge an, wenn man auch noch Rotation und Skalierungseffekte behandeln will.

Eine viel bessere Idee geht davon aus, daß Bilder in dieser oder jener Form „Struktur“ besitzen. Zu diesem Begriff kann man beispielsweise Kanten zählen, die übrigens bei vielen praktischen Anwendungsfällen geradezu ins Auge

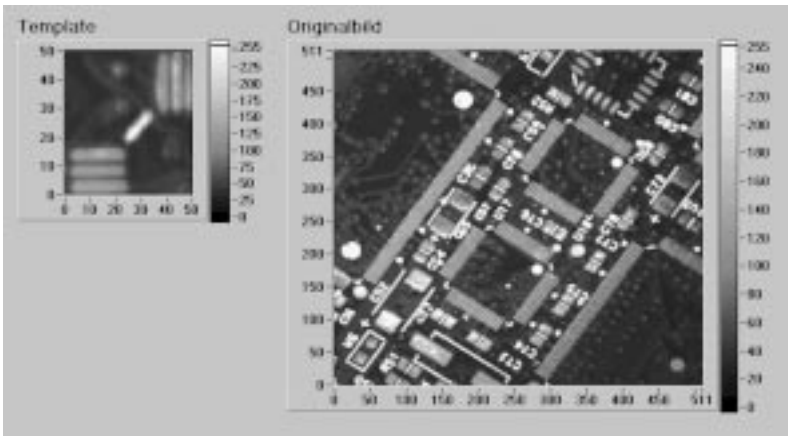

Bild 1. Ein typisches Beispiel für das Pattern-Matching. Das Template ist auf dem Board aufzuspüren. Rotations- und Skalierungseffekte sind dabei einzukalkulieren. Finden Sie es in 100 ms und das 1 Million Mal am Tag?

springen. Man denke in diesem Zusammenhang nochmals an das Computerboard mit den darauf befindlichen Chips, Verbindungslinien oder auch anderen klar strukturierten Bauelementen. Kanten allein sind nicht genug, es gilt diese aufzuspüren und gewissermaßen zu verfolgen.

Besitzt man eine Beschreibung des Kantengerüstes vom originalen Bild und vom Template, so kann man versuchen, anstelle des Vergleiches von ganzen Bildern das Ausgangsproblem auf das Matching von Kantenzügen zurückzuführen. Man erreicht dadurch zweierlei. Zum einen ist die Überführung einer ursprünglich zweidimensionalen Aufgabenstellung in ein eindimensionales Problem gelungen. Die hieraus erwachsenden Vorteile liegen auf der Hand und sind tatsächlich prägnant. Zum anderen, und das ist kaum weniger bedeutsam, lassen sich damit die eingangs genannten erschwerenden Umstände der Skalierung und Rotation fast ohne Zusatzaufwand einarbeiten. Man benötigt hierbei naturgemäß eine Beschreibung von Kantenzügen, die invariant gegenüber Drehungen, Vergrößerungen sowie Verkleinerungen ist.

Somit reduziert sich alles auf den Vergleich von Signalen im gewöhnlichen Sinne. In sehr geradliniger Weise läßt sich das Thema der normalisierten Kreuzkorrelation nochmals, diesmal allerdings für den eindimensionalen Fall aufnehmen. Perfekte Übereinstimmungen gibt es allerdings bei praktisch relevanten Aufgabenstellungen nicht, so daß in dieser oder jener Form immer noch Entscheidungen zu fällen sind.

Selbst wenn man sich absolut sicher ist, daß einem die Lokalisierung eines Templates in einem Bild gelungen ist, bleibt immer noch eine Frage zu klären. Des öfteren sind die Genauigkeitsforderungen nämlich weit höher als die

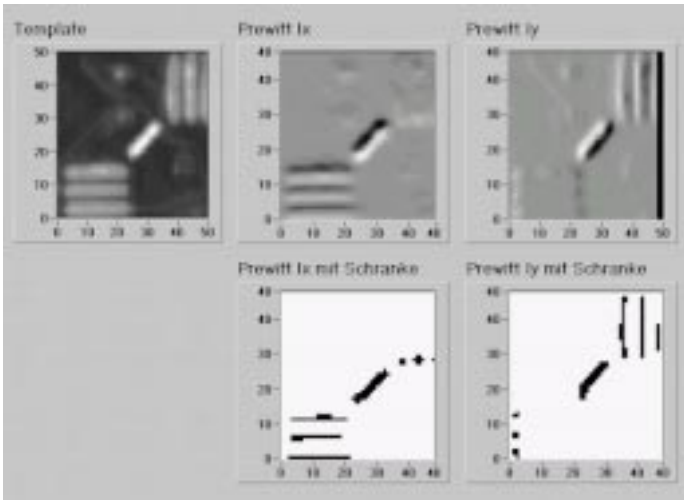

Bild 2. Die Wirkungsweise des Prewitt-Operators anhand des im Bild 1 gezeigten typischen Anwendungsfalles eines Computerboards. Das Template ist genauer untersucht worden. Insbesondere nach Betrachtung der im unteren Teil dargestellten Schrankenbildung kann man die unterschiedliche Behandlung von senkrechten und waagerechten Kanten erkennen.

natürliche Grenze von 1 Pixel. Das mag absurd klingen, da die Auflösung eines Bildes ja exakt 1 Pixel beträgt. Allerdings hilft hier Redundanz wirkungsvoll aus der Klemme. Templates bestehen aus Hunderten bis Tausenden von Einzelwerten, und jeder davon kann (z.B. im Rahmen einer Fittingstrategie) einen Beitrag zur Genauigkeitssteigerung leisten. Hierauf basierend ist man durchaus in der Lage, eine Präzision von 1/10 Pixel in x- und y-Richtung sowie Bruchteile von einem Grad bezüglich der Rotationsgenauigkeit zu erzielen.

Zu guter Letzt steht die Implementierung an, die zumeist eine weitere Herausforderung darstellt. Das trifft insbesondere dann zu, wenn, wie im Falle des Pattern-Matching, die Abarbeitungsgeschwindigkeit hoch sein muß. Auf dem Gebiet des maschinellen Sehens ist ein noch so perfekter, aber langsamer Algorithmus faktisch wertlos, weil der zugrundeliegende reale Prozeß der Taktgeber ist.

Kantendetektion

Die Bestimmung von Kanten in digitalisierten Bildern gehört zu den Standardaufgaben der Bildverarbeitung. Nun sind Kanten ja dadurch gekennzeichnet, daß sie sich aufgrund deutlicher Unterschiede in unmittelbar benachbarten Gebieten eines Bildes ergeben. Das kann aber auch zu Fehlern führen. Auf dieses Kriterium sprechen nämlich nicht nur wirklich vorhandene Kanten an, ebenso gut können verrauschte Komponenten leicht zu einer Fehlinterpretation verleiten. Aus diesem Grund wird man zumeist die Operation der Kantendetektion mit einer passenden zweidimensionalen Filteroperation einleiten. Allerdings wird hierbei der Kern des Filters von der Ausdehnung her eher klein sein. Im anderen Fall läuft man nämlich Gefahr, die wirklichen Kanten zu stark einzuebnen. Wie viele andere Dinge in der Signalverarbeitung, hängt eine geeignete Wahl in erster Linie von der konkreten Situation ab.

An Kantendetektoren besteht in der Bildverarbeitung wahrlich kein Mangel. Zwei der erfolgreichen Vertreter sind mit den Namen Prewitt und Canny verbunden. Schon die Konstruktion des Prewitt-Operators (*Kasten*) läßt erkennen, daß man insbesondere Kanten parallel zu den Bildachsen gut detektieren wird. Weiß man beim Matching-Problem beispielhaft, daß das Template nicht gedreht wurde, dürfte Prewitt das Mittel der Wahl sein. Im allgemeinen Fall hat sich die Kantendetektion gemäß Canny allerdings als leistungsstärker erwiesen. Cannys Algorithmus versucht erfolgreich, auch andere Kantenrichtungen fair zu behandeln. *Bild 2* gibt ein Beispiel für die Wirkungsweise des Prewitt-Operators anhand der Ausgangssituation gemäß Bild 1. Im Mittelpunkt steht hierbei das Template.

Die Ausgaben von Kantendetektionsalgorithmen sind zunächst wieder Grauwertbilder. Allerdings sind die Pixel, die Kanten repräsentieren, so dominant in der Helligkeit, daß man hieraus fast ohne Umwege Kurvenbeschreibungen erzeugen kann. Mit anderen Worten, man erhält Kantenzüge, die durch eine Folge von x- bzw. y-Werten charakterisiert sind.

Codierung der Kantenzüge

Zum besseren Verständnis betrachte man einen einzigen solchen Kurvenzug (*Bild 3*). Im Regelfall sind zunächst einige kosmetische Operationen am Datenmaterial vorzuneh-

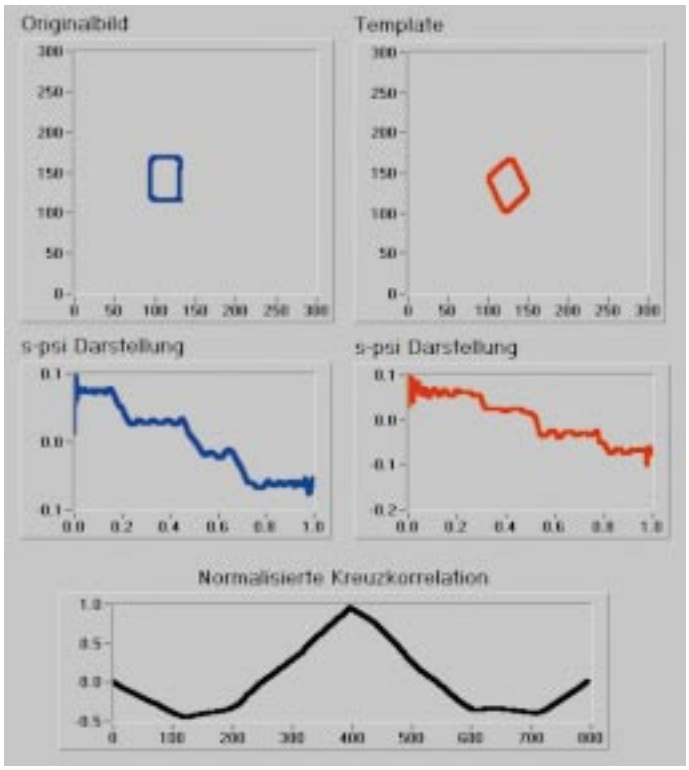

Bild 3. Erfassung und Codierung von Kurvenzügen ist zentrale Aufgabe des hier vorgestellten Pattern-Matching-Algorithmus. Gezeigt ist ein einfaches Beispiel, wobei Original und Template jeweils ein Rechteck darstellen, das nur gedreht wurde. Ferner sieht man die s-ψ-Darstellungen und die normalisierte Kreuzkorrelation. Das Maximum letzterer ist so nahe bei 1, daß ein Treffer faktisch garantiert ist.

men. So kann man etwa Punkte entfernen, aber auch offensichtliche Lücken durch Hinzufügen von geeigneten Pixeln schließen.

Ist das geschehen, notiert man die x- und y-Komponenten des Kantenzuges in der naheliegenden Ordnung, d.h., man beginnt beliebig, läuft dann aber die Kurve ab. Die resultierenden Signale sind im Regelfall nicht störungsfrei, so daß sich Glättungsstrategien wie kubische Splines anbieten. Nun gibt es in der Mathematik den Begriff der natürlichen Parameterdarstellung einer Kurve. Dabei ist einfach für einen bestimmten Punkt der Kurve die zugeordnete Bogenlänge bei entsprechender Winkellage zu ermitteln. Ein perfekter Kreis bildet sich bei dieser Vorgehensweise als gerade Linie ab (*Bild 4*). Wenn man jetzt noch die Bogenlänge auf Einheitsmaß bringt, hat man die Codierungsphase des Kurvenzuges faktisch abgeschlossen. Man bedenke, daß die Analyse des Templates eine ganze Kollektion an solchen codierten Beschreibungen liefert. Die Liste derjenigen des Originalbildes ist gewöhnlich noch länger. Es kommt nun darauf an, Gemeinsamkeiten zwischen Vertretern beider Listen herauszufinden.

Normalisierung und Vergleich

Die gerade erläuterten s-ψ-Darstellungen (Bogenlänge-Winkel) von Kantenzügen (siehe nochmals Bild 3) sind für einen Vergleich verschiedener Exemplare hochgeeignet. Allerdings empfiehlt sich zuvor eine weitere Normalisierungsprozedur, die die Winkellage betrifft. Dabei ist zu bedenken, daß die Bogenlänge s bereits normalisiert vorliegt. Die Normalisierung der Winkelfunktion geschieht durch Mittelwertbildung mit anschließender Nor-

Definition eines wichtigen Kantendetektors

Der Prewitt-Operator stützt sich auf die unterschiedliche Behandlung der beiden Vorzugsrichtungen. Eingesetzt werden zwei Kerne, die durch Rotation auseinander hervorgehen. Diese Operatoren tragen die Namen I_x und I_y, was auf den Zusammenhang mit Ableitungen hinweist:

$$I_x = \begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix}$$

sowie

$$I_y = \begin{matrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{matrix}$$

Im wesentlichen berechnet man demnach den neuen Pixelwert dadurch, daß man die Graustufen der Nachbarn passend summiert und anschließend subtrahiert. Der numerische Aufwand hierfür ist gering, so daß daraus schnelle Verfahren resultieren.

mierung des Energie-Inhaltes (*Bild 5*). Was daraus entsteht, ist eine robuste und invariante Darstellung von Kurvenzügen. Es läßt sich nämlich relativ einfach einsehen, daß gedrehte Kurvenzüge faktisch zur selben normalisierten s-ψ-Darstellung führen werden. Dasselbe trifft zu, wenn die Kurvenzüge durch Skalierungsoperationen auseinander hervorgegangen sind. Ein einziger weiterer Effekt bleibt zu berücksichtigen: Die genannten Darstellungen sind nicht vollständig äquivalent, weil man ja einen Freiheitsgrad noch nicht ausgeräumt hat. Dieser bezieht sich auf den zufällig gewählten Anfangspunkt bei der Beschreibung des Kurvenzuges. Es sei bemerkt, daß die meisten bei Pattern-Matching-Problemen auftretenden Kurvenzüge geschlossen sind, d.h., es gibt weder einen wohldefinierten Anfang noch ein solches Ende.

Diese Unbestimmtheit führt dazu, daß man keinesfalls normalisierte Signale direkt vergleichen kann, vielmehr muß eine potentielle Verschiebungsoperation zwischen zwei Signalen einkalkuliert werden.

Nun bietet aber die eindimensionale Signalverarbeitung die Medizin, die das kurieren kann. Gemeint ist die Kreuzkorrelation. Im konkreten Falle geht es aus den weiter oben beschriebenen Gründen nicht um die übliche Kreuzkorrelation, vielmehr um die viel einfacher zu handhabende normalisierte Variante. Erreicht die Kreuzkorrelation als Signal gesehen einen Spitzenwert in der Nähe von 1, so sind beide Signale weitgehend identisch und nur gegeneinander verschoben. Die Winkellage der Rotation steht im direkten Bezug zur Position dieses Maximums der Kreuzkorrelation. Allerdings muß man von diesem Wissen – aus noch zu erläuternden Gründen – eigentlich gar keinen Gebrauch machen.

Akzeptanz von Kandidaten

Man wird also vernünftigerweise von der Liste der Kantenzüge sowohl des Templates als auch des aktuell vorliegenden Bildes die normalisierten s-ψ-Darstellungen aufbauen. Da das Template voraussetzungsgemäß bereits vor der eigentlich heißen Phase bekannt ist, fällt dieser Teil nicht weiter ins Gewicht. Direkte Vergleiche zwischen all diesen Paaren liefern Hinweise auf Übereinstimmungen. Hierbei sind zwei Hauptstrategien zu verfolgen:

Hat man ein Paar gefunden, das einen maximalen Kreuzkorrelationswert liefert, der sehr dicht an 1 herankommt, so ist man der Lösung bereits sehr nahe und kann sich eigentlich dem Punkt „Genauigkeit der Lösung“ zuwenden. Allerdings ist dieser Idealzustand nur selten zu erleben, und andererseits kann man selbst in diesem Fall etwas zur weiteren Gütesteigerung tun. Gemeint ist damit, daß man bei einem guten Korrelationswert (etwa > 0,8) einen berechtigten Verdacht haben kann, daß man einen Kandidaten aufgespürt hat. Ein solch hoher Wert kann rein zufällig nicht entstehen. Man sollte den oder die Kurvenzüge im Template betrachten und entsprechende Pendants im Bild suchen. Natürlich ist auch das letztlich Sache der normalisierten Kreuzkorrelation. Ein weiterer Vorteil liegt auf der Hand: Da das Template eher von geringerer Größe ist, kann das Pendant nicht weit vom ersten Treffer entfernt sein.

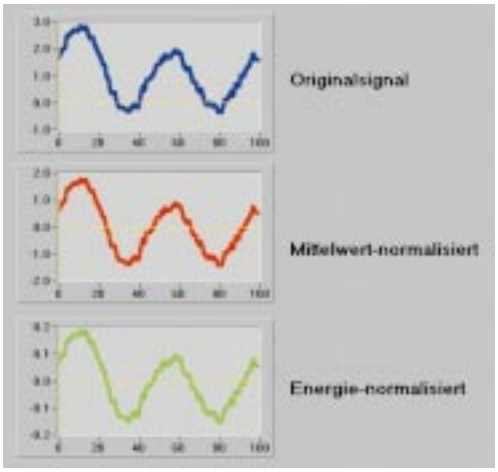

Bild 5. Die eingesetzte Normierungsstrategie ist ein Zweistufenprozeß. Zunächst wird der Gleichanteil beseitigt, woran sich die energetische Normierung anschließt.

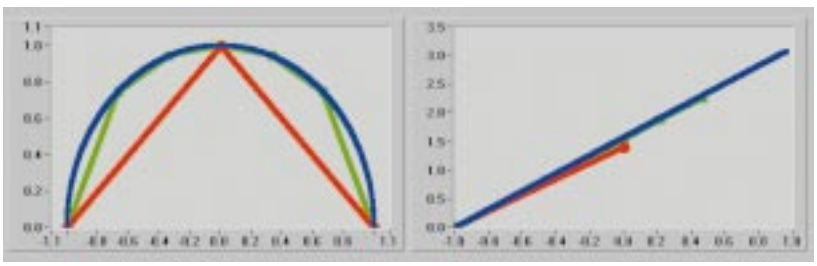

Bild 4. Die Codierung von Kurvenzügen beruht auf der sogenannten Bogenlänge. Kreise werden hierbei als gerade Linien codiert. Wie man rechts sieht, entsteht die Gerade bereits dann, wenn man mit nur sehr wenigen Punkten den Kreis approximiert. Die einzelnen Kurven bestehen aus 3, 7 und 1000 Elementen. Bei Bildverarbeitungsroutinen dürfte man in der Größenordnung von 100 bis 400 Punkten per Kurvenzug liegen.

Durch die letzten Sätze wird klar, weshalb man die Ergebnisse der normalisierten Kreuzkorrelation nicht nutzen muß, um eventuelle Rotationsbewegungen zwischen Template und Vorkommen im Bild zu bestimmen. Hierfür eignen sich die Nachbarschaftsbezüge zwischen verwendeten Kurvenzügen im Template und im aktuellen Bild weitaus besser. Je mehr von diesen Paarbeziehungen hergestellt werden können, um so präziser wird die Abschätzung für den Rotationswinkel sein.

Wenn man auf diese Art und Weise mit hoher Wahrscheinlichkeit einen echten Treffer erzielt hat, empfiehlt sich trotzdem noch ein eigentlicher Test, ob das Template mit dem aktuellen Bild tatsächlich in allen Details übereinstimmt. Das kann z.B. wirklich pixel-weise erfolgen, wobei man auch dabei vorzugsweise mit den normalisierten Varianten arbeiten sollte. Einziger Einwand wäre, daß sich solche Berechnungen vollkommen in der zweidimensionalen Welt abspielen und somit die Tendenz haben, über Gebühr Rechenzeit zu beanspruchen.

Genauigkeit der Lösung

Die bisher geschilderte Lösungsstrategie wird normalerweise Genauigkeiten von ±1 Pixel und ±1 Grad bezüglich der Rotation liefern. Ist man auf qualitativ bessere Resultate angewiesen, muß man andere Verfahren ins Spiel bringen. Genannt wurden bereits geeignete Fittingstrategien, wobei die Parameter aus kleinen Änderungen dx, dy und dϕ bestehen. Die ersten beiden Parameter verschieben den Ort in x- und y-Richtung, der dritte Wert steht für den Winkel. Da alles im Subpixel- bzw. Subwinkel-Bereich abläuft, muß darüber hinaus passend (zumeist linear) interpoliert werden.

Man kann aber auch tiefer in das Reservoir der Signalverarbeitung greifen und Steuerstrategien zum Einsatz bringen. Dabei steht man auf soliden Füßen, weil die zugeordnete Theorie gut ausgebaut ist. Das einzige Manko ist vielleicht, daß Begriffe wie Setpoint oder PID-Regler auf den ersten Blick so schlecht zu Belangen der Bildverarbeitung zu passen scheinen. Dem ist aber keinesfalls so.

Man muß als Setpoint nicht einen speziellen Wert im Auge haben. Das gesamte Template ist das Ziel, das es zu erreichen gilt. Wie die Heizung in einem leicht unterkühlten Raum, muß der Steueralgorithmus versuchen, aus einer relativ guten Anfangslage heraus die perfekte Situation zu erreichen. Das geht überraschend gut und darüber hinaus sogar schnell. Im Grunde genommen genügen hierfür einige wenige Skalarprodukte je Steuerschleife. Ferner ist auch die Anzahl dieser Schleifen bis zur ausreichend guten Übereinstimmung gering.

Die eigentliche Arbeit kann nämlich bereits in der Vorbereitungsphase und basierend allein auf Daten des Templates erfolgen. Vielleicht wird dadurch auch etwas klarer, weshalb man für die genaue Untersuchung des Templates eine Zeitspanne im Sekundenbereich als vernünftig ansieht, während das Pattern-Matching im engeren Sinne im Takt von einigen 100 ms erfolgen sollte. Die unerwartete Verbindung von Bildverarbeitung und Kreuzkorrelation ist vielleicht so etwas wie ein Highlight des Gesamtalgorithmus.

Implementierungsfragen

Eine Algorithmenentwicklung, wie die hier vorgestellte, erfolgt oftmals zunächst als reiner Prototyp. Das kann mit mathematischen Mitteln geschehen oder auch basierend auf Prototyping-Tools. Gemeinsam ist diesen Vorgehensweisen, daß die Geschwindigkeit der Abarbeitung kein Thema ist. Das sieht anders aus, wenn man den Schritt zur anvisierten Anwendung oder gar zum Produkt geht. Ohne Zweifel ergibt sich daraus eine neue Herausforderung, die einen ebensolchen Reiz auf den Entwickler ausüben kann, wie die Ideenfindung selbst. Im Falle des Problems des Pattern-Matching sind es insbesondere folgende Bemerkungen, die verallgemeinerungsfähig sind:

Vieles kann für eine effiziente Implementierung getan werden, wenn man passende Datentypen aufbaut. Solange es irgendwie geht, sollte man mit Integer-Formaten agieren, vorzugsweise mit 16 bit Länge. Letzteres resultiert daraus, daß man zur Erreichung höchster Geschwindigkeiten auf einem PC letztlich auf MMX setzen muß. Diese Si-

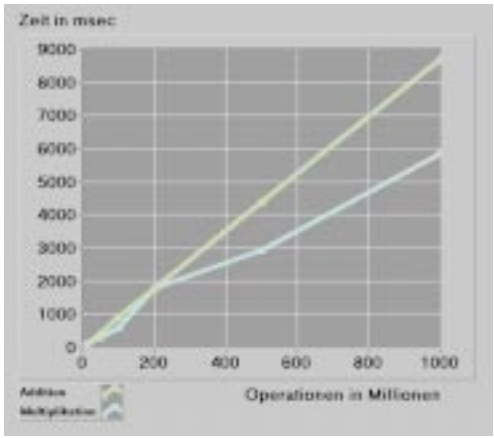

Bild 6. Bei den neueren Prozessorgenerationen besteht kein prinzipieller Unterschied mehr zwischen Additionen und Multiplikationen. Viele der existierenden hocheffizienten Algorithmen gehen allerdings von einem Ungleichgewicht aus. Die Diagramme zeigen einige Resultate von Benchmarks (Pentium II, 350 MHz, ohne MMX, Gleitkomma-Format 32 bit).

tuation hat sich übrigens mit der Freigabe von Intels Pentium III (Katmai) verbessert, so daß 32-bit-Gleitkomma-Typen ebenfalls zum Einsatz kommen können.

Fast immer sind es relativ elementare Operationen, die zum Zeitfresser werden. So können die normalisierten Kreuzkorrelationen durchaus als Nadelöhr des beschriebenen Algorithmus bezeichnet werden. Wenn das so ist, dann lohnt sich fast jeder Aufwand, um diese kritische Stelle zu entschärfen. Neben MMX kann man in diesem Zusammenhang auch an trickreiche Implementierungen der Kreuzkorrelation denken, die sehr wohl existieren. Die damit erzielten Rechenzeitgewinne können signifikant sein.

Eine erfolgreiche Implementierung setzt Wissen über die modernen Prozessorgenerationen voraus. So ist die Behandlung von Cache-Strukturen nicht unproblematisch, und man kann bei ungeschickter Programmierung und ohne es zu wissen für ständige Nachladeoperationen sowohl der Daten- als auch des Instruktions-Caches sorgen. Das fängt schon damit an, daß der Level-1-Cache mit 32 byte aligned werden muß, falls man höchste Verarbeitungsleistung erwartet.

Moderne Prozessoren sind faszinierend schnell (*Bild 6*). Das bedeutet aber keineswegs, daß man nicht Zeit in die Code-Optimierung stecken sollte. Nach Erfahrung des Autors bringt man es von der ersten lauffähigen C-Version bis zum endgültigen Feinschliff auf 10 % der Rechenzeit. Dieser Faktor mag als zu groß erscheinen, hat sich in zahlreichen Anwendungen allerdings bestätigt. Man bedenke, daß hierunter auch Dinge wie optimierte Datentypen fallen. Sieht man von MMX ab, scheint sich ein Umschreiben von C in Assembler allerdings kaum zu lohnen. Moderate Zeitgewinne (nicht besser als Faktor 1,4) werden vollkommen von der schlechten Wartbarkeit und Verständlichkeit ruiniert. Auch Algorithmen unterliegen nun einmal einem ständigen Entwicklungsprozeß.

Interessant sind in diesem Zusammenhang die DSPs und FPGAs. Deren Leistungsstärke ist unbestritten, jedoch kommt insbesondere bei Bildverarbeitungsproblemen die beschränkte Bandbreite zur Wirkung. FPGAs scheinen auf lange Sicht in dieser Hinsicht mehr Erfolg zu versprechen, da sie den Vorteil der inhärenten und massiven Parallelität mit sich bringen. Die atemberaubende Geschwindigkeit, mit der sich die Rechenleistung der Standardprozessoren verbessert, läßt allerdings auch noch eine ganz andere Option zu. Bleibt abzuwarten, ob nicht letztere die ganz großen Sieger der Signalverarbeitung sein werden.

Alternative Strategien

So geschlossen die hier propagierte Lösung für das Pattern-Matching auch aussehen mag, man sollte immer die Frage nach besseren Verfahren stellen. In erster Linie ist die Performance gemeint. Endziel ist ein Echtzeit-Verhalten, das, basierend auf 30 Bildern pro Sekunde, demnach in 30 ms die ganze Angelegenheit vom Tisch haben muß. Nur 15 ms wäre natürlich noch besser, weil ja noch Zeit für abgeleitete Aktionen sein sollte.

Die Wichtigkeit und der kommerzielle Erfolg einer solchen Lösung können als sicher gelten. Vielleicht fühlt sich der eine oder andere Leser hierdurch herausgefordert. Eines dürfte offenbar sein: Eine derartige Lösung muß auf einer geschickten Kombination vieler Teilthemen der Signalverarbeitung beruhen. Aber das ist ja gerade das eigentliche Abenteuer. *gs*

Literatur

[1] *Wenzel, L.*: Digitale Signalverarbeitung ist keine Hexerei, Teil 1. *Elektronik* 1998, H. 23, S. 96ff.
[2] *Wenzel, L.*: Digitale Signalverarbeitung ist keine Hexerei, Teil 2. *Elektronik* 1998, H. 25, S. 86ff.
[3] *Wenzel, L.*: Digitale Signalverarbeitung ist keine Hexerei, Teil 3. *Elektronik* 1999, H. 1, S. 52ff.
[4] *Wenzel, L.*: Digitale Signalverarbeitung ist keine Hexerei, Teil 4. *Elektronik* 1999, H. 3, S. 48ff.
[5] *Wenzel, L.*: Digitale Signalverarbeitung ist keine Hexerei, Teil 5. *Elektronik* 1999, H. 5, S. 60ff.
[6] *Wenzel, L.*: Digitale Signalverarbeitung ist keine Hexerei, Teil 6. *Elektronik* 1999, H. 7, S. 58ff.
[7] *Wenzel, L.*: Digitale Signalverarbeitung ist keine Hexerei, Teil 7. *Elektronik* 1999, H. 9, S. 62ff.
[8] *Wenzel, L.*: Digitale Signalverarbeitung ist keine Hexerei, Teil 8. *Elektronik* 1999, H. 11, S. 62ff.
[9] *Wenzel, L.*: Digitale Signalverarbeitung ist keine Hexerei, Teil 9. *Elektronik* 1999, H. 13, S. 91ff.
[10] *Wenzel, L.*: Digitale Signalverarbeitung ist keine Hexerei, Teil 10. *Elektronik* 1999, H. 15, S. 64ff.
[11] *Wenzel, L.*: Digitale Signalverarbeitung ist keine Hexerei, Teil 11. *Elektronik* 1999, H. 17, S. 62 ff.
[12] *Wenzel, L.*: Digitale Signalverarbeitung ist keine Hexerei, Teil 12. *Elektronik* 1999, H. 19, S. 60 ff.
[13] Die im Rahmen dieser Serie angegebenen Programme stehen zum Download bereit unter: *www.natinst.com/germany*

Dr. Lothar Wenzel ist gebürtiger Berliner und hat Mathematik und Informatik in Greifswald und Dresden studiert. Nach Tätigkeiten in einem Kernkraftwerk und bei der BASF AG, Ludwigshafen, beschäftigt er sich z.Zt. bei National Instruments in Austin, Texas, mit dem Design von Algorithmen auf den Gebieten Simulation, Regelung, Mathematik und Bildverarbeitung.