Echtzeitfähige Analyse hochfrequenter Ultraschalldaten zur Überwachung von Fertigungsprozessen

Quy Raven LUONG [1], Thomas SCHLECH [1], Florian F. LINSCHEID [1], Marcel ACHZET [1], Markus G. R. SAUSE [1]
[1] Universität Augsburg, Augsburg

Kontakt E-Mail: quy.luong@mrm.uni-augsburg.de

Kurzfassung. Für eine effiziente Entwicklung von Software zur Auswertung großer Datenmengen, die typischerweise bei der Überwachung von Prozessen anfallen, wird das sogenannte Ultra High Ultrasonics Framework, kurz UHU, entwickelt [1]. Ziel dieses Frameworks ist es, den Auswertungsprozess bestehend aus der Datenerfassung, Feature-Extraktion und Datenfusion für Daten verschiedener Art zu vereinheitlichen. Dadurch soll sowohl die Entwicklung von Auswerteroutinen und Auswertungssoftware erleichtert, als auch die Übertragbarkeit ähnlicher Konzepte für verschiedene Anwendungen ermöglicht werden. Das Framework übernimmt hierbei die Verwaltung und das Auslesen der Dateien, das Vorverarbeiten der Signale sowie das Extrahieren von Signalcharakteristiken, welche vom Endbenutzer selbst definiert werden können. Das Framework wird bereits für die Auswertung von Körperschallsignalen, Daten von Kraft-Momenten-Sensoren sowie Beschleunigungs- und Maschinendaten eingesetzt. Es ist dabei so ausgelegt, dass es die Möglichkeit bietet, beliebige weitere Messsysteme und Messprinzipien für verschiedene Anwendungen entsprechend kombinieren zu können. Im Kontext der Prozessüberwachung ist insbesondere die Echtzeitauswertung der Daten von großem Interesse, die gerade bei hochfrequenten Daten wie z.B. aus der Körperschallanalyse eine Herausforderung darstellt. Daher wurden in dieser Arbeit Multithreading- und Multiprocessing-Techniken angewendet, um eine echtzeitfähige Auswertung hochfrequenter Daten, wie den Körperschallsignalen, zu ermöglichen. Hierzu wurden verschiedene Strategien basierend auf der Parallel Computing Toolbox von MATLAB getestet. Erhöhte Rechenzeiten aufgrund des zusätzlichen Kommunikationsaufwands und Optimierungen im Auswertungsprozess wurden dabei berücksichtigt. Durch das Parallel Computing können mehrere Datensätze gleichzeitig ausgewertet und folglich der Gesamtdurchsatz an verarbeiteten Daten drastisch erhöht werden.

1. Einführung

Die Neu- oder Weiterentwicklung von Systemen zur Prozess- oder Zustandsüberwachung stellt häufig einen sehr zeitaufwändigen und kostspieligen Prozess dar, da für die Entwicklung solcher Systeme die passenden Sensoren gewählt und eingesetzt werden müssen sowie eine darauf abgestimmte Auswertesoftware geschrieben werden muss. Durch eine Standardisierung des Auswertungsprozesses, welcher die Datenaufnahme, die Signalvorverarbeitung sowie die Feature-Extraktion des Signals beinhaltet, soll die

Entwicklung neuer Überwachungssysteme vereinfacht und beschleunigt werden. Hierzu wurde ein Framework unter dem Namen Ultra-High-Ultrasonics, kurz UHU, entwickelt, welches diese Standardisierung umsetzen soll [2].

Des Weiteren stellt sich als Folge der zunehmenden Digitalisierung und Automatisierung vieler Fertigungsprozesse die Herausforderung, wie mit großen Mengen erfasster Daten, kurz Big Data [3], umzugehen ist und wie Überwachungssysteme inklusive deren Datenverarbeitung den Ansprüchen verschiedener Prozesse bezüglich deren Echtzeitfähigkeit gerecht werden können. Letzteres ist hierbei von besonderem Interesse, da dies nicht nur schnelle Adressierungen und datenbasierte Lösungen von Problemen erlaubt, sondern dem Nutzer auch ein besseres Verständnis des jeweiligen Prozesses durch Überwachung von Echtzeit-Vorgängen ermöglicht. Außerdem soll die Echtzeitfähigkeit mittelfristig die Möglichkeit bieten, die Informationen des Überwachungssystems zurück in die Anlagensteuerung oder Regelung des Prozesses zu bringen. Die großen Mengen auszuwertender Daten erschweren jedoch die echtzeitfähige Analyse, was bei komplexeren, multisensorischen Systemen ein noch größeres Problem darstellt. Um die Echtzeitfähigkeit solcher Systeme zu gewährleisten, werden deshalb Techniken der parallelen Verarbeitung mittels der Parallel Computing Toolbox von MathWorks [4] angewendet. Hiermit sollen die in Echtzeit verarbeitbare Datenmengen erhöht werden. Dies wird in dieser Arbeit anhand mehrerer kontinuierlich erfasster Ultraschalldaten demonstriert.

2. Ultra-High-Ultrasonics-Framework

Das Ultra-High-Ultrasonics-Framework dient der Standardisierung der Auswertung verschiedenster Signale und wurde von Linscheid und Sause [2] für die Auswertung von Signalen der Schallemission, Prüfung mit geführten Wellen, Körperschall- sowie Modalanalyse in der Programmiersprache MATLAB designt. Das Framework ist so ausgelegt, dass die Auswertekonzepte auf weitere Messsysteme erweitert und generalisiert werden können. Durch die Erweiterung können auch Beschleunigungsdaten, Maschinendaten und Signale von Kraft-Momenten-Sensoren eingelesen und ausgewertet werden. Das Framework beschränkt sich daher nicht nur auf die Auswertung von ultraschallbasierten Daten.

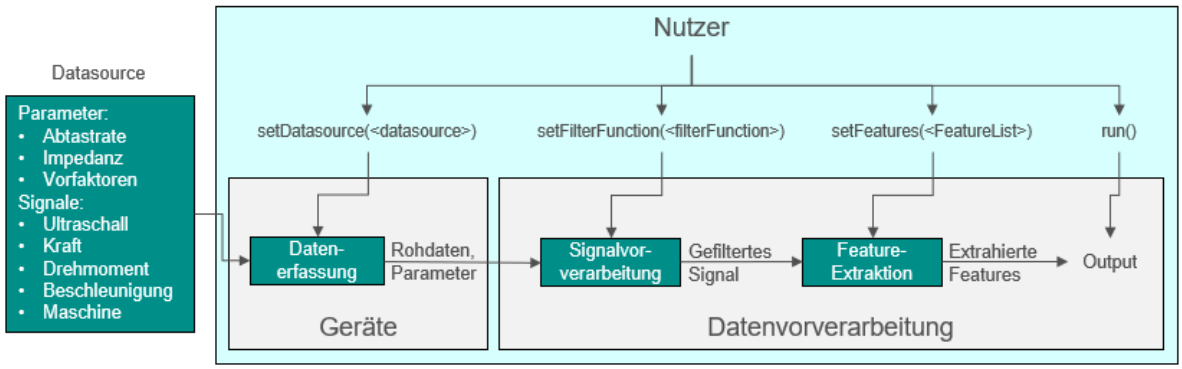

Abb. 1. Struktur des Frameworks

Die Struktur des Frameworks setzt sich aus drei Ebenen zusammen und ist in Abbildung 1 dargestellt. Es unterteilt sich in die Geräte-, Datenverarbeitungs- und Nutzerebene, welche zueinander wichtige Schnittstellen definieren und implementieren.
Die Geräteebene ist dabei verantwortlich für das Aufnehmen bzw. Einlesen der Daten und stellt diese anschließend der Datenvorverarbeitung bereit. Sie übermittelt zusätzlich relevante Informationen wie Abtastrate, Impedanz, Verstärkungs-, Umrechnungsfaktoren und weitere

Parameter an die Datenvorverarbeitung, die für die Auswertung benötigt bzw. vorausgesetzt werden.
Die Datenvorverarbeitungsebene übernimmt aktuell die Signalvorverarbeitung und Feature-Extraktion der von der Geräteebene übermittelten Daten, um die Signale für die nachfolgende Verarbeitung z.B. mittels Methoden der künstlichen Intelligenz zu konditionieren. Innerhalb dieser Ebene können zusätzliche Funktionen für die Feature-Extraktion nach Bedarf vordefiniert werden.
Die Nutzerebene erlaubt, die gesamte Auswertepipeline nach den eigenen Wünschen zu konfigurieren und so dem jeweiligen Prozess anzupassen. Hierfür ist dem Endnutzer die Möglichkeit gegeben, die benötigten Features entsprechend zu wählen, die als Funktion in der Datenvorverarbeitung vordefiniert wurden, und zusätzlich eine beliebige Filterfunktion für die Signalvorverarbeitung zu definieren. Auch die Auswahl und Kombination verschiedener Sensoren kann in der Nutzerebene durchgeführt werden.
Die Entwicklung eines Überwachungssystems für einen neuen Prozess beschränkt sich somit im Wesentlichen auf die Nutzerebene. Sollen neue Sensoriken in das System eingebaut werden, so findet die Entwicklung zusätzlich auf der Geräteebene statt. Im Gegensatz dazu ist die Datenvorverarbeitung in sich abgeschlossen und beinhaltet fest definierte Schnittstellen zu der Geräteebene, sodass die Datenvorverarbeitungsebene für beliebige Prozesse und Sensoren ohne oder mit geringen Veränderungen beibehalten werden kann. Der Entwicklungsaufwand begrenzt sich daher maßgeblich auf die Geräte- und Nutzerebene und wird somit im Vergleich zu einer kompletten Neuentwicklung verringert. Dadurch wird die Softwareentwicklung für ein Überwachungssystem erleichtert und beschleunigt.

In Hinblick auf Echtzeitanalysen wird das Framework nun mit Methoden der parallelen Datenverarbeitung ausgestattet, um die Berechnungszeit zu verringern bzw. die Menge auswertbarer Daten in einem gegebenem Zeitrahmen so zu erhöhen, dass Echtzeitanforderungen an das System erfüllt werden können.

3. Technik

3.1 Parallel Computing

Im Rahmen der Prozessüberwachung spielt die Echtzeitauswertungen für viele Prozesse eine bedeutende Rolle und wird auch in Zukunft weiter an Relevanz gewinnen. Durch die Entwicklung von immer komplexeren und ausgereifteren Prozessen reichen für die Prozessüberwachung relativ einfache Sensorsysteme nicht immer aus und erfordern stattdessen ein komplexeres Netzwerk an Sensoren. Zeitgleich steigen aber auch die Anforderungen an solcherlei Systeme, da vermehrt eine echtzeitfähige Nutzung der Daten angestrebt wird. Die dadurch erhöhten Anforderungen und Mengen erfasster Daten stellen zugleich jedoch eine Herausforderung dar, da diese, ohne Datenverlust in Kauf zu nehmen, in begrenzter Zeit ausgewertet werden müssen.

Um eine echtzeitfähige Analyse zu erreichen, müssen einige Punkte berücksichtigt und von entsprechenden Methoden Gebrauch gemacht werden [5]. Dazu zählt unter anderem die Parallelisierung, die in MATLAB in der von MathWorks entwickelten Parallel Computing Toolbox enthalten ist und im Folgenden eingesetzt wird. Die grundlegende Idee ist hierzu in Abbildung 2 dargestellt. Programme, die in MATLAB geschrieben werden, werden vorwiegend in serieller Verarbeitung durchgeführt. Der gesamte Rechenaufwand wird in solchen Fällen von nur einem CPU-Kern abgearbeitet, während die restlichen Kerne untätig bleiben. Durch Methoden der parallelen Verarbeitung lässt sich ein großer Teil des Arbeitsaufwands auf die zusätzlichen Kerne verteilen, wodurch sich eine insgesamt höhere Auslastung der CPU-Kerne erreichen lässt und infolgedessen die Rechenzeit verringert wird

bzw. eine Verarbeitung von noch größeren Datenmengen in gleicher Zeit erreicht werden kann.

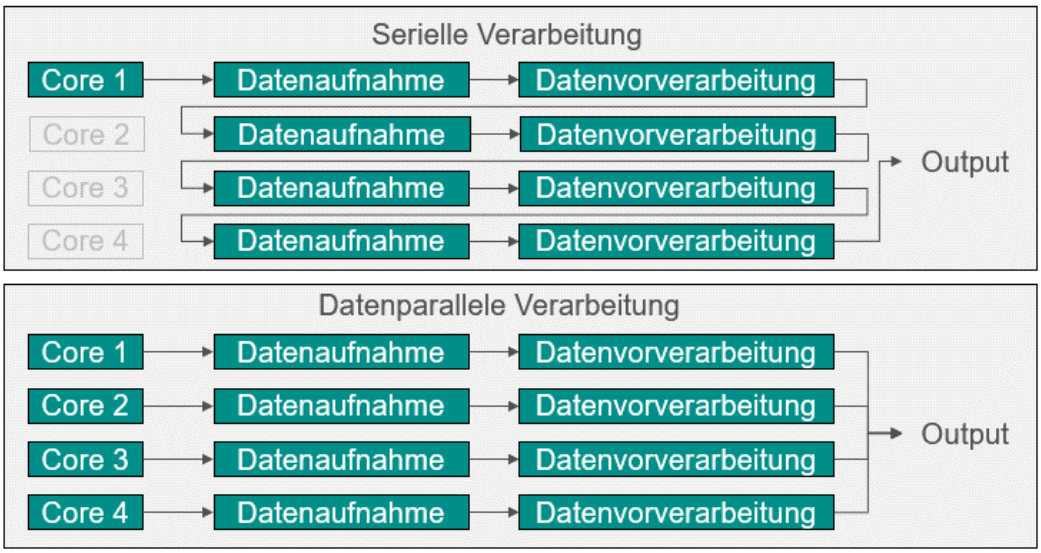

Abb. 2. Serielle und datenparallele Verarbeitung von Signalen. Bei der seriellen Verarbeitung wird der gesamte Arbeitsaufwand auf einen Kern beschränkt. Die datenparallele Verarbeitung ermöglicht die Verteilung des gesamten Arbeitsaufwands auf mehrere Kerne.

Für die Implementierung der parallelen Verarbeitung existierten in MATLAB zum einem das sogenannte Multithreading und zum anderem das Multiprocessing. Die in Abbildung 2 dargestellte Methode verwendet das Multiprocessing zur Umsetzung einer datenparallelen Verarbeitung, sodass jeder Kern die erstellte Auswertepipeline auf seine erhaltenen Daten unabhängig von den restlichen Kernen anwendet.

Ansätze mit Multithreading zur Umsetzung einer aufgabenparallelen Verarbeitung wurden ebenfalls bei der Datenvorverarbeitung bzw. bei der Feature-Extraktion getestet. Die Kerne übernehmen hierbei nur einen bestimmten Satz an Befehlen, sodass eine Datenmenge von allen Kernen gleichzeitig nach Features extrahiert wird. Dieser Ansatz wurde am Ende jedoch verworfen, da sich keine Verbesserungen in der Rechenzeit ergaben. Grund ist eine insgesamt geringere Parallelisierbarkeit der gesamten Auswertepipeline und der damit verbundene Kommunikationsoverhead, welcher beim Initiieren einer parallelen Verarbeitung stets generiert wird und das Programm zusätzlich Ressourcen kostet. Dieser Overhead wird erst durch das Parallelisieren eines genügend hohen Rechenaufwands gerechtfertigt, was für die Feature-Extraktion allein nicht der Fall ist. Aus diesem Grund wurde ein datenparalleler Ansatz mit Multiprocessing gewählt.

Die alleinige Verarbeitung der Daten stellt allerdings an dieser Stelle noch keine Analyse dar, da die Ergebnisse dem Anwender bzw. dem Client zusätzlich noch zur Verfügung gestellt werden müssen. Das Multiprocessing ist allerdings mit zusätzlichem Aufwand verbunden, der explizit berücksichtigt bzw. optimiert werden muss. Die Ursache des Aufwands ist, dass die eingesetzten Kerne ihren jeweils eigenen Speicher reserviert bekommen, auf welche der Client keinen direkten Zugriff hat. Die Ergebnisse der Datenverarbeitung müssen daher vom Kern zum Client als weiteren Schritt konkret kommuniziert werden, wodurch zusätzlicher Kommunikationsoverhead entsteht. Erst nach dieser Kommunikation können die Ergebnisse im Client schließlich datenfusioniert für anschließende Aktionen wie Visualisierungen, Einspeisung in Diagnose- und Prognosemodelle oder zur Rückführung in die Prozessanlage genutzt werden.

3.2 Bewertung der Echtzeitfähigkeit

Die Anforderungen von Echtzeitsystemen sind in der Regel prozessabhängig. Im Rahmen dieser Arbeit wird die Echtzeitfähigkeit so definiert, dass das System in der Lage ist, die von den Sensoren erfassten Daten innerhalb einer vorgegebenen zeitlichen Anforderung zu verarbeiten, wobei die Datenverarbeitung ohne weitere Zwischenschritte direkt nach der Datenaufnahme erfolgen soll. In diesem Fall sind die Anforderungen so gewählt, dass die Berechnungszeit eines Datenblocks nicht länger dauern darf als der Zeitabstand zwischen zwei Datenblöcken, die vom Sensor aufeinanderfolgend erfasst werden. Hierdurch soll sichergestellt werden, dass die Datenvorverarbeitung stets vor der nächsten Datenaufnahme bereits abgeschlossen ist und folglich kein Datenverlust für eine weiterhin funktionierende echtzeitfähige Verarbeitung in Kauf genommen werden muss.

Um eine Analyse noch vor der ersten Anwendung als echtzeitfähig zu bewerten, wird die Auswertepipeline auf bereits lokal abgespeicherte Daten angewendet. Außerdem werden die Ergebnisse nach jedem verarbeiteten Datenblocks auf der Festplatte für zukünftige Zwecke abgespeichert. Da die Daten direkt in MATLAB eingelesen werden können und nicht erst von einem Messgerät mit einer fixen Rate an den Rechner gesendet werden müssen, verkürzt sich die gesamte Rechenzeit aufgrund wegfallender Wartezeiten. Endet also die Auswertung der gesamten Datenmenge in kürzerer Zeit als deren zeitliche Signallänge, so deutet dies auf eine Echtzeitfähigkeit der Analyse hin. Im anderen Fall ist sie auszuschließen, da die Ergebnisse nicht rechtzeitig berechnet und an den Client übermittelt werden können.

Die Echtzeitfähigkeit der Analyse wurde im Folgenden anhand mehrerer Körperschallsignale im Ultraschallbereich getestet, welche an dieser Stelle Signale simulieren sollen, die typischerweise bei Fertigungsprozessen wie beispielsweise dem Bohren/Fräsen mit einer CNC-Maschine oder bei beliebigen anderen Prozessen anfallen können und mit Hilfe von Ultraschallsensoren kontinuierlich erfasst werden. Zur Bewertung der Echtzeitfähigkeit wurden mehrere solcher Signale mit eine Abtastrate von 1 MHz und einer zeitlichen Gesamtlänge von 200 s erstellt bzw. aufgenommen. Für die Messung wurde hierbei das PROfile Ultrasonics-Messsystem der Firma BCMtec GmbH verwendet, wobei die Messwerte als int16-Werte in einer Binärdatei abspeichert wurden. Diese Signale bzw. Dateien wurden anschließend abschnittsweise bzw. in Form von Datenpaketen mit einer zeitlichen Länge von 0,1 ms chronologisch eingelesen, vorverarbeitet und kontrolliert an den Client zurückgesendet, um die Ergebnisse zu visualisieren. Die Berechnungszeit zur gesamten Analyse des Signals wurde abschließend erfasst und ist in Abbildung 3 beispielhaft gegenübergestellt. An dieser Stelle ist anzumerken, dass bei der Datenverarbeitung lediglich eine absolute Datenmenge betrachtet wird. Die Datenmenge kann daher beispielsweise auch als ein Signal mit einer Gesamtdauer von 40 s bei einer Abtastrate von 5 MHz gesehen werden. Sofern die Berechnungszeit die Gesamtsignaldauer abhängig einer festgelegten Abtastfrequenz nicht übersteigt, ist ein echtzeitfähiges System erstmal nicht auszuschließen.

Analysiert wurden bis zu sechs Körperschallsignale, wobei die Berechnungen mit einem sechskernigen AMD-Ryzen-5500U Prozessor durchgeführt wurde. Zu sehen ist, dass die Berechnungszeit mit jedem weiteren Körperschallsignal zunimmt. Beim Multiprocessing nehmen die Berechnungszeiten mit jedem weiteren Signal allerdings langsamer zu als bei der seriellen Verarbeitung. Im Idealfall würde man erwarten, dass die Zeiten konstant bleiben. Doch durch den bereits erwähnten Kommunikationsoverhead ist eine Zunahme der Berechnungszeit unausweichlich und führt im schlimmsten Fall dazu, dass die parallele Verarbeitung trotz der höheren CPU-Auslastung insgesamt langsamer abläuft als der serielle Prozess. Dies erfolgt in diesem Fall, wenn die Ergebnisse unmittelbar nach jeder Verarbeitung eines Datenpakets zurückgesendet werden. Um dieses Problem zu beheben, wird daher diese Rücksenderate der Ergebnisse kontrolliert reduziert, indem die Ergebnisse vom jeweiligen Kern temporär zwischengespeichert und erst nach einer festgelegten Anzahl

an verarbeiteten Datenpaketen von ihm an den Client schließlich zurückkommuniziert werden. Dadurch lässt sich der Kommunikationsoverhead enorm reduzieren und Multiprocessing ermöglicht geringere Berechnungszeiten als die serielle Verarbeitung. Zu berücksichtigen ist jedoch, dass die Daten für die Analyse im Client abgelegt sein müssen. Eine Einschränkung der Kommunikation ist daher auch mit einer Abnahme der Analyserate verbunden, wobei dies für jeden Prozess einzeln zu bewerten ist. Für einfache bzw. langsame Anwendungen wie das Visualisieren der Auswertungen kann dies beispielsweise vernachlässigt werden. Für andere Anwendungen, die sehr schnelle Prozessantworten in hoher Frequenz erwarten, kann dies hingegen möglicherweise zum Problem werden.

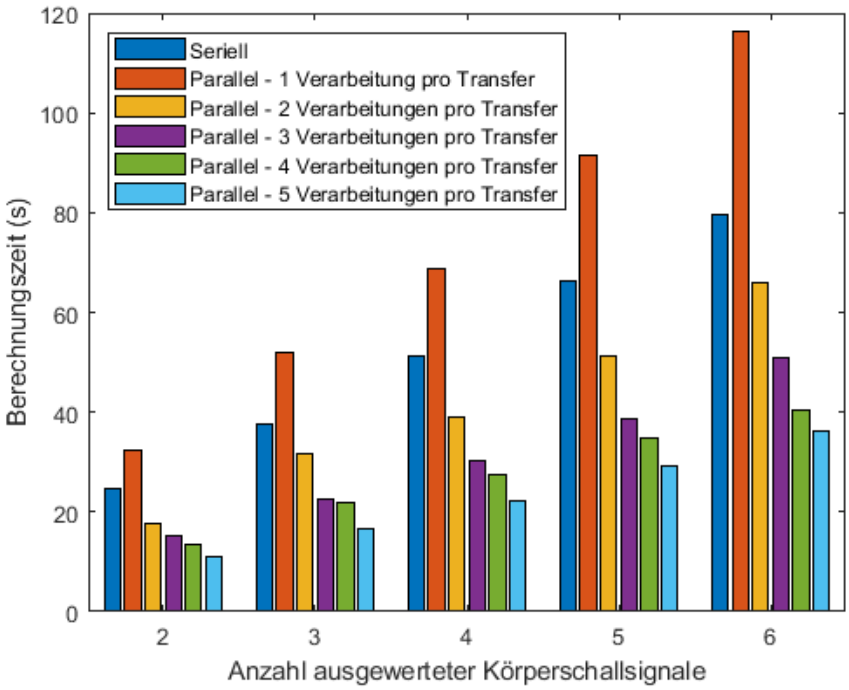

Abb. 3. Vergleich der Berechnungszeit der seriellen und parallelen Verarbeitung von Körperschallsignalen mit einer Größe von jeweils 400 MB. Multiprocessing ermöglicht unter der Vorrausetzung, dass der Kommunikationsoverhead eingeschränkt wird, geringere Berechnungszeiten bzw. höhere Verarbeitungsraten.

4. Echtzeitfähige Analyse

Im Folgenden wird eine echtzeitfähige Analyse demonstriert, welche Datenerfassung, Signalvorverarbeitung sowie Feature-Extraktion umsetzt und schließlich die Ergebnisse an den Benutzer visuell übermittelt.

Zur einfachen Demonstration wird hierzu ein Ultraschallsignal mit einer Abtastrate von 1 MHz bzw. Datenrate von 2 MB/s erfasst und mit einem einfachen Frequenzfilter vorverarbeitet.

Die Feature-Extraktion berechnet aus dem Signal ausgewählte Signalcharakteristiken wie das quadratische Mittel, die Energie des Signals, die Schwerpunktfrequenz, die Peak-Frequenzen und die prozentuale Verteilung der Energie im Frequenzbereich [6]. Die berechneten Features können im Anschluss visualisiert oder weiterverarbeitet werden. Letzteres ermöglicht es, die gewonnenen Signalcharakteristiken in leicht interpretierbare Informationen umzuwandeln, um beispielsweise Aussagen über die bearbeiteten Werkstoffe selbst zu machen oder Anomalien in Prozessen zu erkennen [7], zu klassifizieren und zu quantifizieren.

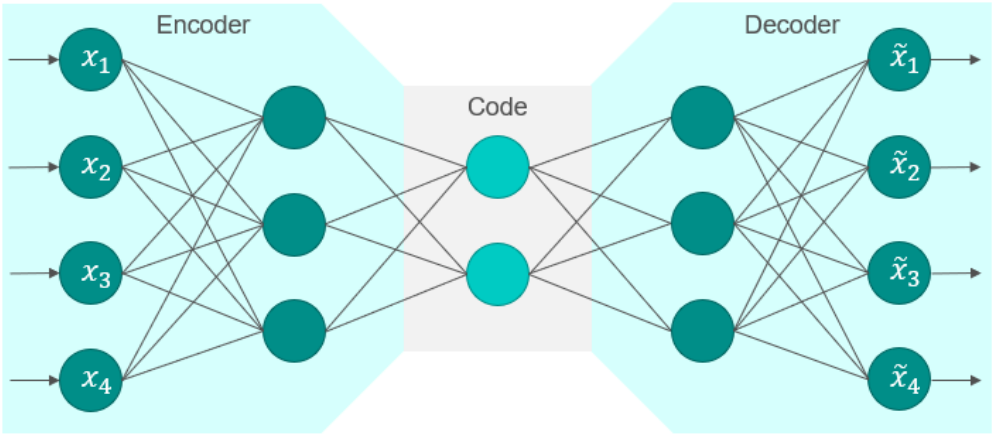

Abb. 4. Struktur eines Autoencoders

Eine der Methoden, die zur Detektion von Anomalien im Signal eingesetzt werden kann, ist ein sog. Autoencoder wie in Abbildung 4 dargestellt. Hierzu werden die berechneten Features als Trainingsdaten für ein neuronales Netz verwendet, wobei die Ergebnisse des Autoencoders den Eingangsdaten entsprechen. Durch die Verengung des neuronalen Netzes in den Hidden Layers werden die eingegebenen Features auf eine kleinere Datenmenge reduziert bzw. enkodiert. Das neuronale Netz erlernt dadurch relevante Muster aus dem Signal zu extrahieren. Die Dekodierung dieser Datenmenge stellt im Idealfall das Eingangssignal als Ausgangssignal schließlich wieder her. In der Realität weicht das Ergebnis jedoch immer vom Input ab. Diese Abweichung ist dabei umso größer, je mehr sich das Eingangssignal von dem angelernten Muster bzw. rekonstruierten Ausgangssignal unterscheidet, und kann daher als Maß zur Detektion einer Anomalie eingesetzt werden [8,9] wie im Folgenden dargestellt.

Das Analyseprinzip ist beispielsweise für die Zustandsüberwachung nutzbar und wird am Beispiel eines Fräsprozesses demonstriert. Hierzu wurde ein Ultraschallsignal mit einer Datenrate von 2 MB/s bzw. Abtastrate von 1 MHz in Zeitabständen von 100 ms verarbeitet und dessen berechneten Features als Input für einen Autoencoder verwendet. Trainiert wurde der Autoencoder mittels Signale, die bei der Fräsbearbeitung mit einem nicht verschlissenen Werkzeug erfasst wurden. Die Trainingsdaten umfassen hierbei sowohl die gefrästen Bahnen als auch den Leerlaufprozess. Als Beispiel für eine werkzeugseitige Anomalie wurde der Autoencoder anschließend auf ein Signal bei gleichen Prozessparametern, jedoch verschlissenem Werkzeugzustand angewendet

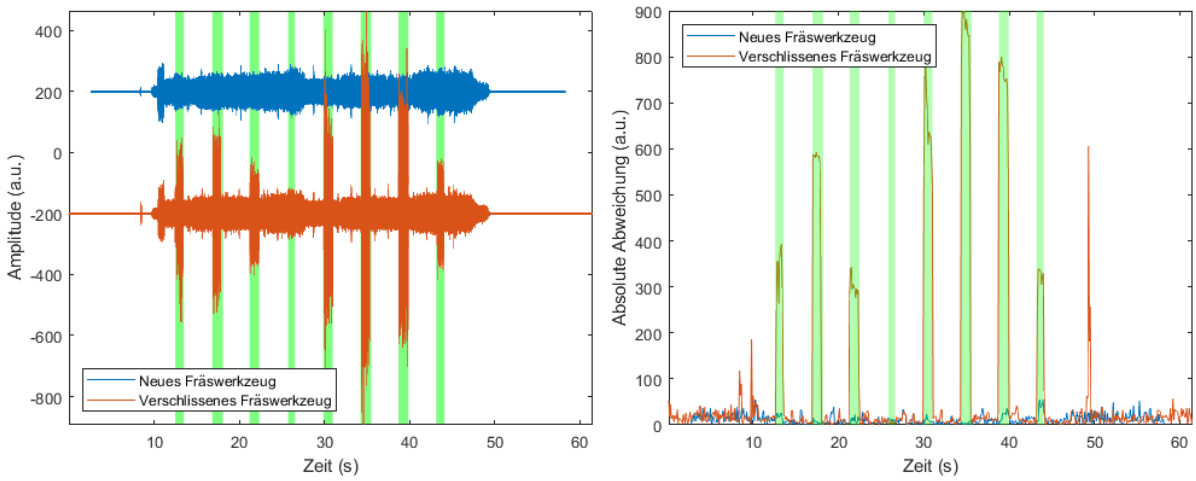

Abb. 4. Detektion einer werkzeugseitigen Anomalie mittels Autoencoder. Die Verwendung eines beschädigten Fräswerkzeugs kann mit Hilfe des Autoencoders als Anomalie sichtbar gemacht werden.

In Abbildung 4 sind auf der linken Seite die beiden Körperschallsignale gezeigt, die beim Fräsen einer Aluminiumplatte mit einem neuen Fräswerkzeug bzw. mit einem verschlissenem Fräswerkzeug (mit abgebrochener Klinge) aufgenommen und zum Trainieren bzw. Testen des Autoencoders als Input eingesetzt wurden. Die Zeiten, zu denen die Fräsbearbeitung stattfand, wurden mit einem grünen Hintergrund manuell markiert.
Rechts in der Abbildung ist der absolute Fehlerwert des Ergebnisses dargestellt. Dieser ist bei der Verwendung eines neuen Werkzeuges gering, da der Autoencoder auf solche Signale gezielt antrainiert ist. Im Fall eines verschlissenen Werkzeuges steigt der Fehlerwert hingegen drastisch an. Über die Wahl eines geeigneten Schwellwert können diese Fehlerwerte daher zum Unterscheiden zwischen Anomalie und Nichtanomalie dienen.

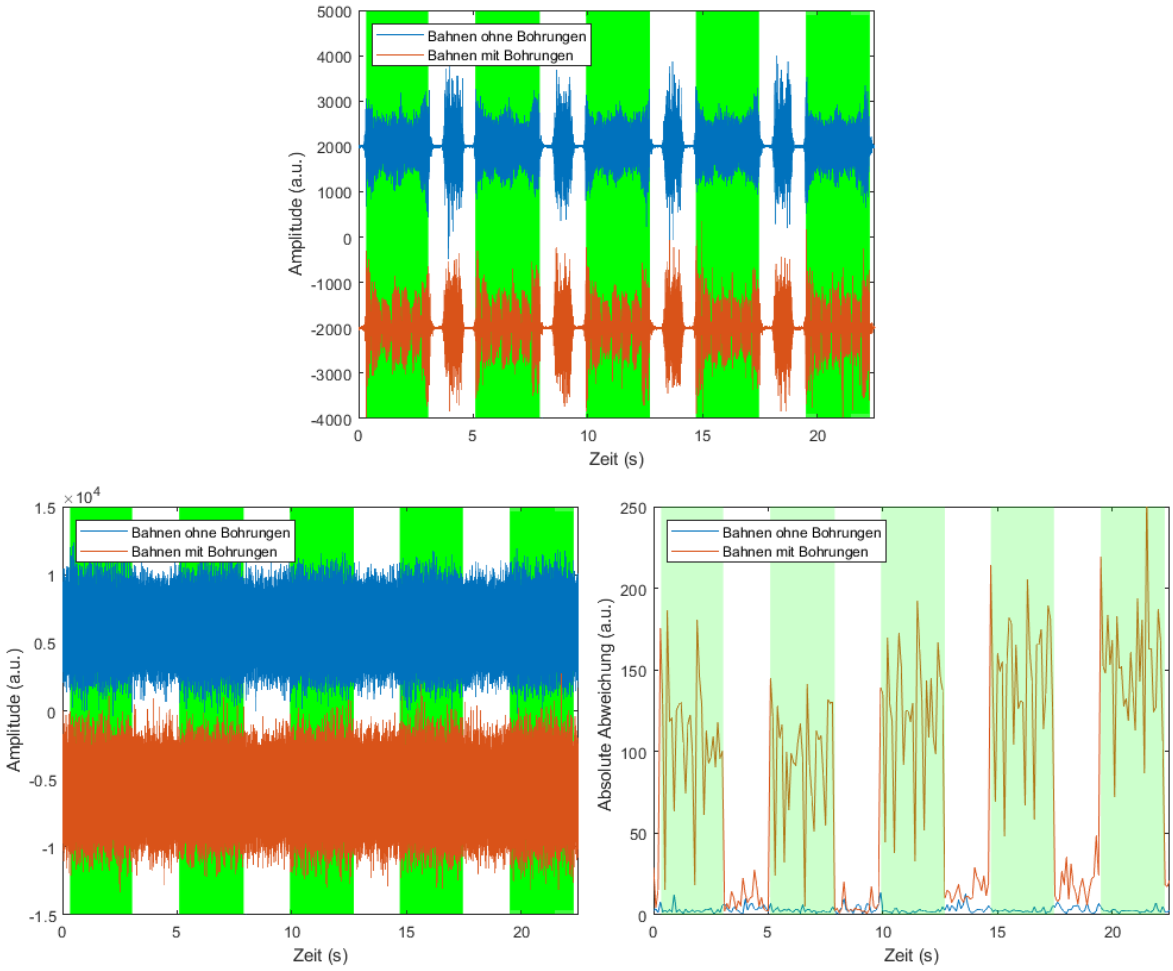

Abb. 5. Detektion einer werkstoffseitigen Anomalie. Oben ist ein Körperschallsignal eines Sensors abgebildet, welches am Spanntisch integriert ist. Unten links ist ein Körperschallsignal eines Sensors abgebildet, welches an der Spindel der CNC-Maschine befestigt ist und dessen Signalcharakteristiken für den Autoencoder verwendet wurden. Unten rechts sind die Rekonstruktionsfehler des Autoencoders zu sehen.

Das gleiche Prinzip kann auch zur Detektion einer werkstoffseitigen Anomalie genutzt werden [10]. Hierzu wurden erneut Körperschallsignale mit einer Abtastrate von 1 MHz bei dem Fräsen einer Aluminiumplatte aufgenommen, wobei in diesem Fall Löcher als Anomalien dienen sollen und zuvor in die geplanten Fräsbahnen gebohrt wurden. Durch die Löcher ergeben sich kurze Unterbrechungen des Bearbeitungsvorgangs wie im oberen Diagramm von Abbildung 5 zu sehen ist. Die Signale stammen hierbei von einem am Spanntisch montierten Sensor und dienen an dieser Stelle ausschließlich zur Visualisierung des Prozesses. Der interessante Sensor, der für die Analyse verwendet wird, ist im

Experiment an der Spindel der CNC-Maschine montiert. Dessen Körperschallsignale sind im unteren linken Diagramm dargestellt. Die Erkennung einer Anomalie mit dem bloßen Auge ist anhand des Signals praktisch unmöglich. Diese Signale wurden nach ihren Signalcharakteristiken untersucht und deren berechnete Features als Input für den Autoencoder verwendet, welcher erfolgreich die mit Bohrlöchern versehenen Fräsbahnen als Anomalien detektieren kann.

Es existieren weitere detektierbare Anomalien. Hierzu zählen z.B. sensorseitige Anomalien wie ein Sensorausfall, welcher zu einem anderen Rauschverhalten des Signals führt. Der Einsatz von Autoencodern beschränkt sich im Allgemeinem nicht auf die Detektion von Anomalien, sondern kann auch für die Detektion bestimmter Prozessschritte oder als Filter [11] genutzt werden. Alternativ kann er auch als Werkzeug zum Einleiten einer weiterführenden Analyse wie dem Klassifizieren von Prozess- oder Werkstückdefekten eingesetzt werden.

5. Zusammenfassung

Multiprocessing erlaubt eine drastische Erhöhung der ausgewerteten Datenmengen und zeigt gerade bei multisensorischen Überwachungssystemen sehr viel Potenzial, welches im Wesentlichen durch die Kommunikation zwischen den Kernen beeinträchtigt wird und der echtzeitfähigen Analyse entgegenwirkt. Doch durch eine kontrollierte Reduzierung der Kommunikationsrate der ausgewerteten Daten kann der Overhead klein gehalten werden, sodass Multiprocessing neben der Verarbeitung großer Datenmengen in Echtzeit auch zur Analyse effektiv eingesetzt werden kann. Die Ergebnisse der Datenverarbeitungen werden schließlich als visuelles Feedback oder als Input für neuronale Netze wie dem Autoencoder verwendet, um Anomalien zu detektieren. In der vorliegenden Arbeit wurden hierzu zwei mögliche Einsätze demonstriert. So wurde im Kontext der Zustandsüberwachung zum einem die Fräsbearbeitung mit einem verschlissenen Fräswerkzeug als Anomalie erkannt, zum anderen wurde im Kontext der Prozessüberwachung Fräsbahnen mit eingebauten Löchern als Anomalie zu lochfreien Fräsbahnen als Regelfall unterschieden.

Referenzen

[1] M. G. R. Sause, F. F. Linscheid, C. Oblinger, S. O. Gade, and S. Kalafat, “Hard-and Software fusion for process monitoring during machining of fiber reinforced materials,” in Munich Symposium on Lightweight Design 2020, 2020. https://doi.org/10.1007/978-3-662-63143-0_6

[2] Florian F. Linscheid and Markus G. R. Sause. "Hard- und Softwarefusion von mehreren akustischen Messmethoden zur Zustandsüberwachung", in DGZfP Jahrestagung 2021: Materialcharakterisierung, Deutsche Gesellschaft für Zerstörungsfreie Prüfung (DGZfP)

[3] Qin, S.J. (2014), “Process data analytics in the era of big data”, AIChE J., https://doi.org/10.1002/aic.14523

[4] MathWorks, Parallel Computing Toolbox Documentation, https://de.mathworks.com/help/parallel-computing/

[5] Martin Hirzel, Robert Soulé, Scott Schneider, Buğra Gedik, and Robert Grimm. 2014. A catalog of stream processing optimizations. ACM Comput. Surv. 46, 4, Article 46 (April 2014), 34 pages. https://doi.org/10.1145/2528412

[6] Markus G. R. Sause, “In situ monitoring of fiber-reinforced composites: Theory, basic concepts, methods, and applications”, Springer International Publishing, 2016

[7] Kraljevski, I., Duckhorn, F., Tschöpe, C., & Wolff, M., “Machine learning for anomaly assessment in sensor networks for NDT in aerospace.”, 2021, IEEE Sensors Journal, 21(9), https://doi.org/10.1109/JSEN.2021.3062941

[8] Jakovlev, S.; Voznak, M. Auto-Encoder-Enabled Anomaly Detection in Acceleration Data: Use Case Study in Container Handling Operations. *Machines* 2022, *10*, 734. https://doi.org/10.3390/machines10090734

[9] Jong Moon Ha, Hong Min Seung, Wonjae Choi, Autoencoder-based detection of near-surface defects in ultrasonic testing, Ultrasonics, Volume 119, 2022, 106637, ISSN 0041-624X, https://doi.org/10.1016/j.ultras.2021.106637

[10] L. Lorenti, G. De Rossi, A. Annoni, S. Rigutto and G. A. Susto, "CUAD-Mo: Continuos Unsupervised Anomaly Detection on Machining Operations," 2022 IEEE Conference on Control Technology and Applications (CCTA), Trieste, Italy, 2022, https://doi.org/10.1109/CCTA49430.2022.9966138

[11] Gao, Fei, et al. "Ultrasonic signal denoising based on autoencoder", Review of Scientific Instruments 91.4, 2020, https://doi.org/10.1063/1.5136269