University of Hasselt

Thesis proposed to achieve the degree of master in computer science

EmbedLLM from legacy documents to embedded XR instructions: automatic context aware content generation to support manual task execution

Author:

Verstappen Bram

Supervisor:

Prof. Dr. Raf Ramakers

Co-supervisor:

Prof. Dr. Kris Luyten

Advisor:

Maties Claesen

Academic year 2025-2026

Acknowledgements

The completion of this thesis would not have been possible without the support, guidance, and encouragement of multiple individuals. I am deeply grateful to everyone who shared their expertise, time, and insights, all of which were instrumental in shaping this research and helping me along the way.

First and foremost, I would like to express my sincere gratitude to my supervisor, Prof. Dr. Raf Ramakers, for his invaluable insights and guidance throughout this thesis. His feedback was crucial not only in refining my ideas and navigating complex challenges, but the lessons I have learned from him both during this project and throughout my entire education will undoubtedly serve me long into my future career.
Furthermore, I would like to extend my sincere thanks to my co-supervisor, Prof. Dr. Kris Luyten, for his encouragement during more challenging periods and for his excellent guidance throughout my education.

In addition, I would like to express my appreciation to my advisor, Maties Claesen, for his daily guidance, constructive feedback, practical insights, and helpful discussions. His expertise and support were an invaluable resource throughout this thesis.

A special word of thanks goes to the Digital Future Lab HCI research group for their welcoming atmosphere and general helpfulness, both during the setup of the user studies and in answering various research questions.

I am also deeply grateful to the participants of my user studies. Thank you for your time, patience, and enthusiasm. Your input was invaluable in shaping the outcomes and validating the insights of this research.

Finally, I would like to express my sincerest gratitude to my parents for their unwavering support throughout my education and, of course, my life. Additionally, I would like to thank my friends for all the support, laughs, serious discussions, and philosophical exercises.

And last but certainly not least, I would like to thank my girlfriend, Silke, for her support during challenging times and the sense of peace she brings.

Abstract

This thesis addresses the cognitive friction associated with executing manual tasks using flat 2D documentation in spatial computing environments, where users must constantly shift attention between instructions, physical tools, and machinery. To solve this, we present EmbedLLM, the first automated, context-aware content reformatting framework that adapts both content design and presentation to the user's environment and specific goal. EmbedLLM converts unstructured step-based PDF manuals into 3D spatially embedded instructions in Virtual Reality (VR). The system operates via a pipeline that abstracts the environment into a semantic scene graph, leverages a multi-stage Large Language Model (LLM) reasoning chain to construct step-based instruction nodes and semantic edges, and instantiates these panels using a novel, semantically aware force-directed 3D label placement algorithm to achieve spatial equilibrium and prevent visual clutter.

We evaluated EmbedLLM through a preliminary preference survey ($n = 20$) across diverse environments and a within-subjects user study ($n = 12$) comparing the framework to a traditional floating PDF window during 3D printer maintenance tasks. The preference survey showed that EmbedLLM achieves significantly higher environmental alignment than traditional baseline documents, with participants strongly favoring spatial orchestration for complex, structurally demanding procedures. However, the task execution study revealed a clear trade-off: active task support with EmbedLLM resulted in significantly longer completion times (996.93 s vs. 580.46 s) and increased cognitive load (3.00 vs. 2.31 on the NASA-TLX scale). Qualitative feedback suggests these performance costs stem from the rigid sequential pacing of the generated interface and novices struggling with hardware terminology.

This work contributes the technical pipeline of EmbedLLM, a semantically enhanced physics layout algorithm, and empirical insights on context-aware XR instruction delivery. Future research directions include real-time situated action updates, user skill level adaptation, and integration with agentic GUI automation and text-to-3D models.

Dutch Summary

Inleiding

Huidige systemen voor spatial computing zijn technisch in staat om rijke, in de scene ingebedde content weer te geven. Voorbeelden hiervan zijn immersieve augmented reality (AR)-games of interactieve demonstraties waarbij virtuele objecten dynamisch door de fysieke omgeving van een gebruiker bewegen[1]. Desondanks erven systemen voor spatial computing overwegend het traditionele desktop WIMP-paradigma (Windows, Icons, Menus and Pointers), waarbij content wordt weergegeven in tweedimensionale vensters die in een driedimensionale ruimte zijn geplaatst. Hoewel spatial computing in staat is tot veel rijkere representaties, zijn legacy-bronnen van content (zoals PDF of HTML) niet ontworpen om onder deze omstandigheden te worden bekeken, waardoor ze de mogelijkheden van Mixed Reality (MR) niet volledig benutten.

Bovendien is een enorme hoeveelheid kritieke, procedurele kennis uitsluitend vastgelegd in deze legacy-documenten, bijvoorbeeld: reparatiehandleidingen die door automonteurs worden gebruikt of onderhoudsgidsen voor operators in fabriekshallen. Bij het uitvoeren van complexe handmatige taken worden operators gedwongen om constant hun aandacht te verplaatsen tussen de fysieke machines, hun fysieke gereedschappen en de content. Deze frequente context-switches verhogen de cognitieve belasting, vergroten fouten bij de lokalisatie van taken en verlengen de totale uitvoeringstijd. Hoewel een nauwe integratie van de content direct in de context van de gebruiker via scene-embedding deze wrijving zou kunnen wegnemen [2, 3, 17, 44], vereist het vertalen van bestaande 2D-content naar een in de scene ingebed alternatief momenteel een intensieve, handmatige herauteur-inspanning.

Daarnaast kan deze herformattering van content naar een contextbewust alternatief niet uitsluitend op de omgeving worden gebaseerd. Als een gebruiker bijvoorbeeld een boodschappenlijst wil samenstellen voor een recept, maar het herformatteringsproces is zich niet bewust van dit specifieke doel, dan zou de in de scene ingebedde representatie uitsluitend de bakstappen tonen. Als gevolg hiervan zou de gebruiker nog steeds gedwongen zijn om de alternatieve representatie handmatig te filteren om de specifieke benodigde informatie te extraheren, wat op zijn beurt het aantal context-switches zou verhogen en het doel van de ruimtelijke aanpassing teniet zou doen. Bijgevolg kan de gebruikerscontext niet worden beperkt tot alleen de omgeving en moet deze ook het doel van de gebruiker bevatten. Dit introduceert de centrale onderzoeksuitdaging van deze masterthesis:

Hoe kunnen stapsgewijze handmatige instructies die in een PDF-formaat worden gepresenteerd, automatisch worden ingebed in en aangepast aan de omgeving en het gebruikersdoel met behulp van Mixed Reality, en welk effect heeft dit op de ondersteunde taakuitvoering?

Om deze uitdaging aan te gaan, is het essentieel om te onderzoeken hoe bestaand onderzoek omgaat met contextbewuste AR en hoe een systeem de omringende omgeving interpreteert. De huidige literatuur introduceert een contextbewuste Augmented Reality (AR) ontwerprichtlijn

[1] https://www.apple.com/apple-vision-pro/

gestructureerd rond twee hoofdpijlers: contextbewust contentontwerp en contextbewuste contentrepresentatie [12]. Het eerste past de inhoud van de informatie zelf aan de situatie van de gebruiker aan, zoals het dynamisch vervangen van een bakrecept door een boodschappenlijst wanneer een gebruiker een supermarkt binnengaat. Omgekeerd past het laatste de manier aan waarop die informatie visueel is georganiseerd of gepositioneerd zonder de kernbetekenis ervan te veranderen, bijvoorbeeld door automatisch een digitaal paneel weg te schuiven van een object om het gezichtsveld van de gebruiker vrij te houden.

Voordat er methoden kunnen worden ontwikkeld om de content aan de context aan te passen, is er een gestructureerd, leesbaar model van de context nodig. Historisch gezien richtten computervisiemethoden zich voor het interpreteren van de fysieke omgeving op ruwe, dichte omgevingsrepresentaties [22]. Ruimtelijke apparaten maken vaak gebruik van Simultaneous Localization and Mapping (SLAM)-algoritmen in combinatie met dieptesensoren om gedetailleerde 3D-geometrische structuren te construeren, zoals puntenwolken of spatial meshes [35]. Hoewel deze ruimtelijke modellen een hoge structurele getrouwheid hebben, zijn ze rekenkundig zwaar en missen ze fundamenteel ingebedde semantische gegevens. Een ruwe spatial mesh kan bijvoorbeeld een geometrisch oppervlak identificeren, maar kan niet intrinsiek herkennen dat het oppervlak behoort tot een "nozzle van een 3D-printer" of een "remschijf van een fiets". Om contextbewuste logica mogelijk te maken, moet een systeem deze dichte gegevens abstraheren naar beschrijvende, semantische representaties. Recente ontwikkelingen pakken deze beperking aan door fysieke ruimtes te organiseren in semantische scene graphs [7]. Een scene graph gebruikt machine learning om een omgeving te destilleren in een hiërarchisch netwerk op hoog niveau waarin knopen (nodes) afzonderlijke objecten met hun semantische attributen vertegenwoordigen, en kanten (edges) de ruimtelijke en conceptuele relaties daartussen definiëren (zoals een specifieke knoop "filamentspoel" die is verbonden met een knoop "werkbank" via een "bovenop"-relatie).
Daarnaast richt een groeiende hoeveelheid onderzoek zich op het herkennen van de huidige activiteit van de gebruiker en het afleiden van het doel van de gebruiker via first-person sensorgegevens [27, 32, 47]. Hoewel deze methoden geschikt zijn voor spatial computing use-cases, dankzij de inherente first-person sensorgegevens, valt doelafleiding buiten het bereik van deze masterthesis. We nemen daarom aan dat het doel van de gebruiker *a-priori* bekend is.

Hoewel deze semantische scene graphs oorspronkelijk zijn ontwikkeld om autonome robots te helpen veilig door menselijke ruimtes te navigeren, toont recent onderzoek aan dat Large Language Models (LLMs) een sterke ingebouwde vaardigheid bezitten om scene graphs te begrijpen [43]. Dit snijpunt van semantische mapping en AI-redenering biedt de ideale basis voor geautomatiseerde, contextbewuste herformattering van content, waardoor een LLM een fysieke omgeving kan evalueren en erover kan redeneren.

Tot slot modelleren we, gebaseerd op deze bevindingen [22, 37, 43], de herformattering van legacy-content naar in de scene ingebedde representaties als een graafprobleem. Concreet wordt, gegeven een gebruikersdoel, een brondocument en een scene graph van de omgeving, een LLM gebruikt om iteratief graafaanvullingen te genereren. Het LLM bouwt zowel de nieuwe knopen op, die de geherformatteerde broncontent bevatten om aan te sluiten bij het doel van de gebruiker, als de semantische kanten die hun relaties binnen de omgeving tot stand brengen.

EmbedLLM

Om deze onderzoeksuitdaging aan te pakken, hebben we EmbedLLM ontwikkeld, wat voor zover wij weten het eerste contextbewuste content-herformatteringsframework is dat zowel contentontwerp als contentrepresentatie aanpast op basis van de omgevingscontext. Omdat het framework automatische herformattering van legacy-documenten naar in de scene ingebedde alternatieven mogelijk maakt, kan het worden gebruikt om verschillende alternatieve representaties van hetzelfde stuk broncontent te maken door alleen de omgeving en het gebruikersdoel te veranderen, waardoor automatische aanpassing van de content louter op basis van de context wordt gerealiseerd. EmbedLLM verwacht drie inputs:

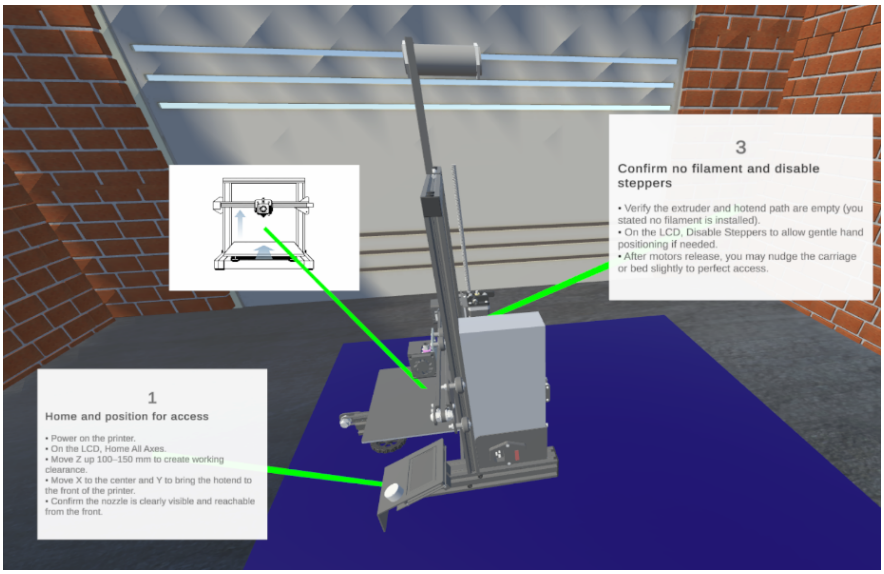


Figure 1: Voorbeeld van de output van EmbedLLM voor een enkele stap. Hier zijn twee tekstpanelen en één afbeeldingspaneel verankerd aan verschillende componenten van een Ender-3 3D-printer. Het eerste paneel legt uit hoe de assen gehomed moeten worden, het afbeeldingspaneel toont de verwachte toestand van de printer na deze eerste stap, en het tweede tekstpaneel vermeldt dat er geen filament in de printer mag achterblijven om door te kunnen gaan.

1. De broncontent.
2. Een tekstuele beschrijving van het gebruikersdoel.
3. En een digital twin van de omgeving vertegenwoordigd door een Unity-scene.

Op basis van deze inputs creëert EmbedLLM een opeenvolgende reeks stappen om de gebruiker naar diens doel te leiden. Voor elke stap genereert het systeem automatisch tekst- en afbeeldingspanelen die semantisch direct verankerd zijn aan relevante objecten in de fysieke omgeving. Gebruikers kunnen vervolgens naadloos vooruit of achteruit navigeren door de stappen met de triggers op hun Meta Quest-controllers. Door de kennis nauw in de omgeving in te bedden en deze te herformatteren om aan te sluiten bij het doel van de gebruiker, beoogt EmbedLLM de cognitieve belasting te verminderen en de afstemming van de content met zowel de fysieke omgeving als het uiteindelijke doel van de gebruiker te verbeteren.
Een voorbeeld van de output is te zien in Figuur 1. De input bestaat uit een brondocument met details over het vervangen van de nozzle van een Ender-3 3D-printer, het doel van de gebruiker is het vervangen van de nozzle, en de scene is een garage met daarin een Ender-3 printer. De eerste stap van deze alternatieve representatie bestaat uit het homen van de printer. Deze stap bevat drie panelen (twee tekstpanelen en één afbeeldingspaneel) die verankerd zijn aan verschillende componenten van de printer. Het eerste tekstpaneel beschrijft hoe de printer gehomed moet worden. Vervolgens toont het afbeeldingspaneel hoe de printer eruit moet zien na het succesvol uitvoeren van deze eerste stap. Ten slotte vermeldt het tweede tekstpaneel dat er geen filament in de printer mag achterblijven om door te kunnen gaan.

EmbedLLM is geïmplementeerd in Virtual Reality (VR) met behulp van Unity. Hoewel implementatie in Augmented Reality (AR) ideaal zou zijn voor fysieke taakuitvoering, introduceert dit visuele computerbeperkingen met betrekking tot scene-begrip. Om deze reden abstraheren we deze technische beperkingen om ons volledig te richten op de kern HCI-bijdragen van het systeem. Tot slot is deze virtuele scene het enige component dat handmatige auteurstaken vereist. Zodra toekomstige ontwikkelingen in computervisie het nodige detailniveau bieden om de scene graph dynamisch te construeren en bij te werken, zal deze handmatige stap niet langer

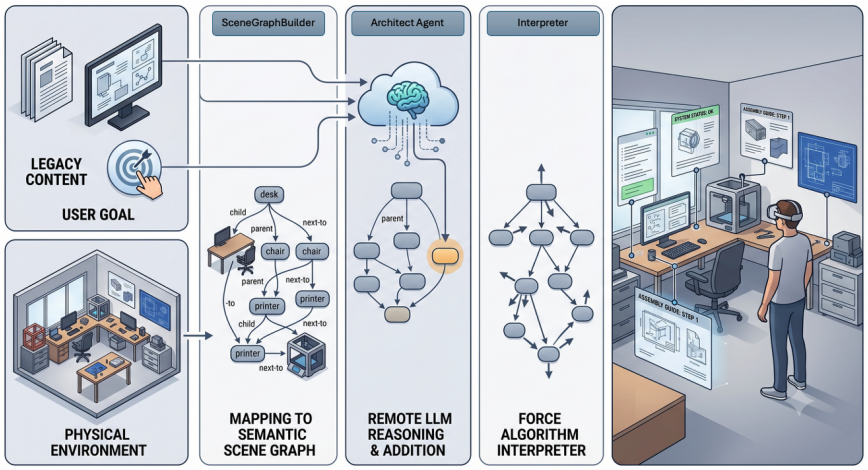


Figure 2: Overzicht van EmbedLLM. Het systeem neemt drie inputs aan, namelijk: het gebruikersdoel, het brondocument en de digital twin van de omgeving. De SceneGraphBuilder converteert vervolgens de virtuele scene in een semantische scene graph. Dit dient daarna, in combinatie met het gebruikersdoel en het brondocument, als input voor de ArchitectAgent. Hier interpreteert een remote LLM de inputs en creëert nieuwe knopen om de scene graph uit te breiden. Deze semantische uitbreidingen worden vervolgens door de Interpreter in de virtuele scene geïnstantieerd, wat resulteert in een in de scene ingebedde en op het doel afgestemde alternatieve representatie.

nodig zijn.

De systeemarchitectuur van EmbedLLM is gerealiseerd via een pipeline die bestaat uit drie verschillende Unity-modules: de SceneGraphBuilder, de ArchitectAgent en de Interpreter (zoals te zien is in Figuur 2). In plaats van te proberen te fungeren als een generieke contextbewuste engine voor willekeurige domeinen, is het bereik van dit framework bewust beperkt tot handmatige, stapsgewijze content. Bijgevolg zijn zowel de onderliggende datamodellen als de prompts voor het remote LLM expliciet gestructureerd om in de scene ingebedde representaties voor stapsgewijze content te faciliteren.

Ten slotte moet de uitvoering deterministisch zijn om een consistente evaluatie van het systeem mogelijk te maken. Om resultaten tussen verschillende deelnemers effectief te kunnen vergelijken, moeten zij exact dezelfde alternatieve representatie ervaren. Om deze vergelijking te vergemakkelijken en tegelijkertijd de uitvoeringstijd te verkorten, bevat EmbedLLM een caching-architectuur. Omdat elke module strikt afhankelijk is van de output van zijn directe voorganger, is voor de output van elke module een specifieke cache geïmplementeerd. Deze gegevens worden op schijf opgeslagen en bij volgende uitvoeringen als input ingeladen. Met name de output van de architect-agent wordt ook gecached, wat zorgt voor een volledig deterministische uitvoering.

De SceneGraphBuilder

De pipeline begint met de SceneGraphBuilder, die de ruimtelijke en semantische context van de omgeving van de gebruiker vaststelt. Omdat standaard virtuele objecten in Unity een intrinsiek begrip missen van hoe ze zich tot elkaar verhouden, parset deze module de geometrie van de omgeving en evalueert deze ruimtelijke eigenschappen. Met behulp van specifieke geometrische functies en een set van nabijheidsheuristieken leidt het systematisch ruimtelijke relaties af zoals ”on top of”, ”underneath of”, ”left of”, ”right of”, ”behind of” en ”in front of” om een uitgebreide semantische scene graph te construeren die de virtuele scene vertegenwoordigt.

De ArchitectAgent

Deze gestructureerde semantische scene graph wordt, samen met de 2D-broncontent (zoals een PDF-handleiding) en het expliciete doel van de gebruiker, vervolgens doorgegeven aan de ArchitectAgent. Deze module fungeert als de centrale redeneerkern van EmbedLLM en biedt de interface naar een remote Large Language Model (LLM). Om ongeleide en onvoorspelbare tekstoutput te voorkomen, dwingt de module het LLM om binnen een strikt gedefinieerd schema te werken dat uitsluitend is geoptimaliseerd voor het procedurele taakdomein. Concreet genereert het LLM een lijst met stappen om het gebruikersdoel te bereiken. Vervolgens kan het binnen deze stappen content in de omgeving inbedden. Het is echter beperkt tot maximaal vier teksten afbeeldingspanelen per stap. Deze beperking is gebaseerd op heuristieken met als doel de visuele rommel te verminderen.
Bovendien is de uitvoering ontkoppeld in drie verschillende, sequentiële aanroepen om bekende cognitieve belastingbeperkingen te overwinnen waarbij de prestaties van het LLM verslechteren wanneer meerdere taken in één prompt worden gecombineerd:

- Call 1 (Stapdefinitie): Vraagt het LLM om het bronmateriaal te analyseren en een logische volgorde te schetsen van de stappen op hoog niveau die nodig zijn om het handmatige doel van de gebruiker te bereiken.
- Call 2 (Knoopgeneratie): Wordt iteratief uitgevoerd voor elke geïdentificeerde stap en vraagt het LLM om kennis uit het brondocument te extraheren en nieuwe knopen voor contenttoevoeging te genereren die specifiek op die stap zijn afgestemd.
- Call 3 (Kantdefinitie): Draagt het LLM op om de bijbehorende semantische kanten (relaties) te genereren die bepalen hoe deze new contentknopen contextueel verbonden moeten worden met bestaande fysieke objecten of structurele oriëntatiepunten in de scene graph.

Hierbij wordt de eerste call eenmalig uitgevoerd, en worden de tweede en derde call iteratief uitgevoerd voor elke stap die in de stapdefinitie-call (call 1) is geïdentificeerd. De gegenereerde toevoegingen over alle opeenvolgende stappen worden uiteindelijk samengevoegd tot een samenhangende lijst van toevoegingen. Deze lijst wordt het bouwplan (build plan) genoemd en fungeert als een gestructureerde reeks van uitgebreide scene graphs voor elke stap, die precies weergeeft welke taakinformatie nodig is en waar deze semantisch verankerd moet worden ten opzichte van de fysieke componenten van de omgeving.

De Interpreter

Het laatste component in de systeem-pipeline is de Interpreter, die verantwoordelijk is voor het parsen van het gestructureerde bouwplan dat door de ArchitectAgent is gegenereerd en het visueel instantiëren ervan binnen de driedimensionale virtuele scene. Een naïeve benadering zou de panelen in het middelpunt van hun ankerobject kunnen positioneren en ze kunnen verplaatsen langs de richting die door hun semantische relaties wordt bepaald, totdat ze niet langer botsen met andere objecten in de scene. Deze benadering houdt er echter geen rekening mee dat de gegenereerde panelen zelf niet als fysieke objecten in de omgeving zijn geregistreerd, wat er vaak toe leidt dat meerdere panelen op exact dezelfde ruimtelijke coördinaten worden verankerd. Dit verslechtert de leesbaarheid ernstig en introduceert visuele rommel. Hoewel men dit zou kunnen oplossen door elk paneel na plaatsing opeenvolgend aan de scene toe te voegen en volgende panelen te dwingen rekening te houden met de nieuw toegevoegde objecten, leunt deze recursieve methode zwaar op rigide structurele afhankelijkheden en kan dit leiden tot onnodig lange ketens van paneel-op-paneel-bevestigingen. Bovendien faalt een naïeve benadering volledig om rekening te houden met de zichtbaarheid voor de gebruiker en mogelijke occlusies.

Om deze lay-outbeperkingen te vermijden en te voorkomen dat panelen overlappen, is een force-directed lay-outalgoritme vereist. EmbedLLM implementeert een nieuwe aanpassing van een op

fysica gebaseerd 3D-labelplaatsingsalgoritme [48]. Hoewel dit geworteld is in gevestigd onderzoek dat gebruikmaakt van natuurkundige principes van aantrekking tot aangewezen ankerpunten en afstoting tussen individuele labels, is het uniek omdat het specifieke semantische krachten in het model introduceert. Door de voorwaartse vectoren van zowel het anker als het paneel te evalueren, berekent het systeem een directionele aantrekkingskracht die specifiek is afgestemd op de onderliggende semantische beperkingen die in het bouwplan zijn gedefinieerd.

De volledige workflow van de Interpreter werkt in opeenvolgende stappen. Eerst stelt de module een initiële plaatsing vast voor elk paneel als een heuristische "beste gok" door het naar buiten te verplaatsen vanaf het middelpunt van het ankerobject langs de richtingsvector van de semantische relatie totdat het vrij is van omgevingsbotsingen. Deze botsingsvrije toestand dient als startpositie voor het force-directed algoritme. Vervolgens gebruikt het systeem de ingebouwde physics-engine van Unity om de aantrekkings-, afstotings- en semantische krachten dynamisch te simuleren en toe te passen. Gegeven het statische karakter van de doelomgeving, wordt het algoritme uitgevoerd totdat de panelen een toestand van ruimtelijk evenwicht bereiken, waarna de fysica-berekeningen worden stopgezet om de prestaties te verbeteren en de contextbewuste representatie te finaliseren.

Evaluatie van EmbedLLM

Om de effectiviteit van onze aanpak bij het beantwoorden van de onderzoeksuitdaging te meten, evalueren we EmbedLLM via twee verschillende gebruikersstudies. Ten eerste, om het vermogen van het systeem te beoordelen om het eerste deel van de onderzoeksuitdaging aan te pakken, meten we de mate waarin de alternatieve representaties die door EmbedLLM worden gegenereerd, worden waargenomen als afgestemd op de fysieke omgeving en de gebruikersdoelen. Daarnaast onderzoeken we de voorkeuren van gebruikers door deze representaties te benchmarken tegen een traditionele smartphone-baseline. Ten tweede evalueren we, om het daaropvolgende deel van de onderzoeksuitdaging aan te pakken, de alternatieve representaties van EmbedLLM tijdens actieve taakuitvoering, met als doel hun impact en effectiviteit bij het bieden van real-time taakondersteuning te kwantificeren.

De eerste gebruikersstudie is uitgevoerd via een online vragenlijst, waarin deelnemers een reeks alternatieve representaties evalueren naast een enkele baseline-conditie. Elke representatie wordt gepresenteerd aan de hand van een korte tekstuele beschrijving van het doel van de gebruiker, de fysieke scene en het brondocument, vergezeld van een set afbeeldingen die de alternatieve representatie zelf tonen. Voor elk scenario beoordelen deelnemers hoe goed de representatie aansluit bij de omgeving, evalueren ze de afstemming met het doel van de gebruiker, en geven ze ten slotte hun voorkeur aan voor het gebruik van de alternatieve representatie boven de traditionele smartphone-baseline.
De resultaten van deze vragenlijst tonen een significant hogere omgevingsafstemming voor de alternatieve representatie in vergelijking met de baseline. De doelafstemming blijft consistent hoog in beide condities, aangezien de baseline ook goed scoort op deze metriek. Interessant is dat de gebruikersvoorkeursgegevens onthullen dat niet elke combinatie van scene, doel en content evenzeer de voorkeur geniet voor scene-embedding. Met name taken die een hoger niveau van ruimtelijk bewustzijn vereisen (zoals het repareren van een fiets) worden sterk verkozen voor scene-embedding. In tegenstelling hiermee vertonen taken met een lager ruimtelijk bewustzijn (zoals het verzamelen van benodigdheden in een supermarkt) een verminderde voorkeur in vergelijking met de traditionele smartphone-baseline.

Voor de tweede gebruikersstudie nemen deelnemers deel aan een sessie van ongeveer 50 minuten. Tijdens deze sessie voeren ze twee reparatietaken uit op een Ender-3 3D-printer: één met een baseline-conditie bestaande uit een zwevend PDF-venster in Virtual Reality (VR), en de andere met behulp van EmbedLLM. Om de taken te vergemakkelijken, zijn beide condities uitgebreid met een video passthrough-modus. Deelnemers bekijken de digitale content in VR en schakelen de passthrough-modus in om met de fysieke printer te communiceren. Na het uitvoeren van elke taak vullen deelnemers een custom vragenlijst in om de omgevings- en doelafstemming te

evalueren, samen met de NASA-TLX om de cognitieve belasting te meten. Taakuitvoeringstijden worden ook geregistreerd.
De resultaten van deze gebruikersstudie bevestigen dat EmbedLLM alternatieve representaties creëert die sterk ingebed zijn in de scene en nauw aansluiten bij het gebruikersdoel. Echter, tijdens actieve taakondersteuning leidt EmbedLLM ook tot een significant hogere cognitieve belasting en langere taakuitvoeringstijden. Kwalitatieve bevindingen en feedback van deelnemers geven aan dat dit kan voortkomen uit complexe terminologie, aangezien geen van de deelnemers voorafgaande ervaring heeft met 3D-printers. Bovendien geven deelnemers aan dat zij zich gedwongen voelen om de gegenereerde instructies strikt stap-voor-stap te volgen, wat hun probleemoplossende flexibiliteit beperkt in vergelijking met het meer vertrouwde en betrouwbare baseline-formaat.

Conclusie

Deze masterthesis onderzoekt methoden om legacy, op handmatige taken gebaseerde 2D-content aan te passen naar contextbewuste, in de scene ingebedde alternatieven voor spatial computing use-cases. We stellen EmbedLLM voor, wat voor zover wij weten het eerste framework is om legacy-content automatisch te herformatteren naar een in de scene ingebed alternatief. Bovendien evalueren we EmbedLLM op basis van gebruikersvoorkeur, doelafstemming en omgevingsafstemming, en beoordelen we de taakondersteunende capaciteiten tijdens handmatige taakuitvoering. Deze evaluaties tonen aan dat omgevings- en doelafstemming hoog worden gewaardeerd en dat gebruikers in specifieke scenario's de voorkeur geven aan EmbedLLM boven de traditionele baseline. De bevindingen geven echter ook aan dat niet alle combinaties van scene, content en doel even geschikt zijn voor contextbewuste aanpassing. Bovendien verhoogt het gebruik van EmbedLLM tijdens actieve taakuitvoering de cognitieve belasting en taakuitvoeringstijden.

Echter, EmbedLLM wordt beperkt door de lange generatietijden en het tokenverbruik van het remote LLM. Daarnaast zijn beide evaluaties die zijn gebruikt om de kwaliteiten van het systeem te beoordelen beperkt in termen van deelnemersselectie. De online vragenlijst mist een strikte screening van deelnemers en meet de digitale geletterdheid van deelnemers niet. Bovendien rekruteert de gebruikersstudie voor taakondersteuning deelnemers zonder voorafgaande ervaring met 3D-printers, terwijl de gegenereerde alternatieve representatie niet expliciet rekening houdt met het vaardigheidsniveau van de gebruiker. Hoewel gebruikerservaring buiten de grenzen van het in deze thesis gedefinieerde contextbereik valt, is het doel van een beginnende gebruiker in dit scenario is niet uitsluitend beperkt tot het uitvoeren van de reparatie, maar omvat het ook het leren van de terminologie. Ten slotte ontbreekt een evaluatie van de prestaties van het systeem met grotere delen broncontent of complexere taken.

Toekomstig werk zou EmbedLLM moeten uitbreiden door methoden te onderzoeken om een uniform inputmodel te creëren voor contextbewuste Augmented Reality (AR). Daarnaast zou het de contextscope moeten uitbreiden en methoden moeten onderzoeken om deze alternatieve representaties te generaliseren buiten op taken gebaseerde broncontent. Bovendien zou integratie met embodied reasoning-modellen, zoals de robotica-frameworks van Gemini, moeten worden verkend om generatie te gronden binnen fysieke uitvoeringsomgevingen. Ten slotte zou toekomstig onderzoek methoden moeten onderzoeken om generatieve text-to-3D-modellen te integreren, wat rijke alternatieve representaties mogelijk maakt die dynamische 3D-modellen en visuele animaties bevatten.

Summary

Introduction

Current spatial computing systems are technically capable of displaying rich, scene-embedded content. Examples include immersive augmented reality (AR) games or interactive demonstrations where virtual assets dynamically traverse a user's physical environment[2]. Nevertheless, spatial computing systems predominantly inherit the traditional desktop WIMP (Windows, Icons, Menus and Pointers) paradigm, displaying content in two-dimensional windows placed in a three-dimensional space. While spatial computing is capable of far richer representations, legacy content sources (such as PDF or HTML) are not designed to be viewed in these circumstances, and thus fail to fully utilize the capabilities of Mixed Reality (MR).

Furthermore, a vast amount of critical, procedural knowledge is captured exclusively in these legacy documents, for example: repair manuals used by car mechanics or maintenance guides used for industry hall operators. When executing complex manual tasks, operators are forced to constantly shift their attention back and forth between the physical machinery, their physical tools, and the content. These frequent context switches elevate cognitive load, increase task localization errors, and prolong overall execution times. While tightly integrating the content directly into the user's context via scene-embedding could eliminate this friction [2, 3, 17, 44], translating existing 2D content into a scene-embedded alternative currently demands an intensive, manual re-authoring effort.

Additionally, this reformatting of content to a context-aware alternative cannot be solely based on the environment. For instance, if a user intends to compile a grocery list for a recipe but the reformatting process remains oblivious to this specific goal, the scene-embedded representation might exclusively display the baking steps. As a result, the user would still be forced to manually filter the alternative representation to extract the specific information required, which would in turn increase the amount of context switches, defeating the purpose of the spatial adaptation. Consequently, the user context cannot be limited to solely the environment and should incorporate the user's goal. This introduces the central research challenge of this thesis:

How can step-based manual instructions presented in a PDF format be automatically embedded in and adapted to the environment and user goal using Mixed Reality and what effect does this have on supported task execution?

To address this challenge, it is essential to examine how existing research manages context-aware AR and how a system interprets the surrounding environment. Current literature introduces a context-aware Augmented Reality (AR) design space structured around two main pillars: context-aware content design and context-aware content representation [12]. The former adapts the substance of the information itself to fit the user's situation, such as dynamically replacing a baking recipe with a grocery shopping list when a user enters a supermarket. Conversely, the latter modifies how that information is visually organized or positioned without altering its core

[2] https://www.apple.com/apple-vision-pro/

meaning, for instance automatically shifting a digital panel away from an object to preserve the user's line of sight.

Before methods can be developed to adapt the content to the context, a structured, readable model of the context is needed. Historically, to interpret the physical environment, computer vision methods focused on raw, dense environmental representations [22]. Spatial devices often utilize Simultaneous Localization and Mapping (SLAM) algorithms alongside depth sensors to construct detailed 3D geometric structures, such as point clouds or spatial meshes [35]. While these spatial models boast high structural fidelity, they are computationally resource-intensive and fundamentally lack embedded semantic data. For example, a raw spatial mesh can identify a geometric surface, but it cannot intrinsically recognize that the surface belongs to a "3D printer nozzle" or a "bicycle brake disc". To enable context-aware logic, a system must abstract this dense data into descriptive, semantic representations. Recent advances address this limitation by organizing physical spaces into semantic scene graphs [7]. A scene graph uses machine learning to distill an environment into a high-level hierarchical network where nodes represent distinct objects along with their semantic attributes, and edges define the spatial and conceptual relationships between them (such as a specific "filament spool" node connected to a "workbench" node via an "on top of" edge).
Additionally, a growing body of research focuses on recognizing the user's current activity and inferring the user's goal via first person sensory data [27,32,47]. While these methods are suited for spatial computing use cases, thanks to its inherent first person sensor data, goal inference is left out of the scope of this thesis. We thus assume that the user's goal is known *a-priori*.

While these semantic scene graphs were originally developed to help autonomous robots safely navigate human spaces, recent research demonstrates that Large Language Models (LLMs) possess a strong native proficiency for comprehending scene graphs [43]. This intersection of semantic mapping and AI reasoning provides the ideal foundation for automated, context-aware content reformatting, allowing an LLM to evaluate and reason over a physical environment.

Finally, based on these findings [22, 37, 43], we model the reformatting of legacy content into scene-embedded representations as a graph problem. Specifically, given a user goal, a source document, and a scene graph of the environment, an LLM is utilized to iteratively generate graph additions. The LLM constructs both the new nodes, which contain source content reformatted to align with the user's goal, and the semantic edges that establish their relationships within the environment.

EmbedLLM

To address this research challenge, we developed EmbedLLM, which is, to our knowledge, the first context-aware content reformatting framework that adapts both content design and representation based on environmental context. Because the framework enables automatic reformatting of legacy documents to scene-embedded alternatives, it can be employed to create different alternative representations of the same piece of source content while only changing the environment and user goal, thus realizing automatic adaptation of the content based solely on the context. EmbedLLM expects three inputs:

1. The source content.
2. A textual description of the user goal.
3. And a digital twin of the environment represented by a Unity scene.

Based on these inputs, EmbedLLM creates a sequential series of steps to guide the user toward their objective. For each step, the system automatically generates text and image panels that are semantically anchored directly onto relevant objects within the physical environment. Users can then seamlessly navigate forward or backward through the steps using the triggers on their Meta Quest controllers. By tightly embedding the knowledge into the environment and reformatting

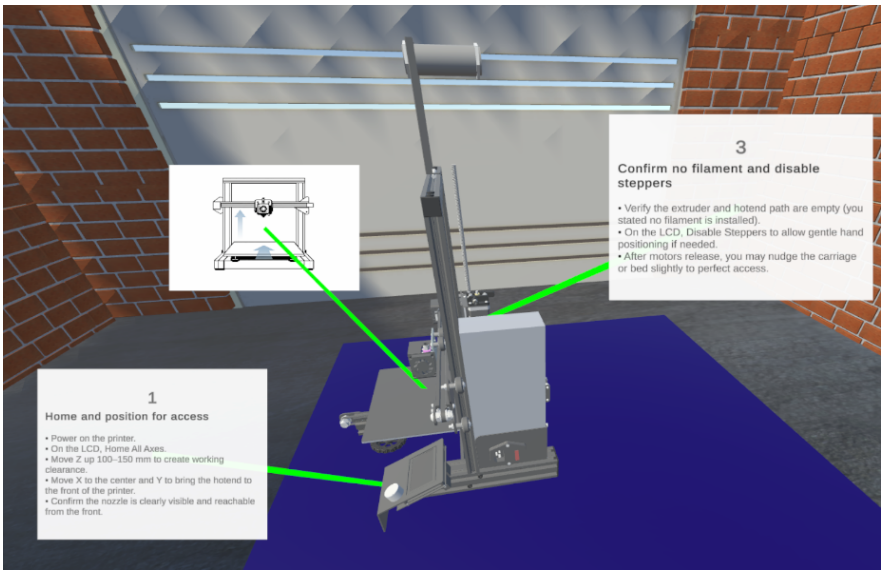


Figure 3: Example output of EmbedLLM for a single step. Here, two text panels and one image panel are anchored to different components of an Ender-3 3D printer. The first panel explains how to home the axes, the image panel depicts the expected state of the printer after this initial step, and the second text panel notes that the filament should not be in place.

it to fit the user's goal, EmbedLLM aims to reduce cognitive load and enhance how effectively the content aligns with both the physical surroundings and the user's ultimate goal.
An example output is shown in Figure 3. The input comprises a source document detailing steps on how to replace the nozzle on an Ender-3 3D printer, the user's goal is changing the nozzle, and the scene is a garage containing an Ender-3 printer. The first step of this alternative representation consists of homing the printer. This step features three panels, two text panels and one image panel, anchored to different components of the printer. The first text panel describes how to home the printer. Subsequently, the image panel shows how the printer should look after successfully executing this first step. Finally, the second text panel notes that no filament should remain in the printer to proceed.

EmbedLLM is implemented in Virtual Reality (VR) using Unity. While deploying in Augmented Reality (AR) would be ideal for physical task execution, doing so introduces visual computing limitations regarding scene understanding. For this reason, we abstract away these technical constraints to focus entirely on the core HCI contributions of the system. Finally, this virtual scene is the only component requiring manual authoring. Once future advancements in computer vision provide the necessary level of detail to dynamically construct and update the scene graph, this manual step will no longer be required.

The system architecture of EmbedLLM is realized through a pipeline comprising three distinct Unity modules: the SceneGraphBuilder, the ArchitectAgent, and the Interpreter (as seen in Figure 4). Rather than attempting to act as a generic context-aware engine across arbitrary domains, the scope of this framework is deliberately constrained to manual, step-based content. Consequently, both the underlying data models and the prompts for the remote LLM are explicitly structured to facilitate scene-embedded representations for step-based content.

Finally, to enable consistent evaluation of the system, the execution must be deterministic. Effectively comparing results across different participants requires that they experience the identical alternative representation. To facilitate this comparison while simultaneously reducing execution time, EmbedLLM incorporates a caching architecture. Because each module depends strictly on the output of its immediate predecessor, a dedicated cache was implemented for

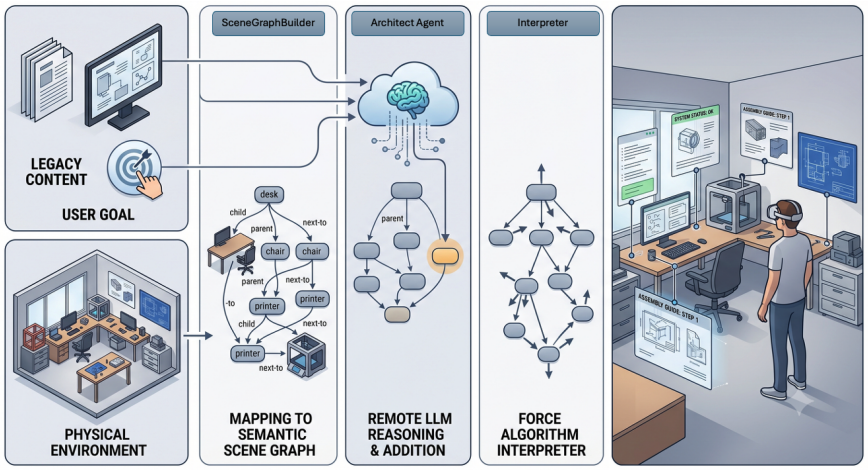


Figure 4: Overview of EmbedLLM. The system takes three inputs, namely: user goal, the source document and digital twin of the environment. The SceneGraphBuilder then converts the virtual scene into a semantic scene graph. This is then served as input, in combination with the user goal and source document, to the ArchitectAgent. Here a remote LLM interprets the inputs and creates new nodes to extend the scene graph. These semantic extensions are then instantiated in the virtual scene by the Interpreter, resulting in a scene embedded and goal aligned alternative representation.

each module's output. This data is persisted to disk and retrieved as input during subsequent executions. Notably, the output of the architect agent is also cached, thereby ensuring fully deterministic execution.

The SceneGraphBuilder

The pipeline begins with the SceneGraphBuilder, which establishes the spatial and semantic context of the user's surroundings. Because standard virtual objects in Unity lack intrinsic understanding of how they relate to one another, this module parses the environmental geometry and evaluates spatial properties. Using specific geometric functions and a set of proximity heuristics, it systematically derives spatial relationships such as "on top of", "underneath of", "left of", "right of", "behind of", and "in front of" to construct a comprehensive semantic scene graph representing the virtual canvas.

The ArchitectAgent

This structured semantic scene graph, along with the source 2D content (such as a PDF manual) and the user's explicit objective, is subsequently passed into the ArchitectAgent. Serving as EmbedLLM's central reasoning core, this module provides the interface to a remote Large Language Model (LLM). Crucially, to prevent unguided and unpredictable text output, the module forces the LLM to operate within a strictly defined schema optimized solely for the procedural task domain. Specifically, the LLM generates a list of steps to accomplish the user goal. Then within these steps it can embed content in the environment. However, it is limited to four text and image panels per step. This limitation is based on heuristics with the aim of reducing visual clutter.
Furthermore, to overcome known cognitive load limitations where LLM performance degrades when multiple tasks are compounded into a single prompt, the execution is decoupled into three distinct, sequential calls:

- Call 1 (Step Definition): Prompts the LLM to analyze the source material and delineate a logical sequence of high-level steps required to fulfill the user's manual objective.

- Call 2 (Node Generation): Iteratively executes for each identified step, prompting the LLM to extract source document knowledge and generate new content addition nodes tailored specifically to that step.
- Call 3 (Edge Definition): Commands the LLM to generate the accompanying semantic edges (relations) that dictate exactly how these new content nodes should contextually connect to existing physical objects or structural landmarks in the scene graph.

Here, the first call is executed once, and the second and third calls are executed iteratively for each step identified in the step definition call (call 1). The output additions generated across all sequential steps are ultimately pooled together into a cohesive list of additions. This list is called the build plan, which acts as a structured sequence of extended scene graphs for each step, representing precisely what task information is necessary and where it should be semantically anchored relative to the physical components of the environment.

The Interpreter

The final component in the system pipeline is the Interpreter, which is responsible for parsing the structured build plan generated by the ArchitectAgent and instantiating it visually within the three-dimensional virtual scene. A naive approach could position the panels at the center of their anchor object and move them along the direction dictated by their semantic relations until they no longer collide with any other objects in the scene. However, this approach does not take into account the fact that the generated panels are not registered as physical objects within the environment themselves, which frequently causes multiple panels to be anchored at the exact same spatial coordinates, severely degrading readability and introducing visual clutter. While one could resolve this by appending each panel to the scene sequentially after placement and forcing subsequent panels to take the newly added objects into account, this recursive method relies heavily on rigid structural dependencies and can result in unnecessarily long chains of panel-on-panel attachments. Furthermore, a naive approach completely fails to account for user visibility and potential occlusions.

To avoid these layout limitations and prevent panels from overlapping, a force-directed layout algorithm is required. EmbedLLM implements a novel adaptation of a force-based 3D label placement algorithm [48]. While it is rooted in established research that utilizes physics-based principles of attraction to designated anchor points and repulsion between individual labels, it is uniquely novel because it introduces dedicated semantic forces into the model. By evaluating the forward vectors of both the anchor and the panel, the system calculates a directional pull specifically tailored to satisfy the underlying semantic constraints defined in the build plan.

The entire workflow of the Interpreter operates in consecutive steps. First, the module establishes an initial placement for each panel as a heuristic "best guess" by translating it outward from the center point of its anchor object along the directional vector of the semantic relation until it clears environmental collisions. This collision-free state serves as the starting position for the force-directed algorithm. Next, the system utilizes Unity's native physics engine to dynamically simulate and apply the attraction, repulsion, and semantic forces. Given the static nature of the target environment, the algorithm executes until the panels reach a state of spatial equilibrium, at which point the physics calculations are halted to improve performance and finalize the context-aware representation.

Evaluation of EmbedLLM

To gauge the effectiveness of our approach in addressing the research challenge, we evaluate EmbedLLM through two distinct user studies. First, to assess the system's ability to handle the first part of the research challenge, we measure the extent to which the alternative representations generated by EmbedLLM are perceived as aligned with the physical environment

and user goals. Additionally, we examine user preferences by benchmarking these representations against a traditional smartphone baseline. Second, to address the subsequent part of the research challenge, we evaluate EmbedLLM's alternative representations during active task execution, aiming to quantify their impact and effectiveness in providing real-time task support.

The first user study is conducted via an online questionnaire, where participants evaluate a series of alternative representations alongside a single baseline condition. Each representation is presented using a short textual description of the user's goal, the physical scene, and the source document, accompanied by a set of images depicting the alternative representation itself. For each scenario, participants rate how well the representation aligns with the environment, evaluate its alignment with the user's goal, and finally indicate their preference for utilizing the alternative representation over the traditional smartphone baseline.
The results of this questionnaire highlight a significantly higher environmental alignment for the alternative representation compared to the baseline. Goal alignment remains consistently high across both conditions, as the baseline also scores well in this metric. Interestingly, the user preference data reveals that not every combination of scene, goal, and content is equally preferred for scene embedding. Notably, tasks that require a higher level of spatial awareness (such as repairing a bicycle) are strongly preferred for scene embedding. In contrast, tasks involving lower spatial awareness (such as gathering supplies from a supermarket) show a decreased preference when compared to the traditional smartphone baseline.

For the second user study, participants take part in a session lasting approximately 50 minutes. During this session, they execute two repair tasks on an Ender-3 3D printer: one using a baseline condition consisting of a floating PDF window in Virtual Reality (VR), and the other utilizing EmbedLLM. To facilitate the tasks, both conditions are augmented with a video passthrough mode. Participants review the digital content in VR and toggle passthrough mode to interact with the physical printer. After executing each task, participants complete a custom questionnaire evaluating environmental and goal alignment, alongside the NASA-TLX to measure cognitive load. Task execution times are also recorded.
The results of this user study confirm that EmbedLLM creates alternative representations that are highly embedded within the scene and tightly aligned with the user goal. However, during active task support, EmbedLLM also leads to significantly higher cognitive load and longer task execution times. Qualitative findings and participant feedback indicate that this may stem from complex terminology, as none of the participants possess prior experience with 3D printers. Additionally, participants report feeling compelled to follow the generated instructions strictly step-by-step, which limits their problem-solving flexibility when compared to the more familiar and trusted baseline format.

Conclusion

This thesis investigates methods to adapt legacy, 2D manual-task-based content into context-aware, scene-embedded alternatives for spatial computing use cases. We propose EmbedLLM, which, to our knowledge, is the first framework to automatically re-author legacy content into a scene-embedded alternative. Furthermore, we evaluate EmbedLLM based on user preference, goal alignment, and environmental alignment, and assess its task-support capabilities during manual task execution. These evaluations demonstrate that environmental and goal alignment are rated highly and that users prefer EmbedLLM over the traditional baseline for specific scenarios. However, the findings also indicate that not all combinations of scene, content, and goal are equally suited for context-aware adaptation. Furthermore, utilizing EmbedLLM during active task execution increases cognitive load and task execution times.

However, EmbedLLM is limited by its high generation times and remote LLM token usage. In addition, both evaluations used to assess the system's qualities are limited in terms of participant selection. The preference questionnaire lacks strict participant screening and does not measure participants' digital literacy. Furthermore, the task-support user study recruits partic-

ipants with no prior 3D printing experience, whereas the generated alternative representation does not explicitly take into account the user's skill level. While user experience falls outside the boundaries of the context scope defined in this thesis, the user's goal of a novice in this scenario is not confined solely to executing the repair, but also includes learning the terminology. Finally, an evaluation of the system's performance with larger pieces of source content or more complex tasks is lacking.

Future work should expand on EmbedLLM by investigating methods to create a unified input model for context-aware Augmented Reality (AR). In addition, it should extend the scope of context and investigate methods to generalize these alternative representations beyond task-based source content. Furthermore, integration with embodied reasoning models, such as Gemini's robotics frameworks, should be explored to ground generation within physical execution environments. Finally, future research should investigate methods of integrating text-to-3D generative models, enabling rich alternative representations that incorporate dynamic 3D models and visual animations.

Contents

Chapter 1

Introduction

1.1 Motivation

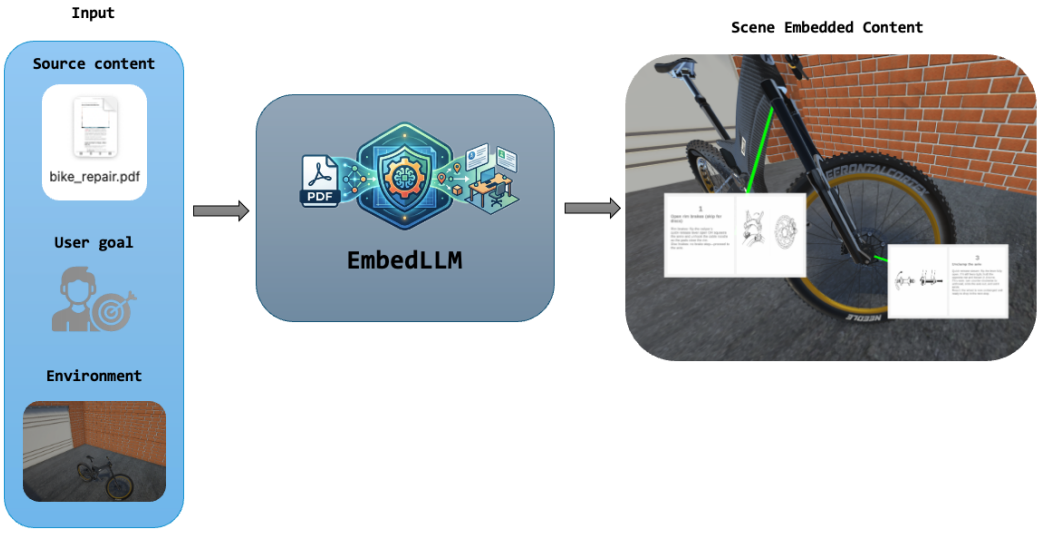


Figure 1.1: Procedural step-based knowledge captured in legacy 2D content turned into a context-aware scene-embedded representation of the same knowledge.

Current Augmented Reality (AR) and spatial computing systems frequently inherit the traditional desktop WIMP (Windows, Icons, Menus, and Pointers) interface paradigm, simply projecting it into three-dimensional space. For instance, applications on the Apple Vision Pro[1] are primarily displayed within conventional 2D windows. Although these windows float within the physical environment, such systems underutilize the true potential of the spatial computing paradigm. However, these platforms possess the technical capacity to present content in a more spatially and contextually embedded manner. The Apple Vision Pro, for example, introduced a novel type of window defined by a three-dimensional volume, which can host arbitrary 3D content. However, generating this content still relies heavily on manual authoring. Another prominent example is the Meta Quest[2] ecosystem. Comparable to the Apple Vision Pro, this platform affords developers the capability to author fully immersive and spatially-aware virtual content. However, due to the inherent complexity of three-dimensional assets, authoring

[1] https://www.apple.com/apple-vision-pro/
[2] https://www.meta.com/be/en/quest/

AR content is significantly more time-consuming than creating traditional 2D media, such as PDFs or web pages. While valuable knowledge is already present within traditional 2D sources (such as PDF repair manuals or HTML setup guides), converting this existing knowledge into augmented reality representations embedded in the scene (as depicted in Figure 1.1) requires a significant manual re-authoring effort.

Additionally, in everyday activities, users do not merely consume content, they coordinate their attention across tools, the environment, and knowledge sources. For example, mechanics executing repair tasks must frequently alternate their attention between their tools, the machinery they are working on, and the manuals providing the necessary information. These frequent context switches elevate workers' cognitive load and prolong execution times. Tightly integrating these knowledge sources directly into the workspace could eliminate this friction, thereby streamlining task execution and improving how technical knowledge is transferred to workers. Augmented Reality (AR) provides a highly effective medium for this integration. Specifically, in task-based repair and maintenance workflows, AR has been proven to reduce workers' cognitive load, task localization times, overall execution times and error rates [2, 3, 17, 44]. Given these advantages, translating technical knowledge captured in traditional knowledge sources into AR content embedded in the scene is exceptionally valuable.

More recently, the enormous volume of research in Large Language Models (LLMs) and other foundational models has shown promising results in the fields of reasoning and content reformatting [23, 46]. This thesis investigates the use of these models to automatically reformat and reason about traditional 2D knowledge sources, with the goal of creating scene-dependent, environment-embedded, and goal-aligned representations for spatial computing purposes.

1.2 Research Challenge

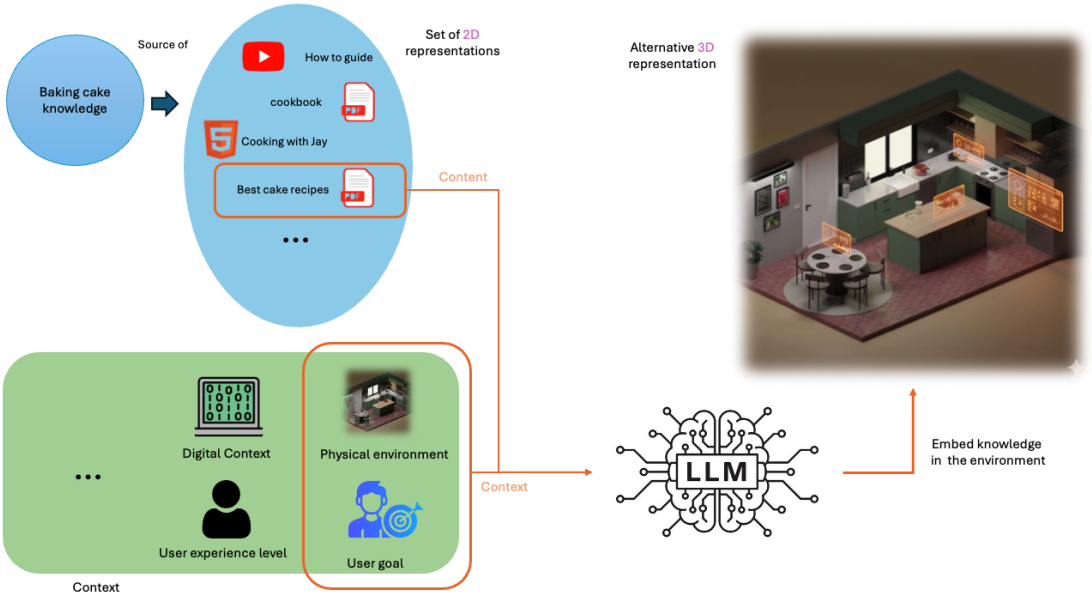


Figure 1.2: High-level depiction of the research challenge to be investigated in this thesis. One of many representations of a piece of knowledge, in this case a guide on how to bake a cake, is reformatted to a three-dimensional alternative representation better suited for XR task execution. This reformatting takes into account the content itself, the scene and the user goal.

To the best of our knowledge, there is currently no framework that extends LLM generation to incorporate contextual data for creating alternative, scene-embedded knowledge representations. While developing a generalized, context-aware XR embedding framework for 2D content

is a promising direction, it requires extensive investigation and ultimately falls outside the scope of this thesis. For this reason, we limit the context to the user's environment and goal. Limiting the scope is necessary due to the inherent complexity and vastness of the domains involved. In addition to the environment and goal, user context includes factors like calendar events and user proficiency, among many others. In parallel, traditional 2D content encompasses a massive spectrum of formats, ranging from static text documents like PDFs and HTML pages to highly dynamic rich media, including video and interactive applications. Additionally, while LLMs enable universal text generation, development of a universal 3D spatial positioning system based on this reformatting is challenging. Furthermore, establishing a universal framework for content-to-context mapping is highly challenging because the type of information within the content heavily dictates the suitable alternative representations and thus its spatial layout. For example, even within a single standardized format like a PDF, the spatial representation required to effectively guide a user through a step-by-step cooking recipe differs fundamentally from the representation suited for reading a continuous narrative, such as a fictional story. Furthermore, the reformatting of this information is highly dependent on both the underlying knowledge and the context. For example, if the content provides an in-depth guide to repairing bicycle disc brakes, but the physical scene features a bicycle with rim brakes, the system should adapt the instructional content to match the scene. Conversely, if the scene lacks a bicycle entirely, a virtual bicycle could be introduced, thereby adapting the spatial context to fit the content. Attempting to generalize a solution across such expansive and variable domains would exceed the scope of this work. For this reason, the scope of this thesis is confined to manual, step-based task execution guides presented in a traditional 2D PDF format. Additionally, the contextual scope is restricted to a scene description and a specific user goal, namely, the execution of the task at hand. Finally, utilizing text-to-3D models could significantly enhance context-embedded knowledge transfer. However, exploring this area would substantially broaden the scope of this thesis and is therefore excluded from the current study. A high-level overview of the scoped research challenge is depicted in figure 1.2, whereby the selected knowledge source is already constrained to a specific step-based guide. This informs the following research challenge:

How can step-based manual instructions presented in a PDF format be automatically embedded in and adapted to the environment and user goal using Mixed Reality and what effect does this have on supported task execution?

While the concept of scene embedding originated from XR, this thesis focuses primarily on Human-Computer Interaction (HCI) contributions, the underlying visual computing challenges are placed out of scope. Specifically, developing such an alternative representation system in Augmented Reality (AR) is currently impractical, as the research would be hindered by the limitations and noise inherent in modern computer vision and scene understanding. Consequently, this thesis abstracts the problem into Virtual Reality (VR). This approach allows us to isolate and evaluate the HCI contributions within a controlled, idealized scenario where all environmental information is known and readily accessible.

The structure of this thesis comprises seven primary chapters. Chapter 2 provides a comprehensive review of the current state of the art, identifying critical research gaps. These gaps are subsequently addressed in chapter 3 and the implementation of EmbedLLM is detailed in Chapter 4. Following this, Chapter 5 describes the hypotheses to address the research question and two different evaluations used to evaluate EmbedLLM on user preference and during task support. After which the results are discussed in chapter 6 and future work is discussed in chapter 7. The thesis concludes in Chapter 8 with a summary of findings and a discussion of their implications.

Chapter 2

Related Work

The concept of spatializing digital content has been extensively explored in prior research. Early systems, such as Magic Cap[1] and Packard Bell Navigator[2], situated desktop computing within a virtual house metaphor. These interfaces spatially segregated tasks. For instance, music and video playback were confined to a virtual multimedia room, while document editing occurred in a virtual office. Switching between these activities required point-and-click mouse interactions, forcing the user to virtually "walk" from one room to another to access different tools. While appealing to novices, this approach ultimately introduced navigation overhead and constrained workflows [16]. In contrast, spatial computing has the ability to abandon the desktop metaphor in favor of scene embedded content constrained by the user's physical surroundings. By grounding the interface in physical reality, this approach facilitates a more seamless interaction with the workspace, which effectively minimizes context switching, reduces error rates, and lowers overall execution times [2, 3, 17, 44].

2.1 Human Computer Interaction Foundations

A context aware content representation embedded in the environment needs to be grounded in Human Computer Interaction (HCI) fundamentals. Specifically, when focusing on task execution support, an understanding of tasks and activity needs to be in place. Activity theory dynamically organizes workspaces according to the user's immediate task [18]. While this thesis focuses on content embedding in the environment instead of organizing existing content, it also focuses on adapting this content to the user's goal and activity, and thus should order the knowledge in the environment based on the current task. Furthermore, context aware systems need to adapt unobtrusively to the evolving environment of the user. When the task of the user changes, a context aware system needs to adapt to this changed activity instead rigidly staying focused on the previous goal. This concept is called situated action [14]. For example, when a user is changing the tire on a bicycle but then suddenly breaks the fork on which the wheel rests, the system should adapt to this new goal. The next step should fix the fork before it can place the new wheel. Furthermore, embedding content directly into the user's environment aligns with the core tenets of ubiquitous computing. Mark Weiser argued that technology should weave itself into the fabric of everyday reality until it is no longer a distinct object of focus for the user [42]. By following this logic a context-aware embedding system allows digital knowledge to merge with the physical world so that the interface itself essentially disappears from conscious thought. The HCI fundamentals discussed lay the groundwork for a framework in which the environment acts as a canvas, knowledge represents the content situated upon it, and attention is viewed as a resource distributed across the entire scene.

[1] https://blog.decryption.net.au/posts/magic-cap.html
[2] http://www.win3x.org/win3board/viewtopic.php?t=14972&view=min

This integration of fundamental HCI principles with XR content embedding aligns closely with established research in the field. Henderson and Feiner researched the benefits of AR documentation in the context of repair task execution [17]. The system developed for this study used text labels, arrows and animated sequences to support task comprehension, localization and execution during a military maintenance job. By anchoring instructions directly to the physical environment, their system allowed mechanics to locate tasks much faster when compared to baseline conditions using standard laptop-based electronic manuals or untracked head-up displays. Furthermore, the study highlighted that participants found the embedded virtual content to be a highly satisfying and intuitive way to execute complex maintenance procedures. Additionally, a significant difference in head rotation was observed when comparing the proposed AR system to the baseline. This implies that AR supporting task execution can have a positive effect on the physical workload of the operator. However, they also highlight the importance of researching the tradeoff between lowered head movement and the additional head strain of wearing an HMD. Importantly the system design was heavily influenced by a cognitive framework. The authors identified that manual repair requires workers to build a mental model of the equipment and mentally transpose instructions onto the physical world. The proposed application automatically executes this transposing step to save time and reduce mental workload. We now commonly refer to this mechanism as cognitive offloading [33].

Other work explores the use of physical objects to manipulate digital content. Schmidt et al. introduce Kickables, a framework that enables the control of digital content using physical objects that the user manipulates using their feet [34]. Kickables builds upon the work of Ullmer and Ishii who investigate the coupling of digital content with the physical environment through the metaDESK system [39]. Their research explores controlling virtual graphical user interfaces via tangible physical objects. To demonstrate this they build a Tangible User Interface desk prototype and evaluate it using a geographical application. By replacing traditional graphical icons with physical objects, such as a tangible model of the MIT dome, they allow users to anchor and manipulate a digital map of the campus. Moving this physical icon correspondingly updates the map position. Furthermore introducing a second physical building model allows the user to scale or rotate the map by moving the objects relative to one another. This spatial relationship leverages physical proximity constraints to provide a highly natural method for interacting with digital information. Furthermore the metaDESK instantiates the traditional GUI window into tangible physical forms. This is achieved using an active lens, an arm mounted screen displaying a three dimensional view, alongside a passive lens, a framed transparent surface placed on the desk to reveal secondary overlays. Both of these tangible windows provide the user with alternative perspectives of the digital space. Although Ullmer and Ishii prioritize interaction over content representation for context-aware AR, their work introduces the principle of using the inherent spatio-temporal qualities of the physical world to manage digital information. Such an approach provides a strong foundation for investigating the advantages of knowledge that is embedded directly within an environment.

2.2 Context Understanding: Sensors to Scene Graph

As this thesis is situated within the Human-Computer Interaction (HCI) domain of context-aware extended reality, establishing a clear definition of context and the data structures required for its representation is paramount. This section highlights the State Of The Art in scene processing and context representation.

Before a system can start representing or understanding the context, it first needs a way to recognize this context. In the context of AR, Simultaneous Localization and Mapping (SLAM) methods use device sensors to create a mapping of the environment and position the system within this environment [35]. More specifically, AR systems primarily rely on visual SLAM methods. These methods use RGB(D) cameras and IMU sensor data combined with visual computing techniques to facilitate the scene mapping and system localization [35]. Based on the scene perception enabled by SLAM, two distinct classes of scene representation models can

be identified: (1) raw and dense representations and (2) abstracted and descriptive representation [22]. Dense, raw scene representations closely mirror the physical environment, inherently demanding substantial data and computational resources for construction and interpretation. Point clouds, for instance, represent scenes through dense collections of spatial coordinates. However, despite their high physical fidelity, they lack embedded semantic information [22]. In applications such as context-aware augmented reality (AR), recognizing the semantic identity of specific point clusters is crucial. For example, a system must identify which points constitute a 'sofa' to accurately anchor a digital textbook to it. Consequently, researchers have developed semantic point clouds, which assign semantic labels to individual points [31]. While this advancement enables basic semantic comprehension of a scene, it fails to capture inter-object relationships. Nevertheless, dense scene representations remain highly resource intensive.

Conversely, descriptive scene representations distill relevant data from the physical environment to construct a high-level model. While these approaches are significantly less resource intensive, they sacrifice the granular detail inherent to raw, dense models [22]. A prominent example is the scene graph, which utilizes machine learning to identify objects and their spatial or semantic relations within a single image [7]. This results in a graph structure where nodes represent objects and their semantic attributes, and edges denote the relationships between them (for example, a 'chair' node connected to a 'table' node via an 'in front of' edge). Although scene graphs successfully capture inter-object dynamics while reducing computational overhead, static image-based models are ill-suited for augmented reality (AR). AR relies on continuous video stream processing. While video-based scene graphs exist, they frequently fail to preserve the crucial spatial and temporal continuity between consecutive frames. This critical gap has driven recent research toward the development of 3D scene graphs. Kim et al. [22] proposed the 3D scene graph and an accompanying framework that processes continuous RGB-D video streams to construct a 3D scene graph, successfully capturing 3D object positions, semantics, and relational dynamics. Building upon this foundation, Hughes et al. [20] introduced Hydra, a real-time spatial perception system that extends 3D scene graphs with hierarchical abstraction. Through advanced incremental spatial perception algorithms—such as extracting topological maps of places and dynamically segmenting them into rooms—their system structurally organizes the environment into ascending levels of abstraction, ranging from macro-level buildings down to rooms, places, and individual objects. While initially developed to facilitate scalable scene understanding for autonomous robots [4, 20, 22], 3D scene graph models translate effectively to augmented reality applications.

Recent work by Yang et al. [43] explores the integration of Large Language Models in interpreting and generating scene graphs. To evaluate this, they introduce the Text-Scene Graph Bench, a novel benchmark assessing LLM proficiency in these tasks. Upon evaluating 11 state-of-the-art models, the authors discover that while current LLMs struggle to generate accurate scene graphs from scratch, they demonstrate strong capabilities in comprehending existing ones. When coupled with 3D scene graph construction, this proficiency in scene understanding paves the way for advanced, LLM-driven spatial reasoning and context-aware content adaptation.

2.3 User Goal Inference

When generating context-aware scene embeddings, the user's underlying goal is of paramount importance. Alongside the physical environment, the user's intent fundamentally dictates the structure and presentation of the resulting spatial representation. A substantial body of research focuses on dynamically deducing this intent through Human Action Recognition (HAR) and user goal inference. Specifically, egocentric sensing for HAR uses first person sensory data (for example RGB and IMU) to recognize actions [27].
Building upon basic egocentric modalities, recent work leverages Large Language Models to interpret user activity. For instance, Rekimoto et al. propose GazeLLM [32], which integrates eye tracking with egocentric RGB video to isolate specific regions of interest within first-person frames. By minimizing the data payload sent to the Multimodal LLM, this approach streamlines

activity recognition, rendering it highly applicable to the resource constraints of Augmented Reality (AR) environments. Related research extends activity recognition to include user goal inference. For example, Zhao et al. [47] investigate the use of LLMs for video-based long-term action anticipation (LTA). They utilize a top-down pipeline where identified actions are aggregated to deduce the actor's objective, which in turn facilitates the prediction of plausible subsequent steps.
This underscores the viability of utilizing egocentric sensing, a prevalent modality in spatial computing, to infer the underlying goals and intentions of the user. However, to support goal inference using foundation models, systems must be capable of processing continuous streams of egocentric sensor data. Huang et al. address this with Vinci [19], a real-time egocentric vision-language model (VLM) designed for embodied smart assistants. To handle constant input streams, they incorporate a memory module indexed by timestamps and textual descriptions, which facilitates temporal grounding. Furthermore, this work introduces the first VLM specifically trained for egocentric video input.
While these works show promising results, user goal inference falls outside of the scope of this thesis. Instead, our work operates under the assumption that user's goal is explicitly known *a priori*. Nevertheless, this limitation highlights a compelling avenue for future work, which should investigate the integration of Human Activity Recognition (HAR) methods to enable situated action in context-aware content reformatting.

2.4 Context-Aware Extended Reality

While previous sections examine theoretical fundamentals and scene understanding in isolation, recent literature increasingly addresses context-aware Extended Reality (XR). Davari and Bowman research a design space for context aware XR interfaces and propose a context aware adaptive content placement strategy [12]. This design space identifies two major fields: (1) content design and (2) presentation design. Here, content design describes the level of detail, availability and focus of the information, while presentation design focuses on the modality, visual elements and arrangement to convey the information to the user. Additionally, they further subdivide the arrangement dimension of presentation design in non-adaptive or adaptive placement strategies. Non-adaptive placement strategies are constructed for a specific scene or context and do not adapt a change in this context (for example body fixed panels). On the contrary, adaptive placement strategies do adapt to a changing environment and context. These adaptive strategies change the placement and arrangement of windows based on the context (e.g a window anchored to a monitor). Furthermore, Davari and Bowman propose a proof of concept system to illustrate an adaptive arrangement strategy, followed by a user study to quantitatively compare this to a non adaptive baseline. The results of this study highlight that in changing environments the adaptive strategy leads to fewer errors, faster navigation times and less cognitive load. Finally, in static environments the qualitative data highlight a preference for the non-adaptive arrangement strategy, however the quantitative data also indicate a better performance for the adaptive strategy in a static environment. While Davari and Bowman highlight promising results for context aware content placement strategies, research in context aware content strategies remains overlooked. Nevertheless, the proposed design space for context aware XR clearly identifies the content design dimension and further underscores the need for research in this field.

2.4.1 Context Aware Presentation

Other works investigate alternative methods for context aware presentation design, Cheng et al. propose InteractionAdapt [9], an optimization-based method to adapt VR workspaces to changing physical environments. Specifically, InteractionAdapt optimizes for: (1) physical affordances, placing UI elements to enable the most suitable input technique; (2) occlusion avoidance, ensuring no UI elements are blocked by others ; and (3) temporal consistency, preferring layouts with minimal changes compared to the original environment. The focus, however, is on

optimizing interaction methods to utilize the changing environment, thereby supporting user posture and passive haptic feedback. Additionally, Cheng et al. describe a user study comparing InteractionAdapt to Surround and Consistency baselines. The Surround baseline places elements relative to the user's position , while the Consistency baseline strictly optimizes for layout similarity. The study consists of two tasks executed in a fixed order. The first task requires participants to select highlighted buttons on virtual panels , while the second requires them to manually adjust the layout to their preference. The results show significantly faster movement times between target buttons , lower selection times , and more frequent use of touch interactions in the InteractionAdapt condition. Finally, during the layout task, participants moved significantly fewer elements compared to the baselines , indicating that InteractionAdapt successfully placed elements closer to the participants' ideal configuration. Notably, prior to InteractionAdapt, Cheng et al. also propose SemanticAdapt [10]. However, SemanticAdapt focuses on optimizing for semantic relations between UI elements and physical objects rather than user interaction and physical affordances. For example, a virtual PDF associated with reading might automatically be placed near a physical notebook. In later work, Li et al. build upon InteractionAdapt with SituationAdapt [28]. This system uses object recognition and LLM reasoning to create an adaptive placement strategy that respects social interactions. SituationAdapt would thus focus on not blocking a person's face or a shared point of interest. While it is never explicitly stated, the research of Cheng et al. also closely fits into the context aware XR design space described by Davari and Bowman [12]. Furthermore it also focuses on the adaptive arrangement techniques of the representation design aspect from the design space.

In related work, Lu et al. investigate different application placement methods for augmented reality (AR) environments [29]. They specifically focus on spatial transitions, aiming to support continuous information availability as users move between physical locations and engage in various activities. The authors argue that prevalent industry systems are restricted to single-environment information retrieval, limiting interactions in mobile scenarios. Current systems typically anchor windows in their original physical locations, which requires significant manual effort to reopen and arrange applications when the user needs to access information in a new space. Additionally, while some devices (such as the Microsoft HoloLens 2) allow an interface to loosely follow the user in mobile scenarios, this feature is restricted to a single window. Furthermore, while some state-of-the-art approaches allow multiple windows to follow the user, Lu et al. note that this approach scales poorly, increasing cognitive load and the risk of occluding crucial real-world information. To address these limitations, they investigate three methods for mobile information orchestration: (1) None: This baseline method mimics current industry standards. When the user switches environments, all previously opened windows disappear, and the user must manually reopen the necessary applications via a menu. (2) Some: This system uses contextual understanding of the new scene to predict and automatically open a subset of relevant applications. It then anchors these windows within the new environment. If the user requires additional information not present in these suggested windows, they must open those applications manually. (3) All: In this condition, the system automatically transfers and places all currently open windows into the new environment. To evaluate these context-aware presentation designs, the authors conduct two user studies. The first study gauges the scalability and performance of the methods by measuring how quickly users could retrieve information using each presentation method. The second study evaluates the methods in a more ecologically valid setting, asking participants to role-play everyday tasks while moving across three physical rooms (e.g., a kitchen, an office, and a living room). Results indicate that the All condition consistently yields faster information retrieval times, whereas the None condition is universally the slowest. However, the Some condition yields better efficiency when the number of open windows is high and system prediction is accurate. Ultimately, the Some condition emerges as the most preferred interface in the realistic role-playing scenario, whereas the All condition is the least favored due to excessive clutter.

In parallel, Lam et al. investigate ways of extending the continuity of Mobile AR (MAR) into our daily lives [24]. They argue that collaboration between MAR interfaces and traditional

desktop workstations could become a common scenario. To investigate this scenario they propose A2W, a three tier framework that facilitates context aware web browsing in AR, context aware recommendations and an integration with the traditional desktop web browser to enable this continuity. The proposed framework uses location data (GPS), visual data in combination with object detection, motion data (IMU) and user preferences and interactions to form an understanding of context. It then uses this context understanding to create a context aware web interface for MAR use cases. More precisely it uses the location data to find relevant web pages in the user's vicinity. In addition the system uses object detection to calculate an area of interest and then makes sure this region is unobstructed by the web panels. Furthermore, when numerous panels are present, the system uses clustering based on categories to avoid visual clutter. The web panels themselves are further inter related based on contextual cues, for example distance to the physical shop and overall relevance to the user. Additionally, when the system recognizes a physical location (for example a restaurant) it will display the relevant information anchored to the restaurant building. Finally the interactions and recommendations encountered in the MAR interface are then leveraged to create a set of recommendations for desktop viewing. In practice, the system takes into account the AR driven journey before reaching this desktop web browser. Although A2W successfully creates a context aware placement of web panels in AR, the main goal of this study is to investigate the use of AR context to enable an efficient continuity between MAR and desktop web browsing. This results in a more limited use of context in AR. For example, the content of the web page could enhance the visual appearance of the shop it referenced to, or be embedded in the environment such as advertisements. In summary, while the web page orchestration is context aware, the web content is not.

2.4.2 Context-Aware Content

While the previously mentioned research focuses on creating context-aware content arrangements, some work adapts the content itself to the context, thus researching context-aware content design. Davari et al. propose a context-aware AR system for Human-Robot Collaboration (HRC) [11]. They argue that in industrial settings, where a worker's hands are frequently occupied and their attention is focused on the task rather than the cobot, traditional HRC methods are restrictive and require interrupting the workflow. To address this, they propose an AR interface for HRC. While the proposed framework does incorporate content design adaptations based on the context, its understanding of context is strictly limited to procedural information. More precisely, the system recognizes the current step in the process and combines this with the location of task-relevant objects to represent the context. If a coworker were to sit on the workbench, the system would therefore not recognize this as a change in context. Additionally, the proposed framework still requires manual configuration for each independent HRC task. While the core AR system remains consistent, re-purposing the system for a different sequential task requires authoring new procedural steps, mapping new task-relevant robot actions, and redefining the spatial locations of physical objects. Nevertheless, while the system is constrained in its contextual representation and requires manual authoring, it successfully alters the content design itself based on specific contextual changes.

In other work, Behravan et al. investigate the use of Generative AI (GenAI) to create context-aware content in AR [5, 6]. Here, a Vision Language Model (VLM) is used to enable environmental understanding, it then suggests contextually relevant objects to be placed within the scene. The suggestions from the VLM are then fed into a text-to-3D GenAI model to create context-aware 3D models. While this work is still ongoing, and thus a user study evaluating the proposed system has not yet been conducted, the authors do mention some possibilities for future work. They explicitly state that the research and development of an automatic 3D-model placement algorithm is needed to enable fully autonomous content creation. This current limitation of the system is also something we encounter in this thesis, and it will be elaborated upon further in section 4.1.3. Finally, the proposed framework is tailored for AI-supported, context-aware content creation. While this would lower the authoring cost of transitioning

knowledge captured in legacy content sources, it still requires the manual re-authoring of this content. Additionally, it is only suited for 3D object generation and thus lacks the ability to convey context-aware content via media other than 3D models.

Related is LLMER [8], a novel framework to create intractable LLM generated XR environments without sacrificing reliability that is associated with LLM code generation. The novel framework proposes the use of structured JSON data to abstract the code generation and facilitate robust LLM scene construction. Additionally, the proposed framework leverages this abstraction to enable context aware content authoring. The user inputs a descriptive prompt via natural language, this is processed by the LLM which outputs a structured JSON response, this response is interpreted to create a context aware interactive scene. Interestingly, an evaluation shows that this method of abstracting code generation away from the LLM and into a sort of interpreter results in lower token usage and faster response times. These results are extensively leveraged in the proof of concept implementation of this thesis, discussed in chapter 4.

Analogous to LLMER [8], Lee et al. introduce ImaginateAR [25]. The tool facilitates seamless, *in situ* AR scene authoring by combining offline scene understanding, LLM-driven natural language interaction, and a custom text-to-3D pipeline for generating bespoke assets. To evaluate the system, the researchers conduct a user study comparing three authoring paradigms: manual, AI-assisted, and AI-decided. Their findings demonstrate that users favor a hybrid authoring approach. While participants rely on the AI-assisted mode for brainstorming and asset generation, they consistently prefer manual tools for the precise fine-tuning of spatial placement. Finally, the authors address system limitations, highlighting that the reliance on simplistic bounding box representations for environment mapping occasionally leads to spatial inaccuracies during model placement. Furthermore, the methods proposed for local scene representation and improved LLM scene comprehension are implemented in the proof of concept for this thesis and discussed in Chapter 4. Finally, while the proposed tool leverages LLMs for natural language processing and a custom 3D-model pipeline, it lacks the ability to create information-dense AR content, as it is limited to generating and contextually placing 3D models.

Similar to LLMER [8] and ImaginateAR [25], CARING-AI [36] uses Large Language Models to enable context-aware content authoring. The proposed system uses GenAI to create context-aware humanoid avatar visualizations that showcase instruction execution. To realize this, the system uses object detection and a custom Motion Diffusion Model to generate animations accurately situated within the environment. The authors evaluate CARING-AI [36] in a user study and find that participants experience significantly lower cognitive load and physical demand compared to a Programming by Demonstration (PbD) baseline. Finally, when compared to LLMER [8] and ImaginateAR [25], CARING-AI has an increased capability to portray denser information, as it allows for animated demonstrations and is inherently step-based.

Another work that aims to enable natural context-aware AR experiences is Retargetable AR [37]. Proposed by Tahara et al., this novel framework seamlessly adapts virtual content to varied physical environments. It relies on online scene processing to construct a 3D semantic scene graph of the user's surroundings, ultimately placing AR content by satisfying predefined semantic relationships rather than relying on exact geometric coordinates. For example, if a content author creates a scene where a digital character sits on a chair to watch television, the author must explicitly define the required context. In this case, a chair facing a TV screen. When an AR user enters a new physical room, the system maps it online to generate a real-world scene graph. The system then matches the author's defined semantic relations to this real-world graph. Consequently, the digital character is positioned based on semantic labels and spatial relationships rather than absolute real-world coordinates. While this framework alters digital model characteristics, such as orientation and positioning, based on the physical environment, it does not adapt the content itself. The content is inherently context-aware, as it is described entirely in terms of context semantics. For example, if a physical scene does not contain a chair, a true content-design context-aware system would adapt the scenario itself (e.g., the character decides to sit on a table or the floor). Because fallback methods are not

explicitly described in the paper, it can be assumed that the semantic relation solver simply fails, leaving the system unable to display the content in that specific scene. For this reason, Retargetable AR should be classified as a presentation-design context-aware system. It adapts the presentation and placement of the content based on the scene, but it does not alter the underlying content based on the context.

All the research described in this section involves content that is either explicitly authored for a specific environment, making it context-aware only within that setting, or relies on context-aware presentation design. However, to the best of our knowledge, no prior work has investigated true context-aware AR which incorporates content and presentation design as defined by Davari and Bowman [12]. The advent of Large Language Models (LLMs) presents a unique opportunity to explore this gap. Because LLMs have demonstrated promising capabilities in reasoning, scene comprehension, and textual reformatting, leveraging these models to dynamically adapt content design based on situational context warrants active investigation.

2.5 Sense-Making of User Interfaces

Finally, another body of work researches the computational sense making of graphical user interfaces (GUIs), focusing not on Augmented Reality or spatial computing environments, but rather on traditional WIMP (Windows, Icons, Menus, Pointer) paradigms. Although these approaches operate entirely outside the context of XR embeddings, their underlying methodologies for automating GUI interactions remain highly relevant. Specifically, the outcomes of this body of work facilitate a deeper computational understanding of interactive UIs. This capability could be leveraged to extend the scope of immersive content, transitioning from traditional static media to fully interactive applications.
For instance, Li et al. [26] address the challenge of 2D interface comprehension through Screen2Vec. This self-supervised technique generates rich GUI embeddings by capturing the underlying structural and visual semantics of flat screen layouts. By converting interface components into computational representations, their method successfully facilitates a range of downstream applications, including nearest neighbor search, intelligent design aids, and the automation of GUI reformatting. Jiang et al. [21] argue that previous methods fail to account for the visuo-spatial relationships among GUI components. To address this, they propose Graph4GUI, a framework for the automatic processing of GUIs in various downstream tasks (e.g., GUI design suggestions). By modeling the interface as a bipartite graph and processing it via a Graph Neural Network (GNN), their framework captures intricate visuo-spatial and semantic constraints that traditional linear embeddings overlook. Expanding on this focus on interface comprehension for task automation, Veuskens et al. [40] introduce Rataplan, a resilient framework for automating sequences of actions in modern GUIs. Rather than relying on underlying structural hierarchies or sometimes incomplete accessibility APIs, Rataplan utilizes a purely pixel-based reverse engineering approach, making it inherently applicable to any application that renders a visual interface.

The recent emergence of agentic AI has catalyzed a resurgence in UI understanding research, as autonomous agents increasingly require the ability to navigate and manipulate standard graphical interfaces. Addressing this challenge, Wang et al. [41] investigate the use of text-based LLMs for conversational agents designed to control third-party mobile UIs. To overcome the models' inability to process rich visual input, their framework utilizes the Android view hierarchy to construct a linearized, text-based abstraction of the interface. This approach transforms the hierarchical structure of a mobile screen into a semantic format, enabling the LLM to interpret the UI for natural language dialogue and task execution. Furthermore, a literature survey by Zhang et al. [45] categorizes the landscape of LLM-based multimodal agents capable of executing complex, multi-step workflows within third-party GUIs. To successfully execute these workflows, agents require a robust understanding of the interface, including available actions and their specific spatial locations. The surveyed agents achieve this environmental perception

by synthesizing pixel-level visual data (screenshots) with structural hierarchies (widget trees). This dual-modality approach allows the agents to ground high-level natural language instructions into precise visuo-spatial coordinates, such as bounding boxes. While many frameworks leverage general-purpose multimodal models, Zhang et al. note an emerging trend toward Large Action Models (LAMs), architectures explicitly fine-tuned for UI interaction. Nguyen et al. [30] further delineate the evolving landscape of UI perception, noting a transition from structural-only methods (e.g., DOM or accessibility trees) to vision-centric and hybrid models. This shift reflects the necessity for agents to interpret GUIs as visual canvases rather than just hierarchical data structures, allowing for greater adaptability across diverse platforms and legacy systems where metadata may be unreliable. While API-based methods are more efficient for agentic interaction, not all third-party applications provide an API for every task. Thus, visual canvas interaction enables necessary generalization. Nevertheless, all of the aforementioned approaches rely on foundation models to create a computational understanding of the GUI.

Although these emerging hybrid and vision-centric models provide a robust foundation for interpreting interactive interfaces, integrating live GUI orchestration exceeds the current scope of this research. This thesis deliberately limits its focus to static legacy content in the form of step-based manual instructions from PDF documents. However, the intersection of agentic UI perception and context-aware spatial computing presents a promising avenue for subsequent studies to explore the automatic translation of traditional 2D WIMP applications into fully immersive, interactive experiences.

Chapter 3

Iterative design process and system overview

3.1 Design Goals and Requirements

The field of context aware content, as described by Davari and Bowman [12], is still in its early stages. As highlighted in the previous chapter, most research focuses on creating content for a specific context, or adapts the spatial layout based on the context but leaves the content unchanged. In this thesis, we present EmbedLLM, to our knowledge, the first framework that adapts both the spatial layout and the content itself based on the context.
The requirements for a framework that automatically parses traditional PDF content to a context aware scene embedded alternative representation are threefold:

- EmbedLLM needs to be able to interpret the environment, the user goal and the source content (PDF).
- EmbedLLM needs to be able to filter and reformat the source content based on the user goal and environment.
- EmbedLLM needs to be able to instantiate this reformatted content in the environment to realize scene embedded content.

Furthermore, to ensure reproducibility and consistency during evaluation, EmbedLLM must save or deterministically regenerate identical alternative representations.

3.2 Iterative Development Process

Translating static source content into situated, context-aware representations by means of Large Language Model (LLM) reasoning requires bridging the gap between LLM semantic understanding and spatial execution. Because the extent to which LLMs can autonomously handle these combined reasoning tasks is not yet fully established, jumping directly into a final system implementation presents significant risks. Therefore, an iterative, prototype-driven development process is adopted. Before finalizing EmbedLLM's architecture, preliminary proof-of-concept prototypes are developed to evaluate the LLM's scene graph matching capabilities. These early iterations allow us to isolate variables and systematically determine which reasoning tasks can be effectively offloaded to the model, and which require rigid system constraints.

3.2.1 Early Proof of Concept 1: Initial LLM Reasoning

To evaluate scene graph reasoning, the first proof of concept utilizes three distinct, self-contained 3D representations of the source content. Manually constructed scene graphs for each representation are supplied to the LLM, alongside the original content, the environmental scene graph, and the user goal. The LLM is then tasked with selecting the most appropriate predefined representation and integrating its scene graph as a subtree at the optimal location within the environment's graph. After which the prototype interprets the addition and places the accompanying 3D representation at the position of its scene graph anchor. In the scenario used for this prototype, the source content is a maintenance guide for cleaning a motorcycle chain. Notably, in instances where the user is located in an environment without a motorcycle and still wishes to consult the content (for example, when sitting in front of a desk or on a couch), the LLM opts for a virtual motorcycle equipped with *in situ* visualizations, strategically anchoring it on the user's desk. Conversely, when a physical motorcycle is present in the environment, the LLM correctly reasons in favor of *in situ* visualizations, anchoring the procedural steps directly onto the real-world object.
Finally, this proof of concept is implemented in Swift for the Apple Vision Pro. Each alternative representation functions as a spatial volume, anchored to real-world coordinates via a set of points corresponding to objects in the scene graph. By doing so, the need to leverage spatial reasoning is explicitly abstracted away, allowing for an exclusive focus on the LLM's semantic and scene graph reasoning capabilities. Although this proof of concept is not exhaustively evaluated, the preliminary results indicate a promising trend regarding the LLM's scene graph reasoning and matching capabilities.

3.2.2 Proof of Concept 2: Agentic AI and Spatial Reasoning Limitations

The promising preliminary results of the first proof of concept confirm the semantic reasoning capabilities described by Yang et al. [43]. Building upon this, the second proof of concept seeks to evaluate the limits of LLM environmental reasoning. To achieve this, we develop a simulation environment within Unity mimicking the agentic AI workflow, featuring a custom Model Context Protocol (MCP) module. A remote LLM is interfaced with this module and tasked with generating a scene-embedded representation of the provided content. Specifically, we supply the model with the source content, the user's goal, and a semantic scene graph encompassing not only semantic relations and object semantics, but also spatial coordinates and bounding box descriptors. Additionally, to gauge the model's creativity, the MCP module includes the functionality to load external 3D models into the scene via a provided URL. When the agent utilizes this tool, the module verifies the link's availability, subsequently downloading and instantiating the 3D model within the environment. The initial findings of this proof of concept demonstrate a clear contrast in the LLM's capabilities. Although the model effectively creates context-aware alternative content design representations, including targeted text reformatting and the instantiation of three-dimensional visualizations to increase user agency, the overall alternative content representation remains inadequate. Specifically, the positioning of text panels and visualizations relative to the scene is suboptimal, and notably, the scale of the models and panels is disproportionate to the surrounding environment.
For example, when generating an embedded alternative representation of a motorcycle chain maintenance guide in a scene lacking a physical motorcycle, the agent correctly reasons to instantiate a virtual chain and sprocket assembly on the user's desk, visualizing the steps via text panels. However, upon actual instantiation, the 3D models are excessively large and positioned far outside the intended physical environment (as seen on figure 3.1). Notably, the guide also includes a section detailing the necessary tools. The agent loads distinct 3D models for each tool and logically infers they should be arranged on a metal plate on the desk, however the spatial execution of this placement is similarly misaligned.

Although the successful integration of 3D models suggests that the agent can generate cre-

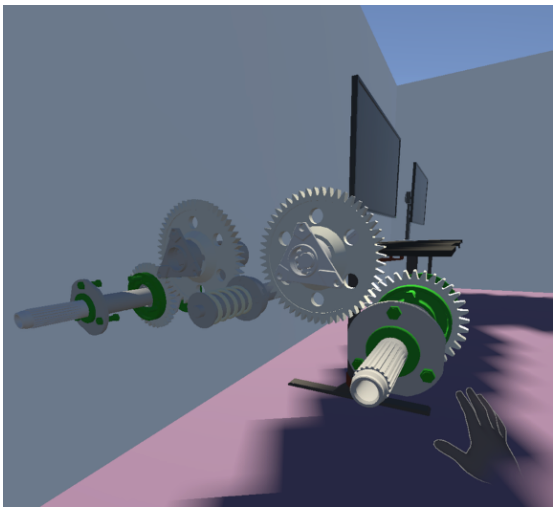


Figure 3.1: The agentic model retrieves a sprocket-and-chain assembly from a remote server on the internet and instantiates it within the scene. However, associated text panels and related models (for example the desk) are severely misaligned, appearing far from the sprocket model upon instantiation.

ative, scene-embedded representations, its deficient spatial reasoning capabilities indicate that fully autonomous agentic scene formatting is currently impractical. Nevertheless, these results remain preliminary. Notably, because creativity, alternative content design, and alternative content representation are not evaluated in isolation, this proof of concept only measures their combined execution within a single prompt. Consequently, the cognitive load limits of LLMs, as described by Adapala [1], may skew these initial findings. Adapala highlights that while LLMs demonstrate robust reasoning capabilities on isolated tasks, their performance systematically degrades when multiple tasks are compounded within the same prompt.
Given that spatial execution emerged as the primary bottleneck in this proof of concept, future iterations of an agentic content-to-context module would likely benefit from integrating the Embodied Reasoning (ER) models discussed previously. Although exploring this integration is beyond the scope of this thesis, further research in this direction promises to yield valuable insights.

3.2.3 Final System

Addressing the research challenge requires establishing a clear mapping between content and context. Inspired by Kim et al. [22], Yang et al. [43] and Tahara et al. [37], we formulate this mapping as a graph-matching problem. By representing the physical environment as a semantic scene graph, we aim to solve this problem by generating an extended graph where source knowledge is integrated as additional nodes and edges. Additionally, this approach should maximize the user's goal, scene coherence, and content coherence. To realize this mapping, we make use of a Large Language Model that iteratively interprets the source content, creates new nodes and edges from the source content and extends the environmental scene graph with these nodes and edges. A high level overview of problem space is represented in figure 3.2. This figure explicitly shows how the input content is but one of many representations for the same knowledge.

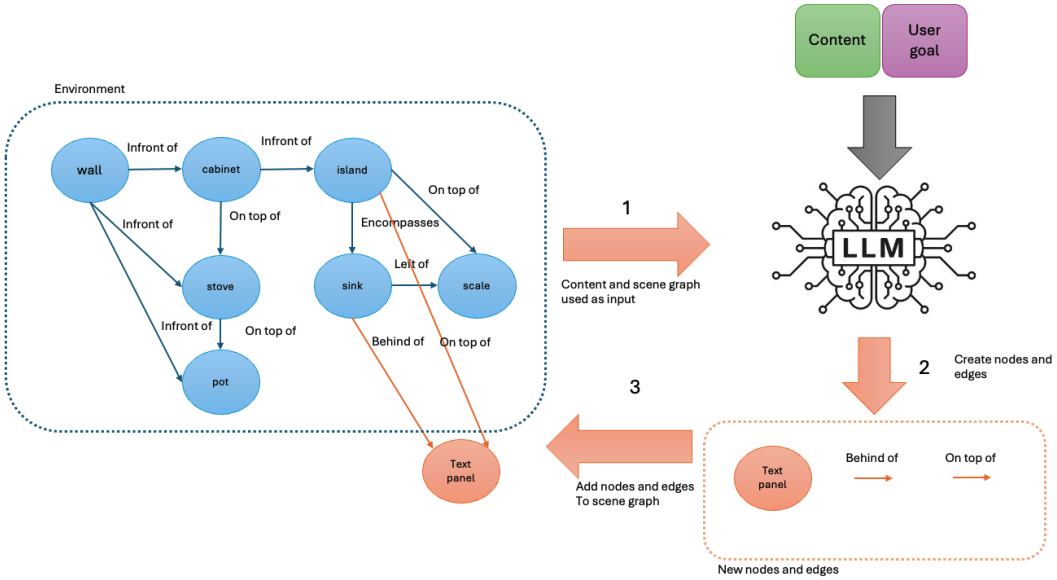
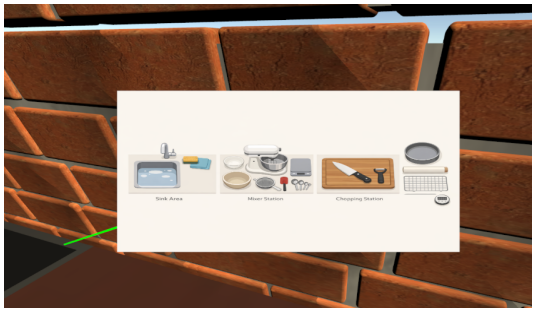


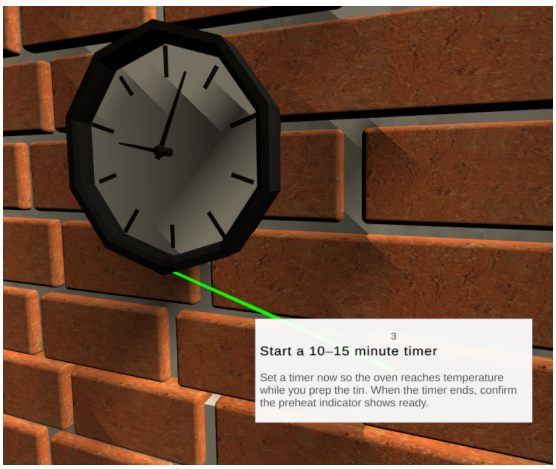
Figure 3.2: A high-level overview of EmbedLLM's workflow. (1) The environment is read by the remote LLM as a scene graph, in combination with the content, and user goal. (2) The LLM identifies the next relevant piece of content based on the user's goal and current state of the scene graph, generating the corresponding node and edges. (3) These elements are appended to the scene graph. (4) This iterative process continues until all necessary content is successfully integrated.

3.3 Walkthrough of EmbedLLM in Different Environments

(a) Overview



(b) Workspace organizer



(c) Reminder

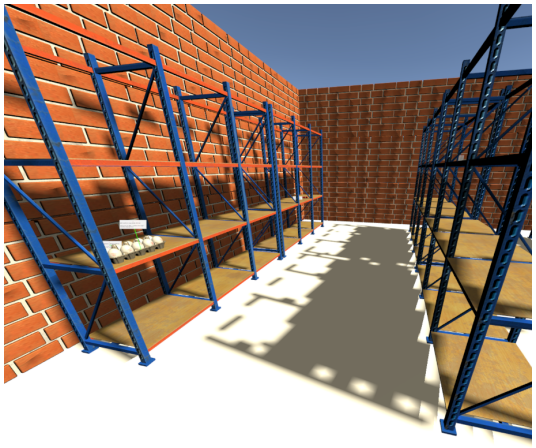
Figure 3.3: Illustration of the initial phase of the generated content within a kitchen environment, where the objective is baking a cake using a recipe. Top left and right: informational panels are spatially anchored to the required physical ingredients and an image positioned at the rear of the countertop demonstrates optimal workspace organization. Bottom: a notification to set a timer is attached to the clock on the wall.

From an alternative perspective, this challenge can be interpreted as a generalization problem within the existing literature. As detailed in the preceding chapter, current research supports the manual creation of content tailored to the physical environment [2, 3, 17, 44]. Consequently, successfully automating this authoring process would allow for the practical realization of context aware content. The following paragraphs detail the behavior of EmbedLLM across diverse environmental contexts and highlights the capabilities of automated scene embedded content generation. By examining these various scenarios, the section illustrates the efficacy of the proposed approach for practical applications.

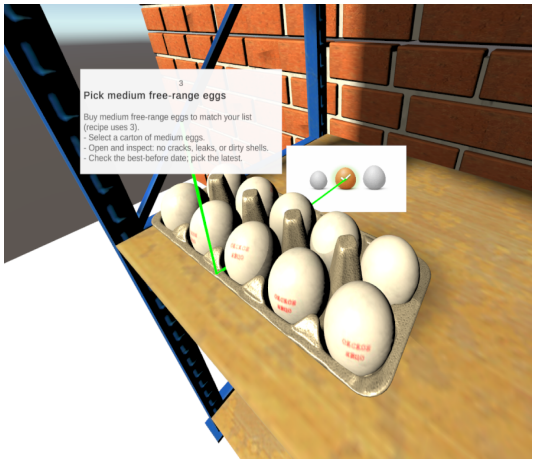
Consider a practical use case in which a user intends to bake an apple cake within a domestic kitchen environment. For this specific scenario, the source content is a recipe, the physical environment is the kitchen, and the overarching objective is baking the cake. Processing these parameters through EmbedLLM yields a scene embedded context aware representation, as illustrated in figure 3.3. The initial phase of this generated content focuses on ingredient collection and workspace preparation.

Specifically, digital panels anchored to physical items like the sugar and flour containers specify the required ingredient types (fig 3.3a). Additionally, an image positioned at the rear of the counter demonstrates the optimal spatial layout for the task (fig 3.3b), while a notification panel attached to the physical clock prompts the user to initiate the oven preheating sequence

(fig 3.3c).



(a) Overview of step 1



(b) Step 1.1: gather eggs

(c) Step 2.1: pick sugar

Figure 3.4: Depiction of the initial phase of content generation situated within a retail supermarket environment. Informational text panels are spatially anchored to the physical ingredients. To optimize the retrieval process, the procedural steps are grouped according to the physical aisles where the items are located. For instance, the first step combines the eggs and apples as they occupy the same aisle. A subsequent step addresses the flour and sugar located in a different aisle. Furthermore, the contextual panel adjacent to the sugar specifies the optimal type required to fulfill the user objective.

Alternatively, consider a situation where the user lacks the required ingredients and must acquire them from a retail location. Under these circumstances, the physical environment transitions to a supermarket and the primary objective becomes gathering supplies, while the source content remains identical. For this context, EmbedLLM produces a different content representation (illustrated in figure 3.4). In this representation, informational panels specifying the required types and quantities of ingredients are anchored to the corresponding physical products on the store shelves (fig 3.4b). Significantly, EmbedLLM optimizes the retrieval process by grouping the ingredients into sequential steps based on their respective aisles (fig 3.4a). For instance, the initial step guides the user to collect the eggs and apples because they are located in the same aisle. The subsequent step then directs the user to a different aisle to obtain the flour and sugar needed for the cake batter. Notably, this specific virtual scene, and consequently the scene graph, does not contain all the required ingredients. The LLM assumes that the baking powder is more likely to be positioned near the sugar rather than near the apples. Thus, it

appends the information regarding the baking powder to the sugar's location. Furthermore, since the user's goal is to bake a specific cake and the source content is the recipe used in the previous scene (Fig. 3.3), the exact measurements required are displayed on the text panels (as seen on figure 3.4c).

To demonstrate the versatility of EmbedLLM, consider an alternative context where a user intends to repair a bicycle within a garage (depicted in figure 3.5). In this scenario, the source content is a maintenance manual, the physical environment is the garage, and the objective is changing a tire. Within this specific representation, informational panels and images are semantically anchored to their corresponding physical components on the bicycle (illustrated in figure 3.5a). The interface depicts various types of brake discs and through axle hubs while detailing the necessary procedures for each variation (figure 3.5b). Additionally, by analyzing the physical bicycle present in the scene, EmbedLLM infers the specific component types the bicycle is most likely equipped with (in this case a disk brake). Furthermore, figure 3.5c depicts a step in which the user is expected to deflate the tire before removal. This panel is anchored to the front wheel, and an image shows how to deflate the valve while a text panel describes the task.
Finally, EmbedLLM is applied to a 3D printer maintenance task within an environment featuring a printer situated on a table. For this specific setting, two distinct combinations of user objectives and source content are evaluated. The first scenario involves replacing the printer nozzle alongside its corresponding instruction manual. The second scenario focuses on changing the filament, similarly paired with its respective guide. These specific contexts are detailed in the subsequent section and serve as the foundation for the task support evaluation presented in section 5.2.

3.4 Walkthrough of EmbedLLM in a Single Environment

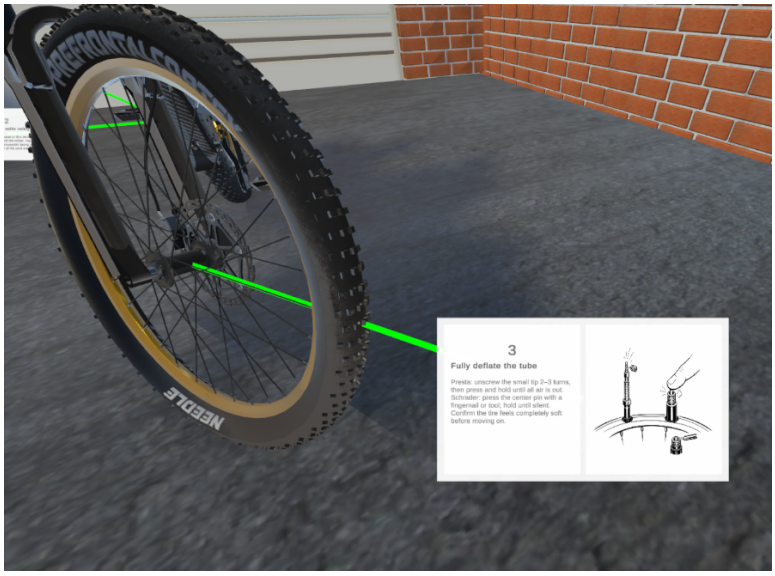
To mitigate visual computing constraints, EmbedLLM is developed within the Unity engine, requiring users to transition between VR and passthrough modes during task execution. The VR environment contains the knowledge required for task completion, whereas the passthrough mode facilitates the physical execution. For this walkthrough, it is assumed that the virtual scene is pre-configured and the input is correctly initialized on the Meta Quest headset. Specifically, the user-side input for this use case comprises three primary components: the source content (a PDF instructional guide for the Ender 3 nozzle replacement[1]), the user's objective ("to change the nozzle on my 3D printer as actionably as possible"), and the pre-configured virtual scene developed in Unity.

Subsequently, the user enters the immersive environment via the VR headset. Within this pre-configured virtual scene, the user is presented with a blue interface panel placed against an empty wall accompanied by a secondary text panel and a green directional line placed on the virtual 3D-printer. This can be seen on figure 3.6a and figure 3.6b. The blue, wall-mounted panel is positioned manually during scene creation, but its contents are generated by the LLM. The panel functions as a central dashboard to visualize EmbedLLM's status. This interface primarily details the progression of the current step, and a comprehensive overview of the step's primary goal. In contrast, the smaller white panels anchored in the environment display the actionable tasks and necessary information required to complete the current step, which, as previously described, is shown on the blue wall-mounted panel. The generation and spatial positioning of these supplementary panels are handled automatically by the LLM. Finally, it should be noted that a single step may comprise multiple actionable tasks prior to its completion. Consequently, one step displayed on the blue panel can correspond to multiple smaller, actionable panels distributed in the environment.
At this stage, the user reviews the high level information of the first step on the blue wall-mounted panel, which is to prepare the workspace by collecting the necessary tools. By moving

[1] https://www.crealityexperts.com/changing-the-nozzle-on-a-creality-printer

(a) Overview

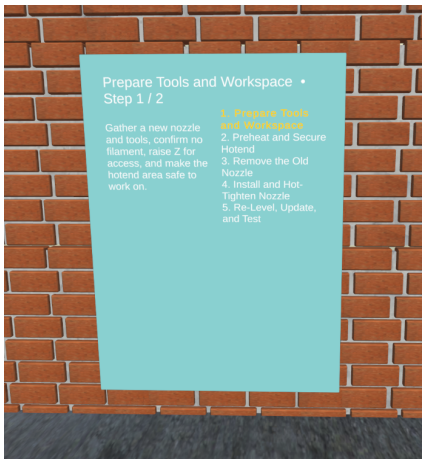
(b) Different part types



(c) Deflate

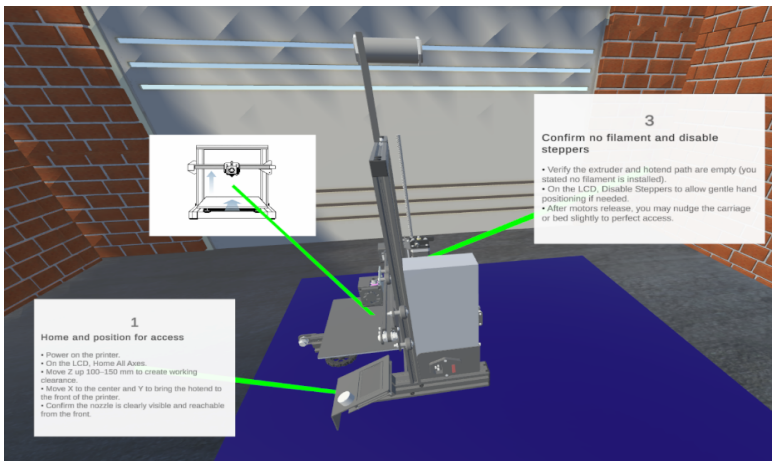
Figure 3.5: Depiction of a garage setting where the user objective is changing a bicycle tire. Within this context, informational panels are semantically anchored to specific bicycle components. The interface details various part types alongside their respective maintenance procedures. Furthermore, EmbedLLM analyzes the physical environment to infer the most probable component type relevant to the user. Finally, both a text panel detailing the tire deflation process and a supplementary image illustrating the procedure are spatially anchored directly to the wheel.

through the steps and using the content displayed in the smaller task panels, the user is supported throughout the execution of the overall task and all the content is embedded in the scene. In theory, this should lower the amount of context switches and the amount of content filtering. Since each piece of content is embedded in the scene, the user can leverage its natural spatiotemporal instincts to filter content. With the goal of offloading the cognitive load of filtering through content back into the environment.
The initial task in the workspace preparation process is the collection of the correct tools, as indicated on the panel. The user then enters passthrough mode and collects all tools specified in the referenced task. Once this task is completed, the user re-enters VR mode and uses the right trigger on the Meta Quest controller to proceed to the second task. As shown in Figure 3.6c, the second step is composed of three distinct tasks. The first of which is anchored to the media control dial of the Ender 3 3D-printer. This panel explains how the user should position all axes of the printer, with the ultimate goal of making the nozzle as visible as possible for the subsequent replacement steps. The user reads this panel and executes the operation it describes. This is done by entering VR mode and following the instructions displayed on the panel. Afterwards, the user remains in or re-enters VR mode to review the next panel. The second panel in this step shows an image of how the printer should appear after the operations described in the first panel have been correctly executed, while the third panel in this step asks the user to confirm that no filament is present in the 3D-printer. After executing these tasks, the user presses the right trigger on the controller to move to the second step, and repeats this process until the nozzle on the 3D-printer has been replaced.



(a) Blue wall-mounted interface panel.

(b) Task panel anchored to the scene.



(c) The user moves to step 2 and reviews its content, starting with task panel marked 1.

Figure 3.6: Example of a user using EmbedLLM. (a) The user reviews the high level steps in the blue wall-mounted panel. This panel shows a high level description of the first step, and listing of all the steps required to reach the user's goal. (b) The first task is represented on the task panels anchored in the scene. The user executes the first task using pass-through mode, (c) then presses the right trigger to move to the second task of step one and reviews its contents in order.

Chapter 4

Implementation

4.1 System Design

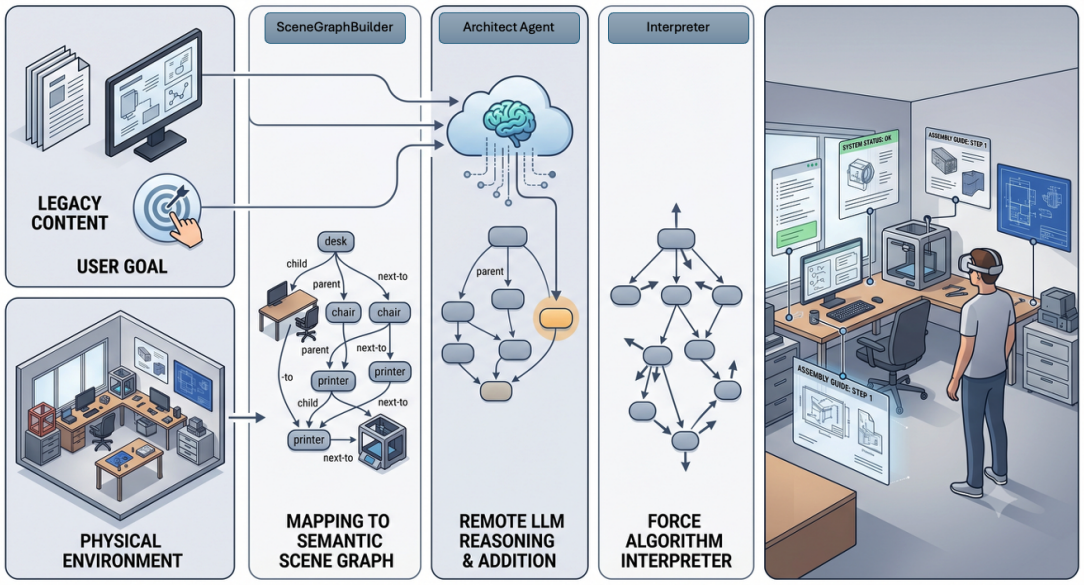


Figure 4.1: Outline of the internal mechanics of the 2D to context-aware content system. Initially, EmbedLLM ingests a virtual scene, a target PDF, and a specific user objective to construct a semantic scene graph representing the environment. Subsequently, an 'Architect Agent,' powered by a remote LLM, generates the necessary content additions and their corresponding semantic relations. Ultimately, the spatial arrangement of these elements is resolved through a force-based algorithm utilizing Unity's physics engine

To determine the viability of our approach in addressing the research question, we develop EmbedLLM, a system that extends LLM reformatting from 2D PDFs to context-aware 3D embeddings. Built as a collection of Unity modules, EmbedLLM is integrated into a Meta Quest application to provide an immersive virtual environment for task execution support. EmbedLLM takes three inputs: a PDF instruction guide, a user goal, and a preconfigured virtual scene. It outputs a set of spatially embedded task panels that connect relevant instruction content to objects in the environment. As previously discussed, EmbedLLM is implemented in VR to bypass current visual computing limitations. This approach necessitates a digital twin of the environment to represent the physical environment. Consequently, EmbedLLM requires a preconfigured scene within Unity. However, this remains the sole component requiring manual

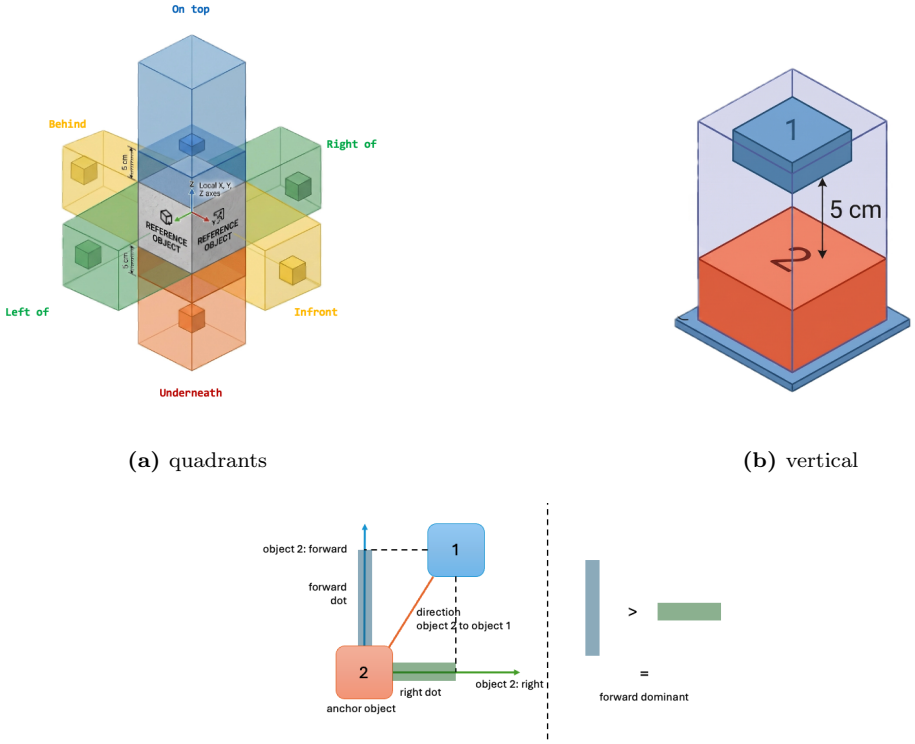
authoring. Furthermore, once visual computing methods enable consistent AR deployment, manual authoring of the digital twin will become obsolete. The first of three Unity modules is called the SceneGraphBuilder. This module, parses the environment and constructs a scene graph derived from the objects, their geometric properties and semantic relations. The resulting scene graph is then processed by the Architect Agent module. Serving as EmbedLLM's reasoning engine, this second module interacts with the remote LLM (OpenAI GPT-5 [1]) to generate and append the content scene graph nodes and edges. Specifically, it creates new nodes and new semantic relations. Finally, the newly generated additions must be instantiated within the virtual scene, a task managed by the Interpreter module. Receiving the updated scene nodes, this module executes a force-based algorithm to dynamically position panels and anchor lines while satisfying the semantic constraints of the extended scene graph created by the architect module. The positions of these additions are the result of executing this force algorithm on the semantic relations of the additions. This pipeline is depicted in figure 4.1.

4.1.1 The SceneGraphBuilder

The purpose of the SceneGraphBuilder is to convert the preconfigured Unity scene into a semantic scene graph that can be used by the remote LLM in the ArchitectAgent module. Because Unity objects do not inherently contain semantic spatial relations, the module derives relations such as "on top of", "underneath", "left of", "right of", "in front of", and "behind of" using geometric heuristics based on object bounds, positions, and local orientation. This can be viewed in figure 4.2a.
The SceneGraphBuilder scans the Unity scene for objects with the tag: environment. Based on this set of objects it constructs a scene graph to represent the Unity scene. The tag filter is added to exclude simulation specific modules such as the camera rig, event systems, other system modules and global lighting objects. This extraction is conducted offline, a design choice informed by the work of Lee et al. [25].
After collection, each object is added to the scene graph as a node parameterized by its label, center world coordinates, forward vector, and spatial bounds. To establish semantic relations, which are represented as edges in the graph, objects are evaluated pairwise. An overview of the pairwise evaluation can be seen in figure 4.2. Specifically to define "on top of" or "underneath of" relations, the SceneGraphBuilder calculates the vertical distance between the top of one object and the bottom of another. A distance of 5 centimeters or less while being in the horizontal bounds of the anchor object defines an on-top-of relationship. Conversely, reversing this calculation establishes an underneath relationship. This is depicted in figure 4.2b. Additionally, one object being contained by another is determined by verifying whether the minimum and maximum bounding coordinates of the first object are entirely enclosed within the bounds of the second.

To calculate the horizontal semantic relations ("left of", "right of", "in front of", "behind of"), the module first determines the horizontal distance by representing the objects as two-dimensional position vectors. Additionally, it computes the vertical difference between the objects. It only checks for horizontal relations if this vertical difference does not exceed 60 centimeters and if the objects are considered "near," which occurs only when the horizontal distance is less than 150 centimeters. Notably, the distance thresholds used to determine if objects are near are based on heuristics and are not explored further in this work. To classify horizontal relations, the module evaluates dot products based on object two's local axes. First, an orthogonal "right" vector is derived from object two's forward vector. The module then determines the relative position vector pointing from object two to object one, computing its dot products with both the forward and right vectors. By comparing the absolute values of these two dot products, the primary axis of alignment is identified. If the absolute forward dot product is greater, the relationship is classified as forward-dominant, requiring only a check to see if object one is "in front of" or "behind" object two. Conversely, if the absolute right dot product is greater, the relationship is right-dominant, and EmbedLLM evaluates whether

[1] `https://developers.openai.com/api/docs/models/gpt-5`



(a) quadrants **(b)** vertical

(c) horizontal

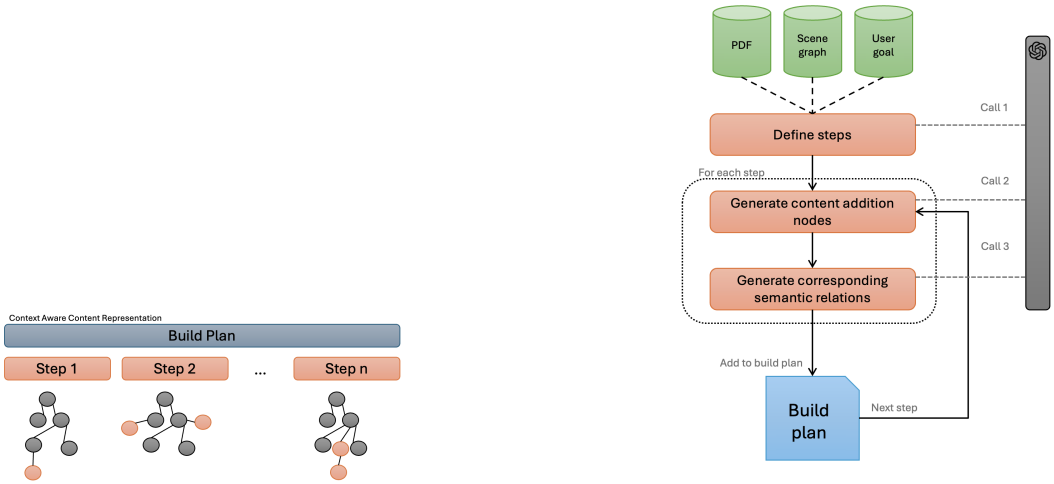
Figure 4.2: Depiction of how the SceneGraphBuilder calculates the semantic relations based on objects in the Unity scene. (a) All possible semantic relations plotted relative to anchor object, (b) visualization of logic to classify a vertical relation (”on top of” and ”underneath of”) and (c) depiction of logic to classify a horizontal relation (”left of”, ”right of”, ”in front of”, ”behind of”).

object one is to the ”left” or ”right.” Finally, these dot products need to be greater than 0.5 to be eligible for classification. Otherwise, a diagonal object could be categorized as in front of or right of. This is depicted in figure 4.2c.

4.1.2 The ArchitectAgent

The second module, called the ArchitectAgent, acts as the central reasoning engine of EmbedLLM. This module facilitates an interface for the remote LLM to interpret the scene graph, user goal, and source content, and to create additions to this scene graph. These additions will later be processed by the Interpreter module to instantiate panels in the scene and realize the alternative, context-aware content representation.
Since this thesis focuses on step-based manual tasks, the ArchitectAgent is specifically designed to generate context-aware representations suited for this domain. Specifically, both the output of the LLM and the structure of this output are constrained to adhere to the structure of this domain. Rather than simply asking the LLM an open-ended question to ”make the content context-aware”, which relies entirely on its unguided reasoning capabilities, the ArchitectAgent forces the LLM to operate within a strictly defined format. Specifically, the ArchitectAgent operates on three distinct calls, each call has a unique prompt and the responsibilities of these calls are discussed later in this section. These prompts restrict the agent to output structured data that adheres precisely to the schema of EmbedLLM. This constrained approach ensures the LLM generates actionable, consistently formatted scene graph additions rather than unpredictable text. Furthermore, our preliminary results indicate that while unguided content generation produces valid alternative representations, displaying all generated information si-

multaneously leads to significant visual clutter. To mitigate this information overload within our test environments, we introduce a temporal dimension by organizing the process into sequential steps. Prior to the embedded content generation phase, the content is structured so that only a single step is displayed at any given moment, with each step comprising a maximum of four tasks. This sequential approach aims to greatly enhance the overall clarity and usability of EmbedLLM.



(a) Buildplan representation.

(b) Architect module workings.

Figure 4.3: A high-level overview of the workings of the ArchitectAgent. (a) The scene embedded representation of the source content consists of a sequence of steps, each of which is defined as an extension on the environment scene graph. (b) The workings of the Architect module. This module receives the source content, scene graph, and user goals as input and creates the collection of extended scene graphs which represent the context-aware content representation. Call one defines a set of steps, call two generates the node additions for the scene graph given a certain step, and call three defines the related semantic relations. Calls two and three are executed iteratively for each step.

Thus to achieve the context aware representation, the agent must first delineate the sequence of steps that form the alternative representation. For each individual step it defines up to four atomic tasks within this step. Consequently it identifies both the new nodes to be introduced to the scene and the semantic relations that act as edges. The prompts for these three calls and the ArhcitectAgent system prompt can be found in appendix A. The ArchitectAgent is initialized with a system prompt detailing its macro-level objectives, but it remains idle until explicitly invoked to execute a specific workflow step. When the system requests step execution, such as during the initial call to define the overall step structure, the necessary contextual data is passed to the Large Language Model (LLM). This triggers the execution of this specific step in the execution loop.
Consequently, EmbedLLM generates a distinct, extended scene graph for every step. Conceptually, the complete representation is simply a sequence of these extended scene graphs (represented in Figure 4.3a). Furthermore to mitigate the cognitive load of Large Language Models [1], we restrict each task to an individual prompt. This design choice aligns with the findings of Lee et al. [25], which indicate that LLM performance deteriorates as the number of assigned responsibilities within a single prompt increases.
Consequently, the ArchitectAgent's workflow is divided into three distinct types of LLM calls. The initial prompt is dedicated exclusively to defining the sequential steps of the alternative representation. Once these steps are stored in memory, the module iterates through them. During this iteration, EmbedLLM executes the remaining two calls per step: one to generate the content addition scene graph nodes, and another to establish the corresponding semantic

edges. This workflow is visualized in figure 4.3b. This results in a list of additions grouped for each step. We call these additions the build plan, which are later interpreted by the interpreter module to instantiate the content embeddings in the scene.

4.1.3 The Interpreter

Finally, once the ArchitectAgent has created the buildplan it's the interpreter's job to parse this buildplan and instantiate the panels and images in the scene. Turning this buildplan into a set of panels necessitates solving the semantic relations while taking into account the three dimensional environment in which these relations need to be solved. For instance, a naive approach might calculate placement relying exclusively on the positions, bounding boxes, and semantic relations of the existing environment objects. However, because the generated panels are not registered as physical objects within the environment themselves, this method frequently causes multiple panels to be anchored at the exact same spatial coordinates. Consequently, this overlap severely degrades readability and reintroduces visual clutter into the scene. Consequently, the interpreter module must go beyond merely resolving semantic relations into spatial coordinates based solely on the environment's scene graph. It must simultaneously account for user visibility and potential occlusions, ensuring that any spatial adjustments adhere to the semantic constraints defined in the build plan.

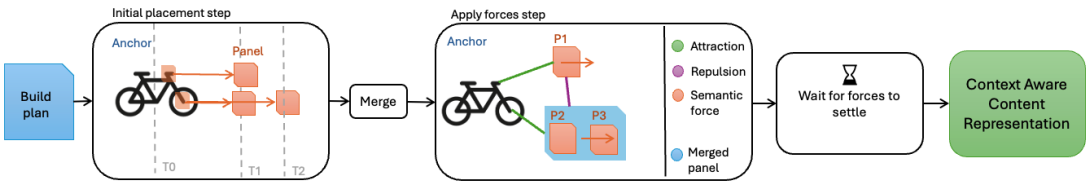


Figure 4.4: Workflow of the interpreter module. From an input build plan, EmbedLLM calculates initial panel positions using anchor-relative semantic vectors. Overlapping panels are merged to prevent occlusion, followed by the application of attraction, repulsion, and semantic forces. The algorithm resolves at spatial equilibrium to output the final context-aware representation.

To address this placement problem, we draw on prior work on annotation layout in 3D models, which explicitly considers spatial readability, overlap avoidance, and visibility constraints. Zheng et al. [48] explore a dynamic, force-directed algorithm designed for the optimal spatial placement of text annotations on 3D models. By leveraging physics-based principles, the algorithm continuously recalculates label coordinates in real time. Individual labels repel one another to prevent occlusion, while simultaneously maintaining an attractive pull toward their designated anchor points. Rooted in established research on optimal label placement, this 3D spatial approach integrates seamlessly into our framework by utilizing Unity's native physics engine to compute the underlying forces. However, the baseline algorithm requires adaptation to successfully incorporate semantic relations. We propose integrating dedicated semantic forces into the existing force-directed model. By evaluating the forward vectors of both the anchor and the panel, EmbedLLM calculates the directional pull necessary to satisfy the underlying semantic constraints. While this approach cannot guarantee perfect adherence to the semantic relations, it effectively balances clear label placement with the overall structure of the build plan.

Prior to the execution of the force-directed algorithm, the interpreter module must establish the initial coordinates for each panel. Because the subsequent physics simulation relies on repulsion and attraction forces to iteratively optimize placement, this initial positioning serves as a heuristic 'best guess.' Specifically, EmbedLLM identifies the center point of the anchor object and translates the panel outward along the directional vector dictated by the semantic relation, continuing until the panel is completely free of environmental collisions. For instance, if a relation dictates that a panel must be placed to the right of a baking tin, EmbedLLM com-

putes the tin's center and its forward vector. The panel is then translated rightward, relative to this orientation, until it clears all spatial obstacles. This first collision-free state becomes the starting position for the force-directed algorithm.
Furthermore, spatial relations within this system are explicitly object-centric rather than universally aligned. Because a constraint such as 'right of' is calculated relative to the anchor object's forward vector, its physical direction is entirely dependent on its anchor's orientation. For instance, 'right of' a baking tin may align with the user's perspective of right, whereas 'right of' a rotated pot might physically manifest on the user's left. While transitioning to viewport-dependent semantics, where relations dynamically update based on the user's perspective, presents a valid alternative, EmbedLLM intentionally maintains a static, scene-relative context, to prioritize environmental consistency. Ensuring that the generated elements remain fixed in their physical locations regardless of how the user moves or interacts. Additionally, these scene dependent semantics in favor of viewport dependent semantics lets EmbedLLM gauge the LLM's efficacy in semantic reasoning. Exploring these viewport-dependent semantics does, however, remain a compelling avenue for future work to further optimize spatial coherence and user experience. Additionally, the force based label placement system proposed by Zheng et al. is dynamic and viewport dependent and keeps the forces active during execution. Given the static nature of the target environment, we terminate the force-directed algorithm once the panels reach a state of spatial equilibrium. Halting these continuous physics calculations yields significant performance improvements, a benefit that becomes particularly pronounced when operating within scenes containing highly complex, multi-component models.
Finally, we observe that upon resolving the semantic relations, multiple panels may be anchored to the exact same spatial coordinates. Although the interpreter's force-directed approach physically separates these panels to prevent direct overlap, visual occlusion can still occur from specific user viewing angles. This phenomenon is a direct artifact of utilizing a scene-dependent, rather than viewport-dependent, placement algorithm. To mitigate this line-of-sight issue, the interpreter module dynamically pools the intersecting panels into a single, consolidated interface, presenting the original contents as distinct sub-panels. The workings of the initial placement and force algorithm adapted to semantic relations is visible in figure 4.4.

4.2 Caching and Token Use

To minimize token usage and generation times during system development and evaluation, we implement various levels of caching within the modules. As detailed in the previous section and illustrated in Figure 4.1, the input of each module relies on the output of the preceding one. Consequently, each module caches its output to disk. Upon initialization, a module verifies the existence of its corresponding cache. If present, it bypasses computation and instead parses and loads the cached state. Due to this caching logic, the context-aware alternative content generation occurs near-instantaneously for a given scene, user goal, and PDF, provided the scene has been previously processed.
Because the ArchitectAgent module executes various reasoning and image generation API calls, EmbedLLM's token usage and generation times are high. The interpreter requires a complete build plan to begin initial placement and force simulation, meaning it cannot start executing until the entire plan is generated. It does not, however, need to wait for the images to be generated. For this reason, we run the interpreter using image placeholders, delaying the actual image generation until the interpreter has begun executing the placement steps. While this approach reduces execution time to some extent, first run token usage and execution time remain considerably high.
Finally, this caching mechanism enables a controlled user study evaluation. Because generating a scene from a populated cache is deterministic, every participant experiences an identical scene content representation. This evaluation is discussed further in the next chapter and in section 5.2.

Chapter 5

Evaluation

5.1 Preliminary Study on User Preferences

While scene embedded content has been evaluated extensively in current research [2, 3, 17, 44], the contribution of EmbedLLM is to facilitate an automatic scene embedded content representation so that true context aware content can be achieved. To evaluate the quality of this automatic scene embedding, we created and distributed a qualitative survey to 20 participants. Furthermore, we conducted a more in-depth user study with 12 participants to evaluate the alternative representations in real task execution environments. This is discussed in the following section 5.2.

5.1.1 Survey Design

To assess the user's preference and gauge EmbedLLM's generalizability, participants complete a survey presenting four distinct contexts (as seen in figure 5.1). For each scenario, participants evaluate the content representation in terms of its alignment with both the environment and the user goal. Furthermore, a reverse coded item is incorporated to compare the EmbedLLM scene embedded approach against a standard baseline (using a web browser on a smartphone). This baseline is represented in figure 5.1a. To ensure the survey remains concise and accessible, each scene is introduced via a brief text description that specifies the environment (for example, a kitchen) and the user's task (for example, baking a cake). This description is accompanied by a sequence of images demonstrating the initial steps of the scene embedded content.
The survey presents four distinct scenarios (as seen on figure 5.1, evaluating each with the same set of three questions measured on a 7-point Likert scale:

- A_E *Alignment with the physical environment*: This content representation is well integrated in the physical environment.
- A_G *Alignment with the user goal*: This content representation is well aligned with the user's goal.
- C_B *Preference compared to baseline*: I would prefer the baseline (smartphone and webpage) over this content representation.

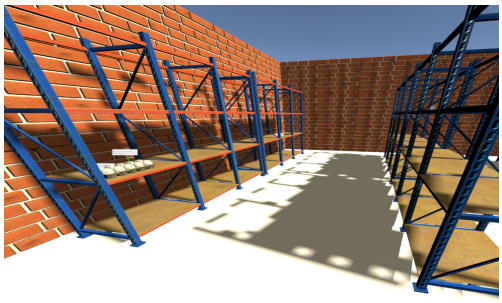
The full survey used in this user study can be found in the appendix B. Based on these metrics we formulate the following hypotheses.

H_1: *Users will rate EmbedLLM's automatically generated scene-embedded content significantly higher on scene coherence (A_E) across varied contexts when compared to the baseline smartphone scenario (baking a cake).*

H_2: *Users will rate EmbedLLM's automatically generated scene-embedded content significantly*

(a) baseline

(b) kitchen



(c) store

(d) garage

Figure 5.1: The four scenes used to assess user preference as well as generalization during the survey. Each scene has a distinct environment and user goal. (a) baking a cake in kitchen, (b) baking cake in kitchen, (c) gathering supplies in store, (d) repairing a bicycle in garage.

higher on task relevance (A_G) across varied contexts when compared to the baseline smartphone scenario (baking a cake).

H_3: *Users will prefer EmbedLLM's automatically generated scene-embedded content (C_B) significantly more across varied contexts when compared to the baseline smartphone scenario (baking a cake).*

Confirming these hypotheses indicates that automatic scene embedding outperforms the baseline, thereby validating the feasibility of context-aware content reformatting. The evaluation procedure utilizes a standardized order of presentation for all subjects. Participants initially evaluate the baseline scenario that involves following a cake recipe on a mobile device (fig 5.1a). Subsequently, they assess the identical baking task utilizing a scene embedded context aware representation (illustrated in fig 5.1b and fig 3.3). The assessment then progresses to a bicycle repair guide presented through a context aware format (as seen in fig 5.1d). The sequence concludes with a grocery store scenario where the objective is to collect supplies using a context aware representation (shown in fig 5.1c and 3.4). An in depth explanation of these different scenes can be viewed in section 3.3
It should be noted that this fixed presentation order introduces an inherent sequential bias, particularly because the baseline condition is only evaluated once at the beginning of the study. To address this potential limitation, we considered evaluating EmbedLLM against a neutral Likert score of 4. However, we ultimately opted to retain the original condition as an empirical baseline for comparison.

5.1.2 Analysis

The preference survey data is collected as a separate CSV file and loaded into a dedicated python dataframe. Each question assesses one of three constructs across the four scenes: physical

environment alignment (A_E), user goal alignment (A_G), and preference compared to baseline (C_B). Prior to analysis, reverse coding is applied to the C_B questions for the three non-baseline scenes, ensuring that higher scores consistently indicate a favorable outcome across all constructs. Descriptive statistics, means and standard errors of the mean, are computed per question.
Furthermore, to examine whether the context-aware representations differ significantly from the smartphone baseline, each construct (A_E, A_G, and C_B) is compared against its baseline counterpart for each of the three non-baseline scenes, yielding nine paired comparisons in total. Because all participants evaluate all scenes, the data are paired at the participant level. For each comparison, normality of the pairwise differences is first assessed using the Shapiro-Wilk test. A two-sided Wilcoxon signed-rank test is used as the primary inferential test, supplemented by a two-sided paired t-test as a parametric reference. Effect sizes are quantified using Cohen's d for the paired t-test and the rank-biserial correlation r for the Wilcoxon test. In addition to the baseline comparisons, a paired test is conducted between kitchen C_B and garage C_B to assess whether preference scores differ significantly between these two scenes. Normality of differences is again verified via Shapiro-Wilk, and the same combination of Wilcoxon signed-rank and paired t-test with effect size estimates is applied.

5.1.3 Results

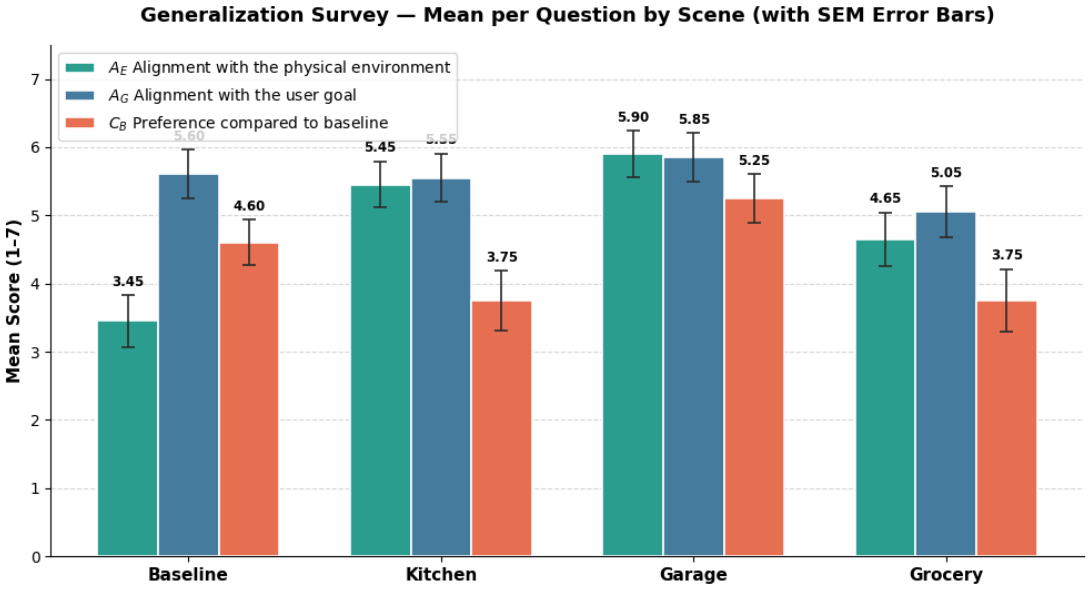


Figure 5.2: Bar chart illustrating the mean scores for each item in the preference questionnaire (n=20). The Standard Error of the Mean (SEM) is visualized at the top of each bar. Furthermore, bars are grouped per scene.

The average value for each question (A_E, A_G and C_B) alongside the standard error for each scene can be viewed in figure 5.2. The spatial integration (A_E) of context-aware scene embeddings is rated significantly higher than the smartphone baseline across all evaluated contexts. In the Garage context, where digital instruction cards are anchored to physical bicycle components, participants report a very large and highly significant improvement in environmental integration, $t(19) = 4.53, p < .001, d = 1.01$ (Alternative $M = 5.90$ vs. Baseline $M = 3.45$). Similarly, the Kitchen context shows a large, significant improvement over the baseline, $t(19) = 3.82, p = .001, d = 0.85$ (Alternative $M = 5.45$). Finally, although the Grocery context yields a slightly lower absolute rating, it still demonstrates a statistically significant and substantial improvement in environmental integration over the traditional smartphone screen, $t(19) = 2.66, p = .015, d = 0.60$ (Alternative $M = 4.65$). Taken together, these findings

provide sufficient evidence to reject the null hypothesis H_{10} and accept H_1.

In user-rated goal utility (A_G) between the baseline smartphone app and the context-aware spatial embeddings no significant differences are observed. The baseline smartphone applications are rated highly for general goal utility ($M = 5.60$). Crucially, these high levels of utility are maintained after transitioning to the spatial alternative representations in the Kitchen ($M = 5.55, t(19) = -0.09, p = .933$) and the Garage ($M = 5.85, t(19) = 0.46, p = .650$). In the Grocery scene, a slight drop in mean utility is observed ($M = 5.05$), but it does not reach statistical significance, $W = 23.5, p = .220, d = -0.25$. Although H_2 is not supported, the absence of a significant decline in goal utility suggests that the transition to spatial embeddings does not compromise task relevance, with ratings remaining consistently high across all contexts.

Finally, in contrast to the uniform trends in environmental integration and goal utility, subjective user preferences (C_B) are highly task-dependent. The Garage context leads to a statistically significant increase in preference for the spatial embedding over the baseline smartphone manual, $W = 28.0, p = .033$ (Alternative $M = 5.25$ vs. Baseline Expectation $M = 4.60$).
Conversely, for the Kitchen and Grocery contexts, preferences shift significantly in favor of the traditional smartphone baseline. In the Kitchen context, preference for the spatial embedding decreases significantly, $t(19) = -2.82, p = .011, d = -0.63$ (Alternative $M = 3.75$). In the Grocery context, a similar significant decrease in preference is observed in the non-parametric Wilcoxon test, $W = 24.0, p = .039$ (Alternative $M = 3.75$), although it does not reach significance in the parametric reference t-test, $t(19) = -1.69, p = .108$. These mixed results do not support H_3 in its general form. While the spatial embeddings are significantly preferred over the smartphone baseline in the Garage context, preferences reverse significantly in favor of the baseline in the Kitchen and Grocery contexts, indicating that subjective preference for spatial embeddings is highly context-dependent rather than uniformly higher. The results of the tests can be viewed in table 5.1.

Table 5.1: Results of the preference survey (n=20)

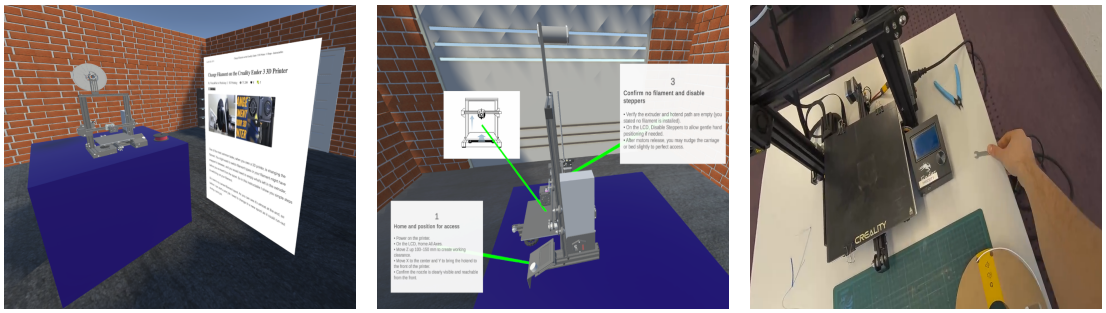
Shapiro–Wilk normality test	$W = 0.9549,\ p = 0.4484$ (normal)
Composite score mean	5.0222 ($neutral = 4.0$)
Wilcoxon signed-rank test (one-tailed)	$W = 189.0,\ p = 0.0008$
Effect size (rank-biserial r)	-0.8000 (≥ 0.7 large)
One-sample t-test	$t = 4.1066,\ p = 0.0003$
Q2.3 vs Q3.3 mean	3.75 vs. 5.25
Q2.3 vs Q3.3 Shapiro–Wilk (differences)	$p = 0.0656$ (normal)
Q2.3 vs Q3.3 Wilcoxon signed-rank test	$W = 9.0,\ p = 0.0060$
Q2.3 vs Q3.3 effect size (rank-biserial r)	0.9143 (large)
Q2.3 vs Q3.3 paired t-test	$t(19) = -3.6268,\ p = 0.0018,\ d = 0.8110$

5.2 Evaluating EmbedLLM During Manual Task Execution

As previously discussed, the survey investigates user preferences by comparing traditional smartphone web browsing with the alternative representations provided by EmbedLLM, alongside an evaluation of environmental and goal alignment. The results demonstrate that EmbedLLM successfully automates the generation of scene-embedded content representations, which are perceived as exhibiting significantly higher environmental alignment than baseline smartphone usage under identical conditions. Moreover, the data highlights nuanced differences in user acceptance of scene-embedded content across varying contexts. Specifically, while users favor the EmbedLLM representation significantly more than the baseline during a bicycle repair task within a garage setting, this preference contrasts starkly with a supply-gathering task within a store environment.
Nevertheless, that initial evaluation does not measure the practical quality or effectiveness of

these alternative representations. To address this limitation, we conduct a formal user study focused on evaluating system performance within an active task support scenario. Although testing EmbedLLM across a vast array of varying contexts is logistically unfeasible, we design a focused evaluation incorporating two distinct combinations of user objectives and source materials, both situated within a single shared environment.

5.2.1 User Study Design



(a) baseline **(b)** EmbedLLM **(c)** passthrough mode

Figure 5.3: User study conditions. From left to right, the baseline condition showing a floating PDF window in the virtual space, EmbedLLM condition showing context aware panels anchored in the scene and a passthrough mode which participants utilize to execute tasks in the physical environment.

The within subject user study design incorporates two distinct maintenance tasks involving an Ender 3 3D printer. To closely mimic real task execution scenarios, participants are tasked with replacing either the printer nozzle or the filament. Specifically, the filament replacement procedure requires participants to first preheat the nozzle to the standard PLA extrusion temperature. They must then actuate the extruder lever to withdraw the existing filament, after which the replacement filament is manually routed through the Bowden tube and into the heated nozzle. For the second task, the nozzle change procedure involves preheating the hotend to the appropriate temperature, removing the old nozzle, and properly seating the new one.
As discussed earlier, EmbedLLM takes a PDF as source input to generate the alternative scene embedded representation. Both scenes utilize a different source document. Both the filament change PDF [1] and the nozzle change PDF [2] are exported as PDF documents. These source documents can be found in appendix C.
To complete the procedures of these scenes, users engage with either the EmbedLLM context aware system or a floating PDF window baseline interface. This baseline window is populated by the source PDF content itself. To ensure experimental consistency, both conditions are implemented utilizing virtual reality and an accompanying passthrough mode. Participants acquire task information within a virtual environment and subsequently execute the physical actions utilizing this passthrough mode (as seen on figure 5.3c). The baseline condition features a single floating digital window situated near the physical workbench (depicted on figure 5.3a). Users interact with this interface via scrolling gestures to simulate a standard touchscreen experience, effectively replicating the conventional spatial computing use of floating PDF documents. Conversely, the EmbedLLM condition integrates textual panels and imagery directly into the physical environment by semantically anchoring them to relevant objects (shown on figure 5.3b). A more in depth discussion of the workings of EmbedLLM and one of the scenes used in the user study can be viewed in section 3.4.
Navigation between procedural steps is controlled via buttons on the meta quest controllers, allowing users to seamlessly advance or return to previous instructions. Furthermore, each participant is required to execute every scenario utilizing one of the two experimental conditions.

[1] https://www.instructables.com/Change-Filament-on-the-Creality-Ender-3-3D-Printer/
[2] https://www.crealityexperts.com/changing-the-nozzle-on-a-creality-printer

To effectively mitigate potential order effects and learning bias, the assignment of these conditions to the respective scenarios is counterbalanced employing a Latin square design.

Because the evaluation involves LLM generated material, maintaining consistency across participants is essential for reliable statistical analysis. To achieve this consistency, the methodology employs the caching mechanism detailed in section 4.2. By loading an identical cache for every subject, EmbedLLM guarantees a uniform content representation. Consequently, this experimental control necessitates a trade off regarding personalization. While a real world deployment could dynamically adapt the content based on the specific user objective, the physical environment, and the source material, this study holds these variables constant. For instance, a novice user might explicitly request to change the 3D printer filament while simultaneously learning the associated terminology. Nevertheless, accommodating such highly individualized goals is deemed unfeasible for this evaluation due to the prohibitively long generation times required.

The evaluation methodology incorporates both qualitative and quantitative data collection. Qualitative insights are gathered through structured questionnaires and open ended participant feedback. Conversely, quantitative performance is tracked via the NASA-TLX questionnaire and total task completion times. Following the conclusion of each individual task, the NASA Task Load Index assesses the cognitive load experienced by the user. This established metric is paired with a custom questionnaire designed specifically to evaluate the coherence of the physical scene, the user goal, and the generated content. Ultimately, upon the completion of both tasks, participants fill out a final comparative survey that directly evaluates EmbedLLM relative to the baseline interface.

As noted before, the purpose of the coherence questionnaire is to measure the scene and goal alignment of a given condition, in combination with the quality of the content itself (for example too much or little information). The questions of the coherence questionnaire are listed below:

1. It was easy to identify and locate the physical components on the machine.
2. The instructions clearly explained what was expected at each step.
3. The order of steps was clear.
4. The information provided was concise and easy to find.
5. I felt that I spent a lot of time finding the necessary information. *(Reverse-coded)*
6. Some of the information I needed was missing. *(Reverse-coded)*
7. The wording of the instructions was confusing or ambiguous. *(Reverse-coded)*
8. Any additional comments or remarks? *(Open ended)*
9. What did you like or dislike about EmbedLLM you just used? *(Open ended)*

Here, questions one and five measure scene alignment, questions two and three measure goal alignment and the quality of the content is gauged using questions four, six and seven. Furthermore, question five, six and seven are reverse-coded, and question eight and nine are open ended questions and facilitate additional user feedback.

In parallel, the comparative questionnaire, shown below, includes three reverse-coded questions (two, four and six). For this questionnaire, a higher score is in favor of EmbedLLM. The questions are listed below:

1. If I had to execute the task at hand, I would prefer using the context aware system over the traditional PDF window.
2. The traditional PDF window made it easier to understand the information and find the knowledge I needed. *(Reverse-coded)*

3. Once I became familiar with the context aware system, it was easier to process information in a task-based context than with the traditional PDF window.
4. I think I performed better using the traditional PDF window. *(Reverse-coded)*
5. The context aware system provided more relevant information.
6. I felt I was less confused when working with the traditional PDF window. *(Reverse-coded)*
7. I felt I was less confused when working with the context aware system

Drawing upon the established metrics, we introduce four additional hypotheses, culminating in the following set:

H_1: *Users will rate EmbedLLM's automatically generated scene-embedded content significantly higher on scene coherence (A_E) across varied contexts when compared to the baseline smartphone scenario (baking a cake).*

H_2: *Users will rate EmbedLLM's automatically generated scene-embedded content significantly higher on task relevance (A_G) across varied contexts when compared to the baseline smartphone scenario (baking a cake).*

H_3: *Users will prefer EmbedLLM's automatically generated scene-embedded content (C_B) significantly more across varied contexts when compared to the baseline smartphone scenario (baking a cake).*

H_4: *During manual task execution, participants will find the EmbedLLM context aware content representation significantly more suitable for their environment when compared to the traditional floating PDF window baseline interface in terms of its environment alignment as seen in questions one and five of the coherence questionnaire.*

H_5: *During manual task execution, participants will find the EmbedLLM context aware content representation significantly more suitable for their active tasks than the traditional floating PDF window baseline interface in terms of its goal alignment as seen in questions two and three of the coherence questionnaire.*

H_6: *During manual task execution, participants will prefer EmbedLLM context aware content representation significantly more when compared to the traditional floating PDF window baseline interface as seen in the comparative questionnaire.*

H_7: *The EmbedLLM context aware representation yields superior support for task execution relative to the floating PDF window baseline interface. We anticipate that this enhanced support will result in significant reductions across two performance metrics, task completion time and cognitive load as measured by the NASA-TLX questionnaire.*

5.2.2 User Study Procedure

The user study procedure is designed to last approximately 50 minutes per participant and is structured into a sequence of standardized phases. To ensure safety and consistency, preparations are made prior to each participant's arrival, including clearing the physical workspace of obstacles and verifying that the Meta Quest headset is fully charged with the correct Unity scenes and data loaded. Logging directories are prepared, and the specific condition sequence for the participant is checked against the Latin square ordering.

Upon arrival, participants undergo a 5-minute introductory session. They are provided with an overview of the study, specifically that they evaluate two different repair task support systems while performing maintenance on an Ender 3 3D-printer. Participants are informed that all necessary knowledge is displayed in the VR headset and that they can freely switch between the virtual content and the physical real-world view utilizing EmbedLLM's passthrough mode. Following this briefing, participants sign an informed consent form and are explicitly reminded that they can withdraw from the study at any time.

To mitigate the "novelty effect" of virtual reality and minimize extraneous cognitive load during the actual evaluation, participants engage in a 5 to 10-minute familiarization phase. The researcher assists the participant in fitting the headset comfortably and adjusting the interpupillary distance (IPD) to ensure clear text visibility. Participants are then loaded into a neutral tutorial scene focused on a bicycle repair task. During this tutorial, participants learn the basic controls and how to interact with both system interfaces:

- EmbedLLM condition: Users learn how to read the overview panel (which displays chapters and step counts), view the actionable task panels, and use the controller triggers to move sequentially through the steps.
- Baseline condition: Users learn to navigate the floating PDF by pressing and holding the controller button or trigger while moving the controller vertically to simulate a scrolling gesture.

Finally, participants are instructed on how to toggle the passthrough view and receive explicit safety warnings regarding the physical dangers of the 3D printer's hotend.

The primary experimental phase begins by launching the assigned application build and recording the participant's point of view. The researcher first anchors and calibrates the virtual printer's position to perfectly align with the physical machine.
Participants are reminded of their explicit goal for the current task and are instructed to use the "think aloud" protocol to vocalize their reasoning as they work. External camera recordings, a point of view recording and timers are initiated. A strict 25-minute time constraint is enforced for completing the task. This limitation is deliberately implemented because the LLM condition is a novel interface. Capping the time creates a sense of urgency, encouraging participants to actively engage with the content for functional understanding rather than simply exploring the environment for the sake of novelty. If the participant achieves the goal before the 25 minutes elapse, the trial concludes early. Throughout the execution, the researcher observes the participant and notes any spoken questions or remarks.

Immediately following the conclusion of the first task, the researcher helps the participant remove the headset. The participant then moves to a laptop to fill out the NASA-TLX to measure cognitive load, followed by the coherence questionnaire corresponding to the first condition.
Once completed, the examination and post-test phases are entirely repeated for the second maintenance task utilizing the alternate assigned condition.

After the completion of both experimental conditions and their respective immediate post-tests, participants fill out the final comparative questionnaire. The session concludes with a brief wrap-up where the true, detailed purpose of the study is explained to the participant, followed by gratitude for their time. Finally, the researcher extracts the VR telemetry logs, aggregates the survey platform data, and collects the video recordings for subsequent analysis.

5.2.3 Analysis

Raw study data are collected from three distinct sources: a `times.json` file recording task execution durations per participant and condition and a directory of per-participant JSON questionnaire exports. Prior to analysis, all raw inputs are processed by a dedicated preprocessing script that normalizes participant identifiers to a canonical format (e.g., P0, P1) and harmonizes condition labels to resolve minor inconsistencies in the source data, such as spelling variations in task names. The output of this step is a single structured `clean_data.json` file, which serves as the sole input to subsequent analyses.
All data analysis procedures are conducted in Python using Jupyter Notebooks to ensure transparent and reproducible documentation. Data manipulation and parsing are performed with the `pandas` and `numpy` libraries, while statistical analyses are carried out with functions from `scipy.stats`. For the questionnaire analyses, responses are first expanded into per-question columns and normalized through reverse-coding where required. In the NASA-TLX questionnaire, item Q4 is reverse-coded, while in the coherence questionnaire items Q5, Q6, and Q7 are

reverse-coded. Qualitative questions Q8 and Q9 of the coherence questionnaire are excluded from analysis.

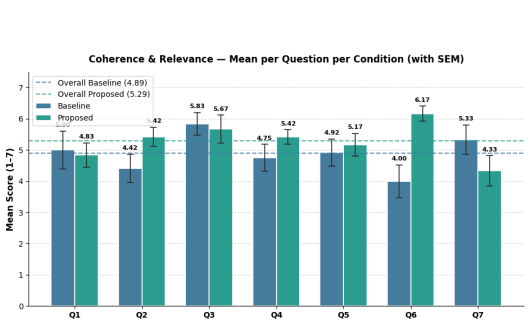
To address hypotheses H_4 and H_5, we analyze the coherence questionnaire responses obtained after each session. Because the study follows a within-subjects design at the condition level, each participant contributes one baseline session and one proposed-condition session, making the overall condition comparison paired rather than independent. After reverse-coding the relevant items, we first examine the questionnaire descriptively by calculating mean scores per question and per condition. In addition, to directly assess the constructs defined in the hypotheses, we compute two coherence sub-scales by averaging the relevant item pairs: an environment alignment score based on questions 1 and 5, and a goal alignment score based on questions 2 and 3. For each session, we also compute an overall composite coherence score by averaging across all analyzed questionnaire items. Statistical significance for the overall coherence score and for both sub-scales is assessed using a paired t-test. As a robustness check, we additionally report a Wilcoxon signed-rank test, and normality of the paired differences is assessed using the Shapiro-Wilk test. Effect size for the paired comparisons is quantified using Cohen's d.
In addition to the sub-scale analyses, we also inspect the item-level coherence results directly. This allows us to evaluate the contribution of the individual questionnaire items underlying H_4 and H_5, namely questions 1 and 5 for environment alignment and questions 2 and 3 for goal alignment. For these item-level comparisons between baseline and proposed conditions, we again report Shapiro-Wilk tests on paired differences, paired t-tests, Wilcoxon signed-rank tests, and Cohen's d.
Furthermore, we conduct task-specific coherence analyses for the filament and nozzle tasks separately. These comparisons are not treated as paired, because participants do not perform the same task under both interface conditions, instead task assignment alternates across conditions via latin square. Consequently, for each task we compare baseline and proposed-condition groups using the Mann-Whitney U test and additionally report the independent-samples t-test as a parametric reference. The effect size for these task-specific analyses is expressed as the rank-biserial correlation.

To address hypothesis H_6, we analyze the comparative questionnaire completed once at the conclusion of the study. Because this questionnaire is not tied to a specific session and does not yield paired baseline-versus-proposed observations, inferential significance testing is executed against Likert neutral (four on a seven scale Likert). As in the original analysis, responses are reverse-coded where necessary, after which mean scores are calculated per question across all participants.

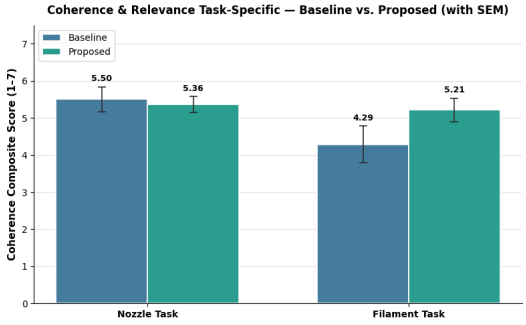
To address hypothesis H_7, we evaluate whether the proposed interface provides superior support for task execution relative to the floating PDF baseline by examining task completion time and perceived cognitive load. Execution time is recorded for each participant in each condition and analyzed at the overall condition level using a paired t-test, reflecting the within-subjects design in which every participant completes one baseline session and one proposed-condition session. Before applying this parametric test, the normality of paired differences is assessed using the Shapiro-Wilk test, and Cohen's d is calculated as the corresponding effect size. In addition, average execution times are also summarized for each task-condition combination, although no inferential test is performed at that level because only six participants contribute to each individual task-condition cell.
Perceived cognitive load is assessed using the NASA-TLX questionnaire after each session. Responses are first expanded into their constituent items, with Q4 reverse-coded prior to analysis. We then calculate mean scores per question for both conditions and derive a composite NASA-TLX score for each session by averaging across all analyzed items. Since each participant contributes one score under each condition, overall baseline-versus-proposed differences are tested using a paired t-test, with Shapiro-Wilk used to assess normality of the paired differences and Wilcoxon signed-rank reported as a non-parametric robustness check. Cohen's d is again used to quantify effect size.
Finally, task-specific NASA-TLX analyses are carried out separately for the nozzle and fila-

ment tasks. As with the task-specific coherence analyses, these comparisons are treated as independent because participants do not complete the same task in both conditions. Accordingly, baseline and proposed groups are compared using the Mann-Whitney U test, with the independent-samples t-test included as a parametric reference and the rank-biserial correlation reported as effect size.

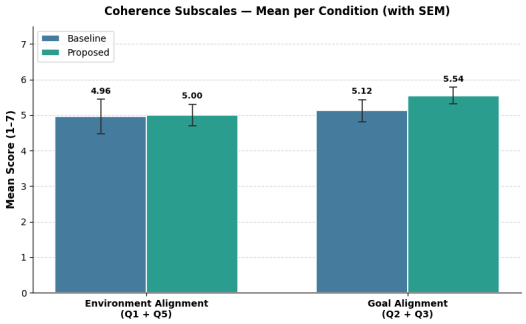
5.2.4 Results



(a) Coherence questions



(b) Coherence tasks

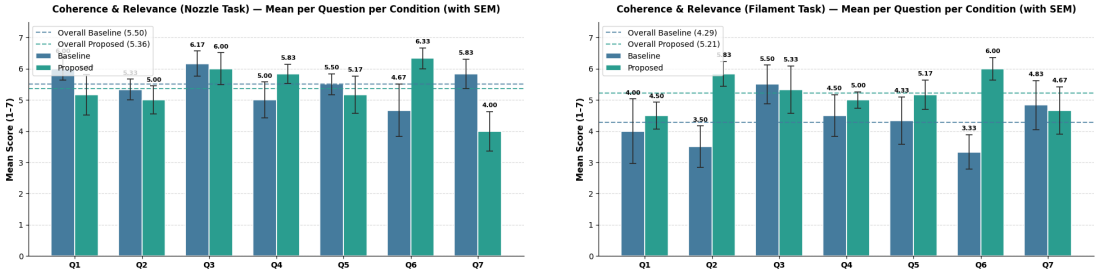


(c) Coherence sub-scales

Figure 5.4: Results of the coherence questionnaire. (a) Shows the average value of each individual question for both EmbedLLM and the baseline condition. The dotted lines display the composite average. (b) Shows the composite average for each task and (c) displays the results for the environment alignment and goal alignment sub-scales.

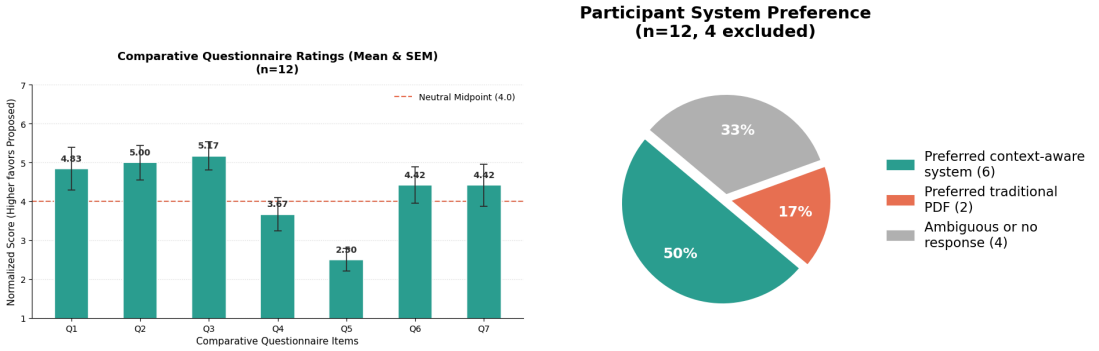
The coherence results (shown in figure 5.4 and table 5.2) show a higher overall mean score for the EmbedLLM condition than for the baseline condition, but this difference is not statistically significant. Specifically, the composite coherence score increases from 4.8929 in the baseline condition to 5.2857 in the EmbedLLM condition, with the paired t-test indicating no significant effect, $t = -1.2008$, $p = 0.2550$, and a small-to-moderate effect size, $d = 0.3467$.
To evaluate H_4 and H_5 more directly, two sub-scales are computed from the coherence questionnaire. The environment alignment sub-scale, defined as the mean of Questions 1 and 5, shows virtually no difference between conditions, with means of 4.9583 for baseline and 5.0000 for EmbedLLM, and no significant effect in either the paired t-test or Wilcoxon signed-rank test. The goal alignment sub-scale, defined as the mean of Questions 2 and 3, shows a larger increase from 5.1250 to 5.5417, but this difference is also not statistically significant. The results for the sub-scales are shown in figure 5.4c.
At the item level, the clearest improvement occurs on Question 6, where the EmbedLLM condition scores significantly higher than baseline, with $p = 0.0005$ in the paired t-test and $p = 0.0020$ in the Wilcoxon signed-rank test. By contrast, Questions 1, 3, and 7 do not show significant improvements, and Question 1 is slightly lower in the EmbedLLM condition overall. The results for the individual questions can be seen in figure 5.4a.
Task-specific coherence analyses reveal a small decrease for the nozzle task, from 5.5000 in

the baseline condition to 5.3571 in the EmbedLLM condition, and a larger increase for the filament task, from 4.2857 to 5.2143, but neither comparison reaches significance under the Mann–Whitney U test. Across both interfaces, the filament task therefore appears to benefit more from the EmbedLLM representation than the nozzle task, although these differences remain non-significant (as seen in figure 5.4b). Furthermore, these differences can also be seen when we isolate the average question results based on task. As seen in figure5.5, for Q1 EmbedLLM dips below the baseline for the nozzle change task
Taken together, these findings do not provide sufficient evidence to support either H_4 or H_5. The EmbedLLM system does not yield a significant improvement in either the environment alignment sub-scale or the goal alignment sub-scale, and the overall coherence score is likewise not significantly different from baseline.



(a) Coherence nozzle change task **(b)** Coherence filament change task

Figure 5.5: Average per question coherence questionnaire results based on task. (a) average coherence score for the nozzle change task, (b) for the filament change task.



(a) Comparative questions **(b)** Comparative preference

Figure 5.6: Results of the comparative questionnaire. (a) Average scores for individual questions, (b) pie chart showing that 6 out of 8 participants of the task support user study preferred the EmbedLLM system. Four responses were deemed ambiguous or missing and were therefore removed from the analysis for this specific question.

The comparative questionnaire is used to evaluate H_6 (results are shown in figure 5.6). After reverse-coding the relevant items, the overall mean score is 4.2857, which lies above the neutral midpoint of 4.0 and therefore indicates a descriptive preference for the EmbedLLM system. However, this preference is not statistically significant, as shown by the one-tailed Wilcoxon signed-rank test, $W = 53.5$, $p = 0.1367$, and the one-sample t-test, $t = 0.9944$, $p = 0.1707$ (seen in table 5.3). Additionally, the per question results are shown in figure 5.6a.
The qualitative preference responses (seen in figure 5.6b) follow the same pattern. After excluding ambiguous or unusable responses, 6 out of 8 participants prefer the context-aware system, while 2 prefer the baseline system. A one-tailed binomial test on this preference count does not reach significance, $p = 0.1445$.
These results suggest that participants tend to prefer the EmbedLLM representation, but the

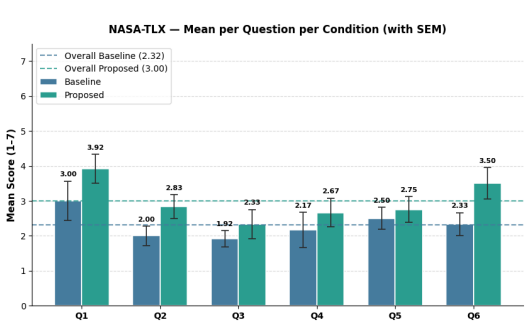
Table 5.2: Coherence and relevance significance results. Higher scores indicate more favorable ratings.

Measure	**Baseline**	**EmbedLLM**	**Statistical test**	**Effect size**
Overall coherence score	4.8929	5.2857	$t = -1.2008$, $p = 0.2550$	Cohen's $d = 0.3467$
Environment alignment ($Q1 + Q5$)	4.9583	5.0000	Wilcoxon $p = 0.6475$, $t = -0.0666$, $p = 0.9481$	Cohen's $d = 0.0192$
Goal alignment ($Q2 + Q3$)	5.1250	5.5417	Wilcoxon $p = 0.1895$, $t = -1.4832$, $p = 0.1661$	Cohen's $d = 0.4282$
Nozzle task coherence	5.5000	5.3571	Mann–Whitney $U = 20.5$, $p = 0.7475$	Rank-biserial $r = -0.1389$
Filament task coherence	4.2857	5.2143	Mann–Whitney $U = 10.0$, $p = 0.2403$	Rank-biserial $r = 0.4444$

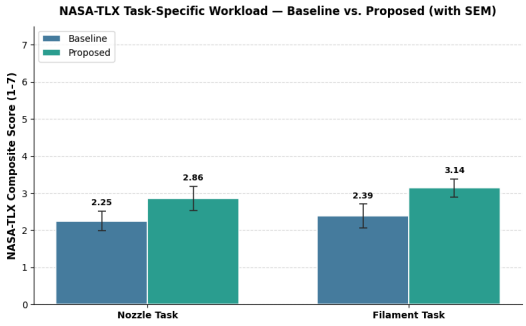
evidence is not strong enough to conclude a statistically reliable preference. Accordingly, H_6 is not supported.

Table 5.3: Comparative questionnaire significance results. Scores above the neutral midpoint of 4 indicate preference for the EmbedLLM system.

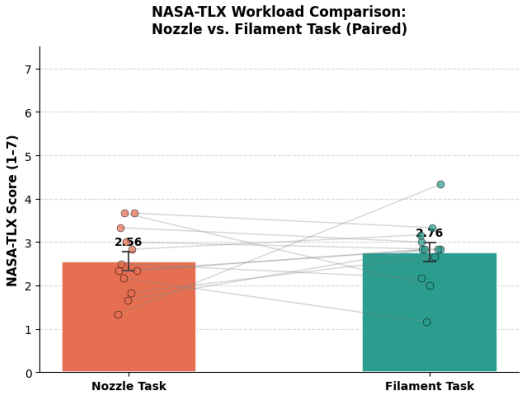
Measure	Value	Interpretation
Overall mean	4.2857	Above neutral midpoint
Shapiro–Wilk	$W = 0.9236$, $p = 0.3175$	Normal
Wilcoxon signed-rank (one-tailed)	$W = 53.5$, $p = 0.1367$	Not significant
One-sample t-test	$t = 0.9944$, $p = 0.1707$	Not significant
Effect size	Rank-biserial $r = -0.3718$	Small effect



(a) NASA-TLX questions



(b) NASA-TLX tasks



(c) NASA-TLX comparing tasks

Figure 5.7: Results of the NASA-TLX questionnaire. (a) The average scores for each question, (b) the results for both the baseline and the EmbedLLM condition for each task and (c) comparing the results for all participants over the two tasks.

NASA-TLX is used to assess the cognitive-load component of H_7. The overall composite NASA-TLX score is significantly lower in the baseline condition than in the EmbedLLM condition, (as seen in figure 5.7) with means of 2.3194 and 3.0000 respectively. This difference is significant in both the paired t-test, $t = -2.4425$, $p = 0.0327$, and the Wilcoxon signed-rank test, $p = 0.0156$, with a moderate effect size of $d = 0.7051$ (as shown in table 5.4).

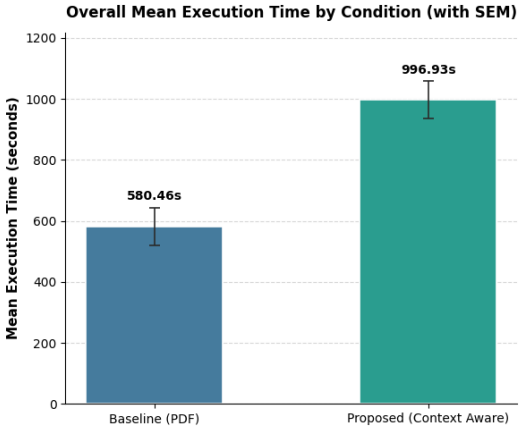
At the item level, the strongest significant difference appears on Question 2, where the EmbedLLM condition scores worse than baseline, $p = 0.0172$ in the paired t-test and $p = 0.0391$ in the Wilcoxon signed-rank test. The remaining NASA-TLX items do not reach significance individually, although most trend in the same direction as the overall composite score. These results are shown in figure 5.7a

When separated by task, the baseline condition again shows lower NASA-TLX scores for both

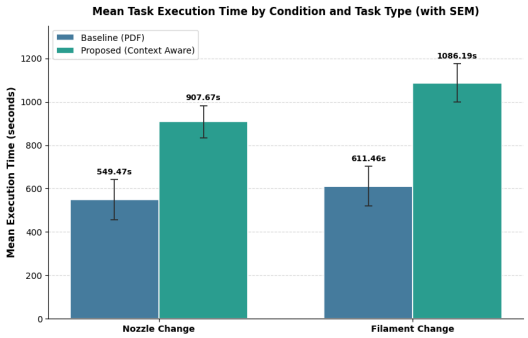
tasks. For the nozzle task, the mean increases from 2.2500 under baseline to 2.8611 under the EmbedLLM system, while for the filament task it increases from 2.3889 to 3.1389. Neither task-specific comparison is statistically significant, but both effect sizes are moderate in magnitude and in the same direction as the overall result. This is depicted in both figure 5.7b and figure 5.7c.
These results indicate that the EmbedLLM system does not reduce perceived workload relative to the baseline. Instead, the NASA-TLX findings suggest that participants experience the baseline interface as less demanding overall.

Table 5.4: NASA-TLX significance results. Lower raw scores indicate lower workload.

Measure	Baseline	EmbedLLM	Statistical test	Effect size
Overall NASA-TLX score	2.3194	3.0000	$t = -2.4425$, $p = 0.0327$; Wilcoxon $p = 0.0156$	Cohen's $d = 0.7051$
Nozzle task NASA-TLX	2.2500	2.8611	Mann–Whitney $U = 10.0$, $p = 0.2281$	Rank-biserial $r = 0.4444$
Filament task NASA-TLX	2.3889	3.1389	Mann–Whitney $U = 9.0$, $p = 0.1659$	Rank-biserial $r = 0.5000$



(a) Execution time



(b) Execution time tasks

Figure 5.8: Results of the execution time, (a) shows the execution time for each condition, while (b) shows execution time based on task.

The second component of H_7 concerns task completion time (results are shown in figure 5.8). The EmbedLLM system produces substantially longer execution times than the baseline condition (as seen on figure 5.8a), with a mean completion time of 996.93 seconds compared to 580.46 seconds for the baseline interface. This difference is statistically significant in the paired t-test, $t = -4.8348$, $p = 0.0005$, and the effect size is large, $d = 1.3957$ (seen in table 5.5).
The task-specific descriptive statistics follow the same trend (shown in figure 5.8b). The nozzle task is completed faster on average than the filament task in both conditions, with mean times of 549.47 seconds and 611.46 seconds for baseline, and 907.67 seconds and 1086.19 seconds for the EmbedLLM condition, respectively. Because only six participants contribute to each

task-condition combination, these per-task differences are retained as descriptive results rather than subjected to statistical testing.
Taken together, the execution-time and NASA-TLX results both run counter to H_7. Rather than improving support for task execution, the EmbedLLM interface is associated with longer completion times and higher overall NASA-TLX scores.

Table 5.5: Execution time significance results. Lower completion time is better.

Measure	**Baseline**	**EmbedLLM**	**Statistical test**	**Effect size**
Overall execution time (s)	580.46	996.93	$t = -4.8348$, $p = 0.0005$	Cohen's $d = 1.3957$
Nozzle task mean (s)	549.47	907.67	Descriptive only	—
Filament task mean (s)	611.46	1086.19	Descriptive only	—

Chapter 6

Discussion

6.1 Discussion of the Results

The acceptance of H_1 in combination with the trends perceived for H_4 and H_5 highlights EmbedLLM's efficacy in automatically producing content that users perceive as deeply integrated into the physical environment and suited for the user goal. Conversely, the specific outcomes of C_B (H_3) reveal a nuanced user preference: not all combinations of contextual scenes and user goals are deemed suitable or beneficial for scene-embedded orchestration. This finding is further supported by the results of hypothesis H_6, for example Scene 3 achieving a significantly higher preference score (C_B) than Scene 2 (fig 5.6b). Consequently, this finding raises the question of whether the specific scene employed in the user study would also be considered effective for scene embedding. Unfortunately, this particular context is omitted from the generalization survey, precluding a direct comparison.
Furthermore, the results from the task support user study show a trend in favor of the EmbedLLM system when compared to the baseline for environment and goal alignment during manual task support. Although the results do not reach statistical significance to formally support H_4 and H_5, the data indicates a favorable trend in the expected direction. More specifically, EmbedLLM is on average gauged as more fitting for the environment and the user task based on the quantitative results of the coherence and the comparative questionnaire. Additionally, the qualitative results of the comparative questionnaire highlights that 6 out of 8 participants prefer the context aware system (depicted on figure 5.6b).
Notably, EmbedLLM scores lower on question one of the coherence questionnaire when compared to the baseline. This question gauges if it is easy to identify and locate the physical components on the machine. Isolation of the tasks reveals that the EmbedLLM system scores lower on Q1 solely during the nozzle change procedure (as seen on figure 5.5). This result may be attributed to the structural demands of the task itself. Because the nozzle change does not involve searching for concealed printer parts, participants achieve higher average scores than in the filament task (as seen on figure 5.4b). The filament task, however, necessitates locating the hidden filament release lever. This requirement likely accounts for the lower question one average observed in that task and highlights the efficacy of the EmbedLLM system, demonstrating that the benefits of context-aware support are most pronounced in visually demanding scenarios. Furthermore, EmbedLLM underperforms on question three, which evaluates the logical flow of instructions, and question seven, which assesses linguistic clarity. Participants note that the phrasing used in the instructions was unfamiliar to them. This disconnect likely occurs because the alternative representations are developed under the assumption that users possess fundamental 3D printing expertise. Consequently, participants without prior experience fail to recognize specific parts. Notably, some are unaware of the "Bowden tube," a critical structural element of the Ender 3 that is omitted in more contemporary consumer models like those from Bambu Lab.

Regarding hypothesis H_2, the survey results reveal no significant differences in user-rated goal alignment between the traditional baseline application and the context-aware spatial embeddings. The baseline smartphone application is already rated highly for general goal utility. Crucially, these high levels of utility are successfully maintained when participants transition to the spatial alternative representations, specifically within the Kitchen and Garage contexts. Therefore, although H_2 is not formally supported, the absence of a significant decline in user-rated utility is a positive outcome. It suggests that transforming legacy documents into scene-embedded representations does not compromise overall task relevance, as goal alignment ratings remain consistently high across these varied environments.

Turning to subjective user preference, the findings from hypotheses H_3 and H_6 reveal a consistent narrative regarding the acceptance of scene-embedded interfaces. The preliminary survey evaluating H_3 demonstrates that preference is highly context-dependent. While participants favor traditional smartphone interfaces for tasks like cooking or grocery shopping, they significantly prefer the EmbedLLM approach for the mechanical bicycle repair scenario. This inclination toward spatial support during complex manual tasks is mirrored in the outcomes of the practical user study evaluating H_6. Although the comparative questionnaire results do not achieve statistical significance, a clear descriptive preference emerges, with 6 out of 8 participants explicitly favoring the EmbedLLM system over the traditional PDF window. Collectively, these results indicate that while automated spatial orchestration is not universally desired across all daily activities, it holds substantial subjective appeal for visually and structurally demanding maintenance procedures.

This subjective preference is further reinforced by the trends observed for H_4 and H_5, which gauge environment and goal alignment, during the active task support sessions. Even though the quantitative differences do not reach formal statistical significance, the coherence metrics consistently trend in a positive direction. Participants generally perceive the generated text fields, images, and semantic relationships as highly coherent with both their physical workspace and their immediate maintenance objectives.

H_7 assesses EmbedLLM's effect on task support during manual task execution. While coherence and comparative questionnaire metrics indicate a favorable trend toward the EmbedLLM system, objective measurements of execution time and NASA-TLX scores (as seen in figure 5.8 and figure 5.7) demonstrate a significant divergence between the conditions. Thus leading to the rejection of H_7. Notably, the baseline condition yields significantly lower task completion times and reduces cognitive load. Participant feedback suggests this outcome may result from the EmbedLLM system's lack of flexibility. Although users subjectively prefer it, they feel constrained by its enforced operational sequence. Conversely, the baseline affords greater autonomy in problem-solving, a structural difference that likely accounts for its lower cognitive demand and execution times.

To explicitly couple these findings back to the primary research challenge of this thesis, *how step-based PDF manuals can be automatically embedded and adapted using Mixed Reality, and the subsequent effect on task execution,* the results demonstrate a clear duality between systemic capability and practical usability. Regarding the first half of the research question, the acceptance of H_1, the sustained goal utility observed in H_2 and the positive trend regarding user preference seen in H_3 and H_6 validate that EmbedLLM can successfully parse and transform legacy 2D instructions into context-aware, scene-embedded representations without compromising task relevance. However, regarding the effect on supported task execution, the impact is highly nuanced. Subjectively, spatial orchestration enhances execution for structurally demanding procedures by assisting users in locating complex or concealed components, a benefit reflected in the qualitative preferences and the positive trends of H_4 and H_5. Objectively, as demonstrated by the rejection of H_7, this adaptation introduces measurable costs, yielding higher cognitive load and longer execution times compared to a traditional baseline. We suspect these performance drawbacks stem primarily from two compounding factors. First, the rigid, sequentially enforced delivery of the instructions likely restricts users' problem-solving autonomy. Second, the participants' limited domain knowledge as complete novices makes them

particularly vulnerable to the complex hardware terminology used in the generated content. Ultimately, while automatic scene embedding is technically achievable and subjectively preferred for complex maintenance, its effectiveness during active execution is currently bottlenecked by a lack of operational flexibility and adaptation to user expertise.

Furthermore, participant feedback highlights a potential learning curve associated with the VR environment. Specifically, P3 notes that while they prefer the context-aware system for Task 1, the baseline is easier to navigate for Task 2. P3 proposes using the baseline system as an introductory step to smoothly transition users into VR, reserving the context-aware system for users who have already acclimatized to VR support tools. Other participant feedback highlights how prior knowledge and familiarity influences system preference. P11 favors the baseline condition, noting that their pre-existing knowledge of the tasks makes the EmbedLLM system's strict chronological pacing feel limiting, though they recognize its potential utility in novel scenarios. Familiarity is similarly cited by another participant as their primary reason for preferring the baseline. This participant additionally criticizes the system for failing to indicate time-sensitive steps, despite the baseline condition sharing this exact limitation. Additionally, participants report difficulties with complex terminology across both conditions, although these complaints are more frequent when using EmbedLLM. Excluding these shared vocabulary concerns, the lack of time-sensitivity warnings is the sole critique directed at the textual content of the alternative representation. Ultimately, the primary advantage cited by those who prefer the EmbedLLM condition is its spatial orchestration, specifically its effectiveness in directing users to the relevant components of the 3D printer. Finally, a visual inspection of the paired data in Figure 5.7c reveals a distinct outlier that warrants consideration. Specifically, one participant reports an unusually high cognitive load during the filament task (scoring above 4.0), accompanied by a steep increase in workload compared to their nozzle task experience. Because the study relies on a relatively small sample size, this single extreme deviation has the potential to positively skew the aggregate data, artificially inflating the mean score for the filament condition.
In conclusion, although EmbedLLM significantly outperforms the PDF baseline in environmental alignment through automated scene embedding, spatial orchestration is not universally suited for every context and goal. Moreover, EmbedLLM's rigidly sequential delivery of information can negatively impact task support, a limitation that it shares with the traditional PDF baseline it seeks to improve upon.

6.2 Alternative Content and Context Mapping Methods

As discussed earlier in chapter 3 , we choose a traditional LLM in combination with scene graph modeling to address the research challenge. This decision is based on the current state of the art research that highlights the LLM's reasoning capabilities [23, 46], LLM scene graph reasoning [43] and scene graphs as an efficient and effective way to represent a physical environment [22, 37]. However, other methods for solving this mapping have also been explored.
An initial alternative approach involves using Graph Convolutional Networks (GCNs) to establish the content-to-context mapping. This method requires constructing a graph of the traditional 2D content and training a GCN to align the two graph representations. We ultimately discard this approach because it necessitates training a dedicated GCN model, which in turn requires a comprehensive dataset of ground-truth graph mappings. The absence of such training data is the primary barrier to pursuing this direction. Nonetheless, EmbedLLM developed in this thesis can facilitate the automated generation of this required data for future research.

Conversely, a different strategy would involve making use of Vision Language Models (VLM) specialized for robotics. Models such as Nvidia Cosmos [1] and Google Gemini Robotics ER [2] are based on traditional foundation models and are further trained on spatialized data [15]. This

[1] https://www.nvidia.com/en-us/ai/cosmos/
[2] https://deepmind.google/models/gemini-robotics/gemini-robotics-er/

allows these models to understand 3D scene and actions and enables their Embodied Reasoning (ER). However, these robotic models are catered for robotic operations. For this reason, the models prioritize low latency, consequently reducing the context window to enable this faster output. While specialized VLMs capable of embodied reasoning offer a compelling avenue for future research, they are based on prior generation foundation models. Since the state of the art prioritizes reasoning, this approach was not explored any further.

6.3 Alternative Interpreter Approaches

When researching methods to facilitate the interpreter module's main objective of instantiating the build plan created by the ArchitectAgent, some alternative interpreter methods are also investigated. The first attempt consists of a recursive algorithm. This algorithm first identifies the group of nodes that are directly connected to the environment and are not dependent on any other content nodes. To resolve the semantic relations for this initial subset, the algorithm positions each panel at its respective anchor's center and translates it along the specified semantic vector (calculated relative to the anchor's forward direction) until all environmental collisions are cleared. Upon successful placement, these panels are formally registered as new objects within the environment. The placement function is then invoked recursively until the set of additions is empty. Consequently the subsequent tier of panels, whose spatial dependencies rely on this updated environmental state, can be resolved. This recursive process is repeated until the entire build plan is instantiated.

A second alternative methodology we investigated for the interpreter module leverages the findings of Dhamo et al. [13], who introduce a Graph Convolutional Neural Network (GCN) designed to generate scenes from semantic scene graphs. Notably, their model has demonstrated the capability to instantiate and position new nodes within an existing scene graph framework. However, despite its theoretical suitability as an interpreter, its reliance on the 3DSSG training dataset presents significant limitations for our application. While the 3DSSG dataset robustly supports macro-level indoor environments (e.g., kitchens), it lacks the micro-level granularity required to distinguish intricate components, such as the individual parts of a bicycle. Furthermore, the model operates within a coordinate space scaled for positioning full 3D objects. Consequently, it is too coarse for our purposes, which require embedding small-scale supplemental content onto finely detailed object sub-components.

6.4 Questions for the Future of Context-Aware Content

When viewed in a broader perspective, the concept of a context aware content future raises multiple questions about what content is and in what way the author can control the representation of this context aware content. While limited to task based PDF guides, EmbedLLM already highlights the discrepancy between the original content representation and the context aware alternative. Another example in the situation of PDF content is that of a narrative story. For example if the user goal is: "to read this story", then the contents of the PDF could be projected "on top of" a physical book. This alternative representation stays consistent with the intention of the original author. However, what if the user goal is: "to experience the story"? This could introduce 3D visuals, animations and even require the user to explore its environment. It is clear that this alternative representation is no longer in line with the intent of the original author, and thus raises the question: in what manner can the author control these context aware content representations?
Additionally, what if we broaden the scope to 2D interactive applications. There is a growing field of research investigating methods to create GUI automation. For example Veuskens et al. proposed Rataplan [40], a pixel based approach to enable automation on Graphical User Interfaces. Since the advent of Large Language Models, this field has seen a surge in research focusing on enabling agentic AI to execute GUI interactions [30]. Both of these works enable GUI understanding in one way or another. If the GUI understanding inherent in this research

is combined with the results of EmbedLLM, this could enable interactive context aware AR applications or experiences. This raises the question: What is the role of the developer in the future of context aware AR?
Furthermore, in the context of industry it is also beneficial to be able to constrain the alternative context aware representations. For example, when executing machinery maintenance in an industry hall, some floating images and text panels could be generated. In this context, it is extremely important that it does not obscure safety features or markings on the machine. Thus it is not only important to facilitate control of the reformatting from an author's position, but also from the position of the end user.

Chapter 7

Future Work and Limitations

7.1 Limitations

From a practical standpoint, EmbedLLM is primarily constrained by high token usage and lengthy generation times, resulting in significant delays when creating alternative representations. This bottleneck makes it unfeasible to incorporate user skill levels as a variable during the evaluation. Nevertheless, because user expertise is not a designated input for EmbedLLM, adapting to it remains beyond the scope of this research. Another point of critique from participants concerns EmbedLLM's strictly sequential content delivery. Since EmbedLLM is explicitly designed to support step-by-step task execution, exploring non-sequential representations is also excluded from the current scope. Finally, while the traditional PDF baseline shares this inherent sequentiality, it naturally affords users the flexibility to skim ahead or dictate their own pace. In contrast, the immersive nature of EmbedLLM causes some participants to feel constrained by the pacing, prompting them to fully offload their cognitive reasoning and passively follow EmbedLLM's immediate prompts.

From a methodological perspective, a notable limitation of this thesis is the absence of preliminary research investigating various context combinations (i.e., pairing specific scenes with diverse user goals). Establishing this foundational understanding could inform a more optimal scene selection for the user study, potentially yielding more profound results. Furthermore, the preference questionnaire lacks strict participant screening and does not measure participants' digital literacy, which limits the generalizability of the preference results. In addition, the specific context utilized in the task support evaluation is absent from the initial preference survey. Consequently, no baseline measurements are established to gauge its inherent suitability for scene embedding. Additionally, the participant demographic selected for the evaluation may act as a confounding variable. By recruiting novices with no prior knowledge rather than individuals with baseline 3D printing experience, participants face the compounded difficulty of learning unfamiliar hardware terminology alongside navigating a novel interface. Finally, an evaluation of the system's performance with larger pieces of source content or more complex tasks is currently lacking. Specifically, processing longer-form PDF documents or more complex procedural workflows is necessary to accurately assess the quality of the generated alternative representations when the model's context window is heavily saturated.

Finally, evaluating EmbedLLM through the theoretical lenses established in Chapter 2 underscores the foundational role of ubiquitous computing. Activity theory is also deeply embedded in the system's design. During execution, tasks are segmented into sequential steps where content is spatially organized around the activity. For instance, a step instructing the user on how to remove a bicycle wheel does not merely extract and isolate relevant knowledge, it actively orchestrates that information in the physical environment relative to the objects involved. Conversely, situated action is largely overlooked. Because the generated content remains static

and fails to update dynamically, the scene-embedded representation cannot adapt to evolving real-world contexts. Consequently, this oversight may directly account for the lack of flexibility reported by participants during use, suggesting that future work must incorporate situated action to enable truly adaptive interactions.

7.2 Future Work

Context aware AR, that adapts both content design and content presentation based on the context, is a relatively novel field of research. EmbedLLM enables automatic reformatting of traditional PDF content to scene embedded content specifically for sequential manual task execution. However, many other directions can and should still be explored. Specifically, future work should expand on EmbedLLM by investigating methods to create a unified input model for context-aware Augmented Reality (AR) and exploring ways to extend the scope of context to generalize these alternative representations beyond task-based source content.

An interesting future direction is researching methods to incorporate the research in agentic AI, specifically for GUI understanding and GUI actions. This body of work directly or indirectly researches GUI understanding [21, 26, 40]. This understanding can be leveraged to investigate methods to automatically transform traditional 2D WIMP applications into scene embedded interactive experiences. Furthermore, when investigating methods for interactive experience scene embedding, this can be combined with the research in affordances to expand upon the semantic scene graph with physical affordances [9,10]. Thus leveraging the physical environment not only as a content canvas, but as an interaction medium used to control the content. Finally, integrating established principles from tangible user interface (TUI) [34, 39] research into the context-aware reformatting of interactive applications presents a promising avenue for future exploration.

Another direction to build upon EmbedLLM involves utilizing text-to-3D models, such as OpenAI's Shap-E [1], to generate richer scene-embedded representations. Compared to 2D images or text, 3D models inherently convey spatial information more effectively [38]. Furthermore, these models can leverage the physical environment to a much greater extent. For instance, incorporating 3D models could enable spatial animations of specific steps, such as pressing the lever on an Ender 3 printer, making task execution significantly more intuitive for novice users. Additionally, this might improve task support for complex tasks for non novice users.

Additionally, further research should explore optimal scene graph ontologies for context-aware content repurposing. While EmbedLLM limits its scope to the user's goal, the physical environment, and the source content, future work must incorporate digital context (e.g., calendars, messages, and photos) alongside broader user-centric facets (e.g., skill level and physical abilities). Consequently, developing a comprehensive knowledge store or ontology for scene graph creation that captures these multidimensional contextual factors warrants further investigation.

Furthermore, future research could explore the integration of spatial reasoning capabilities from embodied reasoning models, as discussed in Section 6.2. This integration could improve the overall generation of semantic nodes within the scene graph. Alternatively, it could facilitate agentic reformatting, enabling the model to position objects and panels using exact spatial coordinates rather than relying solely on abstract semantic relationships

Finally, qualitative feedback from the user study indicated a preference for greater flexibility during sequential task execution. Pursuing this avenue represents a valuable direction for future research. For example, future systems could dynamically update the contextual knowledge store in response to user-executed actions, subsequently adapting the scene-embedded representations to reflect this newly updated state.

[1] https://github.com/openai/shap-e

Chapter 8

Conclusion

This thesis investigates how step-based manual instructions presented in PDF format can be automatically embedded in and adapted to an environment and user goal using Mixed Reality, and what effect such adaptation has on supported task execution. To address this research question we present EmbedLLM, a framework that combines semantic scene graph modeling, LLM-based content reasoning, and a novel physics based 3D label placement algorithm based on scene semantics to transform legacy 2D manuals into context-aware, scene-embedded VR instructions.

In direct response to the research question, this thesis shows that step-based PDF manuals can be automatically transformed into scene-embedded Virtual Reality instructions that adapt to the environment and user goal at the level of both content selection and presentation. This content can then be acted upon using a passthrough mode. However, the results also show that such adaptation does not inherently improve task execution performance: while it strengthens perceived environmental integration and preserves goal relevance, during practical evaluation the scene embedded content leads to higher cognitive load and task execution. These findings suggest that factors such as user expertise, task characteristics, and interaction constraints may influence the effectiveness of such systems, but further research is required to examine these relationships directly.

The contributions of this thesis are fourfold. First, we introduce EmbedLLM, which, to the best of our knowledge, is the first true context-aware content reformatting framework for XR. Second, we present a novel adaptation of a force-based 3D label placement algorithm that incorporates semantic data to dynamically calculate optimal label positions. Third, we provide empirical results evaluating the use of EmbedLLM during task execution. Finally, we present quantitative findings on user perception when comparing EmbedLLM against a traditional smartphone baseline, specifically regarding perceived environmental alignment, goal alignment, and overall preference.

Our evaluation of EmbedLLM reveals a clear trade-off between perceived spatial integration and practical task efficiency. Subjectively, EmbedLLM successfully preserves goal alignment while achieving significantly higher environmental alignment than the traditional PDF baseline. However, user preference proves highly context-dependent, favoring spatial orchestration for structurally demanding manual tasks over routine procedural activities. Despite these subjective benefits, active task execution with EmbedLLM introduces measurable performance costs, yielding longer completion times and higher cognitive load compared to a standard floating PDF. User feedback indicates that these drawbacks stem in part from EmbedLLM's strictly sequential delivery. Although the baseline document is also inherently sequential, users' familiarity with the conventional format likely affords them a greater sense of freedom to explore and skim ahead, whereas the unfamiliarity with EmbedLLM inadvertently constrains their problem-solving flexibility. Additionally, the framework is constrained by high generation times and

remote LLM token usage. Furthermore, both evaluations used to assess the system's qualities are limited in terms of participant selection: the preference questionnaire lacks strict screening and does not measure participants' digital literacy, while the task-support study recruits complete novices. Because the study recruits novices rather than users with prior 3D printing experience, participants face the difficulty of learning complex hardware terminology alongside navigating the interface, an issue compounded by the system's current inability to dynamically adapt content to their specific skill level. While user experience falls outside the boundaries of the context scope defined in this thesis, the user's goal of a novice in this scenario is not confined solely to executing the repair, but also includes learning the terminology. Finally, an evaluation of the system's performance with larger pieces of source content or more complex tasks is currently lacking.

To directly address the limitations of strict sequential delivery and rigid content generation identified during our evaluation, future research should prioritize dynamically updating EmbedLLM's contextual knowledge store in real-time. By responding to user-executed actions, subsequent iterations could break away from forced pacing, enabling flexible task execution that automatically adapts to individual skill levels and is in line with the core tenants of situated-action. Alongside this technical flexibility, future research must systematically investigate which specific types of context, particularly the combinations of user goals and physical environments, are best supported by scene embedding. Because our results clearly demonstrate that spatial orchestration is not universally beneficial across all scenarios, establishing a deeper understanding of context suitability is crucial for future deployments. In addition, future work should investigate methods to create a unified input model for context-aware Augmented Reality (AR) and explore ways to extend the scope of context to generalize these alternative representations beyond task-based source content. To enrich EmbedLLM's representational capabilities, future work should explore the integration of text-to-3D models to generate intuitive spatial animations. As an alternative route for spatial reasoning, the integration of embodied reasoning models could allow EmbedLLM to anchor these elements using exact spatial coordinates, rather than relying on abstract semantic relationships. Finally, expanding upon the goal of supporting both passive documents and interactive applications, future work should explore the use of methods originating from agentic AI for GUI automation to transform traditional 2D graphical interfaces into tangible, scene-embedded experiences. This would complete the transition toward highly interactive, context-aware, spatial computing, effectively utilizing the physical environment not merely as a visual canvas, but as an active medium for interaction.

In conclusion, EmbedLLM establishes a novel framework that bridges legacy 2D documents and context-aware XR by combining content reasoning, scene understanding, and spatial presentation into a single automated pipeline. While this thesis demonstrates that such adaptations successfully integrate content with the physical environment and preserve goal relevance, these subjective benefits do not translate into immediate task performance improvements. Given the observed increases in cognitive load, longer execution times, and highly context-dependent user preferences, it is clear that spatial orchestration requires careful application. Ultimately, this work provides both the technical foundation and the initial empirical insights necessary to drive future advancements in context-aware XR content generation.

Bibliography

[1] Sai Teja Reddy Adapala. Cognitive load limits in large language models: Benchmarking multi-hop reasoning, 2025.

[2] Faisal M. Alessa, Mohammed H. Alhaag, Ibrahim M. Al-harkan, Mohamed Z. Ramadan, and Fahad M. Alqahtani. A neurophysiological evaluation of cognitive load during augmented reality interactions in various industrial maintenance and assembly tasks. *Sensors*, 23(18), 2023.

[3] Doris Aschenbrenner, Florian Leutert, Argun Çençen, Jouke Verlinden, Klaus Schilling, Marc Latoschik, and Stephan Lukosch. Comparing human factors for augmented reality supported single-user and collaborative repair operations of industrial robots. *Frontiers in Robotics and AI*, Volume 6 - 2019, 2019.

[4] Ermanno Bartoli, Dennis Rotondi, Kai O. Arras, and Iolanda Leite. Long-term planning around humans in domestic environments with 3d scene graphs, 2025.

[5] Majid Behravan and Denis Gracanin. Generative multi-modal artificial intelligence for dynamic real-time context-aware content creation in augmented reality. In *Proceedings of the 30th ACM Symposium on Virtual Reality Software and Technology*, VRST '24, New York, NY, USA, 2024. Association for Computing Machinery.

[6] Majid Behravan, Krešimir Matković, and Denis Gračanin. Generative ai for context-aware 3d object creation using vision-language models in augmented reality. In *2025 IEEE International Conference on Artificial Intelligence and eXtended and Virtual Reality (AIxVR)*, pages 73–81, 2025.

[7] Xiaojun Chang, Pengzhen Ren, Pengfei Xu, Zhihui Li, Xiaojiang Chen, and Alex Hauptmann. A comprehensive survey of scene graphs: Generation and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):1–26, January 2023.

[8] Jiangong Chen, Xiaoyi Wu, Tian Lan, and Bin Li. Llmer: Crafting interactive extended reality worlds with json data generated by large language models. *IEEE Transactions on Visualization and Computer Graphics*, 31(5):2715–2724, May 2025.

[9] Yi Fei Cheng, Christoph Gebhardt, and Christian Holz. Interactionadapt: Interaction-driven workspace adaptation for situated virtual reality environments. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23, New York, NY, USA, 2023. Association for Computing Machinery.

[10] Yifei Cheng, Yukang Yan, Xin Yi, Yuanchun Shi, and David Lindlbauer. Semanticadapt: Optimization-based adaptation of mixed reality layouts leveraging virtual-physical semantic connections. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, UIST '21, page 282–297, New York, NY, USA, 2021. Association for Computing Machinery.

[11] Shakiba Davari, Akhil Ajikumar, and Mohsen Moghaddam. Context-aware augmented reality for human-robot collaboration. In *2025 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 436–437, 2025.

[12] Shakiba Davari and Doug A. Bowman. Towards context-aware adaptation in extended reality: A design space for xr interfaces and an adaptive placement strategy, 2024.

[13] Helisa Dhamo, Fabian Manhardt, Nassir Navab, and Federico Tombari. Graph-to-3d: End-to-end generation and manipulation of 3d scenes using scene graphs. *CoRR*, abs/2108.08841, 2021.

[14] Paul Dourish. What we talk about when we talk about context. *Personal Ubiquitous Comput.*, 8(1):19–30, February 2004.

[15] Gemini Robotics Team et al. Gemini robotics: Bringing ai into the physical world, 2025.

[16] Don Gentner and Jakob Nielsen. The anti-mac interface. *Commun. ACM*, 39(8):70–82, August 1996.

[17] Steven Henderson and Steven Feiner. Exploring the benefits of augmented reality documentation for maintenance and repair. *IEEE Transactions on Visualization and Computer Graphics*, 17(10):1355–1368, 2011.

[18] Steven Houben, Jo Vermeulen, Kris Luyten, and Karin Coninx. Co-activity manager: integrating activity-based collaboration into the desktop interface. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, page 398–401, New York, NY, USA, 2012. Association for Computing Machinery.

[19] Yifei Huang, Jilan Xu, Baoqi Pei, Yuping He, Guo Chen, Lijin Yang, Xinyuan Chen, Yaohui Wang, Zheng Nie, Jinyao Liu, Guoshun Fan, Dechen Lin, Fang Fang, Kunpeng Li, Chang Yuan, Yali Wang, Yu Qiao, and Limin Wang. Vinci: A real-time embodied smart assistant based on egocentric vision-language model, 2024.

[20] Nathan Hughes, Yun Chang, and Luca Carlone. Hydra: A real-time spatial perception system for 3d scene graph construction and optimization, 2022.

[21] Yue Jiang, Changkong Zhou, Vikas Garg, and Antti Oulasvirta. Graph4gui: Graph neural networks for representing graphical user interfaces. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA, 2024. Association for Computing Machinery.

[22] Ue-Hwan Kim, Jin-Man Park, Taek-jin Song, and Jong-Hwan Kim. 3-d scene graph: A sparse and semantic representation of physical environments for intelligent agents. *IEEE Transactions on Cybernetics*, 50(12):4921–4933, December 2020.

[23] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc.

[24] Kit Yung Lam, Lik Hang Lee, and Pan Hui. A2w: Context-aware recommendation system for mobile augmented reality web browser. In *Proceedings of the 29th ACM International Conference on Multimedia*, MM '21, page 2447–2455, New York, NY, USA, 2021. Association for Computing Machinery.

[25] Jaewook Lee, Filippo Aleotti, Diego Mazala, Guillermo Garcia-Hernando, Sara Vicente, Oliver James Johnston, Isabel Kraus-Liang, Jakub Powierza, Donghoon Shin, Jon E. Froehlich, Gabriel Brostow, and Jessica Van Brummelen. Imaginatear: Ai-assisted in-situ authoring in augmented reality. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*, UIST '25, New York, NY, USA, 2025. Association for Computing Machinery.

[26] Toby Jia-Jun Li, Lindsay Popowski, Tom Mitchell, and Brad A Myers. Screen2vec: Semantic embedding of gui screens and gui components. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA, 2021. Association for Computing Machinery.

[27] Xiang Li, Heqian Qiu, Lanxiao Wang, Hanwen Zhang, Chenghao Qi, Linfeng Han, Huiyu Xiong, and Hongliang Li. Challenges and trends in egocentric vision: A survey. *Machine Intelligence Research*, 23(1):1–33, February 2026.

[28] Zhipeng Li, Christoph Gebhardt, Yves Inglin, Nicolas Steck, Paul Streli, and Christian Holz. Situationadapt: Contextual ui optimization in mixed reality with situation awareness via llm reasoning. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, UIST '24, New York, NY, USA, 2024. Association for Computing Machinery.

[29] Feiyu Lu, Leonardo Pavanatto, Shakiba Davari, Lei Zhang, Lee Lisle, and Doug A. Bowman. "where did my apps go?" supporting scalable and transition-aware access to everyday applications in head-worn augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 31(9):6112–6129, 2025.

[30] Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, Xintong Li, Jing Shi, Hongjie Chen, Viet Dac Lai, Zhouhang Xie, Sungchul Kim, Ruiyi Zhang, Tong Yu, Mehrab Tanjim, Nesreen K. Ahmed, Puneet Mathur, Seunghyun Yoon, Lina Yao, Branislav Kveton, Jihyung Kil, Thien Huu Nguyen, Trung Bui, Tianyi Zhou, Ryan A. Rossi, and Franck Dernoncourt. Gui agents: A survey, 2025.

[31] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017.

[32] Jun Rekimoto. Gazellm: Multimodal llms incorporating human visual attention. In *Proceedings of the Augmented Humans International Conference 2025*, AHs '25, page 302–311, New York, NY, USA, 2025. Association for Computing Machinery.

[33] Evan F. Risko and Sam J. Gilbert. Cognitive offloading. *Trends in Cognitive Sciences*, 20(9):676–688, 2016.

[34] Dominik Schmidt, Raf Ramakers, Esben W. Pedersen, Johannes Jasper, Sven Köhler, Aileen Pohl, Hannes Rantzsch, Andreas Rau, Patrick Schmidt, Christoph Sterz, Yanina Yurchenko, and Patrick Baudisch. Kickables: tangibles for feet. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, page 3143–3152, New York, NY, USA, 2014. Association for Computing Machinery.

[35] Xingdong Sheng, Shijie Mao, Yichao Yan, and Xiaokang Yang. Review on slam algorithms for augmented reality. *Displays*, 84:102806, 2024.

[36] Jingyu Shi, Rahul Jain, Seunggeun Chi, Hyungjun Doh, Hyung-gun Chi, Alexander J. Quinn, and Karthik Ramani. Caring-ai: Towards authoring context-aware augmented reality instruction through generative artificial intelligence. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25, New York, NY, USA, 2025. Association for Computing Machinery.

[37] Tomu Tahara, Takashi Seno, Gaku Narita, and Tomoya Ishikawa. Retargetable ar: Context-aware augmented reality in indoor scenes based on 3d scene graph. In *2020 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 249–255, 2020.

[38] Melanie Tory, Arthur E. Kirkpatrick, M. Stella Atkins, and Torsten Moller. Visualization task performance with 2d, 3d, and combination displays. *IEEE Transactions on Visualization and Computer Graphics*, 12(1):2–13, January 2006.

[39] Brygg Ullmer and Hiroshi Ishii. The metadesk: models and prototypes for tangible user interfaces. In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology*, UIST '97, page 223–232, New York, NY, USA, 1997. Association for Computing Machinery.

[40] Tom Veuskens, Kris Luyten, and Raf Ramakers. Rataplan: Resilient automation of user interface actions with multi-modal proxies. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 4(2), June 2020.

[41] Bryan Wang, Gang Li, and Yang Li. Enabling conversational interaction with mobile ui using large language models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA, 2023. Association for Computing Machinery.

[42] Mark Weiser. The computer for the 21 st century. *Scientific American*, 265(3):94–105, 1991.

[43] Dongil Yang, Minjin Kim, Sunghwan Kim, Beong-woo Kwak, Minjun Park, Jinseok Hong, Woontack Woo, and Jinyoung Yeo. LLM meets scene graph: Can large language models understand and generate scene graphs? a benchmark and empirical study. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21335–21360, Vienna, Austria, July 2025. Association for Computational Linguistics.

[44] Zhen Yang, Jinlei Shi, Wenjun Jiang, Yuexin Sui, Yimin Wu, Shu Ma, Chunyan Kang, and Hongting Li. Influences of augmented reality assistance on performance and cognitive loads in different stages of assembly task. *Frontiers in Psychology*, 10, 07 2019.

[45] Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Guyue Liu, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. Large language model-brained gui agents: A survey, 2025.

[46] Yang Zhang, Hanlei Jin, Dan Meng, Jun Wang, and Jinghua Tan. A comprehensive survey on process-oriented automatic text summarization with exploration of llm-based methods, 2025.

[47] Qi Zhao, Shijie Wang, Ce Zhang, Changcheng Fu, Minh Quan Do, Nakul Agarwal, Kwonjoon Lee, and Chen Sun. Antgpt: Can large language models help long-term action anticipation from videos?, 2024.

[48] Zijie Zheng, Yu He, Ge Yu, and Xi Xu. Binoforce: A force-based 3d dynamic label layout method under binocular viewpoints. *Electronics*, 14(11), 2025.

Appendix A

ArchitectAgent Prompts

This appendix displays the various remote LLM prompts used by the architect agent. First the overall system prompt is displayed, followed by the two prompts used for step and task generation. Subsequently the prompt for content node additions is displayed and finally the prompt to create semantic relations for each added node.

A.1 System Prompt

The comprehensive interpreter system prompt used to inform the LLM of its responsibilities. This system prompt does not invoke the LLM to execute, execution is reserved for the call specific prompts.

```
You are an assistant that takes a content PDF and a semantic environment representation
(JSON), and produces a creative alternative content representation as a build
plan in service of the user goal described at the end of these instructions.

Your output is driven by the user goal and situated within the provided environment.

Analyze both inputs.  The user goal described in these instructions is the primary
lens:  every design decision must serve what the user is trying to achieve.  The
environment informs where and how this information is anchored to physical objects.

Use the available tools to create the new representation.  Generate tool use calls,
this collection will result in the buildplan.
```

OUTPUT FORMAT

```
Return a brief explanation of how you believe the scene should look after these
changes.  This explanation should include a reasoning for each tool call given
the user's goal.  How does this benefit the user's goal?

CRITICAL: Use the provided tools to build the alternative representation.  Make
a SEPARATE tool call for EACH piece of content or semantic relation you want to
create.  For example, if you plan to create 5 text panels and 3 images, you must
make 8 separate tool calls.  This collection of tool calls is the MOST IMPORTANT
output.

If the PDF or environment JSON lacks information needed to fulfill the request,
explain here and leave the tool calls empty.
```

TOOL USE RULES

- Use the tools provided to build the scene. Make ONE tool call per piece of content. The collection of ALL tool calls will form the complete build plan.
- Example: To create 3 text panels and 2 images, you must invoke CreateText 3 times and CreateTextTo2D 2 times, for a total of 5 tool calls.
- Semantic Relationships: Use objectID's to describe the semantic relations as described in the tool definition. When creating a new object, this objectID can then also be used for further definition of semantic relations.
- Image Generation: For 'textToImage' representations, provide a detailed, descriptive prompt that visually represents a concept from the PDF that aligns with the user's goal. Image generation can only be limited to basic visualizations of information and can not be treated as a medium to create UI elements. There can be no text displayed in the images.

CREATIVITY & REFORMATTING

- Content Selection: Do not dump the PDF content. Select only the specific fragments, data points, or instructions that directly facilitate the user's goal. Prefer fewer, denser panels over many thin ones. Omit anything that does not directly support executing the goal --- background theory, historical context, warnings already known to the user, and supplementary details all get cut unless they are strictly necessary to complete the task.
- Reformatting: Rewrite selected content to fit the context. If the goal is a ''quick fix,'' reformat long paragraphs into concise, imperative steps. If the goal is ''learning,'' reformat data into clear definitions or captions.
- Actionable panels: Panel titles and body text must describe a concrete action or outcome. Use imperative language (''Remove the clip'', ''Check the pressure'') rather than descriptive labels (''Clip information'').
- Semantic Anchoring: Every text or image representation must be semantically linked to a relevant physical or newly created digital object in the JSON.
- Visual Description: For images, describe the style (e.g., ''minimalist infographic,'' ''photorealistic macro shot,'' ''schematic diagram'') to ensure the generated visual aids the user's goal.

WORK PROCESS

1. Select: Internalize the user goal and filter the PDF for the knowledge needed to achieve it.
2. Reformat: Reformat the content as text or as image. Rewrite the content to fit the user goal. E.g. When the goal is task execution short task descriptions are preferred over long explanations. Describe detailed image description if the content should be represented as an image. One piece of content can both be text and image.
3. Relate: Identify the optimal anchors in the environment JSON. Create semantic relations that facilitate ''in-scene'' anchoring (e.g., placing a reformatted ''How-To'' text block directly onto the physical tool it describes). Focus on creating semantic relationships that best enable the user goal. Each created representation should have at least one semantic relationship by specifying the semanticRelation and relationTarget parameters.
4. Iterate: Iterate over step 1-3 until no more knowledge needs to be selected and embedded.
5. Emit: Generate a SEPARATE tool call for EACH piece of content you identified in steps 1-4. Do not combine multiple content pieces into one tool call.

Each CreateText and CreateTextTo2D must be its own distinct tool call. Verify that every tool use and all params strictly match the tool definitions.

A.2 Call 1: Step Definition Prompt

Step Definition Prompt:

You are a content experience planner for an XR environment. Given a PDF document and a user goal, divide the experience into top-level chapters.

IMPORTANT: You are NOT limited to the chapters or sections present in the PDF. Your primary goal is to maximize the user's success at their stated goal. You may freely add preparatory chapters (e.g. gathering tools, safety checks, workspace preparation) or wrap-up chapters (e.g. testing, cleanup, verification) that are not in the source material. Create whatever chapters best serve the user's goal --- the PDF is a source of knowledge, not a constraint on structure.

Guidelines:

- A chapter is a major thematic or workflow block. Think of it like a book chapter --- self-contained, with a clear scope.
- Chapter 0 is always the entry point: orientation, overview, or preparation.
- Chapters should represent natural high-level workflow phases that help the user progress toward their goal.
- Keep chapter titles short (3--6 words). Descriptions are one clear sentence.
- HARD CAP: You must produce at most 6 chapters. Merge related phases before adding a new chapter.
- Prefer fewer chapters. Only split into a new chapter when the content is genuinely distinct and cannot be absorbed into an adjacent chapter. Fewer, denser chapters are better than many thin ones.
- Make every chapter actionable. Each chapter title should describe a concrete action or outcome the user will complete (e.g. ‘‘Replace the Nozzle’’, not ‘‘Nozzle Information’’).
- Only include chapters that directly support executing the user goal. Cut any chapter that covers background theory, supplementary context, or information the user does not need to act on. If it doesn't help the user complete the goal, leave it out.
- Emit exactly one DefineChapters tool call and nothing else.

Atomic Task Per Step Prompt: You are structuring an XR experience into atomic tasks. For the chapter described by the user, emit exactly one DefineTask call with 1--4 atomic tasks. Each task should cover a distinct, coherent portion of that chapter's content. Prefer fewer task --- only split when content is genuinely distinct and cannot be merged. Make every task title actionable: describe what the user will do or achieve (e.g. ‘‘Detach the tube’’, not ‘‘Tube overview’’). Default to 1--2 task unless the chapter clearly requires more.

A.3 Call 2: Generate Content Additions Prompt

Generate Content Additions Prompt:

You are generating content for Chapter <chapter_number> of <total_chapters>: ‘‘<chapter_title>’’
Step <step_number> of <total_steps_in_chapter> in this chapter: ‘‘<step_title>’’
<step_description>

```
Generate ONLY content for this step.  Do not include content from other steps
or chapters.
PANEL LIMITS: Emit at most 3 CreateText calls and at most 3 CreateTextTo2D calls.
Assign each panel a unique integer 'order' starting at 1 (1 = shown first, 2 =
second, etc.).

Full chapter list (for context --- avoid duplicating content across chapters):

    Chapter 1:  <title> --- <description>

        Step 1:  <title> --- <description>

        Step 2:  <title> --- <description>

    Chapter 2:  <title> --- <description>

    ...
```

A.4 Call 3: Generate Semantic Relations Prompt

Generate Semantic Relations Prompt:

```
You are a spatial layout assistant.  Emit exactly one CreateSemanticRelation tool
call and nothing else.
```

User Message Template:

```
User goal:  <userGoal>

Two panels are both attached to anchor ‘‘<anchorLabel>’’.
Panel A: ‘‘<panelA>’’
Panel B: ‘‘<panelB>’’

Emit one CreateSemanticRelation call:  origin_object_name=‘‘<panelA>’’, target_object_name=‘‘<panelB>’’
semantic_relation=<best fit>.  Choose from:  RightOf, LeftOf, InFrontOf, Behind,
OnTop, Under.
```

Appendix B

User Preference Survey

This appendix presents the scenarios, textual descriptions, and questions used in the preliminary qualitative survey ($n = 20$) evaluating user preference, environment alignment and goal alignment.
Note on Scale Formatting: Due to a visual truncation error during the PDF export process, the "Disagree" label on the lower end of the Likert scale appears partially cut off in this document. Please note that the scale was fully visible to participants during the live digital deployment.

Automatic scene embedded content generation

Imagine if the information you needed was anchored directly into the physical space around you, exactly where you needed it. This is called "scene-embedded content." For my Master's thesis, I am researching how to automatically generate these context-aware representations from traditional 2D documents.

I would love to get your thoughts on a few generated examples. The survey is very quick:

- You will be shown **4 different scenes** (using a group of images to represent each environment).
- For each scene, you'll simply answer **3 quick questions** on a rating scale (on a scale from 1 to 7, agree or disagree).

Please note: To minimize survey fatigue, you will not be presented with the complete source material. For instance, rather than displaying an entire procedural manual or recipe, only the initial steps are shown. Consequently, please be aware that incomplete information is an expected part of the study.

Additionally, the scenes are given in a virtual environment. Feel free to imagine the usage in AR to truly embed content in the physical environment.

Your input is highly appreciated and vital to my research. Thank you for participating!

* Indicates required question

Scene 1: **Baking an Apple cake without scene embedding**

In this scene, the user wants to bake an apple cake and is situated in his/her own kitchen. However, this time the content is not context aware and is instead viewed as a website on a smartphone.
- **Environment**: kitchen
- **User goal**: bake an apple cake

Example of the scene.

Note: this image is AI generated.

1. **Question 1.1**
   This content (webpage on a smartphone) representation is wel integrated in the physical environment (kitchen).

   Mark only one oval.

	1	2	3	4	5	6	7	
Disa	◯	◯	◯	◯	◯	◯	◯	Agree

2. **Question 1.2**
This content representation (recipe web page on a smartphone) is wel aligned with the user's goal (bake a cake).

Mark only one oval.

	1	2	3	4	5	6	7	
Disa	◯	◯	◯	◯	◯	◯	◯	Agree

3. **Question 1.3**
I would prefer a scene embedded representation (showing digital panels and images in the environment close to related objects) over this content representation (website viewed on a smartphone).

Mark only one oval.

	1	2	3	4	5	6	7	
Disa	◯	◯	◯	◯	◯	◯	◯	Agree

Scene 2: Baking an Apple cake

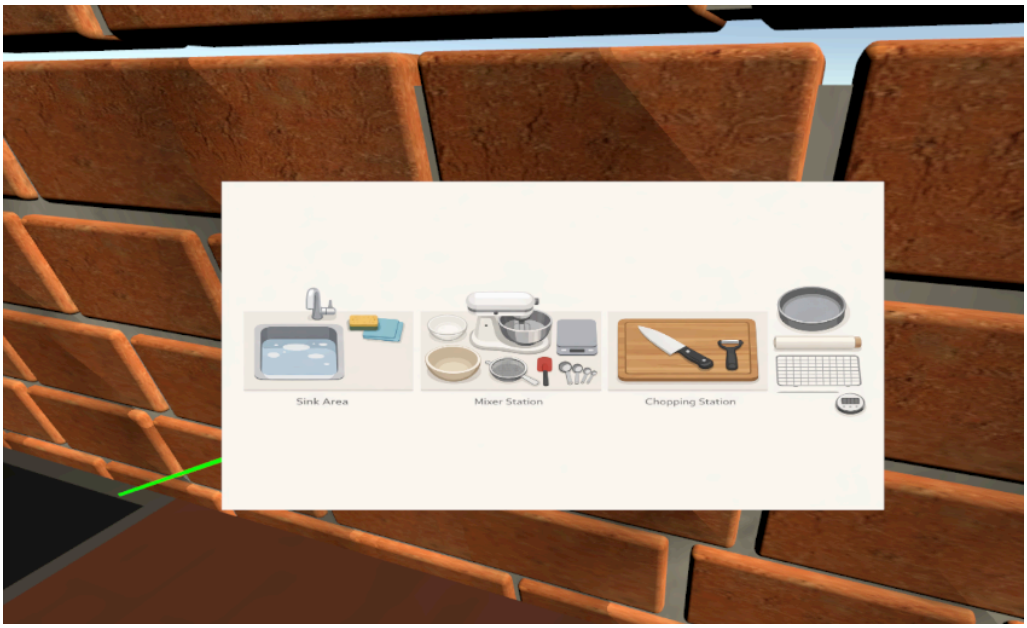
In this scene, the user wants to bake an apple cake and is situated in his/her own kitchen.
- **Environment**: kitchen
- **User goal**: bake an apple cake

Step 1

Overview of the first step: preparing work space.

Step 1.1

The image positioned above the cabinet shows a figure of how the work space should be organized. It identifies three distinct zones, the sink area, a mixing station and a cutting and baki station.

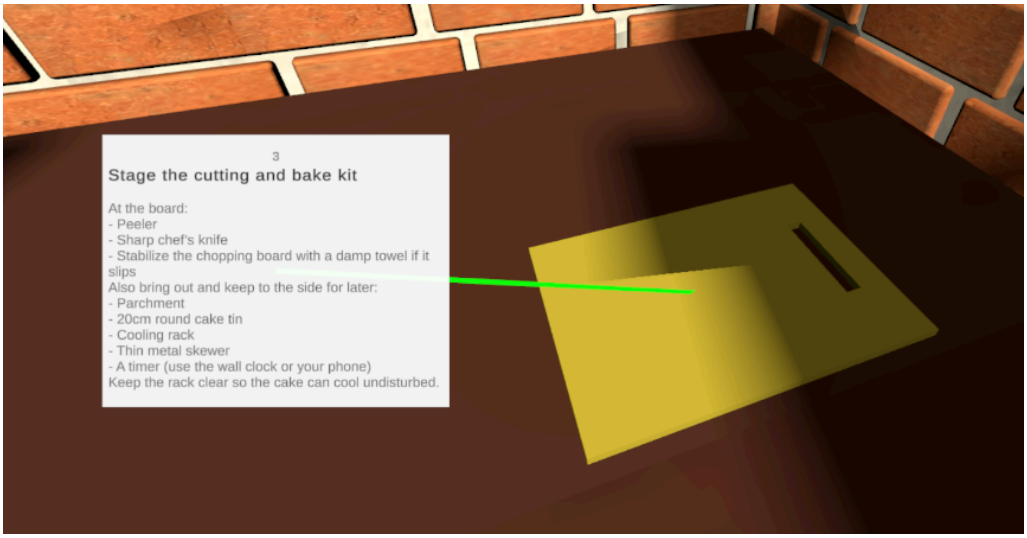


Step 1.2

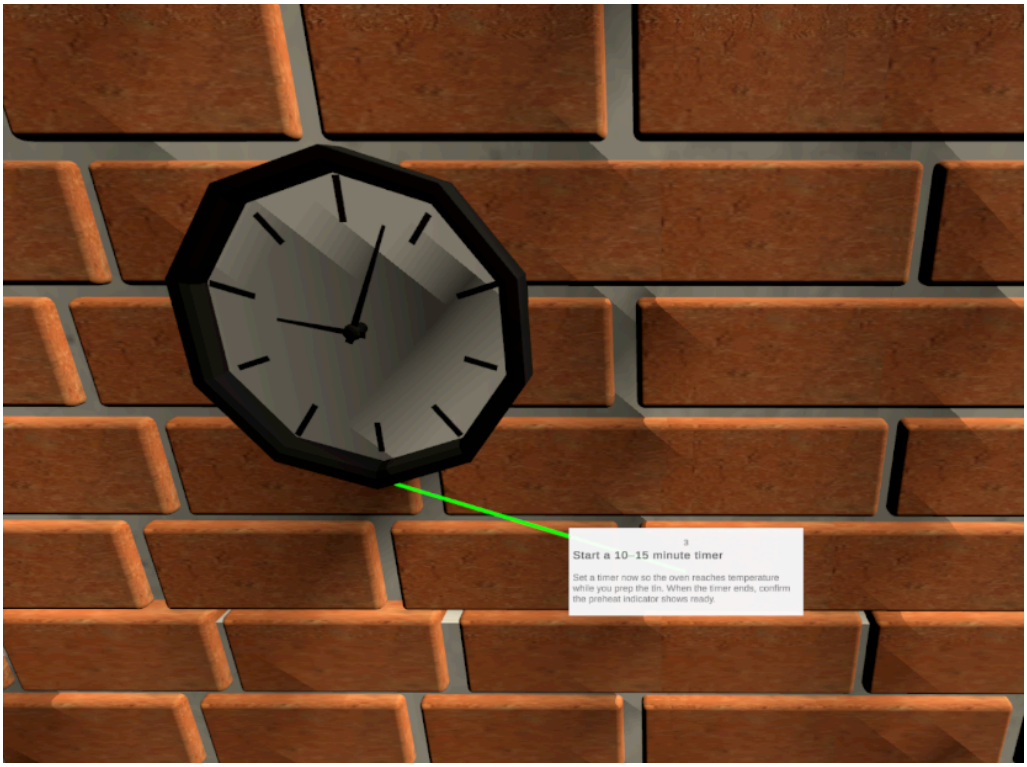
Next to the mixer a text panel is anchored which describes what should be added to the mixing zone.

Step 1.3

Next to the cutting board a panel is anchored describing what should be added to the cutting zo
mixing zone.



Step 1.4
A panel is anchored to the clock reminding the user to set a timer for the oven to reach baking temperature.



4. **Question 2.1**
This alternative content representation is wel integrated in the physical environment (kitchen).

Mark only one oval.

	1	2	3	4	5	6	7	
Disa	◯	◯	◯	◯	◯	◯	◯	Agree

5. **Question 2.2** *
This alternative content representation is wel aligned with the user's goal (bake a cake).

Mark only one oval.

	1	2	3	4	5	6	7	
Disa	◯	◯	◯	◯	◯	◯	◯	Agree

6. **Question 2.3**
I would prefer the baseline (working with a pdf or website on a smartphone) over this content representation (context aware scene embedding).

Mark only one oval.

	1	2	3	4	5	6	7	
Disa	◯	◯	◯	◯	◯	◯	◯	Agree

Scene 3: changing the tire on a bicycle

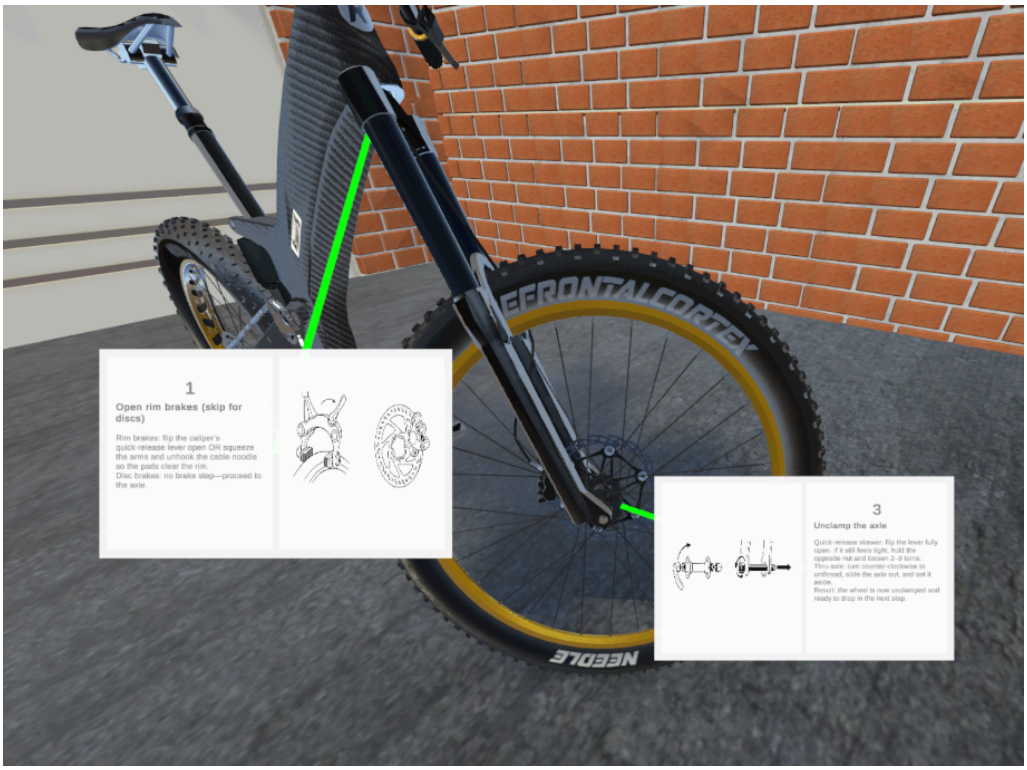
In this scene, the user wants to replace the tire on his/her bicycle and is situated in his/her own garage
- **Environment**: garage
- **User goal**: changing tire on a bicycle

Step 1

Overview of the first step: preparing the bike.

Step 1.1 (left)
A text panel with image anchored to the front fork discussing how to open each type of brake.

Step 1.2 (right)
A text panel with image anchored to the front wheel hub discussing how each type of front axil should be loosened.

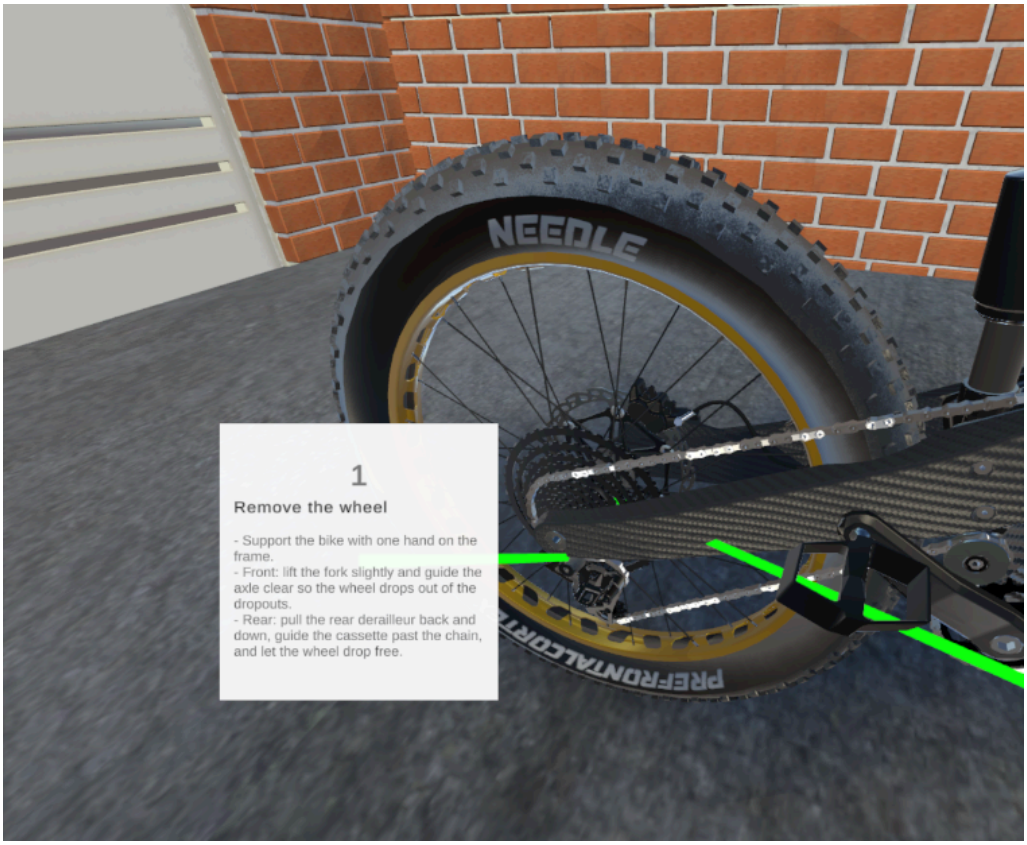


Step 2

Overview of the second step: removing the wheel and deflating the tire.

Step 2.1

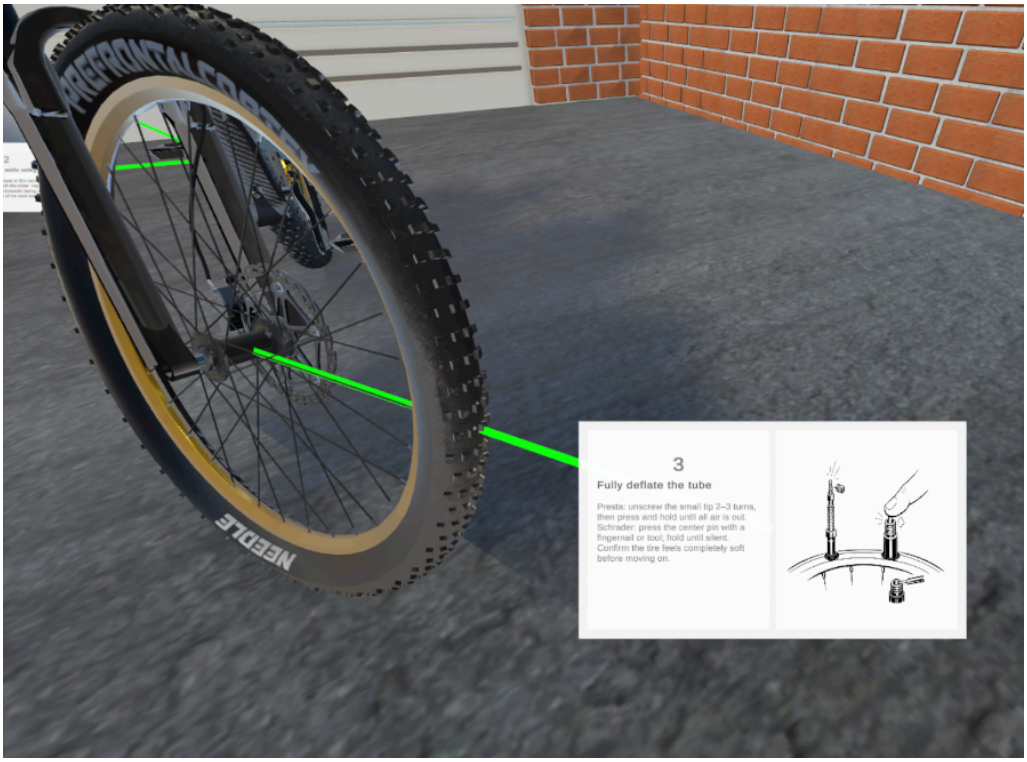
A text panel anchored to the rear wheel axis/hub describing how the user should remove the wheel.



Step 2.2

A text panel anchored to the rear wheel describing how to stow away the old wheel.

Step 2.3
A text panel and image anchored to the front wheel showing and discussing how to deflate the tire.



7. **Question 3.1**
This alternative content representation is wel integrated in the physical environment (garag containing bicycle).

Mark only one oval.

	1	2	3	4	5	6	7	
Disa	◯	◯	◯	◯	◯	◯	◯	Agree

8. **Question 3.2** *
This alternative content representation is wel aligned with the user's goal (change tire).

Mark only one oval.

	1	2	3	4	5	6	7	
Disa	◯	◯	◯	◯	◯	◯	◯	Agree

9. **Question 3.3**
I would prefer working with the baseline (a pdf or website on a smartphone) over working with this content representation (context aware scene embedding).

Mark only one oval.

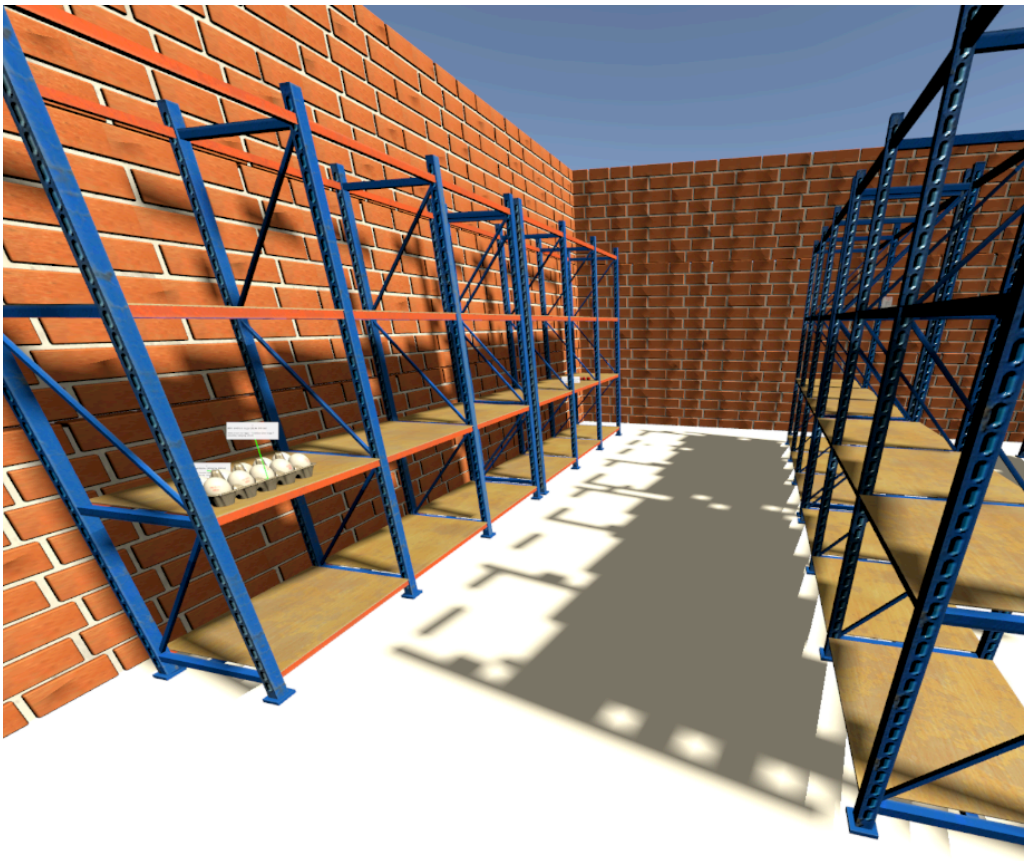
	1	2	3	4	5	6	7	
Disa	◯	◯	◯	◯	◯	◯	◯	Agree

Scene 4: Gathering supplies

In this scene, the user wants to gather the supplies to bake an apple cake and is situated in a grocery store.
- **Environment**: grocery store
- **User goal**: gather supplies to bake an apple cake

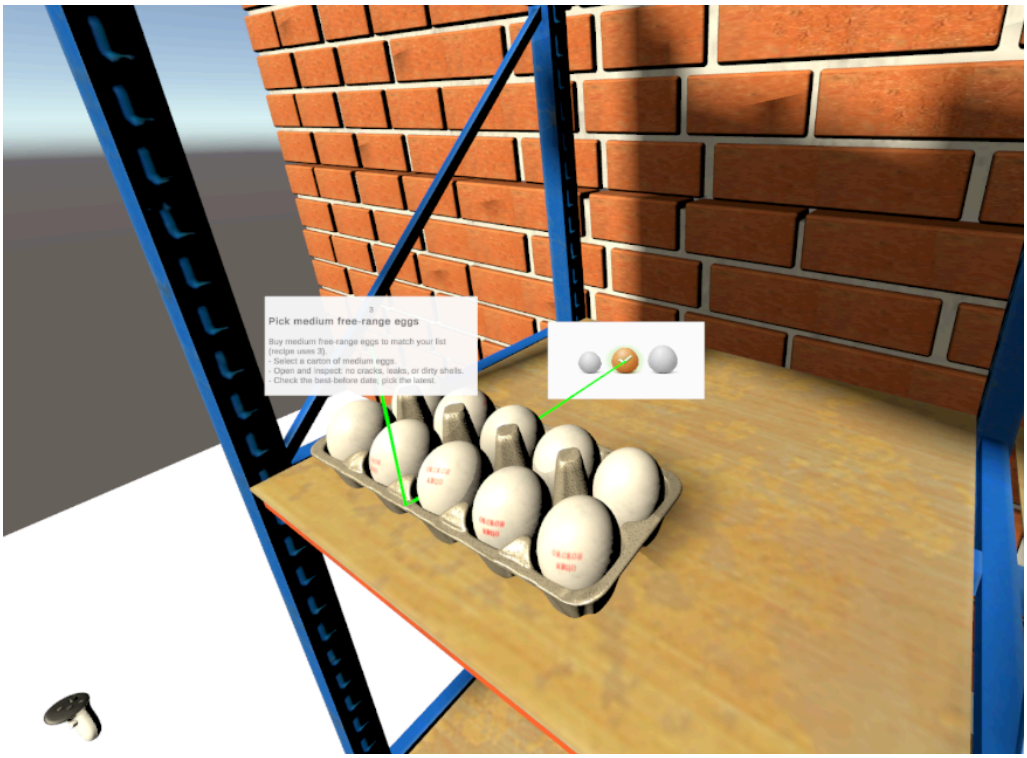
Step 1

Overview of the first step: gathering the supplies in aisle 1.



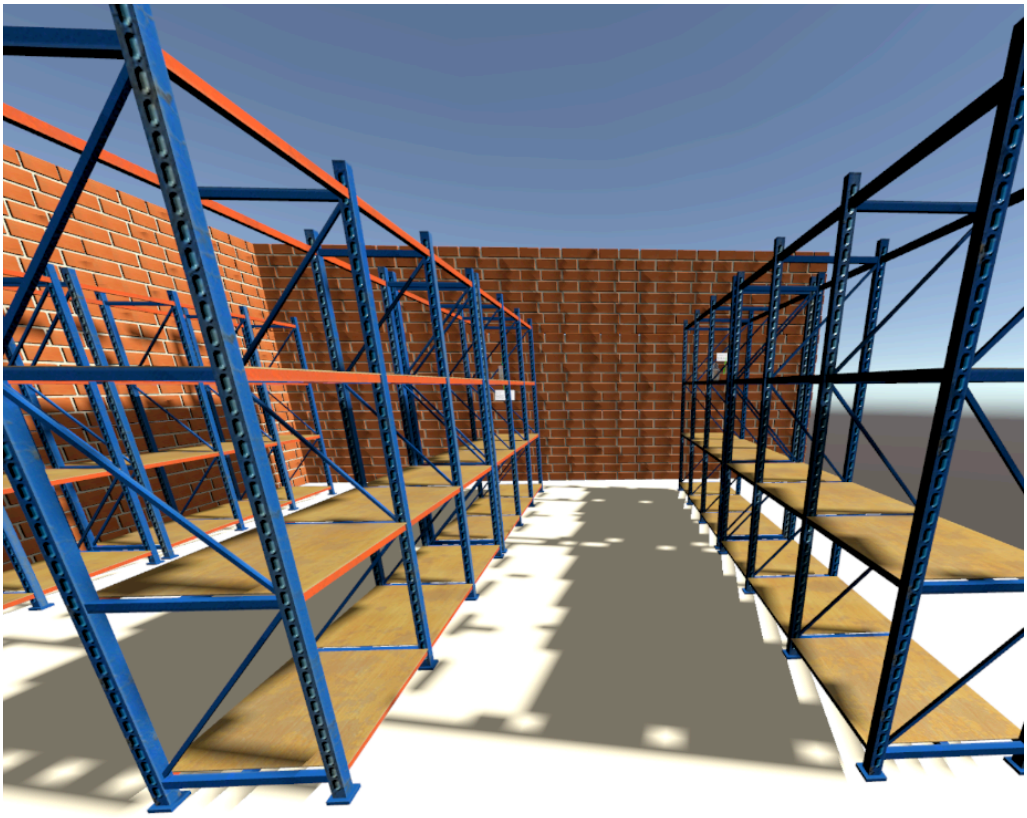
Step 1.1

Text panel and image anchored to the eggs discussing what type of eggs to choose.



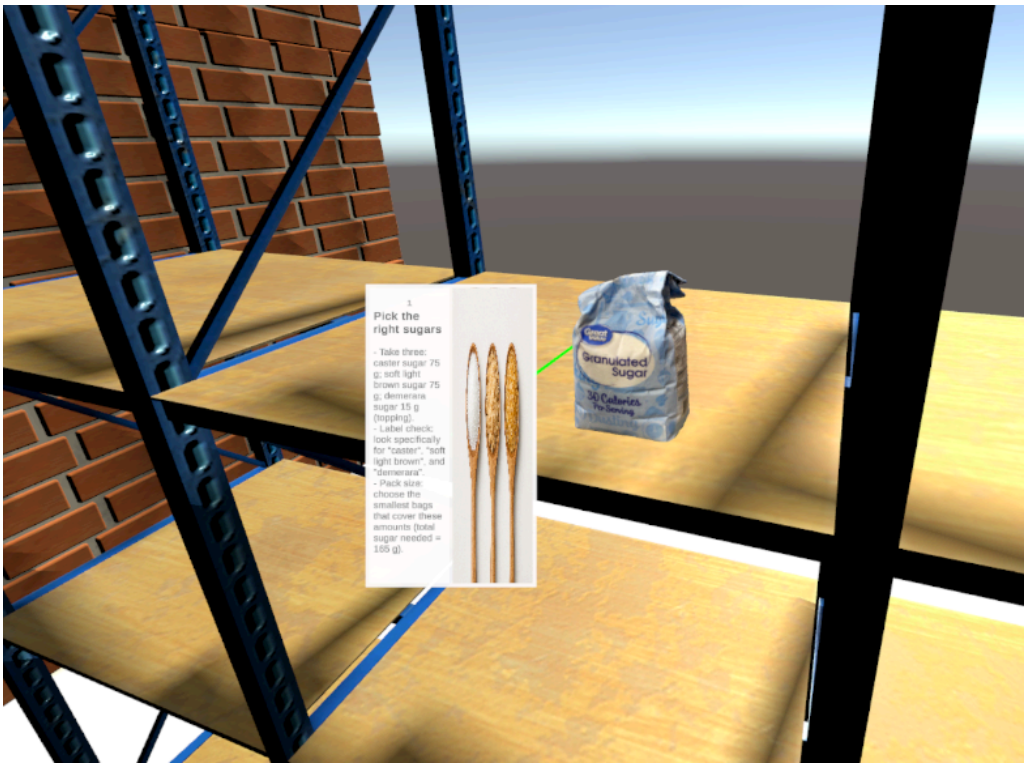
Step 2

Overview of the second step: in the other aisle an overview of the ingredients for the batter are given.



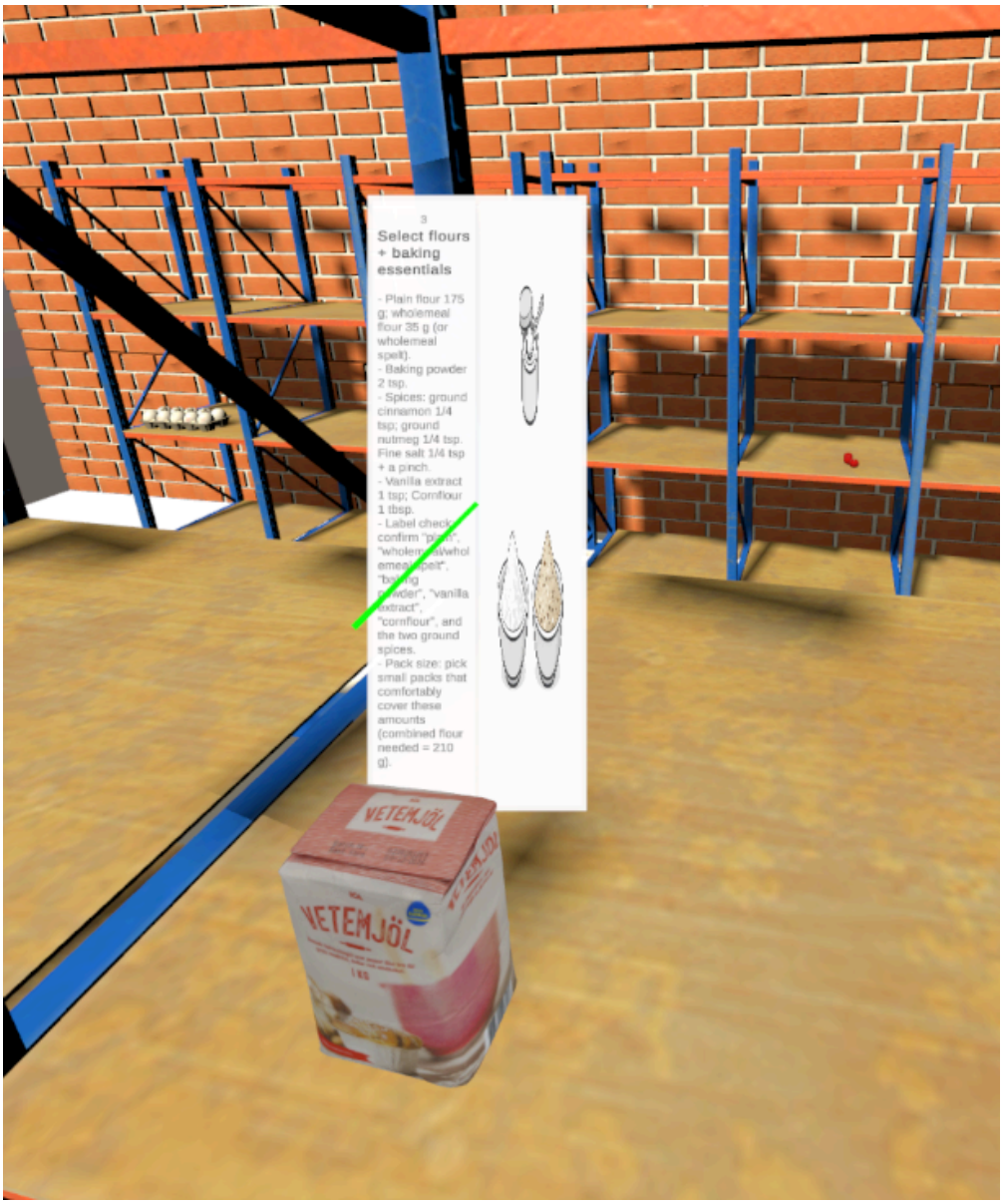
Step 2.1

Text panel and image anchored to the sugar discussing the types of suger and which one to pick



Step 2.2

Text panel and image anchored to the flour discussing which type of flour to pick.



10. **Question 4.1**
This alternative content representation is wel integrated in the physical environment (grocery store aisle).

Mark only one oval.

	1	2	3	4	5	6	7	
Disa	◯	◯	◯	◯	◯	◯	◯	Agree

11. **Question 4.2**
This alternative content representation is wel aligned with the user's goal (gather supplies)

Mark only one oval.

	1	2	3	4	5	6	7	
Disa	◯	◯	◯	◯	◯	◯	◯	Agree

12. **Question 4.3**
I would prefer working with the baseline (shopping list app on a smartphone) over working with this content representation (context aware scene embedding).

Mark only one oval.

	1	2	3	4	5	6	7	
Disa	◯	◯	◯	◯	◯	◯	◯	Agree

Appendix C

Task Support Evaluation: Legacy Source Documents

This appendix presents the source content used as input during the task support user study. The alternative representations used in the user study are based on these documents, combined with the user goal and virtual scene. Both PDF documents are based on publicly accessible websites and were exported to PDF format via a web browser.

C.1 Filament Change Source Document

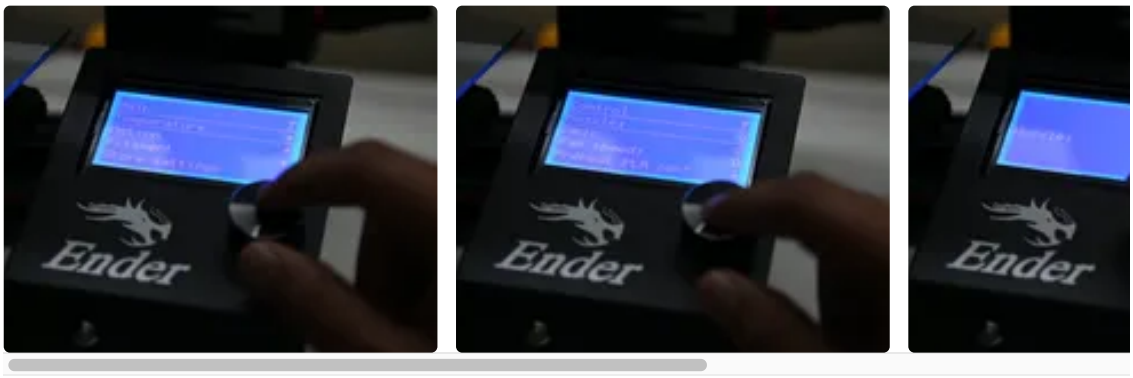
Change Filament on the Creality Ender 3 3D Printer

By VinayakNair in Workshop > 3D Printing 75.200 8 1

One of the most common tasks, when you own a 3D printer, is changing the filament. You might want to switch filament types or your filament might have snapped in between, and you would need to empty what's left in the extruder, before you re-insert from the spool. So in this Instructable I show you simple steps to switching out your filament.

Ok, here’s my current filament spool. As you can see it’s almost at the end, so before I can start a print job I need to change to a new spool as it could run-out during a print job.

Step 1:



So Step Number 1:

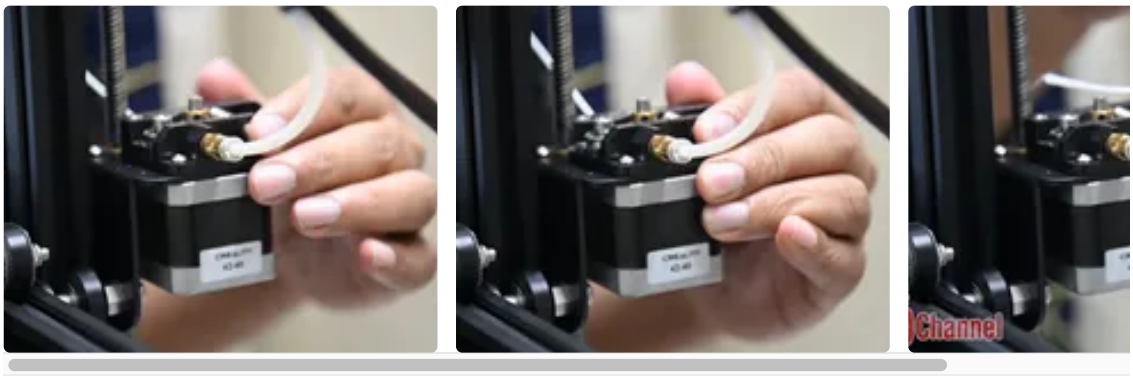
Turn the printer on

In the menu, select Temperature->Nozzle and set a temperature maybe around 200 degrees, as we need the PLA in the extruder to melt.

Click to accept the temperature, and now on the main info screen we can see the extruder heating.

The bed doesn't heat up as we have set the temperature only on the extruder nozzle.

Step 2:



Once the set temperature has reached, we would squeeze the extruder lever, which generally holds the filament and gently pull the filament out.

The end of the filament has melted and that's what has allowed us to pull it out of the extruder as visible on the end.

Do Note, don't try to pull the filament out of the extruder until it has heated properly, give it a few seconds, and once it reaches its optimum temperature it would just slide out. If you’re not careful you could damage the extruder nozzle.

Step 3:

Lodge the filament end into one of the holes on the spool and it’s ready to store. This filament might come in handy for smaller prints.

Step 4:

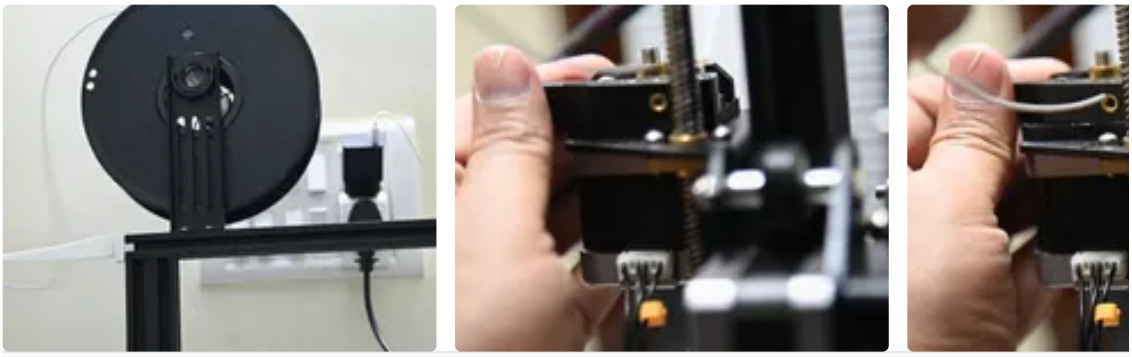
Step 2: Replace filament

I’m replacing the current white filament with a Gray one. Filaments are generally vacuum packed to prevent getting damaged by water.

I’m using PLA with the printer, as they’re cheap and easy to work with, and do not produce toxic fumes as with ABS.

The temperature ranges are marked on the spool, it can work well between 190-230 degrees centigrade for the nozzle and 30-60 degrees on the bed for better adhesion.

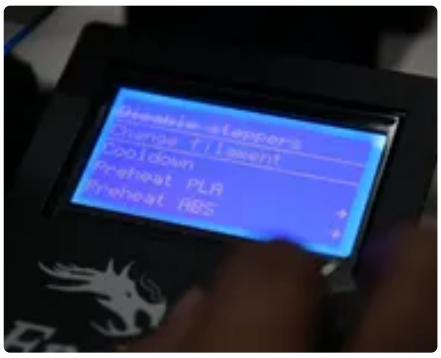
Step 5:



Now we need to feed the filament through to the extruder. Mount the spool onto the spool holder. Hold down the extruder clip and pass the filament through, continue on through the Teflon bowden tube all the way into the extruder until you feel some resistance.

The filament is close to the nozzle.

Step 6:



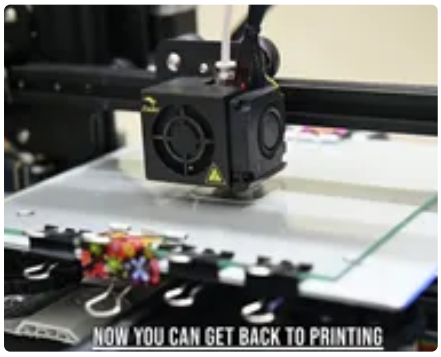
Step 3:

The filament is now at the nozzle, but it will not start printing with it as you would have remaining filament from earlier within.

This old filament needs to be purged from the nozzle, and for this, we have a handy option named “Change Filament” found under Prepare-> Change Filament

If you cannot find this option in the menu open the settings menu and scroll to Move Axis -> 1 mm -> Nozzle -> 15 to 20 instead.

Step 7:

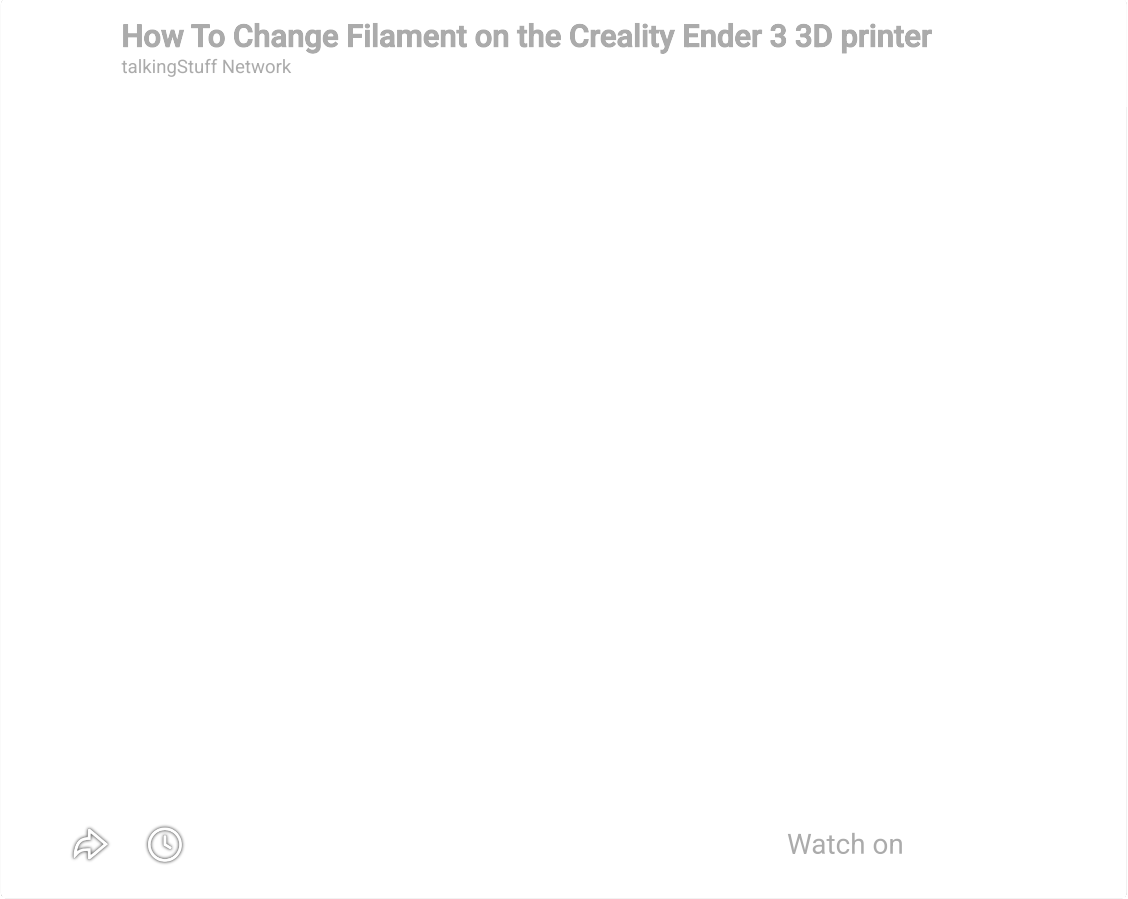


It would heat up the nozzle to 200 degrees and the existing filament is purged as the extruder starts feeding the new filament through the nozzle.

Keep an eye on the colour of the filament being purged, when it changes colour the new filament has taken its place.

In my case it’s not very noticeable, as it’s white to Gray, but if you are switching between colours, it will be quite noticeable. Let it keep purging until all the old filament is replaced by the newer one. There’s quite a lot of the old filament within.

Step 8:



So those were the steps to change the filament on the Ender 3, it’s quite simple and if you would like to watch video instructions, click above.

If you have any questions, do write in to us at tech@talkingstuff.net or whatsapp us at 9652578833.

C.2 Nozzle Change Source Document

Changing The Nozzle On A Creality Printer

The Creality series of printers allows you to change the printing nozzle, which gives you the flexibility to choose the nozzle that best suits your needs. In this guide, we will give you advice on how to choose the best nozzle size for your 3D printer. We'll also show you how to change the nozzle on the 3D printer. This guide uses a Creality CR-10 to demonstrate the process, but the same steps apply to any Creality 3D printer, including the Ender 3, Ender 5, CR-10s Pro, and more.

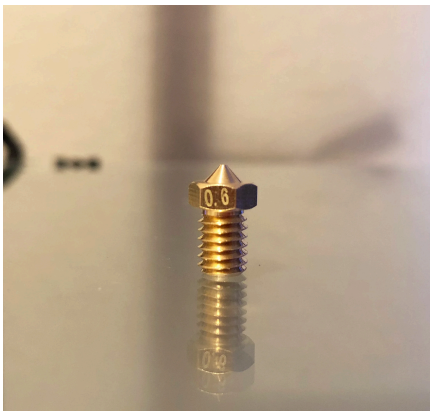
Note: Creality Experts receives a commission for items you purchase from this page, at no additional cost to you. For more information, please see our affiliate link policy.

How To Choose The Best Nozzle Size

- Choose the nozzle size that best suits your 3D printing needs
- If you aren't sure which one to use, get **this variety pack** of nozzles and start with the **0.6mm** nozzle

Nozzle size is an important factor in 3D printing, and it is important to choose the correct nozzle size for your Creality printer. The best nozzle size for you is dependent on what types of models you want to print. There are several trade-offs to different nozzle sizes:

Larger nozzles:



- allow you to use larger layer heights, which means that large models will take substantially less time to print
- generally result in stronger prints
- details and surface finish are more rough

Smaller nozzles:

- excel at printing fine details
- more desirable for prints that use support material because support material is generally easier to remove when a smaller nozzle is used
- More precision is required for leveling the print bed

Common nozzle sizes range from **0.2mm** up to **0.8mm** The standard Creality CR-10 nozzle size is a balanced size of **0.4mm**. While this is a good general nozzle size, you will find it to be frustrating if you are frequently printing larger models. Since we are often printing large models measuring 250mm or more, we prefer to use 0.6mm nozzles on most of our printers. We have found that a 0.5mm nozzle provides a good balance between the speed of a larger nozzle while still maintaining the ability to render details.

If you aren't sure which nozzle size is best for you, we would recommend starting with a **0.5mm** nozzle. If you're confident that most of your prints will be large models that take full advantage of the build volume of the CR-10, the **0.8mm** nozzle could be a potential upgrade.

Buy Your Nozzle

- Recommended variety pack of nozzles: **Amazon Link**

The CR-10 uses a standard size 3D printing nozzle, which means that there a lot of purchasing options available. Our recommendation is **this variety pack** which will give you several nozzle sizes to experiment with.

Note that you can buy nozzles of other materials such as stainless steel. We don't recommend these for general use because while they are more durable, they don't conduct heat as well as brass. They should only be considered for abrasive filaments such as metalfill or woodfill.

By the way, it's a good idea to have a spare nozzle or two. Nozzles wear out over time, which will eventually result in inconsistent printing quality. It's also possible for the nozzle to become jammed with plastic or another obstruction. These issues don't occur often, but when they do happen, it's typically easiest to just replace the nozzle.

Remove The Old Nozzle

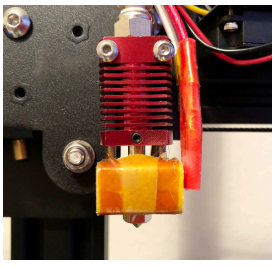
- **Heat the nozzle to 250C, then turn the heat off**
- **Use a wrench to remove the nozzle**

When you're ready to change the nozzle, the first step is to remove the nozzle that is currently installed on your printer. To do this, you should first use the printer controller to heat the nozzle to **250C**. This will loosen the nozzle so that it can be unscrewed later.

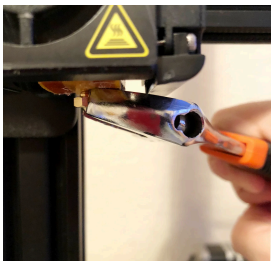
When the nozzle is heated to 240C, you should use the printer controller to set the nozzle heat to **0C** to turn the heat off. Once you've turned the heat off, it's time to remove the existing nozzle.

On the Creality CR-10 or Ender 3, the heater block is secured to the red heatsink assembly with two bolts, so you don't have to worry about breaking the heater block when you unscrew the nozzle. However, some 3D printers such as the CR-10S Pro do not have these bolts. For these printers, you should use a wrench or pliers to hold the heater block as you unscrew the nozzle.

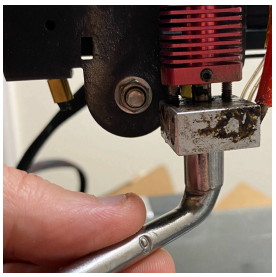
You can use the small wrench included with the printer or a crescent wrench to grip the nozzle and unscrew it. If you will be changing the nozzle often, we recommend **these nozzle wrenches** from Amazon which make it very easy to remove nozzles and install new ones. Be careful once the nozzle comes off of the heater block because it will still be very hot. We typically discard used nozzles since they are so cheap to replace, but if you want to reuse it, set the nozzle aside.



Newer CR-10s have two vertical bolts securing the heater block to the red heatsink. If your CR-10 has these bolts, there is no need to secure the heater block



You can use a wrench to unscrew the nozzle from the heater block.



We recommend **these nozzle wrenches** from Amazon which make this process very simple

while loosening or tightening the nozzle.

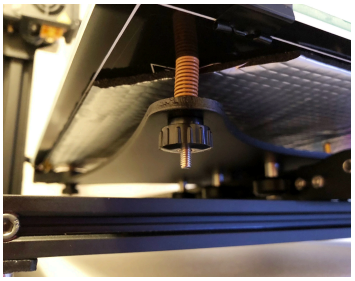
Attaching The New Nozzle

- Screw the nozzle in by hand
- Heat the nozzle to 250C, then turn the heat off
- Use a wrench to tighten the nozzle

You can attach the new nozzle in the same way by screwing it into the heater block. You might be able to use the wrench to do this while the heater block is still hot, but we typically prefer to wait a few moments for the heater block to cool below 100C and then start screwing the nozzle in by hand.

Once the nozzle is partially screwed in, you should use the printer controller to heat the nozzle to 250C again. When the nozzle reaches 250C, turn off the heater and use a wrench to tighten the nozzle. You should tighten the nozzle until it is snug, but don't use extreme force to tighten the nozzle.

Re-Level The Print Bed



- Re-level your print bed using **our guide**

When you change nozzles on the CR-10, you will affect how close the nozzle is to the print bed. Because of this, you will need to re-level the print bed after changing nozzles. Refer to our **bed leveling guide** for step-by-step instructions to level the print bed.

Success!

You've completed the nozzle changing process. Don't forget to change the nozzle size settings in your 3D model slicing software, and then go print something to test the new nozzle out!
