

인공지능 기반 시선 추적을 위한 웹브라우저 확장 프로그램 연구

최 유 경¹ · 박 태 정^{2*}

¹덕성여자대학교 디지털소프트웨어공학 전공 학부과정

²덕성여자대학교 디지털소프트웨어공학 전공 교수

AI-Based Gaze Tracking with a Browser Extension: A Study

Yugyeong Choi¹ · Taejung Park^{2*}

¹Bachelor's Course, Department of Engineering and Converged Technology, Duksung Women's University, Seoul 01369, Korea

²Professor, Department of Department of Engineering and Converged Technology, Duksung Women's University, Seoul 01369, Korea

[요 약]

시선 추적 기술은 사용자의 시선을 실시간으로 추적해 화면에서 바라보는 지점을 파악하는 기술로, 최근에는 고가 장비 없이 스마트폰·태블릿·PC의 RGB 카메라만으로도 정밀 구현이 가능해졌다. 그러나 웹 기반 환경에서의 시선 추적 기술은 일반적으로 소스 코드 수정이나 통합이 가능한 웹사이트에만 적용될 수 있다. 이를 부분적으로 해결하기 위해 웹 크롤링을 활용할 수 있으나, 난독화된 DOM 구조나 동적으로 변화하는 콘텐츠로 인해 한계가 존재한다. 본 연구에서는 이러한 제약을 극복하기 위해 Google Chrome 확장 프로그램 형태의 인공지능 기반 시선 추적 방법을 제안한다. 이를 통해 웹사이트 소스 코드에 직접 접근하지 않고도 실시간 시선 데이터를 수집하고 웹 페이지의 DOM과 결합하여 분석할 수 있다. 성능 평가 결과, 제안한 방법은 속도 저하 없이 효율적으로 시선 데이터를 저장·처리할 수 있으며, 시선 데이터 특성에 최적화된 저장 로직을 제시한다.

[Abstract]

Eye-tracking technology tracks a user's gaze in real time to identify the exact point being viewed on a screen. Recently, accurate gaze tracking has become possible using only the RGB camera of smartphones, tablets, and PCs, without the need for expensive dedicated equipment. However, in web-based environments, such technology is generally limited to websites that allow source code modification or integration. While web crawling can partially address this limitation, it faces challenges due to obfuscated DOM structures and dynamically changing content. To overcome these constraints, this study proposes an AI-based eye-tracking method implemented through a Google Chrome extension. This approach enables real-time gaze data collection and integration with the web page's DOM without direct access to the site's source code. Performance evaluation results demonstrate that the proposed method stores and processes gaze data without degrading speed, and introduces a storage logic optimized for the unique characteristics of gaze data.

색인어 : 시선 추적, 웹 서핑, 인공지능, 콘텐츠 분석, 브라우저 확장 기능

Keyword : Eye Tracking, Web Browsing, Artificial Intelligence, Content Analysis, Browser Extension

<http://dx.doi.org/10.9728/dcs.2025.26.10.2849>



This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (<http://creativecommons.org/licenses/by-nc/3.0/>) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

Received 12 August 2025; **Revised** 04 September 2025

Accepted 16 September 2025

***Corresponding Author; Taejung Park**

Tel: +82-2-901-8339

E-mail: tjpark@duksung.ac.kr

1. 서론

1-1 연구 배경

시선 추적은 사용자의 시선을 실시간으로 추적하여, 사용자가 바라보고 있는 시선 좌표를 연산하는 기술로 최근에는 여러 분야에서 쓰이며 적용 분야가 점차 증가하고 있다. 과거에는 안경이나 헤드마운트 장치와 같은 고가의 전용 장비가 필수적이었으나 인공지능기술 기반 시선 추적 기술이 등장함으로써 정밀도가 더욱 향상되고 있다.

특히, 인공지능 기반 시선 추적 기술 중에서도 Appearance-based 시선 추적 기술[1]을 기반으로, 고가의 전용 장비 없이 사용자가 사용 중인 스마트폰, 태블릿, PC 등 다양한 장치에서 RGB 카메라만 이용해서 사용자가 현재 화면에서 보고 있는 위치를 정확하게 파악할 수 있는 기술이 상용화되고 있다.

이러한 인공지능 기반 시선 추적 기술은 다양한 매체에 적용할 수 있으나 JavaScript를 중심으로 하는 웹 기술 기반에서도 활용 가능하다. 이러한 기술적 특성은 최종 사용자가 모바일 환경이나 데스크탑 환경에서 웹 기술에 기반한 웹서핑, 인터넷 쇼핑, 동영상 이용 등을 주로 사용하는 점을 감안해 볼때 사용자가 많이 본 제품을 추천하거나 광고를 제시[2]하고, 동영상 광고에서 광고주가 의도한 지점을 소비자들이 통계적으로 유의미하게 보고 있는지 정량적으로 제시[3]할 수 있어 상업적, 기술적으로 큰 잠재력을 내포하고 있다.

그러나 웹 기반 시선 추적 기술의 경우 해당 웹 서비스를 제공하는 업체에서 제공하는 웹사이트에 소스 코드 수준까지 코드 접근과 소스 코드 수정, 추가를 지원하지 않으면 적용이 불가능하다.

부분적으로 이러한 문제를 극복하기 위해 외부에서 대상에 대한 정보를 가져오는 크롤링을 사용할 수도 있겠으나, 각 사이트의 웹 접근성에 의해 DOM(Document Object Model) 정보가 난독화되어 얻을 수 있는 정보에 한계가 존재한다. 또한 최근 동적으로 변화하는 웹 페이지 특성 상 데이터 처리에 어려움이 있을 수 있다.

또 다른 접근 방법으로 시선 추적 기술과 적용 대상을 합쳐 하나의 서비스로 구축하는 경우도 생각해 볼 수 있을 것이다. 이 경우에는 시선 추적이 작용하는 것에 따라 대상 서비스의 UI 구조, 내부 정보 등에 접근할 수 있으므로 분석에 이용할 정보가 늘어날 것이다. 그러나 이런 방식은 처음부터 시선

추적이 적용된 서비스를 구축해야 하므로, 해당 분야의 분석 하나만을 위해 전체 서비스를 만들어야 하는 번거로움이 존재하게 된다.

본 연구에서는 이러한 문제를 해결하기 위해서 대표적인 웹브라우저인 Google Chrome 환경에서 작동하는 확장 프로그램 모듈로 인공지능 기반 시선 추적 기술을 구현함으로써 웹서비스의 소스 코드에 직접 접근하지 않고서도 시선 추적 기술을 활용할 수 있는 방법을 제안한다. 또한 이 확장 프로그램을 통해 다양한 웹사이트에서 시선 데이터를 실시간으로 추적하고, 해당 데이터를 웹 페이지의 DOM(Document Object Model) 구조와 결합하여 분석하는 방법을 제시한다.

1-2 논문의 구성

본 논문은 시선 추적 기능을 크롬 확장 프로그램으로 구현하는 과정과 향후 데이터 분석 및 활용을 위한 데이터 전달 및 저장 과정을 제시한다. 1장 서론에서는 연구 배경과 목적, 관련 연구를 제시한다. 2장 본문에서는 크롬 확장 프로그램에 대한 기본적인 구조, 본 연구에서 개발한 시스템 구조, 시선 데이터 전달에 관한 방법론, 성능 측정, DOM 접근에 대한 내용을 논의한다. 3장 결론에서는 해당 연구의 의의와 향후 진행 상황 등을 제시한다.

1-3 관련 연구

1) 인공지능 기반 시선 추적 기술

과거에는 안경이나 헤드마운트 형식의 전용 하드웨어 장비를 이용해서 안구 주변에 적외선 카메라를 배치하고 사용자의 안구에 대한 해부학적, 기하학적 모델을 기반으로 한 model-based 시선 추적 기술이 주로 사용되었으나 Gaze Capture[1]의 등장으로 딥러닝 기술을 활용하여 모바일 기기 사용자의 얼굴 이미지를 실시간으로 포착해서 시선을 추적하는 appearance-based 방식이 시작되었다. DVGaze[4]는 RGB 카메라 2대를 이용하여 추가적인 정보를 확보하고 Transformer 구조[5]를 활용하여 시선 추적 정확성을 향상시켰다. 이렇게 카메라를 2대 이상 사용하면 epipolar geometry를 이용하여 이미지의 중요 영역에 대한 3차원 정보를 복원[6]할 수 있다. 시선 추적에서 3차원 정보의 복원이 중요한 이유는 카메라와 눈까지의 거리 계산이 최종적인 예측 정확성에 큰 영향을 주기 때문이다. 따라서 깊이 카메라

표 1. 기존 시선 추적 방식과 제안된 방법의 비교

Table 1. Comparison of conventional gaze tracking approaches and the proposed method

Category	Conventional Gaze Tracking Approach	Proposed Method (Chrome Extension)
Application Scope	Limited to specific websites requiring source code modification	Applicable to a wide range of web services without source code access
Real-time Interaction	Restricted	Supported
Implementation	Requires development of a dedicated service	Integrated as a browser extension to existing services
Technical Limitations	Affected by obfuscated DOM structures, dynamic page changes, and access restrictions	Mitigates these issues through browser-level integration

표 2. 크롬 확장 프로그램 구성 요소 및 기능

Table 2. Chrome extension component configuration

Component	Filename	Manifest Key	Function
Popup UI handler	popup.js, popup.html	action.default_popup	Provides user interface to trigger gaze tracking and control the module
Background script	background.js	background.service_worker	Runs as a service worker for event-driven tasks and manages gaze module.
Content script	content.js	content_scripts	Injected into web pages to collect DOM info and display real-time gaze data.

(depth camera)를 추가적으로 활용하여 직접적으로 눈과 카메라의 거리를 측정하는 기법[7]도 제안되었다. 이러한 Appearance-based 시선 추적 방식에서는 대량의 학습 데이터가 필요한데 GazeNeRF[8]는 NeRF[9]를 시선 추적 학습 데이터 생성에 활용하기도 하였다. 또한 시선 추적의 전처리 단계로 눈감박임 여부를 판단하는 과정도 매우 중요한데 역광, 조명 부족 등으로 인해 사용자의 얼굴이 거의 보이지 않는 환경에서도 눈감박임을 효과적으로 인식하는 기술이 연구된 바 있다[10]. 최근에는 사용자마다 다른 해부학적 구조의 차이점을 극복하기 위한 개인화 연구[11]도 수행되고 있다.

2) 시선 추적 기술 활용

시선 추적 기술은 차량 내 운전자 안전을 위한 시선 추적 연구[12]-[14], 모바일 상품 주문 사이트에서의 사용자 시선 추적[2], 영상 콘텐츠 분석[3] 등 다양한 분야에서 활용되고 있다.

본격적인 인공지능 기반 시선 추적 기술에서는 시선 추적의 정확성을 증가시키는 측면에 주력하는 것이 일반적이지만 제한적인 시선 추적 기술의 정확도를 보다 잘 활용하기 위한 연구들도 진행된다. DynamicRead[15]에서는 시선 체크, 시선 고정 시간, 시선 추적, 텍스트 화면 전환 등 모바일 환경에서 빈번하게 수행되는 사용자 인터랙션을 최적화하는 방안을 연구하였다.

또한 사우디아라비아의 음식 배달 앱인 HungerStation은 시선 추적 기술을 활용한 캠페인으로 2023년 Canne 영화제 Create Commerce 수상작[16]으로 선정된 바 있다. 방송에도 활용되는 사례가 등장하고 있는데 2024년 글로벌 OTT (Over-the-top media service) 기업에서 방송한 예능 서바이벌 프로그램[17]에서 시선 추적 기술이 활용된 사례도 있다.

1-4 선행 연구와의 차별점

본 연구는 Appearance-based 시선 추적 방식을 웹브라우저 확장 프로그램 형태로 구현하였다. 이러한 접근은 동일한 Appearance-based 시선 추적 알고리즘을 기반으로 하면서도, 다양한 알고리즘을 적용할 수 있는 확장성을 제공한다. 기존의 웹 환경에서는 앞서 논의한 바와 같이 소스 코드 접근 제한 및 동적으로 변화하는 페이지 특성으로 인해 시선 추적 기술을 실용적으로 적용하기 어려웠다. 예를 들어, 아마존(Amazon)이나 넷플릭스(Netflix)와 같은 대표적 상용 웹

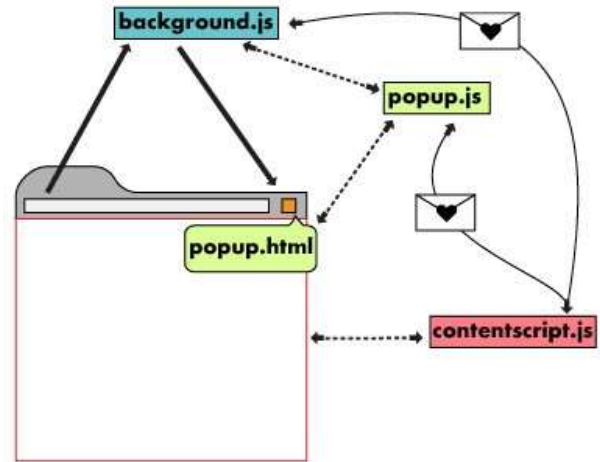


그림 1. 크롬 확장 프로그램의 구조도
Fig. 1. Fig architecture of a Chrome extension

서비스에서는, 서비스 제공자의 협조 및 내부 소스 코드에 대한 접근 권한이 보장되지 않는 한 기존 방식의 시선 추적 기술을 적용하기는 사실상 불가능하다. 이러한 차별점은 표 1에 제시된 바와 같이 기존 접근 방식과의 비교를 통해 보다 명확히 확인할 수 있다.

II. 본 론

2-1 시스템 구성

1) 크롬 확장 프로그램의 기본 동작 구조

본 프로그램은 Google의 확장 프로그램 사양인 Manifest V3[18]를 기반으로 개발되었다. Manifest란 확장 프로그램의 메타 정보와 실행 방식, 보안 정책, 필요한 권한 등이 정의되어 있으며, 이는 확장 기능의 전체 동작을 제어하는 핵심 설정 파일이다. 표 2와 그림 1[19]에서는 크롬 확장 프로그램의 기본적인 구성요소와 각 기능을 제시하고 있다. 예를 들어 현재 접근한 웹페이지의 주소를 저장하고, 그 주소를 페이지 상단에 텍스트박스로 표시하는 확장 프로그램을 실행한다고 해보자. 실행 과정에서의 Popup.js, Background.js, Content.js의 동작 예시는 다음과 같다.

• **Popup.js**

크롬 확장 프로그램 아이콘을 누르면 열리는 팝업창에 해당한다. 팝업에 있는 ‘링크 저장 및 표시’ 버튼을 누르면, 현재 보고 있는 웹사이트 주소를 저장하고 그 주소를 화면에 표시 하라는 명령이 전달된다.

• **Background.js**

사용자의 직접적인 상호작용 없이 백그라운드에서 실행되는 작업을 담당한다. 네트워크 요청, 데이터 저장, 탭의 상태 관리 등을 처리하며, 웹 어플리케이션의 서버 역할과 유사하다. 따라서 popup에서 메시지로 전달받은 주소를 저장하고, content에게 이 주소를 페이지에 띄워달라고 전달하는 과정을 거치게 된다.

• **Content.js**

웹 페이지의 DOM에 직접 접근하여 페이지에서 발생하는 이벤트를 처리한다. background에서 전달된 요청에 따라 현재 웹페이지 상단에 텍스트박스를 만들어 해당 주소를 표시한다.

2) **EyeTracker Extension의 기본 구성**

본 연구를 위해 구현된 크롬 확장 프로그램을 EyeTracker Extension이라고 칭하겠다. EyeTracker Extension은 AI 기반의 Appearance-based 방식으로 사용자의 PC·노트북에 내장된 RGB 카메라를 자동 인식하여 시선 추적을 수행한다. 전체적인 시스템 구조는 그림 2와 같다. 사용자가 임의의 탭을 열고, 해당 탭의 팝업창에서 시선 추적 시작 버튼을 누른다, Background script에서 EyeTracker가 적용된 Iframe을 생성하여 사용자가 상호작용한 탭에 부착한다. 즉 시선 모듈은 popup.js의 작용에 따라 background에서 시선 추적 모듈이 포함된 iframe을 생성해 현재 탭에 부착하는 식으로 구성된다.

• **Popup.js**

사용자가 시선 추적 시작 버튼을 누르면, 입력된 정보에 따라 해당 모듈을 초기화하고, 실행하라는 메시지를 Background.js에 보내게 된다. 시선 모듈 초기화 시 연산의 정확성을 위해 사용자와 모니터와의 거리, 모니터 사이즈에 대한 정보가 필요하다. 따라서 해당 정보도 시선 추적 시작을 지시하는 메시지와 함께 Background.js에 넘겨주게 되어 초기화에 이용된다.

• **Background.js**

시선 추적 모듈을 백그라운드에서 관리하는 데 사용하였다. 백그라운드 스크립트는 탭 상태를 관리하고, 추적 상태를 유지하며, 데이터와 설정을 저장하는 역할을 한다. 시선 추적 모듈이 포함된 iframe을 각 탭에 부착하는 환경 설정을 실행하고, iframe 생성 명령을 Content.js에게 보낸다.

• **Content.js**

해당 프로그램에서는 시선 추적 데이터를 웹 페이지에 실시간으로 표시하고, 사용자의 데이터에 반응하여 페이지를 동적으로 수정하도록 구현하였다. 이와 같이, content script는 다른 트리거 없이 모든 탭에 계속 실행되기 때문에, static하게 유지되도록 background에서 시선 추적 모듈과 데이터를 관리하도록 설계되었다.

3) **Manifest 설정**

위에서 언급한 내용은 기본적인 동작을 위한 시스템 설계이고, 시선 모듈의 원활한 동작을 위해 manifest.json 파일 내부에 추가로 설정해야 할 요소들은 다음과 같다.

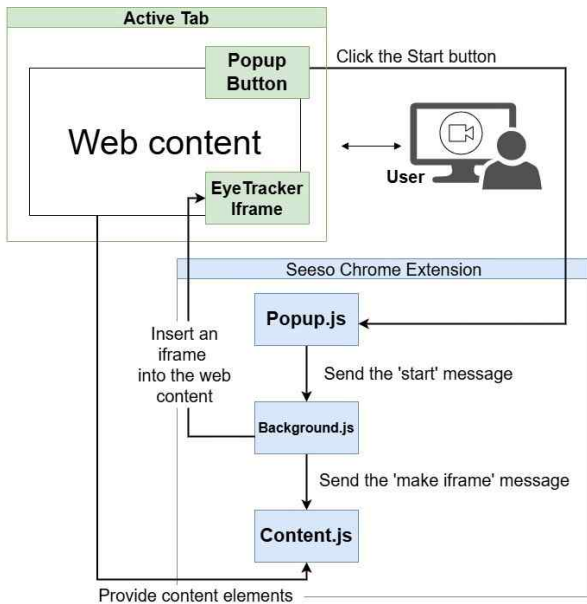


그림 2. EyeTracker Extension의 시스템 구조도
Fig. 2. Fig architecture of a EyeTracker Extension

표 3. 매니페스트 파일 내 주요 키 및 설정값

Table 3. Key attributes and values in manifest file

Manifest Key	value
cross_origin_embedder_policy	require-corp
cross_origin_opener_policy	same-origin
host_permissions	"<all_urls>", "https://console.seeso.io/*"
content_security_policy	"extension_pages": "default-src 'self'; connect-src 'self' https://console.seeso.io https://www.google.com; script-src 'self' 'wasm-unsafe-eval'; object-src 'self';"

• **Content_security_policy**

COEP(Cross Origin Embedder Policy)와 COOP(Cross Origin Opener Policy)는 시선 모듈 실행 시 후술할 SharedArrayBuffer 이동을 위해 Cross-Origin-Isolation 환경 설정이 필수적이다. 보안 설정을 만족하기 위해 적용한 설정 목록은 표 3을 참고하면 된다.

• **Permissions**

Chrome Extensions에 내장되어 있지 않은 API 기능을 이용할 때 허용해야 한다. 본 연구에서 사용한 주요 API는 “tabs”, “scripting”, “storage”, “alarms”, “declarativeNetRequest” 이다.

• **Web_accessible_resources**

Chrome Extensions를 실행하기 위해 manifest에서 등록해놓은 파일 이외에 실행할 파일을 등록한다. 본 프로그램에서는 seeso module 내부의 파일이 여기에 해당한다.

2-2 시선 추적 모듈 구성

Seeso-web-sample[20]은 VisualCamp사에서 배포한 웹 기반 시선 추적 프로그램으로, 시선 좌표를 실시간으로 연산하는 코드를 HTML 페이지에 삽입하여 사용할 수 있다. 기본적으로 이 코드는 localhost 환경에서 호스팅되며, 시선 추적이 적용되는 페이지가 고정되어 있다. 이로 인해 외부 웹사이트에서의 시선 추적 기능 활용에는 제한이 따른다. 현재는 <iframe> 태그를 통해 추적 대상 웹페이지를 삽입하고, 연산된 시선 좌표와 해당 페이지 요소 간 상호작용을 시도할 수 있으나, 보안 정책(CORS, cross-origin isolation 등)으로 인해 외부 페이지의 콘텐츠를 iframe으로 불러오는 데 제약이 발생한다.

본 연구에서 사용한 시선추적 시스템의 기본 구조는 이 Seeso-web-sample을 참고하였다. 실제 시선 추적 연산을 담당하는 모듈로는 VisualCamp사가 npm(node package manager)에 배포한 seeso module 패키지[20]를 사용하였다. 현재 올라와 있는 버전은 개발용으로, 크롬 확장 프로그램의 Manifest V3 보안 설정을 만족하기 위해서는 수정이 필요하다.

1) Seeso module

그림 3은 VisualCamp사에서 Chrome Extensions 용도로 제공받은 Seeso 모듈의 구조도이다. Seeso module은 WASM(WebAssembly) 기반으로 동작하며 실제 시선 연산을 처리하는 부분과, 해당 내용을 실제 적용 코드에 활용하기 편하게 설계된 부분이 존재한다. ‘import’ 및 ‘export’ 키워드를 사용해 시선 추적 모듈을 외부에 제공하고, 실제 적용 환경의 코드에서 import 문으로 활용할 수 있도록 구성되어 있다.

Seeso-web-sample[20]에서도 확인할 수 있다시피, HTML 문서에서 easy-seeso.js를 이용하기 위해서는 반드시

시 <script type=“module”>을 사용하여 모듈을 로드해야 한다. 이는 자바스크립트 사양 중 ES(ECMAScript)모듈의 구조를 따르기 때문이다. 이는 외부 스크립트 로딩 시 CORS 정책의 영향을 받기 때문에[21] 따라서 적용할 HTML 문서에 앞서 서술했던 Cross-Origin-Isolation 보안 헤더를 설정해야 프로그램이 제대로 동작할 수 있다.

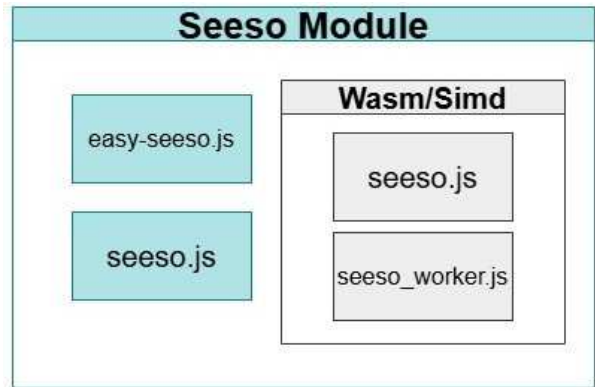


그림 3. Seeso Module의 구성도
Fig. 3. Structure of the Seeso Module

2) WASM 로드 방식 수정

기존 Seeso 모듈은 WASM 연산을 외부 CDN 링크를 통해 로드하는 방식이었다. Seeso 객체를 생성하는 데 사용되는 워커는 Blob[22] 객체를 통해 WASM 파일을 참조했으나, createObjectURL()[23]은 확장 프로그램 보안 정책에 따라 차단된다. 따라서 chrome.runtime.getURL()을 사용하여 확장 프로그램 내부 자원으로 직접 접근하도록 수정하였다[24].

3) Seesoframe 구성 시 보안 전제조건

각 탭에서 동작하는 시선 추적 모듈은 iframe 내부에 시선 추적을 연산하는 스크립트를 포함한 형태로 대상 페이지에 적용된다. 이렇게 iframe 형태로 불러온 시선 추적 모듈을 Seesoframe이라고 부르겠다.

Seeso 모듈은 iframe 기반으로 구성되며, iframe은 부모 문서인 확장 프로그램의 content script의 CSP 및 보안 정책을 따르게 된다. 원본 모듈은 Webpack 빌드 시 ‘devtool: “none”’ 설정 없이 eval() 함수를 포함하고 있어 Manifest V3 환경에서는 로드 자체가 불가능하다[25]. eval 사용은 sandbox iframe을 통해 우회 가능하지만[26], sandbox 속성이 적용된 iframe은 Cross-Origin-Isolation 설정이 불가능하며[27], 이에 따라 SharedArrayBuffer를 사용하는 WebAssembly 연산이 제한된다[28],[29].

따라서 background.js에서 확장 프로그램 설치 시 ‘chrome.runtime.onInstalled’ 이벤트를 활용해 ‘chrome.declarativeNetRequest’ API로 보안 헤더를 동적으로 설정해야 한다[30]. 여기서 updateDynamicRules 메시지를 사용해 보안 헤더를 설정하는데, responseHeaders 인자에 적

용한 각 헤더는 표 4와 같다.

표 4. 보안을 위한 HTTP 헤더 설정

Table 4. HTTP header configuration for cross-origin security

header	operation	value
"Cross-Origin-Embedder-Policy"	set	require-corp
"Cross-Origin-Opener-Policy"	set	same-origin
"Cross-Origin-Resource-Policy"	set	same-origin

4) SeesoIframe 실행

시선 추적 기능을 탭에 삽입하기 위해, chrome.tabs.query ({ active: true, currentWindow: true }, callback)을 이용해 사용자가 현재 보고 있는 활성 탭을 확인한 후, chrome.scripting.executeScript를 통해 해당 탭의 DOM에 SeesoIframe을 생성한다. 시선 추적 모듈은 iframe을 통해 활성 탭에만 적용되며, iframe 생성 시 필요한 속성은 표 5에 정리하였다.

표 5. iframe 생성 시 설정되는 속성 값

Table 5. Attribute values set when creating an iframe

header	value
style.display	"none"
src	chrome.runtime.getURL("/src/seeso/seeso.html")
allow	"camera; cross-origin-isolated"

시선 추적 기능만 수행할 것이므로 해당 iframe은 display: none 속성으로 설정하며, Cross-Origin-Isolation 보안 환경과 카메라 접근이 가능하도록 관련 속성들을 함께 적용해야 한다.

iframe의 src는 seeso.html로 지정되며, 해당 HTML은 ES6 모듈로 작성된 시선 추적 코드를 불러오므로 import가 가능한 구조이다. 이때 경로 설정은 chrome.runtime.getURL ()을 이용하여 확장 프로그램 내부 리소스를 참조하고, 해당 HTML 파일은 manifest.json의 web_accessible_resources 항목에 등록되어야 모든 탭에서 접근 가능하다.

이렇게 생성된 iframe 객체는 전역 변수인 window.seesoiframe에 저장되며, 추후 시선 추적 종료 시 해당 객체를 참조하여 제거할 수 있도록 설계되었다. 이를 통해 iframe의 상태를 통합적으로 관리할 수 있다.

5) 추가 필요 조건 및 고려사항

설정을 모두 마친 후에는 Seeso API의 인증 키 설정이 필요하다. Seeso API는 URL 주소 기반 인증 방식을 사용하므로, 키 고정 과정이 요구된다. 이를 위해서는 먼저[31] Chrome Web Store에 개발자로 등록한 뒤, 새로운 확장 프로그램을 생성해야 한다. 프로그램이 생성되면 해당 URL에 대해 고정된 API 키가 발급되며, 이 키를 manifest.json에 적용하여 테스트 환경에서도 정상적으로 시선 추적 기능을 사용할 수 있다.

단, 해당 확장 프로그램을 Chrome Web Store에 공식적

으로 배포하는 경우에는 manifest.json에서 별도의 키 설정 없이도 정상적으로 작동하며, JSON 내에 키를 명시할 필요는 없다.

2-3 시선 데이터 전달

1) 크롬에서 지원하는 데이터 통신 방법

크롬 확장 프로그램은 Background, Popup, Content Script의 서로 다른 실행 환경간 직접적으로 같은 메모리 공간을 공유하지 않는다. 따라서 데이터를 주고받기 위해 표 6에서 제시하고 있는 통신 방법을 사용해야 한다[32]. 그림 4에서는 구성 요소간 데이터 통신 방법을 도식화하여 제시하였다.

표 6. 크롬 확장 프로그램에서의 통신 방법

Table 6. Communication methods in Chrome extensions

Method	Description	Use Case
Message Passing	Asynchronous message exchange using chrome.runtime.sendMessage, chrome.tabs.sendMessage and listeners.	Event notification and data transfer between background, popup, and content scripts.
Storage API	chrome.storage API allows data sharing across extension contexts; localStorage is isolated per context.	Persisting state in background and accessing it from popup or content scripts.
Long-lived Connections (Port API)	Persistent two-way communication channel using chrome.runtime.connect and port.postMessage.	Real-time data streaming, e.g., continuous gaze data transfer.

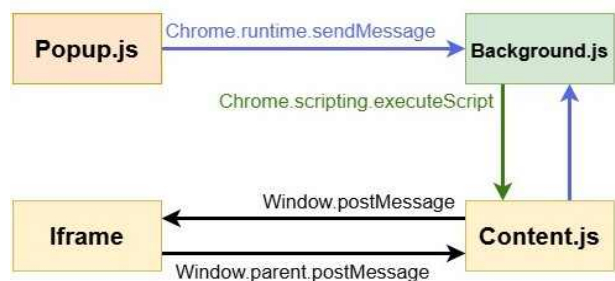


그림 4. EyeTracker Extension에 적용되는 통신 방법 구조도

Fig. 4. Communication architecture applied to the EyeTracker Extension

비유하자면 content script는 현재 탭을 자기 자신처럼 다루기 때문에 DOM 접근이 자유로운 상태라고 할 수 있다. 그리고 시선 모듈이 포함된 iframe은 현재 탭 DOM 내부에 존재하는 요소이다. 그러므로 content script와 iframe 사이에는 자식 관계가 설립한다고 볼 수 있다. 이 경우 표준 웹 API인 window.postMessage()를 통해 직접 양방향 통신이 가능하다.

iframe에서 부모 창인 content script로 메시지를 보내기

위해서는 window.parent.postMessage()를 사용하면 된다.

content script와의 통신은 편리하지만, 페이지 수명에 종속적인 특성 때문에 데이터 손실 위험이 크다. 유지보수와 안정적인 데이터 저장을 위해서는 background script에서 데이터 저장을 처리하는 것이 유리하다.

iframe에서 background로 데이터를 전송하기 위해 chrome.runtime.sendMessage를 사용한다.

2) 데이터 저장 방법

chrome.storage.local[33]은 웹 페이지의 로컬 저장소인 localStorage[34]와는 다르다. 표 7에 관련 내용이 제시되어 있다. localStorage는 background, popup, content script 각 도메인마다 별도의 저장소를 가지고 있어 데이터 공유가 불가능하고, chrome.storage.local은 확장 프로그램의 모든 요소가 데이터를 공유할 수 있도록 설계되었다. 즉 확장 프로그램 전체에서 5MB의 공유 메모리를 가지는 게 된다.

표 7. 크롬 확장에서 사용되는 저장소 종류와 용량 제한
Table 7. Storage types and capacity limits in Chrome extensions

Storage Type	Capacity Limit
chrome.storage.local	5MB per extension for the entire domain
IndexedDB	No fixed limit (depends on the user's disk space and browser implementation)

시선 데이터 특성상 IndexedDB[35]를 이용해 저장하는 것이 적합하다고 판단된다. 키-값 쌍으로 데이터를 저장하며, 데이터의 효율적 관리 및 저장에 특화되어 있다. iframe 내부의 시선 콜백 함수에 indexedDB 저장 로직을 추가하여 구현해본 결과, 시선 추적이 중지되어 iframe이 제거되었을 때는 content script에서 indexedDB를 감지할 수 없게 된다. 시선 추적이 활성화되어 iframe이 부착되었을 때만 indexedDB를 확인할 수 있다. 중지와 시작을 반복해도 이전의 데이터가 손실되지 않고 누적되어 집계되는 것을 확인하였다.

2-4 성능 측정

1) 통신 방법 비교

시선 데이터 저장을 위한 통신 구조로 두 가지 방법을 비교하였다. 그림 5를 보면, A 방법은 iframe 내부에서 indexedDB를 직접 열어 저장하는 방식이고, B 방법은 시선 데이터가 연산될 때마다 Iframe에서 background로 sendMessage를 통해 전송 후 처리하는 방식이다. 이 두 방식에 대한 실제 실험 데이터를 그림 6에서 제시하고 있다. A 데이터는 초반에는 10~20회 정도가 측정되다가, 시간이 경과하면 초당 1초 정도의 데이터를 전달하는 것을 볼 수 있다. B 데이터는 꾸준히 20~30회 정도가 측정되는 것을 볼 수 있다. 이는 시선 추적 모듈의 성능 저하가 아닌, 콜백 함수의 실행에서 병목 현상이 발생한 것으로 볼 수 있다.

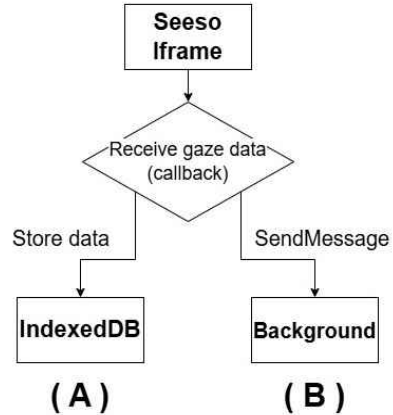


그림 5. 실험 방법 구조도
Fig. 5. Experimental procedure architecture

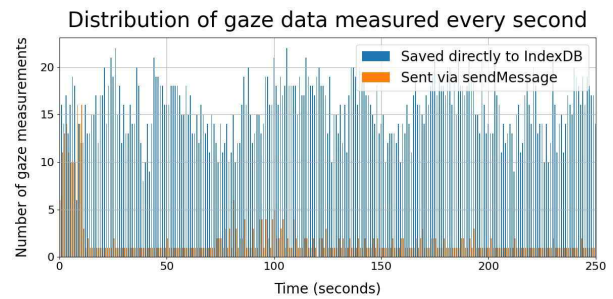


그림 6. 계속된 시선 데이터를 1초당 횟수로 나타낸 그래프
Fig. 6. Graph of measured gaze data in counts per second

시선 좌표가 연산될 때마다 사용자가 등록한 콜백 함수가 실행되며, 이 함수가 초당 처리 가능한 횟수를 초과할 경우, 실행 명령이 누적되어 메모리 병목 현상이 발생할 수 있다. 따라서 콜백 함수는 최대한 간단한 저장 전용 함수로 구성하며, 실행 효율을 높이는 최적화가 필요하다.

2) 저장 방법론

시선 좌표를 indexedDB에 저장하는 과정에서도 일정 수준의 지연이 발생한다. 연산 시점과 저장 시점의 timestamp 차이를 측정한 결과, 1888개의 데이터에서의 평균 저장 지연 시간은 약 34.64ms 정도로 측정되었다. 그림 7은 이 지연시간을 시각화한 것이다.

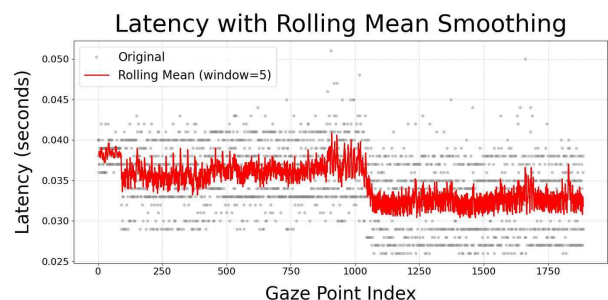


그림 7. IndexedDB에 저장하는 데 걸리는 지연시간 그래프
Fig. 7. Graph of delay time for storing data in IndexedDB

2-5 DOM 접근

일반적인 웹 크롤러를 통해 페이지의 DOM 구조를 추출하면 클래스명이 난독화되어 정보를 얻기 쉽지 않다. 크롬 확장 프로그램을 이용하면 각 페이지의 내부 DOM에 직접 접근할 수 있고, 상호작용도 가능하다. 일치하는 시선 요소를 찾는다는 과정에 있어서도, 동적으로 반응하는 웹 페이지의 요소 하나가 바뀔 때 마다 전체 DOM을 크롤링 한 후 시선 좌표와 비교 연산을 수행하며 각 요소를 검사하는 것보다, 크롬 확장 프로그램에서 내부적으로 검사를 실행한 후 필요한 정보만을 추출하는 것이 더 효율적이다.

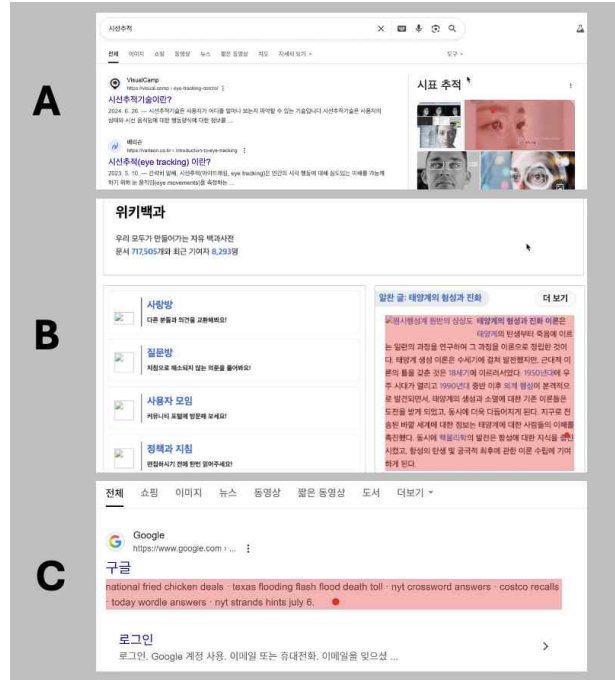
DOM 접근 과정의 실행 화면은 그림 8에 제시하였다. 사례 A와 C는 각각 ‘시선 추적’ 및 ‘구글’을 검색하였을 때의 화면이며, 사례 B는 위키백과(<https://ko.wikipedia.org/wiki/>)에 접속한 결과이다. 사례 B에서는 이미지가 정상적으로 로드되지 않는 현상이 관찰되었다. 이는 시선 추적 모듈 내에서 SharedArrayBuffer 활용을 위해 적용된 Cross-Origin-Isolation 헤더 설정의 영향으로 해석된다. 구글 검색과 같이 동일한 도메인 내에서 제공되는 서비스의 경우에는 정상적으로 동작하였으나, 서로 다른 origin에서 불러오는 외부 콘텐츠가 존재하는 웹사이트에서는 일부 자원이 정상적으로 로드되지 않는 현상이 확인되었다. 특히, 페이지 특성에 따라 CSS 스타일이 손상되어 화면 구성이 왜곡되는 경우도 발견되었다.

1) elementFromPoint() 메서드의 사용

Content에서 접근하며, 본 실험에서는 대부분의 브라우저에서 사용 가능한 elementFromPoint()[36] 메서드를 사용하여 DOM 요소를 계산하였다. 해당 메서드는 document object에서 사용 가능하고, 그림 8의 C 예시처럼 지정된 좌표 점에 의거해 가장 위에 있는 Element를 반환한다. Content에서 대상 웹사이트의 DOM 트리를 받아오고 필터링해 background에서 localStorage에 저장한 뒤, Content에서 현재 시선 좌표와 연산하는 과정을 직접 구현할 수도 있으나, localStorage의 용량이 5MB 정도라는 점과 이 연산 과정에서 속도 저하가 심하게 일어났기 때문에 DOM의 추출과 저장은 IndexedDB를 이용하고, 현재 시선과 일치하는지의 연산은 elementFromPoint()를 사용해 필요한 정보만 수집하는 것이 바람직하다.

2) 사용 시 호출 주기

하지만 이 elementFromPoint()의 호출 트리거 주기는 고려해야 할 사항이다. 앞서 그림 6에서 보였던 것처럼 callback 함수에 데이터 저장, 메시지 송신 등의 과정을 추가하면 속도 저하가 일어나게 된다. 그림 9는 각각 시행한 TPS 결과를 시각화 한 것이다. 동일한 환경 안에 있는 content로의 postMessage 및 console.log로 확인했을 때의 결과는 초당 약 30회 정도가 출력되었다. content에서 elementFromPoint 연산까지 추가했을 때는 초당 1개정도로 출력된다.



*The screenshot contains Korean text because the execution environment was set to the Korean language.

그림 8. 동작 예시 화면 이미지

Fig. 8. Screenshot of operation screen

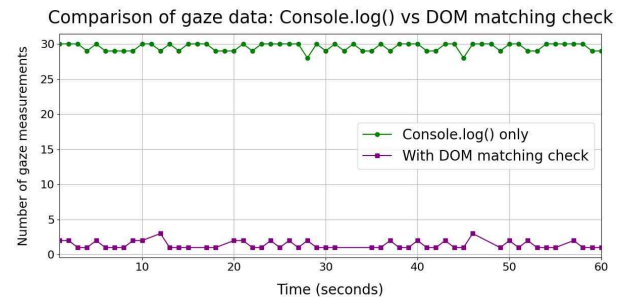


그림 9. console.log와 DOM 매칭 체크 간 시선 데이터 처리 속도 비교

Fig. 9. Comparison of gaze data processing speed: console.log vs. DOM matching check

III. 결론

시선 추적은 하나의 독립된 시스템으로 존재하기보단, 시선 추적을 적용하려는 대상에 융합하는 형식으로 발전되어 왔다. 이를 웹 상에서 다양한 웹페이지 대상으로 실행할 수 있도록 EyeTracker Extension, 크롬 확장 프로그램을 개발하였다.

크롬 확장 프로그램에서 시선 추적 기능이 구동되도록 하기 위해 현재 보안 정책을 알맞은 프로그램 환경을 구성하였다. 1초에 20~30번 이상 빠르게 연산 되어야 하는 시선 데이터 특성상 계산된 데이터를 손실 없이 효율적으로 저장해야 한다. 확장 프로그램 내부에 저장소에 데이터를 모아놓으면 메모리 소모 문제가 있기 때문에, IndexedDB에 저장하는 방식을 고안해 속도 저하 없이 시선 데이터를 보존할 수 있게

하였다. 시선 데이터의 연산과 저장 과정에서 최대한 시간이 소요되지 않는 단순 저장 함수만을 등록해서 이 실행 과정을 최적화할 필요가 있다는 것을 밝혀내었다.

다만 본 논문에서 실험한 속도 측정에는 하드웨어의 성능과 다른 영향이 있었을 가능성이 있기 때문에 향후 이런 요소도 고려해 정확한 성능을 측정해야 한다.

현재 시선과 일치하는, 사용자가 바라보고 있는 DOM 요소가 특정되면 이를 LLM과 연결하여 웹을 이용하는 사용자의 의도와 관심을 자연어로 나타낼 수 있을 것이다. 이는 사용자의 의도와 주의, 관심사 분석에 도움이 될 수 있는 데이터를 제공할 것이다.

다만 시선 데이터와 시스템 구조의 특성 상 효율적인 데이터 저장 로직에 대한 연구가 필요할 것으로 보인다. 시선 데이터만을 일차적으로 필터링하여 fixation만을 구한 후 해당 범위에 있는 dom 요소 데이터만을 저장하는 등의 방식을 구현해야 할 것이다.

감사의 글

본 연구는 덕성여자대학교 2024년도 교내연구비 지원에 의해 수행되었음.

참고문헌

- [1] K. Krafska, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba, "Eye Tracking for Everyone," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2176-2184, June 2016.
- [2] S. Kim, J. Jang, and T. Park, "User Gaze Data Analysis During Mobile Commerce Using Artificial Intelligence-Based Gaze Estimation," *Journal of Digital Contents Society*, Vol. 24, No. 5, pp. 1099-1110, 2023. <https://doi.org/10.9728/dcs.2023.24.5.1099>
- [3] S. Kwak, J. Kim, and T. Park, "AI-Based Video Qualification Using the User's Gaze and Emotion," *Journal of Digital Contents Society*, Vol. 24, No. 3, pp. 463-472, 2023. <https://doi.org/10.9728/dcs.2023.24.3.463>
- [4] Y. Cheng and F. Lu, "DVGaze: Dual-View Gaze Estimation," in *Proceedings of the 2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, Paris: France, pp. 20575-20584, October 2023. <https://doi.org/10.1109/ICCV51070.2023.01886>
- [5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, ... and I. Polosukhin, "Attention Is All You Need," arXiv:1706.03762, 2017. <http://arxiv.org/abs/1706.03762>
- [6] T. Park, "Technique to Estimate Geometry Information from Gaze Detection Datasets Based on Multiple Cameras," *Journal of Digital Contents Society*, Vol. 24, No. 12, pp. 3071-3080, 2023. <https://doi.org/10.9728/dcs.2023.24.12.3071>
- [7] D. Lian, Z. Zhang, W. Luo, L. Hu, M. Wu, Z. Li, ... and S. Gao, "RGBD Based Gaze Estimation via Multi-Task CNN," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, No. 1, pp. 2488-2495, July 2019, <https://doi.org/10.1609/aaai.v33i01.33012488>
- [8] A. Ruzzi, X. Shi, X. Wang, G. Li, S. De Mello, H. J. Chang, ... and O. Hilliges, "GazeNeRF: 3D-Aware Gaze Redirection with Neural Radiance Fields," arXiv:2212.04823, 2023. <http://arxiv.org/abs/2212.04823>
- [9] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," *Communications of the ACM*, Vol. 65, No. 1, pp. 99-106, December 2021. <https://doi.org/10.1145/3503250>
- [10] S. Hong, Y. Kim, and T. Park, "Blinks in the Dark': Blink Estimation with Domain Adversarial Training (BEAT) Network," *IEEE Transactions on Consumer Electronics*, Vol. 69, No. 3, pp. 581-593, August 2023. <https://doi.org/10.1109/TCE.2023.3275540>
- [11] S. H. Choi, D. Son, Y. Ha, Y. Kim, S. Hong, and T. Park, "Looking to Personalize Gaze Estimation Using Transformers," *Journal of Computing Science and Engineering*, Vol. 17, No. 2, pp. 41-50, 2023. <https://doi.org/10.5626/JCSE.2023.17.2.41>
- [12] G. Yuan, Y. Wang, H. Yan, and X. Fu, "Self-Calibrated Driver Gaze Estimation via Gaze Pattern Learning," *Knowledge-Based Systems*, Vol. 235, 107630, January 2022, <https://doi.org/10.1016/j.knsys.2021.107630>
- [13] Y. Wang, G. Yuan, and X. Fu, "Towards Implicit Personal Eye Gaze Calibration in Real-world Driving Scenarios," in *Proceedings of the 2025 Symposium on Eye Tracking Research and Applications*, Tokyo: Japan, pp. 1-2, May 2025. <https://doi.org/10.1145/3715669.3726793>
- [14] Y. Cheng, Y. Zhu, Z. Wang, H. Hao, Y. Liu, S. Cheng, ... and H. J. Chang, "What Do You See in Vehicle? Comprehensive Vision Solution for In-Vehicle Gaze Estimation," in *Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle: WA, pp. 1556-1565, June 2024. <https://doi.org/10.1109/CVPR52733.2024.00154>
- [15] Y. Lei, Y. Wang, T. Caslin, A. Wisowaty, X. Zhu, M. Khamis, and J. Ye, "DynamicRead: Exploring Robust Gaze Interaction Methods for Reading on Handheld

Mobile Devices under Dynamic Conditions,” *Proceedings of the ACM on Human-Computer Interaction*, New York: NY, pp. 1-17, May 2023. <https://doi.org/10.1145/3591127>

[16] Contagious. Cannes Lions: Creative Commerce Winners 2023 [Internet]. Available: <https://www.contagious.com/news-and-views/cannes-lions-creative-commerce-winners-2023>.

[17] Netflix. The Influencer Official Site [Internet]. Available: <https://www.netflix.com/title/81729971>.

[18] Chrome Developers. Extensions Reference - Manifest V3 [Internet]. Available: <https://developer.chrome.com/docs/extensions/reference?hl=ko>.

[19] Chrome Developers. Architecture Overview - Manifest V2 [Internet]. Available: <https://developer.chrome.com/docs/extensions/mv2/architecture-overview?hl=ko>.

[20] VisualCamp. Seeso-Sample-Web [Internet]. Available: <https://github.com/visualcamp/seeso-sample-web>.

[21] MDN Web Docs. CORS (Cross-Origin Resource Sharing) [Internet]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>.

[22] Mozilla Developer Network. Blob - Web APIs [Internet]. Available: <https://developer.mozilla.org/en-US/docs/Web/API/Blob>.

[23] Mozilla Developer Network. URL.createObjectURL() - Web APIs [Internet]. Available: https://developer.mozilla.org/en-US/docs/Web/API/URL/createObjectURL_static.

[24] Chrome Developers. Chrome.runtime.getURL [Internet]. Available: <https://developer.chrome.com/docs/extensions/reference/runtime/#method-getURL>.

[25] Chrome Developers. Content Security Policy [Internet]. Available: <https://developer.chrome.com/docs/privacy-security/csp?hl=ko>.

[26] Chrome Developers. Sandboxing and `eval()` - Chrome Extensions [Internet]. Available: <https://developer.chrome.com/docs/extensions/how-to/security/sandboxing-eval?hl=ko>.

[27] Stack Overflow. SharedArrayBuffer in an Iframe [Internet]. Available: <https://stackoverflow.com/questions/69652019/sharedarraybuffer-in-an-iframe>.

[28] Mozilla Developer Network. SharedArrayBuffer [Internet]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/SharedArrayBuffer.

[29] Web.dev. Cross-Origin Isolation Guide [Internet]. Available: <https://web.dev/articles/cross-origin-isolation-guide>.

[30] Chrome Developers. Chrome.DeclarativeNetRequest API [Internet]. Available: <https://developer.chrome.com/docs/extensions/reference/declarativeNetRequest>.

[31] Chrome Developers. Publish in Chrome Web Store [Internet]. Available: <https://developer.chrome.com/docs/webstore/publish/>.

[32] Chrome Developers. Messaging [Internet]. Available: <https://developer.chrome.com/docs/extensions/mv3/messaging/>.

[33] Chrome Developers. Chrome.Storage [Internet]. Available: <https://developer.chrome.com/docs/extensions/reference/storage/>.

[34] Mozilla Developer Network. Window: LocalStorage Property [Internet]. Available: <https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage>.

[35] Mozilla Developer Network. IndexedDB API [Internet]. Available: https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API.

[36] Mozilla Developer Network. Document: ElementFromPoint [Internet]. Available: <https://developer.mozilla.org/en-US/docs/Web/API/Document/elementFromPoint>.

최유경 (Yugyeong Choi)



2023년 ~ 현재: 덕성여자대학교 디지털소프트웨어공학부 학사 과정

※ 관심분야: 인공지능 기반 시선 추적

박태정 (Taejung Park)



1997년: 서울대 전기공학부 (공학사)

1999년: 서울대 전기공학부 대학원 (공학 석사, 반도체 물리 전공)

2006년: 서울대 전기컴퓨터공학부 대학원 (공학박사, 컴퓨터 그래픽스 전공)

2006년 ~ 2013년: 고려대학교 연구교수

2013년 ~ 2017년: 덕성여자대학교 정보미디어대학 디지털미디어학과 조교수

2018년 ~ 2024년: 덕성여자대학교 공과대학 사이버보안/디지털소프트웨어공학부 부교수

2024년 ~ 현재: 덕성여자대학교 공과대학 디지털소프트웨어공학부 교수

※ 관심분야: 컴퓨터그래픽스, 인공지능, 수치해석, 3차원 모델링