Extended Mathematical Framework for Browser Position Calculation: Coordinate Transformations, Performance Optimisation, and Calibration

Michael R. Malloy

May 18, 2025

Abstract

This paper establishes a comprehensive mathematical framework for calculating positions in browser environments with varying screen sizes, browser windows, and display scaling configurations. Building upon the foundation of coordinate system transformations, we extend the framework to include performance optimisation techniques and calibration methodologies. We rigorously define the mathematical properties of transformation caching and adaptive selection and provide formal proofs of their correctness. In addition, we derive a robust calibration framework that enables accurate parameter estimation and error correction. This expanded framework offers a unified approach for coordinate translation that is mathematically sound and computationally efficient, allowing developers to create responsive and adaptive applications across diverse display environments.

1 Introduction

Modern web applications operate in diverse display environments with varying screen dimensions, window sizes, and scaling factors. Translating coordinates precisely between different reference frames presents significant mathematical challenges, particularly when performance and accuracy are critical requirements.

This paper builds upon the established foundation of coordinate transformation systems, extending it in two significant directions.

1. **Performance Optimisation**: We formalise the mathematical properties of transformation caching and adaptive strategy selection, proving their correctness and analysing their computational complexity.

2. **Calibration Mathematics**: We develop a robust mathematical framework for parameter estimation and error correction, enabling accurate transformation in real-world scenarios with imperfect data.

Our framework combines theoretical rigour with practical utility, providing a complete mathematical foundation for coordinate transformations that is both mathematically sound and computationally efficient.

2 Foundational Coordinate Systems

We begin by recapitulating the five fundamental coordinate systems that form the basis of our framework. These definitions establish the mathematical spaces in which our transformations operate.

Definition 2.1 (Normalised Coordinate System)**.** The normalised coordinate system $\mathbb{N}$ represents positions as ratios of screen dimensions, independent of the actual pixel counts.

- Domain: $\mathbb{N} \subset [0,1]^2$
- For a point $\boldsymbol{p}_n \in \mathbb{N}$, we denote $\boldsymbol{p}_n = (x_n, y_n)$ where $0 \leq x_n, y_n \leq 1$

Definition 2.2 (Screen Coordinate Systems)**.** We define two screen coordinate systems:

- $\mathbb{S}_1$: Screen coordinates in the original display configuration
- $\mathbb{S}_2$: Screen coordinates in the target display configuration

Both are physical pixel coordinate systems with the origin at the top-left corner of their respective screens.

- Domain: $\mathbb{S}_i \subset \mathbb{Z}^2$ where $i \in \{1, 2\}$
- For a point $\boldsymbol{p}_{s_i} \in \mathbb{S}_i$, we denote $\boldsymbol{p}_{s_i} = (x_{s_i}, y_{s_i})$

Definition 2.3 (Browser Coordinate System)**.** The browser coordinate system $\mathbb{B}$ is the physical pixel coordinate system with origin in the upper left corner of the browser window.

- Domain: $\mathbb{B} \subset \mathbb{Z}^2$

- For a point $\boldsymbol{p}_b \in \mathbb{B}$, we denote $\boldsymbol{p}_b = (x_b, y_b)$

Definition 2.4 (Logical Coordinate System)**.** The logical coordinate system $\mathbb{L}$ is used by the browser internally after scaling the DPI.

- Domain: $\mathbb{L} \subset \mathbb{Q}^2$ where $\mathbb{Q}$ is the set of rational numbers
- For a point $\boldsymbol{p}_l \in \mathbb{L}$, we denote $\boldsymbol{p}_l = (x_l, y_l)$

3 Key Parameters

The behaviour of our coordinate transformations depends on several key parameters that characterise the display configurations.

Definition 3.1 (Display Configuration Parameters)**.**

- $\boldsymbol{s}_1 = (s_{w1}, s_{h1}) \in \mathbb{Z}_+^2$: Original screen dimensions in physical pixels
- $\boldsymbol{s}_2 = (s_{w2}, s_{h2}) \in \mathbb{Z}_+^2$: Target screen dimensions in physical pixels
- $\boldsymbol{b}_1 = (b_{x1}, b_{y1}) \in \mathbb{S}_1$: Browser window position in original screen coordinates
- $\boldsymbol{b}_2 = (b_{x2}, b_{y2}) \in \mathbb{S}_2$: Browser window position in target screen coordinates
- $\boldsymbol{v}_1 = (v_{w1}, v_{h1}) \in \mathbb{Z}_+^2$: Viewport dimensions in logical pixels (original configuration)
- $\boldsymbol{v}_2 = (v_{w2}, v_{h2}) \in \mathbb{Z}_+^2$: Viewport dimensions in logical pixels (target configuration)
- $\sigma_1 \in \mathbb{R}^+$: Original DPI scaling factor
- $\sigma_2 \in \mathbb{R}^+$: Target DPI scaling factor

4 Core Transformation Operators

We now define the core transformation operators between our coordinate systems, providing their canonical forms and proving their properties.

4.1 Screen Size Transformation

Definition 4.1 (Screen-to-Normalised Transformation)**.** The screen-to-normalised transformation $T_{S_i \to N} : \mathbb{S}_i \to \mathbb{N}$ maps a point from the screen coordinates to the normalised coordinates:

$$T_{S_i \to N}(\boldsymbol{p}_{s_i}) = \left(\frac{x_{s_i}}{s_{wi}}, \frac{y_{s_i}}{s_{hi}} \right) = \boldsymbol{p}_n \tag{1}$$

where $i \in \{1, 2\}$ indicates the screen configuration.

Definition 4.2 (Normalised-to-Screen Transformation)**.** The normalised-to-screen transformation $T_{N \to S_i} : \mathbb{N} \to \mathbb{S}_i$ maps a point from normalised coordinates to screen coordinates:

$$T_{N \to S_i}(\boldsymbol{p}_n) = (x_n \cdot s_{wi}, y_n \cdot s_{hi}) = \boldsymbol{p}_{s_i} \tag{2}$$

where $i \in \{1, 2\}$ indicates the screen configuration.

Theorem 4.3. *The transformations $T_{S_i \to N}$ and $T_{N \to S_i}$ are linear transformations.*

Proof. We prove this for $T_{S_i \to N}$. For any $\boldsymbol{p}_{s_i}, \boldsymbol{q}_{s_i} \in \mathbb{S}_i$ and scalar $\alpha \in \mathbb{R}$:

1. $T_{S_i \to N}(\boldsymbol{p}_{s_i} + \boldsymbol{q}_{s_i}) = \left(\frac{x_{ps_i} + x_{qs_i}}{s_{wi}}, \frac{y_{ps_i} + y_{qs_i}}{s_{hi}} \right) = \left(\frac{x_{ps_i}}{s_{wi}} + \frac{x_{qs_i}}{s_{wi}}, \frac{y_{ps_i}}{s_{hi}} + \frac{y_{qs_i}}{s_{hi}} \right) = T_{S_i \to N}(\boldsymbol{p}_{s_i}) + T_{S_i \to N}(\boldsymbol{q}_{s_i})$
2. $T_{S_i \to N}(\alpha \boldsymbol{p}_{s_i}) = \left(\frac{\alpha \cdot x_{ps_i}}{s_{wi}}, \frac{\alpha \cdot y_{ps_i}}{s_{hi}} \right) = \alpha \cdot \left(\frac{x_{ps_i}}{s_{wi}}, \frac{y_{ps_i}}{s_{hi}} \right) = \alpha \cdot T_{S_i \to N}(\boldsymbol{p}_{s_i})$

Since both properties are satisfied, $T_{S_i \to N}$ is a linear transformation. The proof for $T_{N \to S_i}$ follows in a similar way.

The canonical matrix representation of $T_{S_i \to N}$ is:

$$T_{S_i \to N}(\boldsymbol{p}_{s_i}) = \begin{pmatrix} \frac{1}{s_{wi}} & 0 \\ 0 & \frac{1}{s_{hi}} \end{pmatrix} \begin{pmatrix} x_{s_i} \\ y_{s_i} \end{pmatrix} \tag{3}$$

And for $T_{N \to S_i}$:

$$T_{N \to S_i}(\boldsymbol{p}_n) = \begin{pmatrix} s_{wi} & 0 \\ 0 & s_{hi} \end{pmatrix} \begin{pmatrix} x_n \\ y_n \end{pmatrix} \tag{4}$$

□

Definition 4.4 (Direct Screen-to-Screen Transformation)**.** The direct transformation $T_{S_1 \to S_2} : \mathbb{S}_1 \to \mathbb{S}_2$ maps a point from the original screen coordinates to the target screen coordinates:

$$T_{S_1 \to S_2} = T_{N \to S_2} \circ T_{S_1 \to N} \tag{5}$$

For any point $\boldsymbol{p}_{s_1} \in \mathbb{S}_1$:

$$T_{S_1 \to S_2}(\boldsymbol{p}_{s_1}) = T_{N \to S_2}(T_{S_1 \to N}(\boldsymbol{p}_{s_1})) \tag{6}$$

$$= T_{N \to S_2}\left(\frac{x_{s_1}}{s_{w1}}, \frac{y_{s_1}}{s_{h1}}\right) \tag{7}$$

$$= \left(\frac{x_{s_1}}{s_{w1}} \cdot s_{w2}, \frac{y_{s_1}}{s_{h1}} \cdot s_{h2}\right) \tag{8}$$

$$= \left(x_{s_1} \cdot \frac{s_{w2}}{s_{w1}}, y_{s_1} \cdot \frac{s_{h2}}{s_{h1}}\right) \tag{9}$$

$$= (x_{s_1} \cdot \alpha_x, y_{s_1} \cdot \alpha_y) \tag{10}$$

Where $\alpha_x = \frac{s_{w2}}{s_{w1}}$ and $\alpha_y = \frac{s_{h2}}{s_{h1}}$ are the scaling factors for width and height, respectively.

Corollary 4.5. *The transformation $T_{S_1 \to S_2}$ is linear.*

Proof. As the composition of two linear transformations ($T_{S_1 \to N}$ and $T_{N \to S_2}$), $T_{S_1 \to S_2}$ is also a linear transformation.

The canonical matrix representation is:

$$T_{S_1 \to S_2}(\boldsymbol{p}_{s_1}) = \begin{pmatrix} \alpha_x & 0 \\ 0 & \alpha_y \end{pmatrix} \begin{pmatrix} x_{s_1} \\ y_{s_1} \end{pmatrix} \tag{11}$$

where $\alpha_x = \frac{s_{w2}}{s_{w1}}$ and $\alpha_y = \frac{s_{h2}}{s_{h1}}$. □

4.2 Browser Window Transformations

Next, we define the transformations related to the position of the browser window.

Definition 4.6 (Screen-to-Browser Transformation)**.** The screen-to-browser transformation $T_{S_i \to B_i} : \mathbb{S}_i \to \mathbb{B}_i$ maps a point from screen coordinates to browser coordinates in configuration i:

$$T_{S_i \to B_i}(\boldsymbol{p}_{s_i}) = \boldsymbol{p}_{s_i} - \boldsymbol{b}_i = (x_{s_i} - b_{xi}, y_{s_i} - b_{yi}) = \boldsymbol{p}_{b_i} \tag{12}$$

Theorem 4.7. *The transformation $T_{S_i \to B_i}$ is an affine transformation.*

Proof. This transformation represents a translation of the vector $-\boldsymbol{b}_i$. Although it does not preserve the origin and is therefore not linear, it is an affine transformation.

The canonical form of an affine transformation is $T(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{c}$, where A is a linear transformation and $\boldsymbol{c}$ is a constant vector. For $T_{S_i \to B_i}$, we have:

$$T_{S_i \to B_i}(\boldsymbol{p}_{s_i}) = I\boldsymbol{p}_{s_i} - \boldsymbol{b}_i \tag{13}$$

where I is the identity matrix and $\boldsymbol{c} = -\boldsymbol{b}_i$ is the translation vector. □

4.3 DPI Scaling Transformations

Now we define transformations related to DPI scaling.

Definition 4.8 (Browser-to-Logical Transformation)**.** The browser-to-logical transformation $T_{B_i \to L_i} : \mathbb{B}_i \to \mathbb{L}_i$ maps a point from the browser coordinates to the logical coordinates in configuration i:

$$T_{B_i \to L_i}(\boldsymbol{p}_{b_i}) = \frac{1}{\sigma_i}\boldsymbol{p}_{b_i} = \left(\frac{x_{b_i}}{\sigma_i}, \frac{y_{b_i}}{\sigma_i}\right) = \boldsymbol{p}_{l_i} \tag{14}$$

Theorem 4.9. *The transformation $T_{B_i \to L_i}$ is linear.*

Proof. This transformation represents a uniform scaling by factor $\frac{1}{\sigma_i}$. Preserves vector addition and scalar multiplication, making it a linear transformation.

The canonical matrix representation is:

$$T_{B_i \to L_i}(\boldsymbol{p}_{b_i}) = \begin{pmatrix} \frac{1}{\sigma_i} & 0 \\ 0 & \frac{1}{\sigma_i} \end{pmatrix} \begin{pmatrix} x_{b_i} \\ y_{b_i} \end{pmatrix} \tag{15}$$

□

4.4 Composite Transformations

We now define composite transformations that combine multiple individual transformations.

Definition 4.10 (Screen-to-Logical Transformation)**.** The screen-to-logical transformation $T_{S_i \to L_i} : \mathbb{S}_i \to \mathbb{L}_i$ maps a point from screen coordinates to logical coordinates in configuration i:

$$T_{S_i \to L_i} = T_{B_i \to L_i} \circ T_{S_i \to B_i} \tag{16}$$

For any point $\boldsymbol{p}_{s_i} \in \mathbb{S}_i$:

$$\begin{aligned} T_{S_i \to L_i}(\boldsymbol{p}_{s_i}) &= T_{B_i \to L_i}(T_{S_i \to B_i}(\boldsymbol{p}_{s_i})) && (17)\\ &= T_{B_i \to L_i}(\boldsymbol{p}_{s_i} - \boldsymbol{b}_i) && (18)\\ &= \frac{1}{\sigma_i}(\boldsymbol{p}_{s_i} - \boldsymbol{b}_i) && (19)\\ &= \frac{\boldsymbol{p}_{s_i} - \boldsymbol{b}_i}{\sigma_i} && (20)\\ &= \left(\frac{x_{s_i} - b_{xi}}{\sigma_i}, \frac{y_{s_i} - b_{yi}}{\sigma_i}\right) && (21) \end{aligned}$$

Theorem 4.11. *The transformation $T_{S_i \to L_i}$ is an affine transformation.*

Proof. As the composition of an affine transformation ($T_{S_i \to B_i}$) and a linear transformation ($T_{B_i \to L_i}$), $T_{S_i \to L_i}$ is an affine transformation.

The canonical form is:

$$T_{S_i \to L_i}(\boldsymbol{p}_{s_i}) = A\boldsymbol{p}_{s_i} + \boldsymbol{c} \tag{22}$$

where $A = \frac{1}{\sigma_i} I$ (a uniform scaling matrix) and $\boldsymbol{c} = -\frac{\boldsymbol{b}_i}{\sigma_i}$ (a translation vector). □

Definition 4.12 (Complete Transformation)**.** The complete transformation $T_{S_1 \to L_2} : \mathbb{S}_1 \to \mathbb{L}_2$ maps a point from the original screen coordinates to the logical coordinates of the target:

$$T_{S_1 \to L_2} = T_{S_2 \to L_2} \circ T_{S_1 \to S_2} \tag{23}$$

For any point $\boldsymbol{p}_{s_1} \in \mathbb{S}_1$:

$$\begin{aligned} T_{S_1 \to L_2}(\boldsymbol{p}_{s_1}) &= T_{S_2 \to L_2}(T_{S_1 \to S_2}(\boldsymbol{p}_{s_1})) && (24)\\ &= T_{S_2 \to L_2}(x_{s_1} \cdot \alpha_x, y_{s_1} \cdot \alpha_y) && (25)\\ &= \frac{(x_{s_1} \cdot \alpha_x, y_{s_1} \cdot \alpha_y) - \boldsymbol{b}_2}{\sigma_2} && (26)\\ &= \left(\frac{x_{s_1} \cdot \alpha_x - b_{x2}}{\sigma_2}, \frac{y_{s_1} \cdot \alpha_y - b_{y2}}{\sigma_2} \right) && (27) \end{aligned}$$

Theorem 4.13. *The transformation $T_{S_1 \to L_2}$ is an affine transformation.*

Proof. As the composition of a linear transformation ($T_{S_1 \to S_2}$) and an affine transformation ($T_{S_2 \to L_2}$), $T_{S_1 \to L_2}$ is an affine transformation.

The canonical form is:

$$T_{S_1 \to L_2}(\boldsymbol{p}_{s_1}) = A\boldsymbol{p}_{s_1} + \boldsymbol{c} \tag{28}$$

Where:

$$A = \begin{pmatrix} \frac{\alpha_x}{\sigma_2} & 0 \\ 0 & \frac{\alpha_y}{\sigma_2} \end{pmatrix} \tag{29}$$

and $\boldsymbol{c} = -\frac{\boldsymbol{b}_2}{\sigma_2}$. □

5 Invertibility of Transformations

An essential property of our transformations is their invertibility, which allows us to move between coordinate systems in both directions.

Theorem 5.1. *All defined transformations are invertible, with the following inverse transformations:*

$$T^{-1}_{S_i \to N} = T_{N \to S_i} \tag{30}$$

$$T^{-1}_{S_i \to B_i}(\boldsymbol{p}_{b_i}) = \boldsymbol{p}_{b_i} + \boldsymbol{b}_i \tag{31}$$

$$T^{-1}_{B_i \to L_i}(\boldsymbol{p}_{l_i}) = \sigma_i \cdot \boldsymbol{p}_{l_i} \tag{32}$$

$$T^{-1}_{S_i \to L_i}(\boldsymbol{p}_{l_i}) = \sigma_i \cdot \boldsymbol{p}_{l_i} + \boldsymbol{b}_i \tag{33}$$

$$T^{-1}_{S_1 \to S_2}(\boldsymbol{p}_{s_2}) = \left(\frac{x_{s_2}}{\alpha_x}, \frac{y_{s_2}}{\alpha_y} \right) \tag{34}$$

$$T^{-1}_{S_1 \to L_2}(\boldsymbol{p}_{l_2}) = \left(\frac{\sigma_2 \cdot x_{l_2} + b_{x2}}{\alpha_x}, \frac{\sigma_2 \cdot y_{l_2} + b_{y2}}{\alpha_y} \right) \tag{35}$$

Proof. We prove invertibility by showing that the composition of each transformation with its inverse yields the identity transformation.

For $T_{S_i \to N}$ and $T_{N \to S_i}$:

$$T_{N \to S_i}(T_{S_i \to N}(\boldsymbol{p}_{s_i})) = T_{N \to S_i} \left(\frac{x_{s_i}}{s_{wi}}, \frac{y_{s_i}}{s_{hi}} \right) \tag{36}$$

$$= \left(\frac{x_{s_i}}{s_{wi}} \cdot s_{wi}, \frac{y_{s_i}}{s_{hi}} \cdot s_{hi} \right) \tag{37}$$

$$= (x_{s_i}, y_{s_i}) \tag{38}$$

$$= \boldsymbol{p}_{s_i} \tag{39}$$

Similar proofs can be constructed for the other inverse transformations, showing that the composition with the original transformation yields the identity transformation in each case. □

6 Performance Optimisation Framework

We now extend our mathematical framework to incorporate performance optimisation techniques. These techniques preserve the mathematical correctness of our transformations while improving computational efficiency.

6.1 Transformation Caching

Caching is a technique that stores the results of expensive operations to avoid redundant computations when the same operation is requested multiple times. We formalise this concept for coordinate transformations.

Definition 6.1 (Cached Transformation)**.** Let $T : X \to Y$ be a transformation between coordinate spaces X and Y. A cached transformation $T_C : X \to Y$ associated with T is a transformation that uses a storage mechanism C to avoid redundant calculations:

$$T_C(\boldsymbol{p}) = \begin{cases} C(\boldsymbol{p}) & \text{if } \boldsymbol{p} \in \text{domain}(C) \\ T(\boldsymbol{p}) & \text{otherwise} \end{cases} \tag{40}$$

where $C : X' \to Y$ is a partial function with $X' \subset X$ representing the subset of points for which the results are cached.

After computing $T(\boldsymbol{p})$ for a previously uncached point $\boldsymbol{p}$, the result is stored in C, extending the domain of C to include $\boldsymbol{p}$:

$$C' = C \cup \{(\boldsymbol{p}, T(\boldsymbol{p}))\} \tag{41}$$

Theorem 6.2 (Correctness of Cached Transformation)**.** *A cached transformation T_C produces the same result as the original transformation T for all inputs:*

$$\forall \boldsymbol{p} \in X : T_C(\boldsymbol{p}) = T(\boldsymbol{p}) \tag{42}$$

Proof. We prove this by considering two cases: 1. If $\boldsymbol{p} \in \text{domain}(C)$, then $T_C(\boldsymbol{p}) = C(\boldsymbol{p})$. By the definition of the caching mechanism, $C(\boldsymbol{p}) = T(\boldsymbol{p})$.

2. If $\boldsymbol{p} \notin \text{domain}(C)$, then $T_C(\boldsymbol{p}) = T(\boldsymbol{p})$ directly.

Therefore, $T_C(\boldsymbol{p}) = T(\boldsymbol{p})$ for all $\boldsymbol{p} \in X$. □

Theorem 6.3 (Invertibility of Cached Transformations)**.** *If a transformation $T : X \to Y$ is invertible with inverse $T^{-1} : Y \to X$, then a cached version T_C of T is also invertible, and its inverse is a cached version of T^{-1}.*

Proof. Let $T_C^{-1} : Y \to X$ be a cached version of T^{-1}. We need to prove that $T_C^{-1}(T_C(\boldsymbol{p})) = \boldsymbol{p}$ for all $\boldsymbol{p} \in X$.

From Theorem 6.2, we know that $T_C(\boldsymbol{p}) = T(\boldsymbol{p})$ for all $\boldsymbol{p} \in X$. Similarly, $T_C^{-1}(\boldsymbol{q}) = T^{-1}(\boldsymbol{q})$ for all $\boldsymbol{q} \in Y$.

Therefore:

$$\begin{aligned} T_C^{-1}(T_C(\boldsymbol{p})) &= T_C^{-1}(T(\boldsymbol{p})) && (43) \\ &= T^{-1}(T(\boldsymbol{p})) && (44) \\ &= \boldsymbol{p} && (45) \end{aligned}$$

This proves that T_C^{-1} is the inverse of T_C. □

Theorem 6.4 (Computational Complexity of Cached Transformations)**.** *Let $T : X \to Y$ be a transformation with computational complexity $O(f(n))$ for some function f, and let $T_C : X \to Y$ be a cached version of T. The amortised computational complexity of T_C over m operations with k unique inputs is $O(k \cdot f(n)/m)$.*

Proof. In the worst case, where all m operations involve unique inputs, the complexity is $O(f(n))$ per operation, or $O(m \cdot f(n))$ total.

In the best case, where all operations use the same input, only the first operation incurs the $O(f(n))$ cost, and subsequent operations have the $O(1)$ cost for cache lookup. The total cost is then $O(f(n)+(m-1)\cdot 1) = O(f(n)+m-1) = O(f(n)+m)$.

In the general case with k unique inputs where $1 \leq k \leq m$, the total cost is $O(k \cdot f(n) + (m-k) \cdot 1) = O(k \cdot f(n) + m - k) = O(k \cdot f(n) + m)$.

The amortised cost per operation is therefore $O((k \cdot f(n) + m)/m) = O(k \cdot f(n)/m + 1) = O(k \cdot f(n)/m)$ when $f(n)$ is non constant. □

6.2 Adaptive Strategy Selection

In addition to caching, we can optimise performance by adaptively selecting the most efficient strategy for a given transformation based on its characteristics and performance benchmarks.

Definition 6.5 (Transformation Strategy)**.** A transformation strategy S_T is an implementation of a transformation $T : X \to Y$ with specific performance characteristics. Multiple strategies $S_T^1, S_T^2, \ldots, S_T^n$ can implement the same mathematical transformation T but with different computational approaches.

Definition 6.6 (Strategy Performance Metric)**.** For a transformation strategy S_T and a set of inputs $P \subset X$, the performance metric $\mu(S_T, P)$ measures the computational efficiency of S_T when applied to points in P. Lower values indicate better performance.

Definition 6.7 (Adaptive Strategy Selector)**.** An adaptive strategy selector A_T for a transformation $T : X \to Y$ is a function that selects the optimal strategy from a set of available strategies $\{S_T^1, S_T^2, \ldots, S_T^n\}$ based on performance metrics and input characteristics:

$$A_T(P) = \arg\min_{S_T^i} \mu(S_T^i, P) \tag{46}$$

where $P \subset X$ is a representative set of inputs.

Theorem 6.8 (Correctness of Adaptive Strategy Selection)**.** *Let $T : X \to Y$ be a transformation, and let $\{S_T^1, S_T^2, \ldots, S_T^n\}$ be a set of strategies implementing T. If each strategy S_T^i correctly implements T, then the adaptive strategy selector A_T produces results equivalent to T for all inputs.*

Proof. By definition, each strategy S_T^i correctly implements T, so:

$$\forall \boldsymbol{p} \in X, \forall i \in \{1, 2, \ldots, n\} : S_T^i(\boldsymbol{p}) = T(\boldsymbol{p}) \tag{47}$$

The adaptive strategy selector chooses one of these strategies, say $S_T^j = A_T(P)$, based on performance considerations. Since S_T^j correctly implements T, we have:

$$\forall \boldsymbol{p} \in X : A_T(P)(\boldsymbol{p}) = S_T^j(\boldsymbol{p}) = T(\boldsymbol{p}) \tag{48}$$

Therefore, the adaptive strategy selector produces results that are equivalent to T for all inputs. □

Theorem 6.9 (Optimal Performance of Adaptive Strategy Selection)**.** *Under the assumption that past performance is indicative of future performance for similar inputs, the adaptive strategy selector A_T achieves optimal asymptotic performance among the available strategies.*

Proof. Let $\{S_T^1, S_T^2, \ldots, S_T^n\}$ be the set of available strategies for transformation T. The adaptive strategy selector chooses the strategy with the minimum performance metric:

$$A_T(P) = S_T^j \text{ where } j = \arg\min_i \mu(S_T^i, P) \tag{49}$$

By definition, $\mu(S_T^j, P) \leq \mu(S_T^i, P)$ for all $i \in \{1, 2, \ldots, n\}$. If we assume that the performance metric is a good predictor of future performance for similar inputs, then S_T^j will have the best performance for inputs similar to those in P.

Therefore, among the available strategies, A_T achieves optimal asymptotic performance for the given class of inputs. □

7 Calibration Framework

We now extend our mathematical framework to incorporate calibration techniques, which enable accurate parameter estimation and error correction.

7.1 Parameter Estimation

In real-world scenarios, the exact values of parameters such as browser position and DPI scaling may not be known. We develop a mathematical framework for estimating these parameters from observed data.

Definition 7.1 (Reference Points)**.** A set of reference points consists of pairs of corresponding points in different coordinate systems. For example, a set of screen-logical reference points is:

$$R_{S,L} = \{(\boldsymbol{p}_{s,i}, \boldsymbol{p}_{l,i}) | \boldsymbol{p}_{s,i} \in \mathbb{S}, \boldsymbol{p}_{l,i} \in \mathbb{L}, i = 1, 2, \ldots, n\} \tag{50}$$

where $\boldsymbol{p}_{s,i}$ is a point in screen coordinates and $\boldsymbol{p}_{l,i}$ is the corresponding point in logical coordinates.

Theorem 7.2 (Browser Position Estimation)**.** *Given a set of screen-logical reference points $R_{S,L} = \{(\boldsymbol{p}_{s,i}, \boldsymbol{p}_{l,i}) | i = 1, 2, \ldots, n\}$ and a known DPI scaling factor σ, the browser position $\boldsymbol{b}$ can be estimated as:*

$$\boldsymbol{b}_{est} = \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{p}_{s,i} - \sigma \cdot \boldsymbol{p}_{l,i}) \tag{51}$$

Proof. From the definition of the screen-to-logical transformation, we know that:

$$\boldsymbol{p}_{l,i} = T_{S \to L}(\boldsymbol{p}_{s,i}) = \frac{\boldsymbol{p}_{s,i} - \boldsymbol{b}}{\sigma} \tag{52}$$

Rearranging, we get:

$$\boldsymbol{b} = \boldsymbol{p}_{s,i} - \sigma \cdot \boldsymbol{p}_{l,i} \tag{53}$$

In the presence of measurement errors, we can take the average over all reference points to minimise the impact of random errors:

$$\boldsymbol{b}_{\text{est}} = \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{p}_{s,i} - \sigma \cdot \boldsymbol{p}_{l,i}) \tag{54}$$

□

Theorem 7.3 (DPI Scaling Estimation)**.** *Given a set of browser-logical reference points $R_{B,L} = \{(\boldsymbol{p}_{b,i}, \boldsymbol{p}_{l,i}) | i = 1, 2, \ldots, n\}$, the DPI scaling factor σ can be estimated as:*

$$\sigma_{est} = \frac{\sum_{i=1}^{n} \boldsymbol{p}_{b,i} \cdot \boldsymbol{p}_{l,i}}{\sum_{i=1}^{n} \boldsymbol{p}_{l,i} \cdot \boldsymbol{p}_{l,i}} \tag{55}$$

where $\boldsymbol{p}_{b,i} \cdot \boldsymbol{p}_{l,i}$ denotes the dot product of the two points, treating them as vectors.

Proof. From the definition of the browser-to-logical transformation, we know that:

$$\boldsymbol{p}_{l,i} = T_{B\to L}(\boldsymbol{p}_{b,i}) = \frac{\boldsymbol{p}_{b,i}}{\sigma} \tag{56}$$

This implies:

$$\boldsymbol{p}_{b,i} = \sigma \cdot \boldsymbol{p}_{l,i} \tag{57}$$

We can formulate this as an overdetermined system of linear equations and solve it using the least squares method to minimise the sum of squared errors:

$$\min_{\sigma} \sum_{i=1}^{n} ||\boldsymbol{p}_{b,i} - \sigma \cdot \boldsymbol{p}_{l,i}||^2 \tag{58}$$

Taking the derivative with respect to σ and setting it to zero:

$$\frac{d}{d\sigma} \sum_{i=1}^{n} ||\boldsymbol{p}_{b,i} - \sigma \cdot \boldsymbol{p}_{l,i}||^2 = 0 \tag{59}$$

$$\sum_{i=1}^{n} -2 \cdot \boldsymbol{p}_{l,i} \cdot (\boldsymbol{p}_{b,i} - \sigma \cdot \boldsymbol{p}_{l,i}) = 0 \tag{60}$$

$$\sum_{i=1}^{n} \boldsymbol{p}_{l,i} \cdot \boldsymbol{p}_{b,i} - \sigma \cdot \sum_{i=1}^{n} \boldsymbol{p}_{l,i} \cdot \boldsymbol{p}_{l,i} = 0 \tag{61}$$

Solving for σ:

$$\sigma_{\text{est}} = \frac{\sum_{i=1}^{n} \boldsymbol{p}_{l,i} \cdot \boldsymbol{p}_{b,i}}{\sum_{i=1}^{n} \boldsymbol{p}_{l,i} \cdot \boldsymbol{p}_{l,i}} \tag{62}$$

Since $\boldsymbol{p}_{l,i} \cdot \boldsymbol{p}_{b,i} = \boldsymbol{p}_{b,i} \cdot \boldsymbol{p}_{l,i}$ (dot product is commutative), the result follows. □

Theorem 7.4 (Complete Parameter Estimation)**.** *Given a set of screen-logical reference points $R_{S,L} = \{(\boldsymbol{p}_{s,i}, \boldsymbol{p}_{l,i}) | i = 1, 2, \ldots, n\}$, we can estimate both the browser position $\boldsymbol{b}$ and the DPI scaling factor σ by solving the following system:*

$$\min_{\boldsymbol{b},\sigma} \sum_{i=1}^{n} \left|\left| \boldsymbol{p}_{l,i} - \frac{\boldsymbol{p}_{s,i} - \boldsymbol{b}}{\sigma} \right|\right|^2 \tag{63}$$

Proof. We want to find the parameters $\boldsymbol{b}$ and σ that minimise the square difference between the observed logical coordinates and those predicted by our transformation:

$$\min_{\boldsymbol{b},\sigma} \sum_{i=1}^{n} ||\boldsymbol{p}_{l,i} - T_{S\to L}(\boldsymbol{p}_{s,i})||^2 = \min_{\boldsymbol{b},\sigma} \sum_{i=1}^{n} \left|\left| \boldsymbol{p}_{l,i} - \frac{\boldsymbol{p}_{s,i} - \boldsymbol{b}}{\sigma} \right|\right|^2 \tag{64}$$

This nonlinear optimisation problem can be solved using numerical techniques such as gradient descent or the Levenberg-Marquardt algorithm.

Alternatively, we can use an iterative approach:

1. Initialise σ to a reasonable value (e.g., 1.0 or an estimated value from other sources).
2. Estimate $\boldsymbol{b}$ using Theorem 7.2.
3. Use the estimated $\boldsymbol{b}$ to calculate the browser coordinates: $\boldsymbol{p}_{b,i} = \boldsymbol{p}_{s,i} - \boldsymbol{b}$.
4. Estimate σ using Theorem 7.3 with the calculated browser coordinates.
5. Repeat steps 2-4 until convergence.

This iterative approach converges to a local minimum of the objective function. □

7.2 Error Correction

Even with accurate parameter estimation, systematic errors in coordinate transformations may occur due to factors such as nonlinear scaling or inaccuracies in hardware reporting. We develop a framework for correcting these errors.

Definition 7.5 (Error Correction Function)**.** An error correction function $E : X \to X$ for a coordinate space X is a function that maps a point in X to a corrected point in X that better corresponds to the actual physical position.

Definition 7.6 (Corrected Transformation)**.** Given a transformation $T : X \to Y$ and error correction functions $E_X : X \to X$ and $E_Y : Y \to Y$, the corrected transformation $T_E : X \to Y$ is defined as:

$$T_E(\boldsymbol{p}) = E_Y^{-1}(T(E_X(\boldsymbol{p}))) \tag{65}$$

where E_Y^{-1} is the inverse of E_Y.

Theorem 7.7 (Affine Error Correction)**.** *If the error correction functions E_X and E_Y are affine transformations, and the original transformation T is affine, then the corrected transformation T_E is also an affine transformation.*

Proof. Let $E_X(\boldsymbol{p}) = A_X\boldsymbol{p} + \boldsymbol{c}_X$ and $E_Y(\boldsymbol{q}) = A_Y\boldsymbol{q} + \boldsymbol{c}_Y$, where A_X, A_Y are linear transformations, and $\boldsymbol{c}_X, \boldsymbol{c}_Y$ are constant vectors.

The inverse of E_Y is given by $E_Y^{-1}(\boldsymbol{q}) = A_Y^{-1}(\boldsymbol{q} - \boldsymbol{c}_Y)$.

Let $T(\boldsymbol{p}) = A_T\boldsymbol{p} + \boldsymbol{c}_T$ be the original affine transformation.

Then:

$$\begin{aligned} T_E(\boldsymbol{p}) &= E_Y^{-1}(T(E_X(\boldsymbol{p}))) && (66)\\ &= E_Y^{-1}(T(A_X\boldsymbol{p} + \boldsymbol{c}_X)) && (67)\\ &= E_Y^{-1}(A_T(A_X\boldsymbol{p} + \boldsymbol{c}_X) + \boldsymbol{c}_T) && (68)\\ &= E_Y^{-1}(A_TA_X\boldsymbol{p} + A_T\boldsymbol{c}_X + \boldsymbol{c}_T) && (69)\\ &= A_Y^{-1}(A_TA_X\boldsymbol{p} + A_T\boldsymbol{c}_X + \boldsymbol{c}_T - \boldsymbol{c}_Y) && (70)\\ &= A_Y^{-1}A_TA_X\boldsymbol{p} + A_Y^{-1}(A_T\boldsymbol{c}_X + \boldsymbol{c}_T - \boldsymbol{c}_Y) && (71) \end{aligned}$$

This is in the form $A\boldsymbol{p} + \boldsymbol{c}$ where $A = A_Y^{-1}A_TA_X$ and $\boldsymbol{c} = A_Y^{-1}(A_T\boldsymbol{c}_X + \boldsymbol{c}_T - \boldsymbol{c}_Y)$, which is an affine transformation. □

Theorem 7.8 (Error Correction Parameter Estimation)**.** *Given a set of reference points $R_{X,Y} = \{(\boldsymbol{p}_{x,i}, \boldsymbol{p}_{y,i}) | i = 1, 2, \ldots, n\}$ and a transformation $T : X \to Y$, we can estimate affine error correction functions E_X and E_Y by solving:*

$$\min_{E_X, E_Y} \sum_{i=1}^{n} ||E_Y(T(E_X^{-1}(\boldsymbol{p}_{x,i}))) - \boldsymbol{p}_{y,i}||^2 \tag{72}$$

Proof. The goal is to find error correction functions E_X and E_Y such that the corrected transformation T_E accurately maps the reference points:

$$T_E(\boldsymbol{p}_{x,i}) \approx \boldsymbol{p}_{y,i} \text{ for all } i \tag{73}$$

This is equivalent to:

$$E_Y^{-1}(T(E_X(\boldsymbol{p}_{x,i}))) \approx \boldsymbol{p}_{y,i} \tag{74}$$

Applying E_Y to both sides:

$$T(E_X(\boldsymbol{p}_{x,i})) \approx E_Y(\boldsymbol{p}_{y,i}) \tag{75}$$

If we parameterise E_X and E_Y as affine transformations:

$$\begin{aligned} E_X(\boldsymbol{p}) &= A_X\boldsymbol{p} + \boldsymbol{c}_X && (76)\\ E_Y(\boldsymbol{q}) &= A_Y\boldsymbol{q} + \boldsymbol{c}_Y && (77) \end{aligned}$$

We can formulate this as an optimisation problem:

$$\min_{A_X, \boldsymbol{c}_X, A_Y, \boldsymbol{c}_Y} \sum_{i=1}^{n} ||T(A_X\boldsymbol{p}_{x,i} + \boldsymbol{c}_X) - (A_Y\boldsymbol{p}_{y,i} + \boldsymbol{c}_Y)||^2 \tag{78}$$

This can be solved using numerical optimisation techniques. □

8 Conclusion

This paper has extended the mathematical framework for browser position calculation to include performance optimisation and calibration techniques. We have rigorously defined and provided formal proofs for:

- The core coordinate systems and transformations between them
- The mathematical properties of transformation caching and adaptive strategy selection
- The parameter estimation and error correction methods for calibration

Our framework provides a unified approach to coordinate translation that is mathematically sound and computationally efficient. It enables accurate positioning in diverse display environments, even in imperfect parameter knowledge or measurement errors.

The mathematical foundations established in this paper form the basis for implementing reliable, efficient, and accurate coordinate transformation systems for browser environments. By combining theoretical rigour with practical concerns, we have created a comprehensive framework that addresses the full spectrum of challenges in position calculation.