Optimising Automatic Calibration of Electric Muscle Stimulation

Graeme Gange,[1] **Jarrod Knibbe**[2]
[1]Monash University, Victoria, Australia
[2]The University of Melbourne, Victoria, Australia
graeme.gange@monash.edu, jarrod.knibbe@unimelb.edu.au

Abstract

Electrical Muscle Stimulation (EMS) has become a popular interaction technology in Human-Computer Interaction; allowing the computer to take direct control of the user's body. To date, however, the explorations have been limited to coarse, generalised examples, due to the low resolution of achievable control. To increase this resolution, the EMS needs to increase significantly in complexity - using large numbers of electrodes in complex patterns. The calibration of such a system remains an unsolved challenge. We present a new SAT-based black-box calibration method, which requires no spatial information about muscular or electrode positioning. The method encodes domain knowledge and observations in a constraint model, and uses these to prune the space of feasible control signals. In a simulated environment we find this method can scale reliably to large arrays while requiring only a modest number of trials, and preliminary tests on real hardware show we can effectively calibrate an electrode array in a few minutes.

Introduction

Functional Electrical Stimulation (FES) has long been used and explored for the upper limb rehabilitation of stroke and spinal cord injury patients (e.g., (Usman et al. 2020; Bouton et al. 2016; De Marchis et al. 2016)). More recently, FES has also been adopted (under the guise of Electric Muscle Stimulation - EMS) as a popular input-output modality in Human Computer Interaction (HCI), affording the computer physical control of the user. Through EMS, the computer can dynamically alter the user's physical actions, such as to improve drawing (Lopes et al. 2016), teach the use of new objects (Lopes, Jonell, and Baudisch 2015), or provide 'forced' walking directions (Pfeiffer et al. 2015).

To date, EMS use in HCI has remained predominantly conceptual, considering themes of opportunities within the space of computer-human control (e.g., for 'proprioceptive interaction' or as haptics for VR (Lopes et al. 2015, 2017)), rather than presenting real world, deployable systems with tangible performance gains, per more traditional HCI. One reason for this, we suggest, is the challenge of achieving high-resolution and fine-grained control.

The bio-medical literature, however, reveals that more complex movement patterns (i.e., non-ballistic, combining agonist and antagonist muscles) can be achieved through dense electrode arrays and combinations of complex signals (Popović-Bijelić et al. 2005; Popović and Popović 2011). As electrode numbers increase, however, so too does the complexity of calibrating the system - determining what electrode patterns and signal parameters will achieve the desired movement. This is not prohibitive in the medical and rehabilitation domain, where practitioners can take their time to optimally calibrate bespoke electrode arrays for the best outcomes for the patient (Bouton et al. 2016). The move to HCI and interaction design, however, necessitates a simplified, general-purpose approach to system calibration.

Currently, the calibration problem remains unsolved (Knibbe et al. 2017). Previous work has explored automated calibration techniques that demonstrate only limited success (Knibbe et al. 2017), or don't scale to support complex stimulation patterns across multiple electrodes (Usman et al. 2020). Without a calibration solution, the further exploration and use of EMS for human-machine interaction remains heavily constrained. Such an automated calibration system should be adaptable to different users, and electrode configurations, with minimal effort. As such we cannot rely on spatial information about muscle positions or electrode layouts. However, a blind exploration of $O(2^{|E|^2})$ activation patterns (for E electrodes) is also clearly impractical. Instead, we need a calibration procedure which carefully explores the search space, learning as much as possible from its previous attempts.

We present a novel approach to auto-calibration of high-density EMS systems. We use a fast, intelligent walk-through of a sub-set of possible configurations, to identify the total resolution of an EMS system. Through a subsequent reduction of the walked-through space, we identify the specific patterns that trigger a given movement; both the agonist and antagonist muscles involved. We report on a technical evaluation of our calibration.

Background

We contribute a new automatic calibration technique for Electric Muscle Stimulation (EMS) for interaction design.

We begin by introducing current approaches to calibration in HCI and beyond, before discussing relevant related work around propositional inference.

Calibrating EMS

EMS systems require both *spatial calibration*, determining the correct electrode placement, and *signal calibration*, determining the correct signal parameters. EMS practitioners typically begin by choosing an initial placement for the 2-8 electrodes being used, selecting stimulation parameters (for pulse width - μs, frequency - Hz, and amplitude - mA), and then engaging in a trial-and-error led procedure, where electrodes are placed, tested, repositioned, tested, and so on. This process can easily take a few minutes for a pair of electrodes (Lopes, Jonell, and Baudisch 2015), scaling linearly as more electrodes are added.

As we look to achieve higher-resolution movement control (such as individual finger actuation, for example), and increase the numbers of electrodes in use, manual calibration quickly becomes impractical and an automated process is required. Knibbe et al. (Knibbe et al. 2017), for example, coupled EMS to electromyography (EMG - reading signals from muscles during actuation), to facilitate a read-and-write approach to calibration, where participants perform a movement that the computer seeks to replicate. While this technique scales easily to many electrodes, initial results only achieved limited accuracy (60%). Furthermore, even assuming perfect calibration, we argue that there are conceptual flaws that limit this idea. Namely, the system learns to produce *certain* gestures, rather than determining anything about the joint-level control, which would facilitate the performance of *any* gesture.

More recently, research has considered sequential explorations of electrode combinations (e.g., (Usman et al. 2020; Popović and Popović 2009)). These works calibrated arrays of 20-60 small electrodes, for individual finger actuation, testing all pairs of electrodes until individual finger configurations were found. Another similar approach is the twitch protocol (Malešević et al. 2010, 2012), wherein individual pulses across electrodes are repeated multiple times, and any resultant finger movement is measured. Whilst promising, these processes scale linearly with electrode count, and only consider simple configurations of electrode pairs.

We look to optimise the auto-calibration of complex electrode arrays, facilitating multi-electrode pairings, whilst constraining the search space that needs to be explored.

SAT and Inference

To guide our automatic calibration, we will rely on automated reasoning techniques based on *Boolean satisfiability* (SAT) to draw inferences from our observations. Given a finite universe V of Boolean variables, a SAT problem asks whether there exists an assignment of each $v \in V$ such that a formula φ (in conjunctive normal form) is satisfied.

Modern conflict-directed clause learning (CDCL) (Zhang et al. 2001)-based SAT solvers are impressively scalable, and frequently used as oracles for inference and verification tasks. Many solvers offer an *assumption* (Eén and Sörensson 2003) interface: given a set A of *literals* (variables, or their negations), the user may ask whether $\varphi \wedge A$ is satisfiable. If not, the solver will return an *unsatisfiable core*: a small subset A' of A such that $\varphi \wedge A'$ is unsatisfiable. Lazy clause generation (LCG) (Ohrimenko, Stuckey, and Codish 2009) constraint solvers extend SAT techniques with support for integer variables and complex constraints. In this paper, we shall be using an LCG solver to maintain and answer queries over a propositional knowledge base. We encode knowledge by adding new constraints to the solver. To check whether $Q \models_{\mathcal{E}} x$ (that is, does Q entail x given knowledge-base $\mathcal{E}$), we check satsfiability of $repr(\mathcal{E}) \wedge A \wedge \neg x$. The entailment holds if this is *unsatisfiable* (there is no assignment consistent with $\mathcal{E}$ where Q holds, and x does not). Similarly, if we want to *find* Q such that $Q \not\models_{\mathcal{E}} x$, we ask for an assignment satisfying $repr(\mathcal{E}) \wedge \neg x$, and read Q out of the result.

Algorithm 1 Pseudocode for QuickXplain

```
function QUICKXPLAIN(oracle, S)
    return QX(oracle, S, ∅, ⊥)
function QX(oracle, F, B, δ_B)
    if δ_B = ⊤ then
        if oracle(B) then return ∅
    if F = {x} then
        return {x}
    Choose F_1 ⊂ F with |F_1| = |F|/2
    F_2 ← F − F_1
    F'_1 ← QX(oracle, F_1, B ∪ F_2, ⊤)
    F'_2 ← QX(oracle, F_2, B ∪ F'_1, F'_1 ≠ ∅)
    return F'_1 ∪ F'_2
```

Minimizing Powersets

A common task when reasoning about powersets is, given a set S satisfying some monotone property p, to identify some subset-minimal set $S' \subseteq S$ satisfying p. A classic algorithm for this problem is QUICKXPLAIN (Junker 2004). Given some oracle for testing p, QUICKXPLAIN identifies a minimal subset of S in $\mathrm{O}(k \log|S|)$ queries, where k is the size of the identified subset.

Pseudo-code for QUICKXPLAIN is given in Algorithm 1. Each call identifies the minimum subset of a *foreground* set F given an assumed (background) It operates by splitting the foreground set S into equal-size subsets F_1, F_2, then recursively computes first a minimal subset of F_1 *given* $F_2 \cup B$, then a minimal subset of F_2 given the *remaining* elements of F_1 (with B).

Formulating the Calibration Problem

We consider a setting where we have a set of actions $\mathcal{A} = \{a_1, \ldots, a_n\}$, responses $\mathcal{R} = \{r_1, \ldots, r_n\}$. Responses can be activated by (initially unknown) sets of actions. We assume activation is monotone: so if A activates response r, $A \cup A'$ also activates r.

However, each response r_i has a (possibly unknown) set of *antagonists* $S_i \subseteq \mathcal{R} - \{r_i\}$. If r_i is activated without any antagonists, response r_i will be observed. But if both r_i

variable	$\mathcal{D}$	meaning
$selected(a)$	$\{0,1\}$	Is action $a \in P$?
$seen$	$\{0,1\}$	Has P already been observed?
$activated(r)$	$\{0,1\}$	Is P known to activate r?
$suppressed(r)$	$\{0,1\}$	Is P known to suppress r?
$pattern(r)$	$\{0,1\}$	Does P contain a pattern for r?
$pattern\_id(r)$	$\mathbb{Z}^+$	*Which* pattern for r is in P?

Table 1: Variables introduced to represent knowledge about the calibration process when choosing a set of actions P to observe. $\mathcal{D}$ indicates the *domain* of each variable.

and some antagonist are activated, the observed response is indeterminate: r_i may or may not be observed.

Let a *goal pattern* G be a pair (T, F) of *target* responses T, and *forbidden* responses F. Given a set of goal patterns $\mathfrak{G} \subseteq \mathcal{P}(A) \times \mathcal{P}(A)$ and oracle O, we wish to find for each $(T, F) \in \mathfrak{G}$ a set A_T such that $T \subseteq O(A_T) \subseteq \mathcal{R} - F$.

Observations and Inferences

Given that each observation involves activating a pattern of electrodes and physically observing the response, it is essential that we infer as much as possible from our observations.

To this end, we will maintain our knowledge base as constraints in the LCG solver `geas` (Gange et al. 2020) – we will refer to this, together with the history of past observations, as the *environment* $\mathcal{E}$. Table 1 summarizes the variables appearing in the constraint model. Each time we make an observation, we add new constraints to the model relating the $selected(a_i)$ variables (for the pattern we observed) to the other variables. For example, if $O(P) = R$, we add $\bigwedge_{a \in P} selected(a) \rightarrow activated(r)$ for $r \in R$. If a strict subset of P is known to activate $r' \notin R$, we also add $\bigwedge_{a \in P} selected(a) \rightarrow suppressed(r')$. We may augment this knowledge-base with additional background knowledge (e.g. $activated(r) \rightarrow suppressed(r')$ for a known antagonist). We query $\mathcal{E}$ by asking the solver for solutions to the constraint problem which satisfy additional constraints. In a slight abuse of notation, the following sections will use $A \models_{\mathcal{E}} x$ to denote the query $\bigwedge_{a \in A} selected(a) \models_{\mathcal{E}} x$.

Achieving a Goal Pattern

To achieve a goal (T, F), we need to find action set A which:

- Contains some activation set for each $r \in T$
- Activates no (observed) antagonist for any $r \in T$
- Does not contain any (observed) activation set for $r \in F$.

Activation sets for elements $R - T$ can manifest in two ways: if we evaluate query Q and observe $r' \in R - T$, then Q necessarily contains an activation set for r'. But if query Q produces response r, and query $Q \cup A$ does not, then $Q \cup A$ must contain an activation set for *some* antagonist of r.

Our calibration procedure works by building up a pool of known sub-patterns, which we have observed activating subsets of the target responses. It then attempts obtain the overall goal by combining previously observed patterns.

The procedure alternates among three phases:

Exploration Attempting to quickly find action-sets which cover all target responses.

Contraction Shrinking action sets to eliminate unwanted responses.

Aggregation Attempt to achieve the target by combining sub-patterns.

Throughout this process, we update our constraint knowledge-base with the results of any observations we have made. Pseudo-code for the overall calibration procedure is given in Figures 2 and 3.

Aggregation Given known patterns P and current environment $\mathcal{E}$, the aggregation phase we attempt to select some subset P_T of P which (a) covers target T, and (b) is not known to contain forbidden patterns. If we find a candidate, we query the oracle with P_T. There are three possiblities:

- $T \subseteq \text{OBSERVE}(\mathcal{E}, P_T) \subseteq \mathcal{R} - F$, so we have succeeded.
- $\text{OBSERVE}(\mathcal{E}, P_T)$ produced a response $r' \in F$, thus we find a minimal activation set for r', which we then forbid.
- $\text{OBSERVE}(\mathcal{E}, P_T)$ failed to produce a response $r \in T$. Since there was some subset P_r which *did* activate r, we search for a minimal superset of P_r which *does not* activate r. If, during this step, we make an observation which instead activates a forbidden response r', we switch to finding a minimal activation for r'.

In the first case, we return the successful pattern. Otherwise, we update our knowledge base with the new observations – which blocks the current combination of patterns – then ask the solver for a new candidate pattern. Eventually, we either successfully achieve the target pattern, or the solver returns failure, and a core indicating *which* subset T_f of targets could not be simultaneously achieved. T_f is used to guide the exploration process: there is no point resuming aggregation until we have found a *new* pattern for some $r \in T_f$ which does not activate any responses in F.

Example 1 *Consider a case with five actions* $\{a_1, \ldots, a_5\}$ *and responses* $\{r_1, \ldots, r_4\}$*, with goal* $(T = \{r_1, r_2, r_3\}, F = \{r_4\})$ *After making several observations, we have discovered the following patterns:*

$$\{a_1, a_2\} \vdash r_1, \{a_3, a_4\} \vdash r_2, \{a_5\} \vdash r_3$$

and have made the additional observations:

$$\{a_1, a_2, a_5\} \vdash \{r_1, r_3\}, \{a_1, a_3, a_5\} \vdash \{r_1, r_3\}$$

In addition, there are as-yet unidentified patterns:

$$\{a_1, a_5\} \vdash r_1, \{a_2, a_4\} \vdash r_4$$

Given these observations, AGGREGATE *will choose candidate pattern* $S = \{a_1, \ldots, a_5\}$*. Which we then observe.*

If we observe $S \vdash \{r_1, r_2, r_3\}$*, we are done: while* S *contains a subset which activates a forbidden response, we still successfully achieved the goal response. But if instead we find* $S \vdash \{r_1, r_2, r_3, r_4\}$*, we will call* REFINE-PAT *to isolate the sub-pattern that produced* r_4*.*

Once REFINE-PAT *isolates* $\{a_2, a_4\}$*, we add this information to the knowledge-base. Then* AGGREGATE *attempts to find a replacement candidate. It fails, returning* $\{r_1, r_2\}$ *as a conflict: indicating that every known combination of patterns for* r_1 *and* r_2 *activates some forbidden response.*

Algorithm 2 The aggregation phase: attempting to trigger goal (T, F) by combining some collection of existing patterns.

function CALIBRATE($\mathcal{E}, T, F$)
  while *true* **do**
    match AGGREGATE(M, P, T) **with**
    case OKAY(A): ▷ Found candidate total pattern.
      if CHECK-PAT$(\mathcal{E}, T, F, A)$ **then**
        return OKAY(A)
    case FAILED(T'): ▷ No consistent candidates yet
      match SEEK$(\mathcal{E}, T', F)$ **with**
      case OKAY(A, R):
        $(A, R) \leftarrow$ SEEK$(\mathcal{E}, T', F)$
        $(A_T, R_T) \leftarrow$ REFINE-PAT$(\mathcal{E}, T', A, R)$
        if $R_T \cap F \neq \emptyset$ **then**
          $(A_F, R_F) \leftarrow$ REFINE-PAT$(\mathcal{E}, F, A_T, R_T)$
          RECORD-PAT$(\mathcal{E}, A_F, R_F)$
        RECORD-PAT$(\mathcal{E}, A_T, R_T)$
      case FAILED: ▷ Search space exhausted
        return FAILED

Algorithm 3 Given a candidate query, check whether it achieves current goal. If not, identify what went wrong and update the knowledge-base.

function CHECK-PAT$(\mathcal{E}, T, F, A)$
  $R \leftarrow$ OBSERVE$(\mathcal{E}, A)$
  if $T \subseteq R \wedge R \cap F \neq \emptyset$ **then**
    return True
  if $R \cap F \neq \emptyset$ **then**
    $(A', R') \leftarrow$ REFINE-PAT$(\mathcal{E}, F, R', A')$
    RECORD-PAT$(\mathcal{E}, A', R')$
  else ▷ $T - R \neq \emptyset$
    Let $r \in T - R$, and A_r be the corresponding activation set.
    $(A_s, R_s) \leftarrow$ REFINE-SUP$(r, F, R, A - A_r, A_r)$
    if $R_s \cap F \neq \emptyset$ **then**
      $(A_f, R_f) \leftarrow$ REFINE-PAT$(\mathcal{E}, F, R_s, A_r, R_f)$
      RECORD-PAT$(\mathcal{E}, A_s \cup A_f, R_f)$
    else
      RECORD-SUP$(\mathcal{E}, r, A_s \cup A_r)$
  return False

Exploration If aggregation fails, we must find a new pattern for at least one of the failed responses T_f. But the space of action-sets is extremely large, so we wish to – as far as possible – avoid blindly trying sets of actions. During previous steps (finding or minimizing patterns for other goals), we may already have found, but not refined into a pattern, action-sets triggering the desired response.

When aggregation fails, then, we first search our history of observations. If we find some previous observation that (a) triggers some response $r \in T_f$, and (b) is not a superset of some already-registered pattern for r, we return that pattern for refinement.

Example 2 *In Example 1,* AGGREGATE *failed with conflict* $\{r_1, r_2\}$. *Looking at our previous observations, we see both* $\{a_1, a_2, a_5\}$ *and* $\{a_1, a_3, a_5\}$ *activated* r_1.

$\{a_1, a_2, a_5\}$ *already contains a known pattern for* r_1*, so is ignored. However* $\{a_1, a_3, a_5\}$ *contains a known pattern for* r_3*, but* not r_1*: thus* SEEK *return* $\{a_1, a_3, a_5\}$ *as the next pattern to refine.*

Algorithm 4 SEEK is called when there is no feasible combination of patterns for a subset T of the current targets. It attempts to identify a set of actions which activate some $r \in T$ which is not already covered by a pattern for r.

function SEEK$(\mathcal{E}, T, F)$
  if $\exists\ r \in T,\ (A, R) \in observed(\mathcal{E}).\ r \in R, A \not\models_{\mathcal{E}} pattern(r)$ **then**
    return (A, R)
  while SAMPLE$(\mathcal{E}, T, F) =$ OKAY(A, R) **do**
    $R \leftarrow$ OBSERVE$(\mathcal{E}, A)$
    if $R \cap T \neq \emptyset$ **then**
      return (A, R)
  return FAILED

$pattern\_id(r)$ is unconstrained in this phase (and the next), allowing the solver to choose an unspecified 'future' pattern for responses.

Sampling Eventually, SEEK may exhaust the set of previously observed responses. At this point, the algorithm must choose a new set of actions to test, in the hopes of activating a target response.

Due to the impact of antagonists, we can only draw weak inferences from failed queries: if A fails to activate r, we cannot tell whether we need more actions, or fewer. However, we can do better than blind guessing by exploiting our previous observations. We consider only sets A satisfying:

$$A \not\models_{\mathcal{E}} seen \tag{1}$$
$$\forall\ r \in F.\ A \not\models_{\mathcal{E}} activated(r) \tag{2}$$
$$\forall\ r \in T.\ A \not\models_{\mathcal{E}} suppressed(r) \tag{3}$$
$$\exists\ r \in T.\ A \not\models_{\mathcal{E}} pattern(r) \tag{4}$$

(1) prevents us from choosing any set we have already observed, also indirectly ensuring we eventually terminate. (2–3) ensures the selected subset is not known to contain any forbidden activations. Finally, (4) restricts our search to sets which are not covered by existing patterns.

Note that we cannot replace (4) with the simpler $\forall\ r \in T.\ A \not\models_{\mathcal{E}} activated(r)$, as the minimum pattern for target r may be a *superset* of an existing pattern for some other target r'.

But it is also worth considering the cardinality of set A. Consider the case where there is a single minimum set P_r which activates response r, and a single antagonist Q_r. If we choose very small sets ($\simeq |P_r|$), it may take many queries before we pick some $A \supseteq P_r$. But if we choose very large sets ($\simeq |\mathcal{A}|$), it is very likely that $A \supseteq Q_r$.

Unfortunately, if A fails to produce a response, we have no way of determining whether A was too large, or too small. As such, we adopt a simple approach: we use the

Algorithm 5 Choosing a new subset to explore, given target set T and forbidden set F.

function SAMPLE($\mathcal{E}, T, F$)
  $card \leftarrow$ SELECT-CARD($\mathcal{E}$)
  $\varphi \leftarrow \begin{cases} \neg seen \\ \wedge \quad \bigwedge_{r \in T} \neg suppressed(r) \\ \wedge \quad \bigwedge_{r \in F} \neg suppressed(r) \\ \wedge \quad \bigvee_{r \in T} \neg pattern(r) \end{cases}$
  if $\varphi \models_{\mathcal{E}} \bot$ **then**
    return FAILED
  while $A_{sel} \neq \emptyset \wedge |A| \leq card$ **do**
    Choose $a \in A_{sel}$ uniformly.
    $A_{sel} \leftarrow A_{sel} - \{a\}$
    if $\varphi \wedge (a \in A) \not\models_{\mathcal{E}} \bot$ **then**
      $A \leftarrow A \cup \{a\}$
      $\varphi \leftarrow \varphi \wedge a \in A$
  while $A_{sel} \neq \emptyset$ **do**
    Choose $a \in A_{sel}$ uniformly.
    $A_{sel} \leftarrow A_{sel} - \{a\}$
    if $\varphi \wedge a \notin A \not\models_{\mathcal{E}} \bot$ **then**
      $\varphi \leftarrow \varphi \wedge a \notin A$
    else
      $A \leftarrow A \cup \{a\}$
    $R \leftarrow$ OBSERVE($\mathcal{E}, A$)
  if $R \cap T \neq \emptyset$ **then**
    REWARD-CARD($\mathcal{E}$, $card$)
  return OKAY(A, R)

UCB1 (Auer, Cesa-Bianchi, and Fischer 2002) multi-armed bandit algorithm to select a candidate subset size s (capped at $\frac{A}{2}$), then use the knowledge-base to select a candidate set of actions of the appropriate size. If the response R contains any un-covered targets, the bandit receives reward $\frac{1}{|R|}$ (in this stage, we prefer queries which yield fewer responses).

Pseudo-code for SAMPLE is given in Algorithm 5. Note that the last component of φ, representing (4), is disjunctive so cannot be directly imposed as an assumption. Instead, we track a single witness r_w which we post as an assumption. If the formula is satisfiable assuming $\neg pattern(r_w)$, it is also satisfiable under (4). Otherwise, we try remaining candidates $r \in T$, and conclude unsatisfiability iff all candidates fail. SAMPLE attempts to select $card$ random actions to add to A. But for each action, we first check that it *can* be added to A without violating φ. If it cannot (e.g. action a' is already selected, and $\{a', a\}$ is a known pattern for the current target), we skip a and try a different action. A consequence of this is that the set returned by SAMPLE does not *necessarily* have cardinality $card$.

Contraction

It is unlikely that the exploration phase will find a pattern which activates only target responses. Instead, once we make an observation with *interesting* behaviour, we will attempt to prune irrelevant actions, to find a smaller set exhibiting the same behaviour. There are two kinds of behaviour we find interesting: observed target responses not covered by existing patterns, and *missing* responses known to contain an existing pattern.

Algorithm 6 Given $F \cup B \vdash R$, we wish to identify a minimal subset of F which still triggers something in T (having $R \cap T \neq \emptyset$).

function REFINE-PAT($\mathcal{E}, T, R, A$)
  return REFINE-T($\mathcal{E}, T, R, A, \emptyset, false$)
function REFINE-T($\mathcal{E}, T, R, F, B, \delta_B$)
  if δ_B **then**
    $R_B \leftarrow$ OBSERVE($\mathcal{E}, B$)
    if $R_B \cap T \neq \emptyset$ **then**
      return $(\emptyset, R_B)$
  Select non-empty F_1, F_2 s.t. $F = F_1 \uplus F_2$
  $(F_1', R_1') \leftarrow$ REFINET($\mathcal{E}, T, R, F_1, B \cup F_2, true$)
  $(F_2', R_2') \leftarrow$ REFINET($\mathcal{E}, T, R_1', F_2, B \cup F_1', |F_1'| > 0$)
  return $F_1' \cup F_2', R_2'$
function REFINE-SUP($\mathcal{E}, r, T, R, A, A_r$)
  return REFINES($\mathcal{E}, r, T, R, A, A_r$, `true`)
function REFINES($\mathcal{E}, r, T, R, F, B, \delta_B$)
  if δ_B **then**
    $R_B \leftarrow$ OBSERVE($\mathcal{E}, B$)
    if $r \notin R_B \vee R_B \cap T \neq \emptyset$ **then**
      return $(\emptyset, R_B)$
  Select non-empty F_1, F_2 s.t. $F = F_1 \uplus F_2$
  $(F_1', R_1') \leftarrow$ REFINES($\mathcal{E}, r, T, R, F_1, B \cup F_2, true$)
  if $R_1' \cap T \neq \emptyset$ **then**
    $(F_2', R_2') \leftarrow$ REFINET($\mathcal{E}, T, R_1', F_2, B \cup F_1', F_1' \neq \emptyset$)
  else
    $(F_2', R_2') \leftarrow$ REFINES($\mathcal{E}, r, T, R_1', F_2, B \cup F_1', F_1' \neq \emptyset$)
  return $F_1' \cup F_2', R_2'$

In the first case, we call REFINE-PAT($\mathcal{E}, T, R, A$) to generate a new pattern. Here A is the set of actions, R the observed response to A, and T is the set of *relevant* targets – those without known patterns covered by A. REFINE-PAT is a direct implementation of QUICKXPLAIN, augmented to return the observed responses to the returned actions.

In the second case, we know actions A_r produced response r, but $A = A_r \cup A'$ did not. However, we want to *generalise* the cause of failure, to prune more infeasible candidates. But this *negative* information is weak: we can only block *supersets* of A_r. We thus call REFINE-SUP($\mathcal{E}, r, F, A_{ex}, A_r$), another modified QUICKXPLAIN, to identify a minimal subset $A_{\overline{r}} \subseteq A_{ex}$ such that $A_r \cup A_{\overline{r}}$ *does not* result in r. If, during this process, we observe some forbidden response $r' \in F$, we switch to blocking r' instead, in hope of obtaining a more re-usable constraint. In either case, after identifying a minimal subset we update the knowledge-base, (indirectly) blocking the current aggregate pattern.

Example 3 *In Example 1, we evaluated candidate* $S = \{a_1, \ldots, a_4\}$ *and obtained forbidden response* r_4. AGGREGATE *must now block some subset of* S.

Starting with foreground set $F = S$ *and empty background* $B = \emptyset$, REFINE-TARGET *splits* F *into two equal-size*

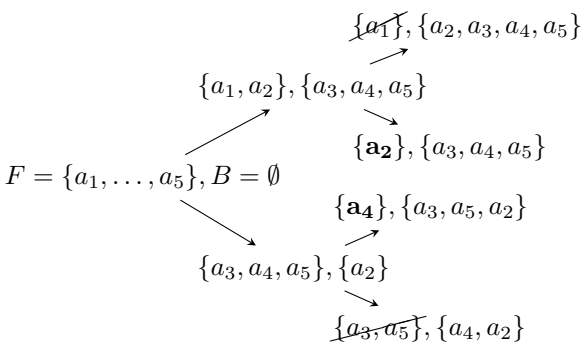

Figure 1: REFINET isolating a pattern for r_4 in Example 3.

random subsets, say $F_1 = \{a_1, a_2\}, F_2 = \{a_3, a_4, a_5\}$, then recursively tries to find the minimum subset of F_1 given F_2.

F_2 by itself only yields r_2, so we need at least one element from F_1. We again split, choosing $F_{11} = \{a_1\}, F_{12} = \{a_2\}$, and recurse. On the left branch, evaluating $B_{\{}a_2, a_3, a_4, a_5\}$ yields $\{r_2, r_4\}$: so we can discard all of F_{11}. On the right branch, we then minimize $F_{12} = \{a_2\}$ given $\{a_3, a_4, a_5\}$. We already know $\{a_3, a_4, a_5\}$ is not sufficient by itself, and F_{12} contains only one element, so we return $\{a_2\}$.

Returning to the top-level, we now wish to minimize F_2 given our (now reduced) $F_1' = \{a_2\}$. We observe $\{a_2\}$ obtaining no response, so partition F_2 into $F_{21} = \{a_4\}, F_{22} = \{a_3, a_5\}$. On the left branch, $B_{21} = \{a_2, a_3, a_5\}$ yields no response, so $\{a_4\}$ is kept. Then on the right branch, $B_{22} = \{a_2, a_4\}$ yields r_4, so we return $F_2' = \{a_4\}$.

At the top level, we then return our newly identified pattern $\{a_2, a_4\}$ and the corresponding response $\{r_4\}$.

Technical Evaluation

Calibration Ecalability

We evaluated calibration on a simulated arm/electrode array system. We could have used a FEM model (per (Keller et al. 2006), for example), however this would not have been able to separate the effects of signal- and spatial- calibration. As such, we adopted a simplified response model. The anterior side of our simulated forearm consists of 4 parallel strands of muscle (fingers) on the anterior side, crossed by three other strands (thumb flexor, and wrist flexor on each edge), laid out as illustrated in Figure 2. The same arrangement is reflected on the posterior side. Each selected pair distributes 1 unit of energy uniformly along a line between its endpoints. A joint is considered flexed (or extended) if all corresponding muscles are receive at least 0.2 units of energy. If both flexed and extended, the observed response is determined randomly in proportion to the difference in activation level.

We calibrated simulated fixed arrays with 7 and 12 electrodes on each side (configurations illustrated in Figure 2), and random arrays with 20 and 30 electrodes on each side. The set of possible actions are any (unordered) pair of electrodes on the same side.

We calibrated three sets of target patterns:

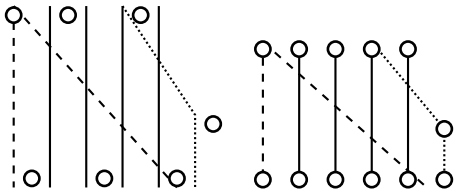

Figure 2: Simulated forearm topology, with flexors for fingers (solid), forearm (dashed) and thumb (dotted). Circles show electrode positions in the 7- and 12-electrode arrays.

- `joint`: flexion of each joint (with others permitted).
- `joint-only`: flexion of each joint independently (all others forbidden).
- `gestures`: a set of four gestures (derived from (Knibbe et al. 2017)). A fist (all fingers and thumb flexed), 'thumbs-up' (fingers flexed, thumb extended), pointing (one finger extended, three flexed, neutral wrist), and a 'stop' gesture (all fingers and wrist extended).

We conducted 200 tests for each array/target configuration, different random seeds and permuted action numbering. The tests are conducted with a budget of 200 queries for each target. Targets in a set are run with a shared knowledge base, so observations and inferences made for a previous target can be reused. If any target in a set exceeds its budget, we continue with the next target. Computational experiments were performed on an AMD Ryzen 5 3600 with 16Gb memory running Ubuntu 18.04. [1]

Figure 3 reports the number of queries evaluated during each test, and how successful we are at achieving target patterns. We see that our method is very robust on the small electrode array, achieving all target sets within the budget. As the number of electrodes (hence search space) and target complexity grows, the number of queries required to identify a pattern increases correspondingly. Nevertheless, for the (20 + 20) electrode configuration, we remain able to achieve complex gestures $> 50\%$ of the time.

The computation time for selecting queries unsurprisingly also grows with the number of possible actions, but remains quite modest. In practice we must wait for the muscles to return to a neutral state between queries, so we expect the introduced latency to be minimal.

Live Technical Evaluation

Following the simulated technical evaluation, we replicate our evaluation on the second author, for two purposes: (1) to verify the findings of our technical evaluation, and (2) to enable reflection on the limitations and future opportunities around this calibration approach.

Equipment We used a 24-channel EMS stimulation board, with uniquely controllable parameters per channel. In this case, we fix the stimulation parameters to 17V, 100Hz, and 300μs. We calibrate an array of seven $2.5cm^2$ randomly placed surface electrodes. We set a budget of 50 queries per

[1] Code is available at https://bitbucket.org/gkgange/stim-cal

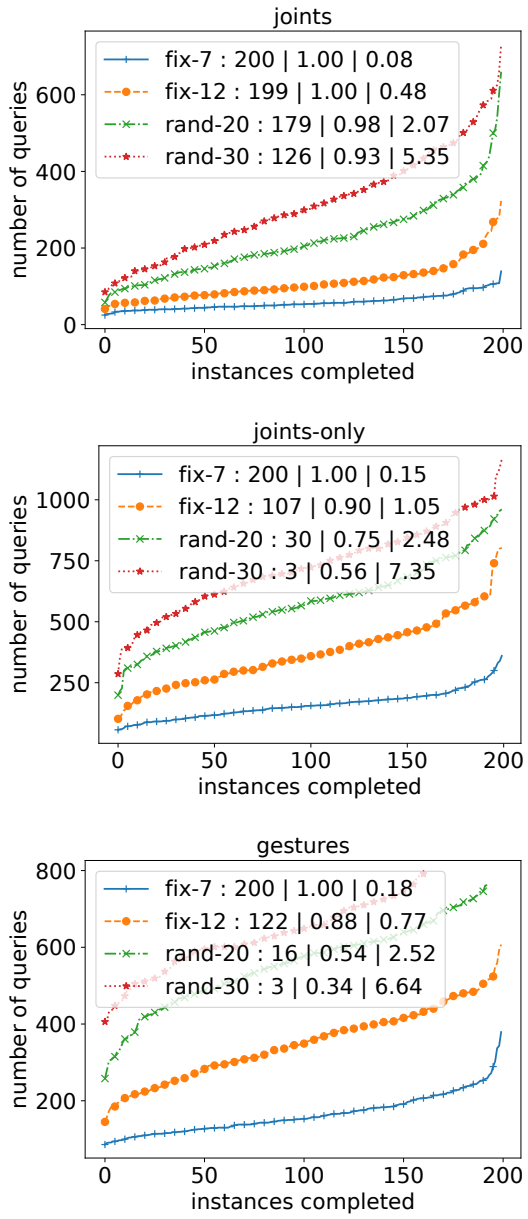

Figure 3: Query count during each run for simulated electrode layouts (ordered by increasing queries). We also report the number of complete successes (tests achieving all goals), mean success rate per test, and max time (in s) for any test.

target. We use a Leap Motion controller to monitor and feed back actuation results into the system. We consider a joint actuated if it demonstrates $> 30°$ angular change.

All `joint` patterns were found within 9 minutes. This includes the time taken to communicate with hardware and allow stimulation and actuation to occur. In that time, 35 electrode combinations were tested ($\sim$12s per test). A manual calibration process would have taken similar total time. The manual process would likely, however, stop at the first occurrence of a desired movement, where our technique seeks to reduce co-activations. For example, by the time our approach settled on the stimulation pattern for the ring finger, it had already observed that actuation ten times. In our 7-electrode configuration, we were unable to locate individual finger actuation (in the `joint-only` set, during 14 minutes of calibration and 74 tests) or two of the gestures (in the `gestures` set, during 15 minutes of calibration and 82 tests). We would not expect to be able to achieve that level of control from seven randomly-placed electrodes.

Next, we explored the stimulation resolution of an untested custom 18-electrode fabric array. We configured the query for wrist and finger abduction, following the `joint` sets above. The procedure identified all movements other than thumb abduction in 65 tests across 14 minutes. Anecdotally, as the participant, you become aware that a gesture will not be achievable (based on proprioception and haptic feedback), prior to the procedure reaching the same conclusion. This provides an interesting opportunity for user input.

Our approach relies on a numerous query budget to refine the electrode configuration for a joint, seeking to reduce noise and electrodes used. This optimises calibration at the expense of user fatigue. Whilst stimulation can be brief during calibration (e.g., 1-2 seconds per test), fatigue co-occurs with acclimation to the stimulation signal (Knibbe, Alsmith, and Hornbæk 2018). This would require the signal to be adjusted over time, necessitating user input.

To operate akin to manual calibration and lessen fatigue, the query budget could be reduced and the procedure configured to more aggressively select electrode configurations that fulfil the query criteria, rather than pruning towards an optimum. However, for complex configurations of electrodes and joint patterns, reducing the query budget will likely lead to missed results. In turn, what is needed is a more tightly-coupled human-in-the-loop approach, where users can dynamically alter stimulation parameters, veto uncomfortable electrode pairings, and manually break out of calibration queries. Further HCI-centric work should explore how best to couple the user to an auto-calibration approach.

Conclusion and Further Work

We have presented a novel automatic calibration procedure for EMS electrode arrays. The calibration procedure makes no assumptions about muscle position or electrode layout, instead using a SAT-based constraint solver to draw inferences from earlier observations and guide the search. In a simulated evaluation, our method reliably achieves complex gestures on electrode arrays of practical size using a modest number of attempts; and the required number of attempts scales reasonably with the number of electrodes. A live evaluation of the system shows the strength of our approach lies in its ability to explore and reveal the capabilities of electrode arrays, where complex stimulation patterns may reveal new joint control resolutions. Manual calibration approaches do not scale beyond the 8 electrodes currently seen in the HCI literature. Our procedure can enable calibration of higher-density arrays, where the calibration time requirements are dependent on the specifics of the hardware and the desire for human-in-the-loop integration.

References

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning* 47(2-3): 235–256. doi:10.1023/A:1013689704352. URL https://doi.org/10.1023/A:1013689704352.

Bouton, C.; Shaikhouni, A.; Annetta, N.; Bockbrader, M.; Friedenberg, D.; Nielson, D.; Sharma, G.; Sederberg, P.; Glenn, B.; Mysiw, W.; and Others. 2016. Restoring cortical control of functional movement in a human with quadriplegia. *Nature* 533(7602): 247–250.

De Marchis, C.; Monteiro, T. S.; Simon-Martinez, C.; Conforto, S.; and Gharabaghi, A. 2016. Multi-contact functional electrical stimulation for hand opening: electrophysiologically driven identification of the optimal stimulation site. *Journal of neuroengineering and rehabilitation* 13(1): 1–9.

Eén, N.; and Sörensson, N. 2003. Temporal induction by incremental SAT solving. *Electr. Notes Theor. Comput. Sci.* 89(4): 543–560.

Gange, G.; Berg, J.; Demirović, E.; and Stuckey, P. J. 2020. Core-guided and Core-boosted Search for CP. Technical report, Monash University. https://bitbucket.org/gkgange/geas/downloads/core-guided-cp.pdf.

Junker, U. 2004. QUICKXPLAIN: Preferred Explanations and Relaxations for Over-Constrained Problems. In McGuinness, D. L.; and Ferguson, G., eds., *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA*, 167–172. AAAI Press / The MIT Press.

Keller, T.; Lawrence, M.; Kuhn, A.; and Morari, M. 2006. New multi-channel transcutaneous electrical stimulation technology for rehabilitation. In *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, 194–197. IEEE.

Knibbe, J.; Alsmith, A.; and Hornbæk, K. 2018. Experiencing electrical muscle stimulation. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2(3): 1–14.

Knibbe, J.; Strohmeier, P.; Boring, S.; and Hornbæk, K. 2017. Automatic calibration of high density electric muscle stimulation. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1(3): 1–17.

Lopes, P.; Ion, A.; Mueller, W.; Hoffmann, D.; Jonell, P.; and Baudisch, P. 2015. Proprioceptive Interaction. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, 939–948. New York, NY, USA: ACM. ISBN 978-1-4503-3145-6. doi:10.1145/2702123.2702461. URL http://doi.acm.org/10.1145/2702123.2702461.

Lopes, P.; Jonell, P.; and Baudisch, P. 2015. Affordance++: Allowing Objects to Communicate Dynamic Use. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI 15, 25152524. New York, NY, USA: Association for Computing Machinery. ISBN 9781450331456. doi:10.1145/2702123.2702128. URL https://doi.org/10.1145/2702123.2702128.

Lopes, P.; You, S.; Cheng, L.-P.; Marwecki, S.; and Baudisch, P. 2017. Providing Haptics to Walls & Heavy Objects in Virtual Reality by Means of Electrical Muscle Stimulation. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, 1471–1482. New York, NY, USA: ACM. ISBN 978-1-4503-4655-9. doi:10.1145/3025453.3025600. URL http://doi.acm.org/10.1145/3025453.3025600.

Lopes, P.; Yüksel, D.; Guimbretière, F.; and Baudisch, P. 2016. Muscle-Plotter: An Interactive System Based on Electrical Muscle Stimulation That Produces Spatial Output. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST 16, 207217. New York, NY, USA: Association for Computing Machinery. ISBN 9781450341899. doi:10.1145/2984511.2984530. URL https://doi.org/10.1145/2984511.2984530.

Malešević, N.; Popović, L.; Bijelić, G.; and Kvaščev, G. 2010. Muscle twitch responses for shaping the multi-pad electrode for functional electrical stimulation. *Journal of Automatic Control* 20(1): 53–58.

Malešević, N. M.; Maneski, L. Z. P.; Ilić, V.; Jorgovanović, N.; Bijelić, G.; Keller, T.; and Popović, D. B. 2012. A multi-pad electrode based functional electrical stimulation system for restoration of grasp. *Journal of neuroengineering and rehabilitation* 9(1): 66.

Ohrimenko, O.; Stuckey, P.; and Codish, M. 2009. Propagation via Lazy Clause Generation. *Constraints* 14(3): 357–391.

Pfeiffer, M.; Dünte, T.; Schneegass, S.; Alt, F.; and Rohs, M. 2015. Cruise Control for Pedestrians: Controlling Walking Direction Using Electrical Muscle Stimulation. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI 15, 25052514. New York, NY, USA: Association for Computing Machinery. ISBN 9781450331456. doi:10.1145/2702123.2702190. URL https://doi.org/10.1145/2702123.2702190.

Popović, D. B.; and Popović, M. B. 2009. Automatic determination of the optimal shape of a surface electrode: selective stimulation. *Journal of neuroscience methods* 178(1): 174–181.

Popović, D. B.; and Popović, M. B. 2011. Advances in the use of electrical stimulation for the recovery of motor function. In *Progress in brain research*, volume 194, 215–225. Elsevier.

Popović-Bijelić, A.; Bijelić, G.; Jorgovanović, N.; Bojanić, D.; Popović, M. B.; and Popović, D. B. 2005. Multi-field surface electrode for selective electrical stimulation. *Artificial organs* 29(6): 448–452.

Usman, H.; Zhou, Y.; Metcalfe, B.; and Zhang, D. 2020. A Functional Electrical Stimulation System of High-Density Electrodes With Auto-Calibration for Optimal Selectivity. *IEEE Sensors Journal* 20(15): 8833–8843.

Zhang, L.; Madigan, C. F.; Moskewicz, M. W.; and Malik, S. 2001. Efficient Conflict Driven Learning in Boolean Satisfiability Solver. In Ernst, R., ed., *Proceedings of the 2001 IEEE/ACM International Conference on Computer-Aided Design*, 279–285. IEEE Computer Society.