Relating Spatial Skills and Expression Evaluation

Jack Parkinson
jack.parkinson@glasgow.ac.uk
University of Glasgow

Quintin Cutts
quintin.cutts@glasgow.ac.uk
University of Glasgow

Steve Draper
s.draper@psy.gla.ac.uk
University of Glasgow

ABSTRACT

Work connecting spatial skills to computing has used course grades or marks, or general programming tests as the measure of computing ability. In order to map the relationship between spatial skills and computing more precisely, this paper picks out a particular subset of possible programming concepts and skills, that of expression evaluation. The paper describes the development of an expression evaluation test, which aims to identify participants' ability to perform evaluations of expressions across a range of complexity. The results indicate participants' expression evaluation ability was significantly correlated with a spatial skills test (r=0.48), even more so when only considering those with less prior programming experience (r=0.58). Thus, we have determined that spatial skills are of value in expression evaluation exercises, particularly for beginners.

CCS CONCEPTS

• **Social and professional topics → Computing education**.

KEYWORDS

spatial skills, introductory computing, programming test, expression evaluation

1 INTRODUCTION

There is a connection between spatial skills and computing science [1, 4, 6, 7, 9, 10, 16], but we still have a very limited understanding of how this connection manifests. Spatial skills are more strongly associated with some areas of computing than others [6, 9], but typically work in this area consists of large-scale studies which use broad measures of ability. Each of these studies is important in painting a picture of the relationship: we now know that spatial skills are associated with code navigation [7], they are more strongly associated with programming specifically than other areas of computing [6] and they can be trained to improve computing outcomes [2, 4, 10]. They also correlate with success at many different levels in higher education [2, 6, 9, 10]. Every new piece of research expands our understanding of the nature of the relationship.

Our intention is to expand further by observing the interaction between spatial skills and a fine-grained computing task. Spatial skills are associated with broad measures of computing ability, like exam and course grades, but challenges arise in using these measures: they cover large portions of course content (often weeks of learning) and while they may be effective tools for identifying ability levels at individual institutions, they are high-level, specific to an institution and not available for inspection. They may be testing some fine-grained skills, but are likely testing much more.

By *fine-grained*, we mean to strip away the abstract layers of computing and observe specific processes of program comprehension. This isn't because we consider other elements of computing to be unimportant: rather we wish to determine whether spatial reasoning is applied broadly in programming or if its effects can be isolated to highly specific activities. We are also aware that there is more to computing than programming, but we do consider the ability to program to be a core part of computing education. Therefore, we present the development of a fine-grained test of one aspect of computing ability, focusing on a highly specific but widely applied facet of programming. We then use it in an experiment to compare results with spatial skills testing outcomes.

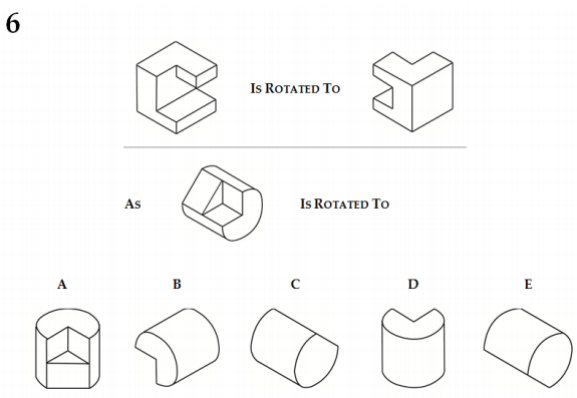

Figure 1: An item from the Revised PSVT:R *(answer: C)*

2 BACKGROUND

2.1 Defining Spatial Skills

A breakdown of spatial skills was a contribution of Parkinson & Cutts [9], who give an extensive, well-developed breakdown of the many factors within spatial skills. In brief, spatial skills lack a concise definition, but are fundamentally related to one's understanding of space and objects, particularly how they enable one to mentally construct internal representations of structures and manipulate them. Many spatial skills tests involve parsing flat figures, constructing mental models of represented structures and performing some change in their structure or orientation.

Different factors of spatial skills are best understood by considering tests used to measure them. The Revised Purdue Spatial Visualisation Test of Rotations (PSVT:R) is a test of spatial visualisation widely used in engineering and computing research, and is used in this study as a test of spatial skills (see figure 1) [18].

2.2 Measures of Computing Ability

Roughly in order of decreasing granularity, the following measures of computing ability have been correlated with spatial skills: **Academic attainment** by Parkinson & Cutts [9], who determined that spatial skills increase with academic progression; **Retention and GPA** by Veurink & Sorby [15], who observed that first years

students in computing and engineering showed higher average retention by spatial ability; **Programming module marks** by Jones & Burnett [6] and Parkinson & Cutts [10], who observed that introductory course module marks correlated with a spatial skills test; **Source code navigation** by Jones & Burnett [7], who used a combination of time taken and movements around and between files required for students as a proxy for navigation effectiveness, which correlated with spatial ability; **Programming tests** by Cooper et al. [4] and Bockmon et al. [2], who observed a correlation between spatial skills and the Advanced Placement (AP) Computer Science test and the SCS1R [1] respectively. Note that in most of these studies the authors examine more factors than those described, but for the purpose of this research we are interested specifically in the measures of computing ability used.

2.3 Understanding the Relationship

One may question, if spatial skills training has already been shown to be effective for computing students and the relationship has already been established with the SCS1R, why is it useful to expose the relationship with more fine-grained skills?

The first obvious reason is scientific curiosity. Even with all the evidence for the relationship which now exists, it is still an unexpected pairing. Indeed, Matti Tedre quotes an early learning psychologist in computing in his ITiCSE keynote this year [14] as having stated that practically one is never likely required to determine "whether two programs were the same if one was rotated 90 degrees" [17]. One would probably assume that Weinberg was being facetious; in this context, rotation of *programs* is not a requirement, but it appears that rotation *itself* does indeed correlate with success in computing, as ridiculous as it may sound.

The study presented in this paper also shows that the impact of spatial skills training is not simply affective, ruling out anecdotal criticism which could be levelled at Sorby's interventions [12]: one could argue that, by taking a proportion of students aside, dedicating more time to them and giving them more face-to-face contact with faculty, their engagement with their studies will improve regardless of the intervention's content. This experiment is designed to be completely removed of any affective factors, concerned with only a short test on either side of the relationship to see if the relationship still holds.

This kind of work helps us to determine not only a subset of *what* skills in computing depend upon spatial skills, but also *when* training may be of most value. For example, if it could be determined that spatial skills are only of value in navigating large programs, first-year students are not likely to need good spatial skills in order to grapple with their early learning. However, if spatial skills can be observed as interacting with one's ability to perform simple variable assignments, it would be beneficial for the students to have their spatial skills developed as early as possible, given how central variable assignments are to early-stage programming. Such a finding would also corroborate Margulieux's theory that spatial skills are of more value to novices who have yet to develop domain-specific strategies and rely on transferable skills which are exposed by spatial skills tests [8].

Finally, this kind of work opens up the possibility of finding more targeted methods of improving programming ability than a spatial skills training course. The intervention devised by Sorby [13] and used in computing [2, 4, 9] covers several areas of visualisation – we do not know, however, how computing skills interact specifically with each chapter. Given how widely it has been tested, 3D mental rotation clearly correlates with computing, but in what ways? In what areas of computing? What about symmetry, or construction from flat patterns? With answers to these questions we can refine and focus interventions to remove redundancy which may exist.

2.4 Expression Evaluation as a Fine-Grained Domain of CS

We selected **expression evaluation** as our starting point. We consider this to involve the comprehension of expressions at the structural and text-surface level, demonstrated (in this context) by the ability to hand execute them. A structural understanding of expressions is required to construct an internal model of execution. The ability to parse the text surface (i.e. elements in the atomic layer in the Block model [11]) is important to piece together which operations need to be compiled into the internal mental model of execution. We selected expression evaluation as the computing activity for the following broad reasons:

Flexibility of complexity. Expressions can range from trivial arithmetic combinations to highly complex data manipulations. There is flexibility within the domain of expression evaluation to generate a range of tasks which can vary substantially in complexity whilst still being rooted in the same procedures. Expressions act as a framework around which to build layers of difficulty so that we can observe how differently-skilled readers will evaluate them.

Contained, limited context. Evaluating expressions needn't rely upon a wide range of CS skills or apply a wealth of practical activities. One does not need to have a good understanding of problem-domain analysis, for example, which is often a requirement (or is at least of benefit) in code comprehension exercises. Expression evaluation appears to us as comprehension at its lowest, least-bloated level, with no need for reliance on other areas of, or skills involved in, CS. We can further constrain this context by limiting the operations involved.

Limited domain knowledge required. Even if students have the skills required to complete computing exercises but do not have the specific knowledge of the execution domain, they are unlikely to perform well. Computing has developed a multitude of jargon and domain-specific language that we often take for granted and would mean very little to outsiders. The keywords `print` or `println` are fairly common in programming languages, but someone without the relevant domain knowledge may associate this with a physical, paper printing function rather on-screen output. Any attempt at completing programming tasks requires at least two distinct parts: skills related to execution, and domain knowledge. While we acknowledge the importance of domain knowledge, we do not believe it is related to spatial skills (following the theories of Parkinson & Cutts [9] and Marguliuex [8]), so the authors intend to eliminate it as a factor as much as possible, and expression evaluation gives us a context in which we can do this effectively. As long as a reader has a basic understanding of the mechanical operations involved and the language-specific syntax, even novices can successfully complete expression evaluation tasks. Therefore, we can generate expression

evaluations which are complex and cognitively challenging to solve, but do not require extensive domain knowledge and thus add an extra factor of variance into the study.

Easy to practically develop and deliver. Expressions require very little setup or context to be valid. Expressions can be evaluated in isolation, in just a few lines of code. This makes them easy to develop en masse and also limits the materials and resources required to deliver the tasks as exercises. There is no need for a development environment, a codebase or a fully formed program to be developed: because of the highly limited context, expression evaluation tasks can be developed and delivered easily.

3 RESEARCH QUESTIONS

Based on prior work, which indicates that:

- a hypothetical cognitive relationship exists between spatial skills and program comprehension [8, 9]
- spatial skills are more strongly correlated with programming than non-programming modules [6]
- spatial skills correlate with success in a dedicated set of programming test questions [2]
- spatial skills are likely to be of more value to novices than experts in solving problems [8]

we hypothesise that a correlation between spatial skills and programming exists on a cognitive level, relating spatial skills to the ability to mentally execute programs, specifically program comprehension tasks. This leads us to ask two research questions:

- **RQ1:** Is there a correlation between spatial skills and the structural component of expression evaluation?
- **RQ2:** If so, does this correlation hold for both beginners and experts?

4 DEVELOPMENT OF A TEST OF EXPRESSION EVALUATION

Due to the selected participant pool (first year computing students at the authors' institution) the test is in Python, as our students are well-versed in Python and should have the appropriate domain knowledge to understand the expressions mechanically. The formulation of the test had two primary guiding principles:

- **"Low Python."** We wanted to limit prior understanding of Python specifically as a factor for accurately completing the test as much as possible.
- **"High Cognition."** Instead, we focused on the cognitive procedures involved in expression evaluation. E.g. interpretation of syntax, forming strategy, incorporating complex data structures and dynamically executing internally.

In short, although Python is a requirement for completing the test accurately, we wanted to flatten it as much as possible and develop a test which tapped into the cognitive strategies being applied.

Given that the participants' ability to complete the test would be directly correlated with the PSVT:R, it made sense to structure the tests similarly to reduce bias towards a participant's preferred style of testing. Therefore, we present in this section the development process for a test of Expression Evaluation in Python (EEP) consisting of 30 multiple choice questions, tailored to take 20 minutes.

4.1 Developing Expressions of Varied Complexity

Much like the PSVT:R, the EEP was intended to be developmental: each question was to be more difficult or complex than the previous. This process involved the authors generating a large number of expression evaluation tasks and devising a complexity ranking factor. The ranking factor was generated from two facets of complexity: operational and data complexity.

4.1.1 Operational complexity. This relates primarily to the number of distinct operations required to evaluate an expression and how "complex" the operations are. The number of operations was the primary metric considered, with the expectation that this would be the main factor determining the structure of the abstract syntax tree that the participant must construct and maintain, considerably impacting the developed mental model of the expression.

The operational complexity factor was also influenced by how "difficult" the operations involved were. In general terms a Python addition operator (+) was considered less "difficult" than exponentiation operations (**), and indexing into a list to extract one element is easier than extracting several elements with a slice. No such compounded complexity exists in the PSVT:R: every operation is a rotation in one axis by 90 degrees. The increased complexity of later questions arises from the combined number of operations required, but each individual operation is equally complex.

In order to generate authentic expressions, it simply was not possible to completely flatten the complexity and reliably order by the number of operations alone. However, we did select a limited subset of operations to be used in the questions to reduce the overall complexity and the effect of domain knowledge on the test. The subset of permitted operations was limited enough that we could group them into three tiers according to their perceived complexity. The only operations appearing in the test are:

- Tier 1: arithmetic operators +, - and negating variables
- Tier 2: indexing into lists with the form `list[i]`, using keys to extract values from dictionaries, built-in functions `str()` and `int()` and the arithmetic * operator
- Tier 3: extracting slices from lists and strings (`data[i:n]`)

The test also involved some very small locally defined functions, whose complexity was determined by the number of operations involved in the function body. For each question, each operation was counted and multiplied by its tier to give it an operational complexity metric. For example, consider the Python expression:

```
x = a * 3 + 7 - 5 + b - c
```

which has four Tier 1 operations (+, -, +, -: 4×1) and a Tier 2 operation (*: 1×2) so has an operational complexity of 6. Now consider the Python expression:

```
x = s[3] + s[1] - s[4]
```

which has two Tier 1 operations and three Tier 2 operations, giving it an operational complexity of 8.

4.1.2 Data Complexity. The other factor used in determining the order of questions was the complexity of the data structures involved. According to Parkinson & Cutts' model [9], particularly referencing notional machines [5], the structure of the expression (operations used and order of precedence) is not the only thing

which needs to be internally formulated in expression evaluation: the reader must also have a model of the data involved as part of the dynamic mental model. Again, the number of structures referenced in each expression was considered to be the most important metric in increasing complexity. Once again the data structures permitted were limited and arranged into tiers:

- Tier 1: scalar variables
- Tier 2: strings, lists and dictionaries
- Tier 3: nested lists (max depth 1) and dictionaries within lists

As with operational complexity, the number of distinct data structures used was the primary factor, multiplied as necessary by the each structure's tier to provide a data complexity metric.

The PSVT:R includes some figures which have enhanced complexity because they can be difficult to parse, i.e. in identifying the 3D structure from the 2D representation. Figure 2 includes some examples. In our test, we replicate the difficulty of hard-to-parse complex 2D representations of 3D shapes with hard-to-parse complex 2D representations of data, like nested lists. It is worth noting that, while claiming to be a test of rotation, the presentation of the PSVT:R has a persistent factor of ambiguity; some extreme cases are shown in figure 2, but there is an unwritten expectation upon the participant to parse line drawings as solids – this is not a trivial task, so the representation of "data" in the PSVT:R is fairly complex.

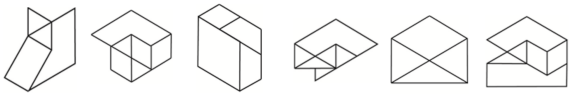

Figure 2: A selection of complex representations of 3D shapes from the Revised PSVT:R

4.1.3 Combined Complexity Factor. Once each question had been allocated an operational and data complexity metric, these were multiplied to provide the combined complexity factor. We use multiplication rather than addition because an additive system does not account for the compounding complexity of having to develop a robust mental model of both a program (a sequence of operations) and data demonstrated by the expressions listed in table 1.

We consider the third example to be considerably more complex than the other two since a complex model of operations *and* data must be generated, whereas in the other two the complexity is loaded in only one domain. If the combined complexity metric were additive, these three expressions would be considered of equal complexity, but this does not feel accurate. Combining multiplicatively adds more complexity to expressions which require robust mental models of both data and operations to be built.

4.2 Final Structuring and Presentation

To maximise similarity to the PSVT:R and reduce confounds which could have arisen from differences in test delivery, more steps were taken to align the tests. The EEP was made multiple choice, with possible answers A-E. In the PSVT:R, all multiple choice options presented to the subject are *possible* orientations of the object being rotated (i.e. they represent the same object) but cannot be achieved by following the correct sequence of rotations. We replicated this in the EEP by making most incorrect answers potentially achievable if the subject had misinterpreted or incorrectly applied operations (i.e. there were no completely irrelevant answers).

Data complexity: 2	Operational complexity: 8
`a = [3, 6, 7, 1, 5, 1, 4, 9]` `x = a[4] * a[5 - 3] + 6`	

Data complexity: 8	Operational complexity: 2

`l1 = [[4, 3, 9], [8, 3, 6]]`
`l2 = [[7, 3], [5, 1], [6, 6]]`
`a = 2`
`b = 5`
`x = l2[a]`	

Data complexity: 5	Operational complexity: 5

`l1 = [[6, 2, 8, 5], [9, 3, 6, 1], [3, 1, 7, 3]]`
`a = 3`
`b = 2`
`x = l1[b][a] + b`	

Table 1: A selection of sample expressions with the metrics of data and operational complexity noted

The questions were then arranged on paper to match the PSVT:R in style, down to the typesetting used and the format of the example questions provided. The purpose of this was to try and make sure that participants didn't get a distinct "head start" on either test due to the nuances of the way they were formatted or the examples provided. The final version of the EEP can be found at http://ccse.ac.uk/expressionevaluation.believe

5 TESTING THE RELATIONSHIP BETWEEN THE EEP AND THE PSVT:R

With the EEP developed, we went on to see if there was a connection between expression evaluation and spatial skills. We did this by having participants take both the EEP and the PSVT:R and performing an analysis of scores achieved on both sides.

5.1 Participants

Participants were invited from a pool of level 1 students (in both CS1 and CS0 courses) at the authors' institution. Students were invited to sign up during a lecture and through an online, cohort-wide communique. Amazon vouchers were offered as an incentive to students who completed both tests. 38 students took part.

5.2 Instruments

The two main instruments were the Revised PSVT:R and the EEP. There were three other artefacts given to the participants:

(1) **Python guidance sheet**, which gave the participants guidance on how each of the operations they would encounter worked and their order of precedence.
(2) **Initial Python test**, which required determining the outcome of individual operations (24 questions).
(3) **Post-test questionnaire**, a 6-question free-text question designed to extract skills and strategies employed.

Recall the guiding principles in section 4. One of the main principles this test was designed to fulfil was to reduce load on Python knowledge and focus on the cognitive process of performing an

expression evaluation regardless of the context. The guidance sheet was included to clarify any nuances of the language which could impact the cognitive process, like specifics of order of precedence. The test was included to identify participants who simply and clearly did not apply the mechanical Python operations properly; for the purpose of this study, we wished to eliminate participants who are unable to comprehend simple Python expressions since we are concerned more with their cognitive processes. In addition, participants were explicitly instructed not to mark any of the instruments with anything but the letter-response to answer questions.

5.3 Delivery

All materials were printed and issued in person under the supervision of a researcher who ensured that no conferring took place. Participants were provided with five minutes to complete the initial Python test and read through the guidance sheet before being issued one of the tests. Counterbalanced testing was conducted: half the participants received the PSVT:R first while the other half received the EEP first. Counterbalancing was necessary to account for test fatigue and the possibility that either of the tests had a training element on the other. Both tests were timed for 20 minutes, so after 20 minutes the participants were allowed a short break before starting the next 20 minute test.

Participants were asked not to write on the the question booklets for either of the tests, and were instead expected to supply the multiple choice letter answer to each question on dedicated answer sheets. The main purpose of this was so to avoid participants using the sheets as a cognitive aid to write intermediate values on.

6 RESULTS & DISCUSSION

6.1 Removed Participants

Before analysing the data, any participant who performed poorly on the initial Python test was to be removed. In this instance, no participant got more than one question wrong, so no participants were removed as a result of poor Python knowledge.

We also eliminated any participants who had excessive working-out on their answer sheets. The reasoning behind this decision was based on the expectation that all the working for these tests should be internal, as per instructions issued to the students before beginning the test. By writing out solutions step-by-step, the participants were eliminating the cognitive load required to complete the exercises. All of the working-out was in Python (intermediate values in expressions were written out, but no intermediate shapes in rotations were sketched), indicating that this would likely skew results in favour of the EEP. It is unclear whether participants who wrote out their answers did so because they did not understand the verbal instructions given at the start of the testing period, or because they felt that they could not complete the test without writing our intermediate steps. 5 participants were removed as a result of this analysis, resulting in a final cohort size of 33, which is the cohort described in the rest of this section.

6.2 Counterbalancing

Roughly half of the cohort (16 participants) took the PSVT:R first and the other took the EEP first. The difference between the scores of each group was not found to be significant in either test by an ANOVA, so we can assume that taking one test or the other first did not impact on the overall results of the experiment.

6.3 Correlation

In interpreting the effect size of correlations, we turn to Cohen's definitions of effect sizes according to r, with small, medium and large effect sizes considered to vary around .10, .30 and .50 respectively [3] – we also note, however, that context is important, and what Cohen determines as generally a large or small effect size may not be interpreted as such by other researchers. Since the data showed a skewed distribution, a non-parametric Spearman's rank correlation was determined between the PSVT:R and the EEP. The resulting r statistic was found to be 0.48 where $p<0.01$, indicating a significant, large-positive correlation between the results of the EEP and the PSVT:R. This is strong evidence that there is a connection between one's spatial skills and their ability to perform expression evaluation, which affirmatively answers **RQ1**.

6.3.1 Ceiling Effect / Homogeneous Upper Division. Table 2 shows a breakdown of the scores on each test. By chance, the spatial skills of the level 1 students who volunteered to take the tests were exceptionally high. According to other literature which has used the PSVT:R in computing related studies, this score is more comparable to Parkinson & Cutts' masters level students than level 1 students.

	Mean	Max	Min	SD
PSVT:R	24.61	30	16	3.81
EEP	23.36	30	13	3.92

Table 2: Breakdown of scores achieved in the Revised PSVT:R and the EEP

Sorby, who has been using the PSVT:R and other spatial visualisation tests for years to determine whose spatial skills are not at an acceptable level to take part in entry-level engineering (and thus require supplementary training) has the following breakpoints: a student scoring 18 and below needs training, a student scoring 19-21 is marginal (would likely benefit from training but does not necessarily require it) and scores of 22+ are considered safe passes [13]. As can be seen in figure 3, the vast majority of the scores attained were solid passes, indicating that if the students were in Sorby's engineering classes they wouldn't require training. If we consider this group to be more or less homogeneous – as Sorby does – then we must assume that attempting to observe a correlation between these high scores and the EEP scores will not be as strong as one observed with participants with more spread out scores.

It is possible that upon hearing the description of the experiment, some students opted not to volunteer because they have an aversion to spatial tasks (i.e. have poor spatial skills) and therefore only students with strong spatial skills opted to take part, thus raising the average substantially. We expected that by offering a cash-equivalent incentive this would not be an issue, but this appears not to be the case. As such, we expect a ceiling effect to have occurred in the spatial skills test, potentially lowering the correlation.

6.4 Splitting the Cohort

In an attempt to answer **RQ2**, we collected data from the students on which course they elected to take at the start of the year: CS1

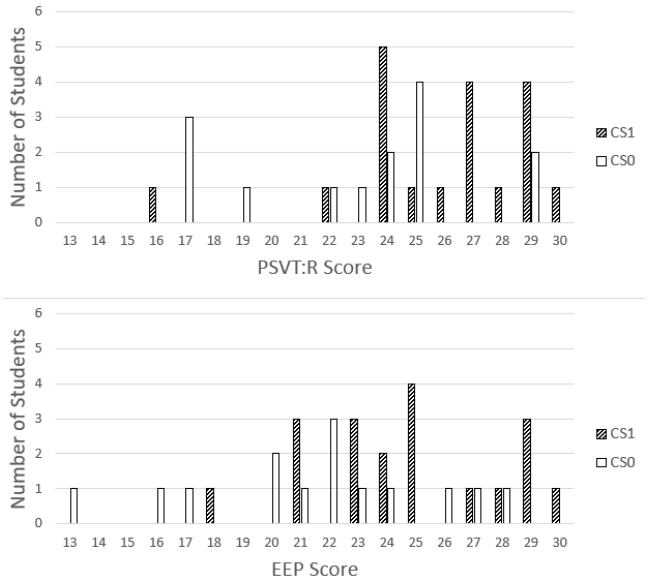

Figure 3: Scores achieved on each test, separated by cohort

or CS0. This is not a perfect distinction between *novice* and *expert*, but in terms of expression evaluation, our knowledge of the cohorts dictates that CS0 students are unlikely to have performed any expression evaluation in a formal programming setting before beginning the academic year, whereas the CS1 students are likely to have been programming for more than a year prior to starting, so are likely to be proficient – even expert – in expression evaluation.

Cohort	Spearman's r	p	n
CS0	0.58	<0.05	14
CS1	0.28	0.24	19
Total	0.48	<0.01	33

Table 3: Correlation coefficients, p values and participant numbers for correlations between the revised PSVT:R and the EEP, broken down by self-selected student cohort

The correlation is substantially higher for the CS0 group, and the CS1 group show a medium, insignificant correlation. If we consider that a large proportion of CS1 scored highly to begin with and are essentially indistinguishable from noise at the high end, this result is to be expected. Conversely, the slightly "less-noisy" CS0 group is showing a much stronger correlation. This aligns with Margulieux's theory of novices being more likely to apply transferable skills than experts, who use domain specific strategies. This suggests that we can answer **RQ2** with a "no", however we are hesitant to make strong claims due to a) the possibility that the upper division is largely homogeneous in spatial ability and b) the distinction between expert and novice is not specific to expression evaluation.

A supplementary observation which can be made regarding these cohorts: the CS1 cohort elected to take a more advanced programming course because they already had programming experience; the CS0 cohort is designed for complete beginners, who self-identify as having little to no programming experience. Although the average PSVT:R scores of both of these groups are high (25.8 and 22.9 respectively) the CS1 cohort scored substantially higher. This is in line with Parkinson & Cutts' findings: those further along in their academic computing career have better spatial skills on average.

6.5 Summary

We have observed that there is a large correlation between participants' spatial skills and their ability to perform expression evaluation when controlling as much as feasibly possible for prior knowledge, affirmatively answering **RQ1**. We also observed that the correlation is higher for CS0 students, which could be described as novices compared to CS1 experts; this answers **RQ2**, however we would want to repeat the experiment with a clearer distinction between expert and novice before making any firm claims.

7 CONCLUSION

We have examined existing research on spatial skills and computing science to date, which associates spatial skills with success in programming and specifically with a pseudo-code programming test. The aim of this research was to dig deeper into a very fine-grained area to see if spatial skills can be associated with one's ability to perform expression evaluation. This has been demonstrated, and is an interesting discovery because it shows that spatial skills – or the underlying cognitive abilities which spatial skills expose – have a strong connection with a central part of programming (even in the early stages) which in turn is central to studying computing. It also appears that this relationship holds much more strongly for novices than it does for experts, though our distinction between these groups is imprecise. This discovery adds another piece of the puzzle representing our understanding of the relationship between spatial skills and computing, serving as part of a body of work contributing towards utilising this relationship for the benefit of computing students and beyond.

REFERENCES

[1] Ryan Bockmon, Stephen Cooper, Jonathan Gratch, and Mohsen Dorodchi. 2019. (Re) Validating Cognitive Introductory Computing Instruments. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. 552–557.

[2] Ryan Bockmon, Stephen Cooper, William Koperski, Jonathan Gratch, Sheryl Sorby, and Mohsen Dorodchi. 2020. A CS1 Spatial Skills Intervention and the Impact on Introductory Programming Abilities. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 766–772.

[3] Jacob Cohen. 1992. A power primer. *Psychological bulletin* 112, 1 (1992), 155.

[4] Stephen Cooper, Karen Wang, Maya Israni, and Sheryl Sorby. 2015. Spatial skills training in introductory computing. In *Proceedings of the eleventh annual International Conference on International Computing Education Research*. ACM, 13–20. https://doi.org/10.1145/2787622.2787728

[5] Benedict Du Boulay. 1986. Some difficulties of learning to program. *Journal of Educational Computing Research* 2, 1 (1986), 57–73.

[6] Sue Jones and Gary Burnett. 2008. Spatial ability and learning to program. *Human Technology: An Interdisciplinary Journal on Humans in ICT Environments* (2008). https://doi.org/doi/10.17011/ht/urn.200804151352

[7] Sue Jane Jones and Gary E Burnett. 2007. Spatial skills and navigation of source code. *ACM SIGCSE Bulletin* 39, 3 (2007), 231–235. https://doi.org/10.1145/1268784.1268852

[8] Lauren E Margulieux. 2019. Spatial Encoding Strategy Theory: The Relationship between Spatial Skill and STEM Achievement. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. 81–90. https://doi.org/10.1145/3291279.3339414

[9] Jack Parkinson and Quintin Cutts. 2018. Investigating the Relationship Between Spatial Skills and Computer Science. In *Proceedings of the 2018 ACM Conference on International Computing Education Research*. ACM, 106–114. https://doi.org/10.1145/3230977.3230990

The version of copy of this manuscript appears in UKICER '20 at https://doi.org/10.1145/3416465.3416473

[10] Jack Parkinson and Quintin Cutts. 2020. The Effect of a Spatial Skills Training Course in Introductory Computing. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (Trondheim, Norway) *(ITiCSE '20)*. Association for Computing Machinery, New York, NY, USA, 439–445. https://doi.org/10.1145/3341525.3387413

[11] Carsten Schulte. 2008. Block Model: an educational model of program comprehension as a tool for a scholarly approach to teaching. In *Proceedings of the Fourth international Workshop on Computing Education Research*. ACM, 149–160.

[12] Sheryl Sorby, Norma Veurink, and Scott Streiner. 2018. Does spatial skills instruction improve STEM outcomes? The answer is 'yes'. *Learning and Individual Differences* 67 (2018), 209–222. https://doi.org/10.1016/j.lindif.2018.09.001

[13] Sheryl A Sorby. 1999. Developing 3-D spatial visualization skills. *Engineering Design Graphics Journal* 63, 2 (1999).

[14] Matti Tedre. 2020. From a Black Art to a School Subject: Computing Education's Search for Status. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. 3–4.

[15] Norma L Veurink and Sheryl A Sorby. 2011. Raising the bar? Longitudinal study to determine which students would most benefit from spatial training. In *American Society for Engineering Education*. American Society for Engineering Education.

[16] Jonathan Wai, David Lubinski, and Camilla P Benbow. 2009. Spatial ability for STEM domains: Aligning over 50 years of cumulative psychological knowledge solidifies its importance. *Journal of Educational Psychology* 101, 4 (2009), 817. https://doi.org/doi/10.1037/a0016127

[17] Gerald M Weinberg. 1971. *The psychology of computer programming*. Vol. 29. Van Nostrand Reinhold New York.

[18] So Yoon Yoon. 2011. *Psychometric properties of the revised purdue spatial visualization tests: visualization of rotations (The Revised PSVT: R)*. Purdue University.